Schema-based design and the AKIRA Schema Language: An Overview

Giovanni Pezzulo[1] and Gianguglielmo Calvi[2]

[1] ISTC-CNR, Via S. Martino della Battaglia, 44 - 00185 Rome, Italy
giovanni.pezzulo@istc.cnr.it
[2] Noze s.r.l., Via Giuntini, 25 int.29 56023 Navacchio, Cascina (PI), Italy
gianguglielmo.calvi@noze.it

Abstract. We present a theoretical analysis of *schema-based design (SBD)*, a methodology for designing autonomous agents architectures. We also provide an overview of the *AKIRA Schema Language (AKSL)*, which permits to design schema-based architectures for anticipatory behavior experiments and simulations. Several simulations using AKSL are reviewed, highlighting the relations between pragmatic and epistemic aspects of behavior. Anticipation is crucial in realizing several functionalities with AKSL, such as selecting actions, orienting attention, categorizing and grounding declarative knowledge.

1 Introduction

In the last two decades several theoretical and computational models have been inspired, directly or indirectly, by theories of sensorimotor and cognitive development [9, 11, 53] that describe schematic structures, or 'schemas', as crucial in behavior and cognition. The term 'schema' was firstly introduced by Bartlett [8] to mean a map or structure of knowledge stored in long-term memory. Successively Piaget [53] described schemas in a more operational sense, roughly as mental representations of some physical or mental actions that can be performed on an object or event. He considered schemas as the building blocks of thinking, and the basic structure underlying behavior and cognition (in a process that he described as 'assimilation and accommodation').

One tenet of schema theory is that schemas are specialized subsystems realizing a tight coupling between perception and action. A schema can be used for recognizing a specific entity (say a dog) or a class of entities (say journalists or swimmers), or for controlling a specific action (say opening a door or skying). Some of these schemas may require parameters to be filled in. There can also be more complex schemas for planning sequences of actions, as well as for more complex cognitive operations such as doing inferences. Central to schema theory is not only what schemas can individually do, but also how they are organized and what they can collectively do.

This view has inspired several other researchers in cognitive science, artificial intelligence and cognitive robotics. In these fields several schema-like structures

have been proposed, including *frames* [43], *scripts* [57], *schemas* [3, 4, 21, 44, 46, 60], *neural schemas* [39], *semiotic schemas* [55], and *behaviors* [12, 38]. Architectures including distributed and competitive functional units are often referred to as 'behavior-based' or 'schema-based'. Several integrated frameworks have been proposed for designing them; among the most popular ones, we can mention the behavior-based approach proposed in [6], the *NSL/ASL* in [65] and the *Robot Schema (RS)*, a formal language for designing robot controllers proposed in [37] which includes perceptual and motor schemas.

Since the term 'schema' has been used in several contexts, it has assumed several senses, too. For example, Piaget referred mainly to *sensorimotor schemas*, highlighting their action-oriented nature in contrast with other data structures that only include conceptual knowledge. Schemas for processing stimuli or controlling the perceptual apparatus are often referred to as *perceptual schemas*, while those for controlling locomotion, reaching or grasping are often referred to as *motor schemas*. Another important distinction is between *anticipatory schemas*, that include predictive components, and *reactive schemas*, that do not; and consequently between anticipatory and reactive schema-based architectures.

Reactive vs. Anticipatory Schema-Based Architectures. One important distinction among schema-based architectures is their *reactive* or *anticipatory* nature. Originally, the label 'behavior-based' has been used as a synonym of 'reactive' [5, 12]. Reactive schema-based architectures, that respond quickly to dynamic environments, have challenged traditional AI models which rely on slow and costly deliberation. They are now de facto a standard in autonomous robotic systems [56]. However, recently several schema-based architectures have been proposed which include anticipatory mechanisms, such as inverse and forward internal models, and which generate and exploit expectations about the next sensory stimuli [13, 18, 21, 47, 67]. These anticipatory aspects are inspired by psychological theories of action control [30, 34], indicating that anticipated effects of (possible) actions play a fundamental role in regulating the agent's behavior. Several neurobiological evidences also suggest internal models and in particular forward models as plausible candidate mechanisms [40, 66]. See also [22] for a comprehensive review of neural correlates of anticipation in the brain.

Aims and structure of the paper. The main contribution of the paper is twofold: illustrating the schema-based design methodology and its peculiarities, and presenting a comprehensive framework and an implementation environment for anticipatory schema-based architectures. Accordingly, in the rest of the paper we firstly introduce *schema-based design (SBD)* as a methodology for building autonomous agents architectures. We then present the *AKIRA Schema Language (AKSL)*, a framework for designing and implementing anticipatory schema-based architectures, and we review simulations realized with it.

2 Schema-Based Design (SBD)

Schema-Based Design is a methodology for designing artificial systems. It is inspired by ethological and neuroscientific empiric evidence [3, 4]; and many schema-based architectures are directly inspired by ethological models, such as the praying mantis in [5], the computational frog in [2], and the computational cockroach in [10]. In SBD the functional aspects are more stressed than the actual realization and localization of schemas in the brain: several researchers find schema useful exactly because they provide an intermediate level of representation between the neural and the personal level [3]. In SBD cognition and behavior are explained in terms of schemas and their dynamics: behavior is not controlled by an unique process, but emerges from the dynamic competition and cooperation of several active schemas. More complex cognitive functionalities can emerge both by sophisticating the schemas and by permitting them to interact in more complex ways.

2.1 What's in a schema?

Schemas are coarse-grained functional units, being approximately at the same level of description of ethological and neurobiological units of automatic action control such as *detect prey* or *escape* [5, 47]. They do not only contain conceptual knowledge, but are strongly action-oriented and include perceptual and motor elements. Schemas consist of actions and sensory information organized around, and serving to realize, a *goal* or a set of related *goals*. In their simplest form, schemas can be described as sets of rules having the form *condition* → *action* or *condition* → *action* → *expectation* (respectively in the case of reactive and anticipatory schemas), that can act in parallel or in series, and whose success corresponds to the achievement of a goal.

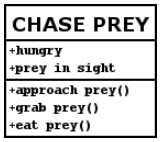

Fig. 1. A sample schema: *chase prey*

As an example, Fig. 1 illustrates the main functional components of a sample motor schema, which is named after its goal: *chase prey*. It includes two sample triggering conditions: *hungry*, that indicates the value of a drive, and *prey in sight*, that indicates the presence of specific stimuli in the visual field. It also includes three actions[3]: *approach prey*, *grab prey*, *eat prey*. They can be imple-

[3] In this example each action is represented as a localist sub-unity of the schema. However by using a distributed representation scheme, multiple actions can be embedded implicitly, say in a single neural network; see for example [62, 63].

mented as rules, or set of rules, which receive perceptual input and send motor commands such as 'go left' or 'go right' to the motor apparatus of the agent.

Schemas have four main properties: goal-orientedness, flexibility, selectivity, and excitability.

Goal-orientedness. The goal-centered behavioral organization of a schema is its first property. This aspect also distinguishes schema-based systems from production systems and classifier systems [13, 21, 31, 45], which also use rules and rulesets. Related views are the *ideomotor principle* [34] in psychology, the *TOTE* [42] model in cybernetics, and the definition of goal-orientedness provided by Gallese and Metzinger [24]: "Action control actually equates to the definition of the action goal: the goal is represented as a goal-state, namely, as a successfully terminated action pattern".

Flexibility. Goal-orientedness does not imply, however, that schemas have only one way to realize their goals: they can flexibly realize their goals under variable contingent conditions and by exploiting a (limited) repertoire of actions. This property can be called *flexibility*.

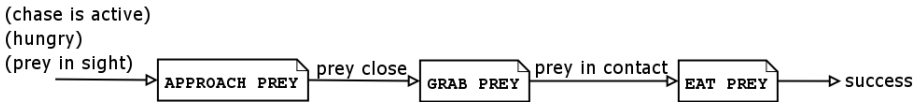

Fig. 2. A sample sequence of actions realized by the *chase prey* schema

When *chase prey* is active, its actions can be triggered in a different timing and order, or be skipped, depending on the context. In the example illustrated in Figure 2, the three actions are simply concatenated. *Approach prey* is immediately triggered, since the two preconditions, *hunger* and *prey in sight*, still hold if the schema is active. If *approach prey* succeeds, it produces as a consequence *prey close*, which in turn triggers *grab prey*, and so on. The success of the last action, *eat prey*, also entails the success of the whole schema. The failure of one of the actions can instead trigger another action, or produce a context in which no actions are suitable, and thus lead to the failure of the whole schema.

Selectivity. An important consequence of schemas' goal-orientedness is their *selectivity*: in order to realize their goal, schemas do not need (and can not process) all the possible information from the environment. On the contrary, they select, attend to, and use only stimuli which are relevant for its specific goal. This implies that when a schema is operating the action-perception loop of the agent has both pragmatic effects (realizing the goal via triggering actions) and epistemic ones (gathering relevant stimuli).

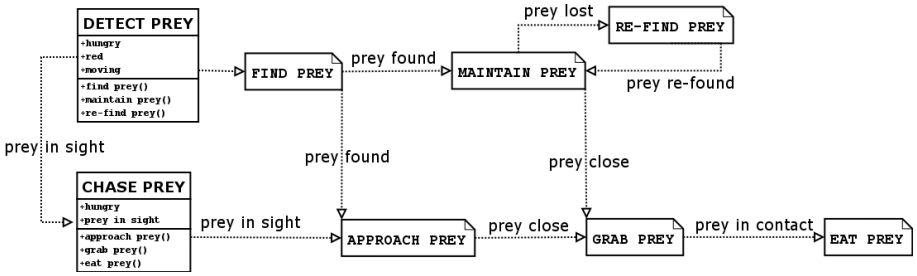

Fig. 3. Possible sequences of actions realized by a perceptual and a motor schema

In several schema-based frameworks epistemic and pragmatic aspects are implemented in different schemas: perceptual and motor. However, these schemas are either embedded one in another, [3, 4], or they can pass sensory information [47]. The dotted lines in Figure 3 indicate the functional relations between one perceptual and one motor schema (*detect prey* and *chase prey*) which can pass sensory information. *Detect prey* includes as triggering conditions specific stimuli such as *red* and *moving*. It includes as actions three specialized strategies for finding preys the first time (*find prey*), maintain it in the visual field (*maintain prey*), and find it again if it is temporarily lost (*re-find prey*). The edge from *detect prey* to *chase prey* indicates that the former schema can trigger the latter (this is the meaning of the triggering condition *prey in sight*).

In this example the actions of the *chase prey* motor schema can be triggered by (the success of) other actions both in the same schema or in the related perceptual one, *detect prey*. Moreover, the perceptual schemas can convey to the motor schemas sensory information, for example the position of the prey to approach (not shown in the picture). Several courses of actions can emerge by the interactions of the two schemas in different contexts, depending on which functional relations are actually exploited. For example, a prey can be lost during tracking: the failure to *maintain prey* triggers *re-find prey*. Or, the prey can be captured without being lost, and in that case *re-find prey* is never activated. Notice that not all the functional relations are shown; in particular, those resulting from failure are not (for example, if a prey is lost both *maintain prey* and *approach prey* fail).

Excitability. The last important property of schemas is their *excitability*: they have a variable *activity level*. As we will discuss in detail, the activity level represents its relevance and desirability in the current situation. It can depend on motivational factors, such as active drives, and on contextual factors, such as the presence of appropriate stimuli or of other active schemas. Each schema gains resources, such as access to sensors and effectors, and as a consequence the possibility to influence the overall behavior of the agent, in a measure dependent on its relative activity level. This leads us to the next topic, which is the organization and functioning of a whole schema-based architecture.

2.2 Schema-based architectures and cooperative competition

A schema-based architecture is distributed; it includes several schemas specialized for different goals, such as *detect prey* and *detect predator*, or for realizing the same goal under different contexts, such as several instances of *detect prey* specialized for different kinds of preys. All schemas cooperate, triggering one another and exchanging sensory information, and compete for gaining priority over sensors and effectors: only some of them can be (partially) active at once. Such *cooperative competition* from which behavior emerges is considered a fundamental brain principle [3, 4, 43]. Competitive cooperation can be implemented in multiple ways, but it exploits three principles of self-organizing systems: (1) local excitation (e.g. active drives and schemas can excite other schemas); (2) global inhibition (e.g. schemas and drives inhibit concurrent ones); (3) emergence: behavior emerges from the influences of several schemas that can be active at once.

Since commands from several schemas can be fused, a schema-based architecture can realize complex patterns of actions, most beyond the possibilities of single schemas. An important aspect of schema-based design is the possibility to realize systems which can fulfill more that one functionality, selecting the most appropriate one on the basis of contextual factors such as current drives/goals, stimuli and expectations. We can say that the most challenging aspect of SBD is not implementing one specific functionality, but understanding how all them coordinate and realize a complex system having habits and fulfilling its drives/goals while remaining responsive to opportunities and affordances in the environment. In several architectures schemas can also be arranged hierarchically [18, 27, 51, 54]; in this case top-down influences channelize behavior in accordance with expectations generated at the higher level, while the system remains responsive to stimuli-driven bottom-up dynamics.

2.3 Pragmatic and epistemic aspects of SBD

Schemas are evolved by organisms to successfully interact with entities in their environment, realizing their own goals. In producing behavior, schemas are thus selected for their expected success in action; if there are several schemas for realizing the same goal in different contexts, the most fit is selected. Since schemas are selected for action according to their activation level, a high activation level of a schema encodes a high confidence that it is succeeding or it will succeed in realizing its goal state (e.g. in *detect prey*: *the prey can be detected*). In summary:

> The activity level of a schema encodes a degree of confidence that it will succeed.

Although schemas are evolved for pragmatic reasons, their functioning also entails several factors which can be considered epistemic, in the sense that they are directed to acquire or process information. In traditional AI architectures these operations are traditionally dealt with by manipulating explicit representations, for example comparing an observation with an expectation, assigning a

confidence level to an assumption, or matching an expectation. These operation can instead be dealt with *implicitly* and *procedurally* in SBD.

In the literature of dynamical systems it is often assumed that embodiment and structural coupling permits using information that is not explicitly represented in the system. For example, the environment can be used as an external memory, provided that the agent's sensors can access it often –a prey continues to serve as a trigger of some schemas as long as it remains visible. In a similar way, certain pragmatic states of the schemas, such as their successes or failures, implicitly encode epistemic content. For example, it is possible to interpret the success of a *follow prey* schema as an indication that there is a prey, without any need to explicitly represent such state of affairs.

Therefore we discuss two functional equivalences: the degree of activation of a schema corresponds to (1) the truth of its assumptions and (2) to the desirability of its consequences, without any need to explicitly represent them (but, as we will see, we can derive explicit knowledge from them).

The first functional equivalence There is a functional equivalence between the success of a schema and the assumption that the state of affairs that it permits to deal with is true. This is due to two reasons. First, action success or failure depends on epistemic assumptions and conditions that are verified or falsified by acting. Second, since the activity level of the schema depends on its success rate, it also indicates a confidence level that the behavior is appropriate and thus the entity to deal with is indeed there. As an example consider again the *detect prey* schema; in order to successfully track a kind of entity, the preys, the schema must be specialized to deal with prey-relevant features. Since in order to gain activation the schema has to succeed in actually matching these features, success of action is also a confirmation of such assumptions and expectations: *a prey is here, or will be here in the near future*. The functional equivalence between the success of the action and the truth of its assumptions and predictions is the *first functional equivalence*:

> The success of a schema indicates that specific (actual or expected) states of affairs, encoded in its assumptions and expectations, are true.

We can now come back to the example in Figure 3. When the preconditions of *detect prey* are verified (e.g. *red* is verified by the compliance of a visual routine), the schema triggers its actions. In turn, the compliance of its actions continuously verifies its preconditions (the success of *find prey* verifies *red* and *moving*) and produces new conditions in the same or in other schemas (the success of *find prey* produces *prey found*, which is a precondition of *approach prey*). This means that actions in two schemas are triggered by epistemic assumptions, which in turn are verified by the compliance of other actions.

As discussed above, several information can be implicitly dealt with. All the conditions shown in the picture, such as *prey found* or *prey close*, do not need to be explicitly represented, nor it is needed that symbolic information is passed among the schemas or the actions (all the labels are only for the designer's sake).

Their functional meaning is implicitly encoded in the functioning of the schema mechanism, and in particular in the schema's activity level. The activity level of a schema, in fact, implicitly encodes the degree of confidence in its implications. If a schema can access the activity level of other ones, it can use this information 'as if' it was an explicitly reported condition. For example, an high activity level of *detect prey* implicitly encodes conditions (e.g. *prey found*), that are informative for *chase prey*. Further contextual elements, such as the state of the schema or the presence of specific stimuli, help *chase prey* disambiguating the information and triggering different actions such as *approach prey* or *grab prey*.

Very often epistemic information is graded and not crisp. For example I can be more or less sure that a prey is in front of me. As a corollary of the first functional equivalence, the degree of certainty, or confidence, in an assumption can be formulated according to the degree of success of the shcema or action:

> The activity level of a schema is a measure of confidence in its assumptions.

Categories and Beliefs. Thanks to the first functional equivalence schemas can not only be used for acting: their success also entails an implicit categorization of the entities to deal with, and implicitly represents beliefs such as 'there is a prey now'. Bickhard's interactivism [11] suggests a similar perspective: if an active interaction fails, then the 'indication for action' (and the content of the representation) is false. This fact has two main implications. Firstly, contrary to the typical pipeline information-processing scheme *perception* → *categorization* → *action*, a prey is categorized as a prey because of the compliance of the *detect prey* schema, and not vice versa (actually, there is a loop between all these factors). Secondly, categories and beliefs do not need to be explicitly represented anywhere, since the current activity level of schemas already indicates them. As an example, in Section 4 we present a simulation in which such dynamical, action-related categorization is realized.

As proposed by Piaget [53], in humans there is a progressive conceptualization of information which is initially only procedural[4]. However, only part of information implicitly used by the schemas becomes explicitly available, for example for categorization, and internally manipulable when coupling is broken; some information remains instead procedural. For a discussion of accessibility and awareness of procedural information used in the control of action, see [23].

Epistemic Actions. We have discussed how actions can also have, as a side effect, an epistemic value for the system. For example, knowing that the ball is there, is round, is soft, etc., is a form of implicit knowledge that in some cases can

[4] Some assumptions which are much more 'profound' and invariant are conceptualized very late, when they are. Consider the assumption 'under normal circumstances, the world is quite stable'. In order to remain successful for a while, several (if not all) schemas implicitly rely on this assumption, which is not however conceptualized. We could say that also the functioning of the whole schema-based system encodes several important assumptions about the world.

be internalized. But there is another, more sophisticated way for a system to obtain information by exploiting the first functional equivalence: performing an action with the aim to know something about the world (e.g. 'control if' or 'look whether'). That is, I can turn on the light in order to know whether or not the circuit works well, and not because I need light. In this case, we can distinguish between the *pragmatic action* (action in the most common sense) and the *epistemic action*, that is aimed at gathering information: the pragmatic action (turning on the light) is only a vehicle of the epistemic one (knowing if the circuit works well). This example illustrates that an action can have both pragmatic and epistemic value, and it can be executed for the former or the latter necessity. It is also worth noting that in order to know something about the world we need to act on it (either actually or 'in simulation', see [26, 28]).

In Section 4 we will present a simulation showing the two main consequences of the first pragmatic principle: (1) on the basis of pragmatic actions, either real or simulated, epistemic states such as beliefs can be formulated; (2) epistemic actions are possible, too: some pragmatic actions, actually performed or simulated, can be triggered for knowing something, and not for their pragmatic effects. In both cases we interpret a belief as the result of an epistemic action, which can be either implemented through a pragmatic action, or explicitly executed by means of a pragmatic action. This view has an important implication: all cognitive operations involving beliefs, such as reasoning, refer to (and have their meaning thanks to) actual or possible pragmatic actions.

The second functional equivalence Organisms have motivations, and their actions are determined by their needs. In schema-based design this is modeled through motivational units, such as drives, causing schema activations: in this way there is no need to manipulate and reason explicitly on utility and values of entities in the world. As a consequence, typically a high activity level of a schema also encodes the fact that it has been learned to, and is expected to be effective for satisfying the organism's needs: a primitive, implicit form of means-ends reasoning. Again, success of action strengthens and tends to confirm the relationship between a schema and the satisfaction of an organism's need. Since the organism can have competing motivations, typically the schemas for realizing the most important or urgent ones are assigned the highest activity level and are thus selected (but of course, as far as they do not penalize one another, more than one schema can be selected). This implies that:

> The activity level of a schema is a measure of desirability of (the consequences of) its success.

In some cases the organism faces challenges that have to be dealt with very quickly. As an example, consider an organism successfully following a prey. If an unanticipated danger occurs, such as a predator, or the organism is near an abyss, it has to quickly change its behavior and activate another schema, whichever the activity level of the *follow prey* schema is. This means that a sudden change in

allocation of resources among schemas has to occur, depending on how promptly the new situation has to be dealt with:

> The rapidity of increase in activation of a schema is a measure of its urgency.

The two factors, one epistemic and one motivational, which correspond to a high activity level in a schema seem to be at odds: for example, a high activity level of the schema *detect prey*, which depends on the organism being hunger, also corresponds to the belief or expectation that there is a prey somewhere, which could not be the case. However, in organisms there is a relation between the epistemic criterion, that is maintaining true assumptions, and the motivational criterion, that is pursuing its needs. In order to succeed (and consequently to satisfy its goals) an organism needs to maintain its epistemic states, and this is why schemas are designed for implicitly checking their conditions. For this reason an organism motivated by hunger can 'bet' on the success of the schema for *catching preys*, maintaining artificially (against evidence) a high activity level even if it is unsuccessful at the moment. Since activating a schema also means inhibiting other ones, this strategy is only good if the schema will indeed succeed in the near future, otherwise the organism will die. This means that the organism is selected by evolution to have 'good guesses' and to fuel schemas that will succeed and, in that way, verify their assumptions: the first functional equivalence is not broken, only postponed. The correspondence between the desirability of a behavior and the truth of its assumptions is the *second functional equivalence*:

> The activity level of a schema is a measure of confidence that it will be successful, and consequently that its assumptions and expectations will become true.

Notice that this principle works due to the fact that, at the end, each schema predicts its own success, too. The teleonomic structure of a schema can be thus represented as *condition* $\rightarrow$ *action* $\rightarrow$ *expectation* $\rightarrow \ldots \rightarrow$ *expectation* $\rightarrow \ldots \rightarrow$ *action* $\rightarrow \ldots \rightarrow$ *success*. A schema is selected by evolution because of the adaptive advantage of (the implications of) its success. Its structure guarantees the desirability of its intermediate actions as well as the meaningfulness of the assumptions and expectations it produces during its execution. Schemas are learned for dealing successfully with the environment: maintaining correct representations and betting that they will be useful are two sides of the same coin.

3 The AKIRA Schema Language (AKSL)

The AKIRA Schema Language (AKSL) has four main components: schemas, drives, routines, and actuators. It is build on the top of the AKIRA simulation framework [1] and integrated with the Irrlicht 3D engine [33] and the Ikaros simulation framework [32][5].

[5] The sourcecode of AKSL is available in the AKIRA website [1].

3.1 Schemas

Schemas can be described as tuples (det,inv,for,urg,rel,app,con,act,thr). The first three parameters represent their components:

- det is a detector[6], i.e. the selector of a certain kind of stimuli (each schema only processes some information which has been learned to be significant)
- inv is an inverse model, deciding a motor command to send to an effector
- for is a forward model, calculating the expected next stimuli

Basically each schema has a cycle in which: (1) the detector collects sensory information (received by perceptual routines) and the sensory expectation (received by the forward model), compares them (dotted circle in Fig. 4: the degree of mismatch is used for calculating rel, see later), and sends a sensory input to the inverse model; (2) the inverse model, on the basis of the input received, calculates a motor command and sends it to the effector (camera or wheel motors); (3) the forward model receives an efference copy of the final motor command (of the camera or wheel motors), generates a sensory expectation and sends it to the detector.

The cycle of each schema is run asynchronously and in parallel with each other, with an amount of computational resources (speed and memory) that depends on its *activity level*. Five other parameters are used for calculating the activity level at the beginning of each schema's cycle:

- urg is the urgency value, representing how promptly the schema has to be executed when its contextual conditions are met. This parameter is very high in schemas which have to deal with risky situations, in which an immediate action is needed.
- rel is the reliability value, representing how much that schemas is (expected to be) successful in the current situation. The reliability value is set according to the degree of match of the expectations generated by the forward model for with respect to the actual stimuli.
- app represents the appropriateness with respect to currently active drives and goals. It is a learned parameter.
- con is a learned contextual parameter that depends on the activity level of other schemas. Schemas can in fact evolve links with an hebbian-like mechanism explained in [49], which permit to transfer activation. This associative mechanism permits mutual priming of schemas that are often active in the same situations.

[6] Optionally more sophisticated operations can be realized inside the detector. For example, as in Kalman filtering [36], a reliability value can be assigned to stimuli and expectations, and the final input be calculated as their weighed sum. In this case, if the stimulus is lacking or inaccurate, the expectation can be used for (partially) replacing it. Another possibility is to erase from the stimulus the self-generated part, predicted by the forward model. This functionality is useful e.g. for avoiding tracking our own hand when it is in the visual field. Moreover, as in Smith predictors [61], the prediction of the forward model can be fed at different time intervals for compensating long loop delays.

– *act* represents the total activity level of the schema, which sums up the epistemic and motivational factors. It is calculated as $urg + rel + app + con$. Thus, the final activity level of a schema represents how much the schema is expected to be both effective (successful) and desirable in the given context (according to current drives/goals and other contingent factors).
– *thr* is a threshold for sending motor commands. Under the threshold the schema functions normally but its motor commands to the actuators are inhibited.

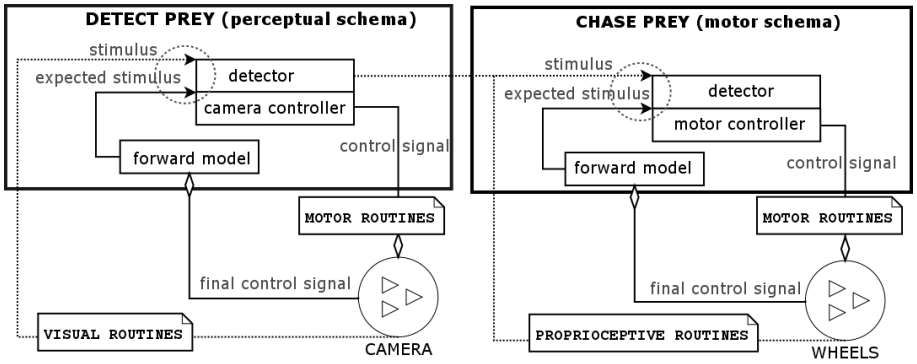

Fig. 4. Example of the coupled perceptual and motor schemas.

Coupled perceptual and motor schemas Perceptual and motor schemas can be *coupled*. The functioning of two sample coupled schemas, *detct prey* and *chase prey*, is illustrated in Figure 4. The perceptual schema receives as input perceptual information from the camera (data are preprocessed by perceptual routines). As indicated by the dotted circle, sensed stimuli are compared with sensory information that is predicted by the forward model, and the error is used for setting the reliability value *rel* of the schema. The detector thus sends sensory stimuli to the controller (inverse model), which in turn generates a motor command and sends it to the camera motor (via motor routines), and (optionally) sensory information (e.g. the position of the detected prey) to the coupled motor schema. The motor schema receives as input the activity level of the coupled perceptual schema, proprioceptive information about the current state of the wheels' motor, and optionally additional sensory information from the perceptual schema. Like in the perceptual schema, sensed and predicted stimuli are compared and reliability values are assigned. Sensory information is conveyed to the controller, which sends motor commands to the wheels' motor (via motor routines). Notice that in both schemas, in order to generate predictions, the forward models receive an efference copy of the (final) motor commands received by the camera or wheel motors, and learn to predict their sensory effects.

Three modalities Schemas can operate in three distinct modalities: (1) generation; (2) simulation; (3) imitation.

The mode *generation* is the default one and serves for generating behavior appropriate to the context. The functioning is the one previously explained.

The mode *simulation* is used for predicting the long-term effects of schemas. When a schema runs in simulation the motor commands generated by its inverse model are inhibited and not sent to the actuators; however, efference copies are sent as usual to the forward model, which generates expectations and sends them to the detector. Inside the detector, expectations are not matched against stimuli (and their accuracy can not be calculated), but directly fed to the inverse model. The simulation mode thus produces a loop between the forward and inverse model, generating simulation of possible courses of events which extends over several steps. Schemas in simulation mode can also be run faster than real time, thus actually simulating several steps beyond[7]. The simulation mode can be used for producing, testing and selecting in advance multiple alternative courses of events 'proposed' by different schemas. For example, if an agent has alternative schemas for navigating, by running them in simulation it can explore 'virtually' (and not by trial and error) its environment. This permits to foresee possible dangers that can arise during navigation, to anticipate if by following a path it is actually possible to reach a target location, or to calculate which is the shortest path to a target location by comparing the time spent for simulating the alternative ones. In principle all schemas can be run in simulation, regardless of their actual reliability and activity level (but notice that simulating is a costly operation). However, only schemas whose reliability value is significantly high are able to generate predictions which are adapt to the current context.

The mode *imitation* serves for understanding (and possibly reproducing) behavior observed in a demonstrator (see [17]). When schemas run in imitation mode the perceptual state (observed in the demonstrator) is fed to the inverse models which generate the motor command that would have produced in that situation. The motor commands to the actuators are inhibited, but the efference copies are fed to the forward models which generate the next predicted perceptual state, which is thus compared with the next perceptual state (observed in the demonstrator). This process roughly corresponds to the question: "which of the schemas could have generated the perceptual states I observe?", the answer being the schema(s) which are accurate in predicting. By knowing that, the agent is now able both to understand the demonstrator's actions, and to imitate them. Of course depending on the differences between the agent's and demonstrator's behavior repertoires, imitation can be more or less accurate.

[7] A more complex possibility is running in simulation the whole schema-based system. In this case, predictions generated by one schema are also fed to other schemas. The effects of the motor commands on actuators and sensors are simulated, too, and then predictions replace sensory information in the whole system. This mechanism permits to test in advance not only the long-term effects of single schemas, but also their combinations. Notice that it is impossible to use the schema-based system in generation and simulation at the same time.

3.2 Determining the activity level of schemas

Schemas in AKSL couple perception and action via anticipation: they learn to associate stimuli to behavior which produces appropriate expected results[8]. As already discussed, thanks to their dual nature, schemas are well suited for both pragmatic activity such as guiding behavior, and for epistemic activity such as categorization. This is due to the fact that their activity level depends both on their reliability, and on their appropriateness with respect to current drives/goals.

Schemas are assigned a reliability level depending on how well they predict the sensorimotor flow. The reliability of a perceptual schema is a confidence level that a certain entity, encoded in the schema, is expected to be present. For example, if the schema *detect prey* has a high activity level, not only it can be assumed that it is successfully tracking the prey (with the camera), but also that there is, or there will be, a prey in the visual field. For this reason, the most important aspect of perceptual schemas is their epistemic side: if an architecture has several perceptual schemas, they can be seen as competing hypotheses for representing/categorizing the current perceptual situation, such as *detect prey* vs. *detect predator*. The reliability of a motor schema is instead a confidence level that the behavior encoded in the schema is applicable in the context.

As an example, consider the *catch prey* schema. The most important aspect of motor schemas is pragmatic: they can be seen as competing behaviors, or as different means to realize the same behavior. However, even perceptual schemas have relevant pragmatic aspects (they orient attention by moving the camera) and motor schemas have epistemic aspects (they contribute to categorization). Each schema has thus both aspects, since its success implies both achievement (of action) and categorization (of object/event).

The current motivational state of a schema-based agent also influences the activity level of schemas. This means that when a schema-based agent has *hunger* or is *fearful* its schemas for detecting and catching preys in the former case, or for detecting and escaping from predators in the latter case, will gain activation. In this way the agent's pragmatic activity is oriented toward desired goal states, such as preys or hiding places. Attention is channelized toward relevant stimuli, too: the agent spends many resources in deciding whether or not there is a prey than, say, deciding whether or not there is an hiding place. This is obtained by drives (such as *hunger*) providing activation to relevant perceptual schemas (such as *detect prey*). This happens even when there are not preys in the visual field; and since a high activity level of *detect prey* can in turn be interpreted as an evidence that there is a prey, this process causes visual imagery: the agent 'imagines' what it is searching for. However, this is only temporarily: although *hunger* can artificially maintain a high activity level for some time, if there are

[8] Several machine learning methodologies have been used in literature for learning the inverse and forward models, and for evaluating the degree of mismatch between stimuli and expectations; for example, in [67] responsibility signals are used. AKSL currently permits to use both fuzzy logic and feed-forward or recurrent neural networks libraries; see [49].

no preys the schemas for detecting and catching them will still be not relevant and thus will lose activation.

There is also a general constrain on how much activation can be assigned to schemas. Activation, to be divided among schemas, is in fact limited. Schemas compete for acquiring resources: they can not access them while they are used by other schemas, but have to wait until they are released (the mechanism is based on the AKIRA Energetic Model, explained in [49]). This means that active schemas inhibit one another via the allocated resources but without lateral inhibitions. By modifying the total amount of activation available it is also possible to channelize behavior in different ways, since few or many schemas can be active at once.

3.3 Motivations and Routines

AKSL permits also the design of simple motivational systems: as in several ethological studies, drives such as *hunger* and *fear* can be implemented. We have also included in AKSL several routines such as *detect red* or *move left* for pre-processing information (e.g. sensory data).

Although they have very different roles, drives and routines are implemented in a similar way. Each drive and routine embeds a simple operation. In the case of drives, this may consist in a 'biological clock' (e.g. raising the activity level of *hunger*). In the case of routines, this may consist in reading the value of a sensor, or sending a motor command to an actuator. Schemas do not receive input from sensors and do not send output to actuators: three kinds of routines (perceptual, motor and proprioceptive) have the role to mediate between them.

Drives and routines have an activity level *act* and can exchange activation with other components. For example, drives typically fuel appropriate schemas, and schemas fuel routines (and vice versa). The more a drive is active, the more it can fuel the schemas which satisfy it; thus drives introduce a motivational influence on the agent's behavior. The more a routine (e.g. *detect red*) is active, the more it reliably finds a pattern in the sensor (e.g. *red* is detected). Energetic links between schemas, drives and routines can be learned via a hebbian-like system. The rationale is that schemas whose success reliably satisfy drives become associated with them, while schemas become associated with routines that provide useful input or output facilities. See [47] for the details.

3.4 Sensors and Actuators

Perceptual and motor routines receive input from sensors such as a camera. The actuators (camera or wheel motors) receive asynchronous commands from the motor routines and perform command fusion (libraries based on fuzzy logic and mixture of Gaussians are available; see [49]). In many systems in the literature (see [14] for a review) several schemas can be partially active at once but only one is selected for commanding the actuators. In our model each active schema sends concurrently its motor commands to the actuators via the motor routines. Since more active schemas receive more resources and can perform their operations

faster, they also have a higher firing rate when sending motor commands. The actuators fuse all the motor commands asynchronously received; this means that a schema with an higher activity level (and as a consequence sending commands with higher firing rate) also influences the actuators more. Notice that this has not necessarily only impact on the agent's movements. Since one of the actuators is the camera motor, that actuator also determines which sensory information the agent pays attention to.

3.5 Comparison with related literature

AKSL shares resemblances with other schema-based architectures such as [3, 37]. With respect to them, the two main differences are the presence of internal models and the fact that perceptual and motor schemas are separate units which can however be coupled. AKSL shares resemblances with MOSAIC [67] and HAMMER [18], too. Differently from them, AKSL uses a parallel architecture in which the activity level influences directly the amount of computational resources (speed and memory) assigned to each schema, without explicitly calculating responsibility values. The second difference is that schemas compete for limited resources and active schemas inhibit other ones. The third relevant difference is that commands are received and fused asynchronously by the effectors; a high activity level permits schemas to have a higher firing rate of and thus to send more commands. Lastly, AKSL also permits hebbian-like learning and spreading activation between the schemas, which can thus provide activation to one another, being in the same or in different hierarchical layers. See [49] for the details of the architecture.

4 Exemplar Capabilities of AKSL

We have used AKSL for addressing several research fields. Here we review our simulations in the fields of (1) action selection and attention, (2) category formation, (3) simulation of future behavior, (4) grounding, and (5) hierarchical control of action.

4.1 Action selection and attention

Recently several anticipatory schema-based systems have been proposed for action control in robotics which base action selection on predictive success, both in distributed approaches [62, 63] and in localist ones, such as MOSAIC and HAMMER (but other exist, [64]). They use a combination of forward and inverse models for generating competing motor plans for the same or for different targets, and the models predicting better are selected for the control of action. This responds to two related questions: which action is preferable given the sensory and goal context? Which schema can successfully actuate the action? These questions become related if success of prediction is used for action selection: a *successful* schema performs an action (and satisfies a drive/goal), thus a schema

predicting well also predicts its own success. On the contrary, not only schemas which fail, but also schemas which are expected to fail, can be assigned less activity. Differently from several schema-based models, in MOSAIC and HAMMER there is not a one-to-one correspondence between a schema and a behavior, but each behavior (e.g. 'grasp teapot') is realized by the cooperation and competition of several schemas, specialized for different contexts (e.g. 'light' or 'heavy' cup). Thus, competing models generate alternative motor plans, such as *grasp full teapot* vs. *grasp empty teapot*, which are selected according to the basis of how accurately the models predict the right sensorimotor flow. When a motor command is generated for grasping a teapot, an efference copy is used by the forward models in the two modules for generating the sensory consequences under two different contexts. These predictions are thus compared with actual sensory feedback, and the most appropriate one is selected for action control. Commands of the two modules can be combined linearly, providing generalization.

In [47] we have described a schema-based architecture (see Fig. 5), inspired by an ethological model of the praying mantis, which shares resemblances with MOSAIC and HAMMER but uses AKSL. In that architecture predictions generated by the forward models are not only used for determining schemas' reliability, but also used for orienting the motor apparatus (camera and wheels). *Perceptual schemas* gather information relevant for the current task, and orient the camera toward relevant inputs (e.g., relevant colors and trajectories), also determining part of the next stimuli, like in *active sensing*. *Motor schemas* select the most appropriate motor action (e.g., specialized for following or escaping from quick or slow, big or small entities). The first novelty of this architecture is the possibility to deal with multiple concurrent drives, whose urgency changes over time (depending on internal regulatory mechanisms or by external stimuli). The challenge is generating the appropriate behavior for satisfying the currently active motivation. Depending on the current motivational state, affordances offered by the environment are selected: for example, an hunger agent tries to catch prey, while a fearful agent that is escaping from a predator avoids prey. The second novelty is that perceptual and motor processes are integrated in the same framework and coupled, and the agent is able to orient attention for gathering information necessary for satisfying its current needs. The two aspects, determining behavior and orienting attention, are closely related in AKSL thanks to the coupling of perceptual and motor schemas.

4.2 Category Formation

Schema activity has (real or anticipated) epistemic implications; it can be exploited for categorizing and distinguishing objects from background. In an anticipatory framework, since the activity level of schemas depends on their predictions, objects and categories are defined by the typical, coherent patterns of (expected) transformations under a given set of agent's actions. For example, in [21] it is investigated how to build $action \rightarrow effect$ sensorimotor schemas through interaction with a simple environment. Moreover, the agent interac-

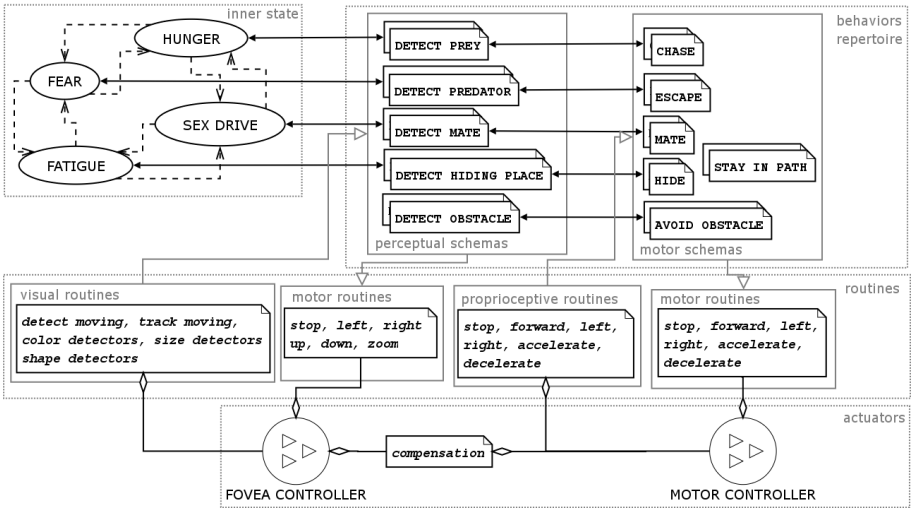

Fig. 5. The schemas-based architecture of the praying mantis in [47]

tively enlarges its ontology by learning new objects (called *synthetic items*), and this in turn permits to learn incrementally new abilities and competences.

AKSL has been used for categorization, too. In [48] we have demonstrated how to develop *perceptual categories* (such as different types of insects) and *abstract categories* (such as two roles played by those insects, predators and preys), on the basis of the theory of *perceptual symbol systems* [7]. Shortly, the agent learns to activate the most relevant schemas, specialized for dealing with features of the entities (such as color and size) and is able to track, follow or escape the entities by means of the dynamics of collaboration and competition among the schemas. The specific novelty of AKSL comes from the possibility to evolve energetic links among schemas in an hebbian-like way. In such a way 'clusters' of active schemas emerge during interaction with a given entity (say insect_one or predator) behave as *simulators* in the sense of [7]. They can then generate a simulation of categories of objects or events by rehearsing and priming the associated schemas, even in absence of environmental cues. Thanks to simulative capabilities, specific runs of a simulator reenact the multimodal experience of a category, while adapting to the current situation.

4.3 Simulation of Future Behavior

Simulative theories of cognition [7, 26, 28] suggest that by rehearsing the motor programs for interacting with an object an agent can anticipate the sensorial stimuli it will receive and simulate the consequences of its motor commands one or many steps beyond current time. Recently several neuroscientists [20, 41] proposed that the cerebellum and the basal ganglia could create a loop permitting

to simulate and select multiple alternative courses of actions, providing neural support for these theories.

Simulation permits to realize multiple functionalities. Some of them are related to the immediate control of action: for example, actual stimuli can be replaced when the sensors are unavailable or unreliable [19]. More complex and future-oriented capabilities are possible, too. The alternative courses of events can be evaluated before acting, for example with a *somatic marker* mechanism [15]: if the sensory predictions have already been experienced and had been categorized as negative, the schemas generating them can be stopped by a 'command from the future' (see Fig. 6). A simulated exploration of the environment can also be exploited for selecting a plan: the effects of several competing plans can be produced off-line, and their expected sensory consequences evaluated against actual or expected drives/goals. Neuroscientific evidence indicates that reward prediction is used for selecting, for example, a path in a maze [58].

Fig. 6. Simulation and Long Term Effects. The long term effects of schemas are predicted, and the expected states evaluated. This mechanism can be used for exciting/inhibiting the schemas depending on their long-term effects. If the long-term effects of several schemas are generated and compared, this mechanism can also be used for selecting among alternative courses of actions (simulative planning).

Simulative capabilities have been used in cognitive robotics, too. For example, in [25, 68] internal simulation of the sensory consequences of multiple possible motor actions is used to perform robust planning in the presence of noise. Similarly, in [16] a simulative process is used for checking in advance if the selected behavior will cause problems in the future. This method permits to avoid the costs of performing complete planning, since only the usual behavior path is checked in anticipation. Simulative capabilities have also been used in social tasks, such as imitation, joint attention, plan recognition, perspective taking, prediction of intent, etc.; for example, the HAMMER architecture has been used for modeling all these aspects [18, 17].

Schemas in AKSL, if used in the *generation mode* already simulate few steps in the future for the sake of predicting the next sensory stimuli. If a prediction of the long-term effects of the agent's actions is instead needed, they can be used in *simulation mode*. Running schemas in simulation permits to realize several novel functionalities, such as realizing a 'somatic marker' mechanism, and simulative planning. Up to the moment we have only conducted preliminary experiments on these topics; in our future work we plan to continue investigating them.

4.4 Grounding

Since schemas can be run in simulation, the agent is able to self-generate sensory information that would be provided by the environment as a consequence of its actions. This means that not only actually performed actions provide an epistemic access to the environment, but also simulated ones; as a consequence, not only experienced objects and events can be grounded, but also these 'virtually' experienced via simulated interaction. In [29], for example, internal simulation of possible trajectories is used for grounding concepts related to navigation; for example, distance from obstacles is grounded and estimated by running simulations until they encounter the obstacle. Dead-ends are recognized through simulated obstacle avoidance, while passages are grounded in successfully terminated simulations of navigation.

In the schema-based framework proposed by [55] expectations about the sensorimotor flow are used instead for grounding the meaning of words and sentences in natural language. Words for perceptual features are grounded into sensory information; for example, 'red' is grounded in some (expected) values of the robot's sensors. More complex attributes are grounded thanks to (actual and potential) actions. Concepts for objects which are for example reachable or graspable are grounded by schemas which regulate actual behavior and at the same time encode predictions of the consequences of expected interaction. For example, the meaning of 'sponge' is the set of expected consequences of own actions over a sponge (e.g. the anticipated softness) which constitutes the grounding of the word.

In [50] AKSL has been used for designing a 2-layered architecture, corresponding to the two systems for automatic and willed control of action in [46]. The lower layer, which is called *sensorimotor*, is very similar to the already presented architecture of the praying mantis, but includes schemas for navigating in a simulated house. The higher layer, which is called *deliberative*, includes instead declarative knowledge (beliefs and goals states) and pre-compiled sequences of schemas (plans) for navigating the house scenario; this layer is used for reasoning. One novelty of this system is that beliefs are dynamically added to the deliberative layer on the basis of how schemas perform in the sensorimotor layer. For example, the belief *the door is open* is added when the schema(s) for traversing the door is being successful, or is expected to be successful. Not only actual actions, but also simulated ones have been used for forming beliefs: in this case schemas have been used in the *simulation mode*. The rationale is that 'I can believe that the door is open since I expect that, if I try to traverse it, my attempt will succeed' (I also have to assume that the context for acting will be appropriate; for example, I have to start the action in front of the door). The system can also perform explicit epistemic actions (for example in order to check if a belief is true). By exploiting the same machinery that serves for building up the belief x, I can know under which conditions I will come to believe x by using counterfactual reasoning (e.g., what do I have to do in order to know whether or not the door is open?). We have argued that beliefs which are built in this

way are grounded, and their verofunctional value can be verified or falsified on the basis of success or failure of schemas in the sensorimotor layer[9].

4.5 Hierarchical Control of Action

Several hierarchical architectures exist in literature for action control and orienting of attention [18, 27] in which schemas that include representations at different level of abstraction are used. We have used AKSL for hierarchical control is the above mentioned 2-layered architecture. Plans in the deliberative layer are simply pre-compiled sequences of schemas. If a plan is selected by reasoning, it influences the dynamics of the sensorimotor layer by triggering schemas in sequence, much in the way drives do. Depending on the amount of resources assigned to the deliberative layer (and to plans), this influence can constrain more or less the behavior of the agent.

Another example of use of AKSL for hierarchical control is the architecture for visual search in [51]. Similarly to 'pandemonium' models [59], schemas at the higher layers encode increasingly abstract representations and expectations. They learn to predict the activity level of those at the lower layers, and expectations produced at the high level canalize in a top-down way search at the lower level, while bottom-up error signals serve mainly to confirm or disconfirm concurrent running hypotheses. Empirical evidence exist for a hierarchical organization of the visual apparatus; a comprehensive theoretical framework and implementation is *predictive coding* [54]. This approach is also consistent with simulative theories of cognition. According to Grush [26], simulations can nest to produce increasingly abstract levels of description, in which the criteria for 'matching' are increasingly distant from perceptual matching, although they remain grounded on (actual or simulated) sensorimotor interaction.

5 Conclusions

AKSL permits to design and implement *anticipatory* schema-based architectures, and anticipation plays a crucial role in realizing several functionalities. For example, action selection is influenced by the anticipatory capabilities of the schemas, and in particular of their forward models. One of the elements for assigning activity level to a schema is its accuracy in predicting the next sensory input in case one pattern of actions is selected. Attention has an anticipatory component, too: predictions generated by the forward models permit to orient attention toward the expected position of entities such as a prey to detect, or toward parts of the environment in which the agent expects to find information relevant for its current task. Category formation depends on anticipation, too, and in particular by expectations generated by several schemas at once. Schemas specialized for features of the same entity are likely to have coordinated patterns of prediction

[9] Some beliefs are about other beliefs and not about stimuli. In all cases, however, a relation (direct or indirect) can be identified with the sensorimotor layer.

and for this reason can evolve energetic links in an hebbian-like way. Categories thus emerge as clusters of schemas which are expected to be more active during interaction with the same entities. Simulative capabilities, that are based on the substitution of actual stimuli with self-generated expectations, permit a number of other functionalities, such as generating and comparing alternative courses of events on the basis of their long-term effects, or grounding concepts on the basis of self-generating sensory stimuli. Lastly, in designing hierarchical architectures, an interplay of top-down and bottom-up signals are used which convey sensory expectations and prediction errors.

Anticipation is thus crucially involved in several functions, from simplest to more complex ones. But is it really advantageous to anticipate in all circumstances? Running the forward models, and especially using schemas in simulation mode, is very costly in terms of time and computational resources, and the adaptive advantage of predicting the future could be lost is this means less responsiveness to the contingencies of the environment. We have conducted several preliminary experiments (reported in [47, 48]) and compared schema-based models having anticipatory and reactive strategies (i.e. without the forward models) in several tasks. We have found a significant adaptive advantage of anticipatory strategies when the agent has to deal with complex and dynamical environments offering multiple possibilities for action, while this could be not the case if only simpler tasks are required to the agent; see also the experiments reported in [35].

These experiments seem to indicate that a selective pressure for developing anticipatory capabilities, and consequently using anticipation as a 'lever' to develop increasingly complex functionalities, could be the increased level of complexity and dynamicity of the environment. More anticipation also permits an agent to deal successfully with more drives and motivations, since it can allocate attentive resources and orient its behavior by taking into account present but also future needs. In turn, more motivations and more anticipation make the environment of the agent increasingly complex, and thus demand for even more anticipatory capabilities; for a discussion, see [52]. Of course, it is an open challenge to understand which is the level of complexity and dynamicity of the environment which makes anticipation advantageous, and for this reason we plan to continue using AKSL in the future in a number of simulations comparing reactive and anticipatory strategies.

6 Acknowledgments

This work is supported by the EU project **MindRACES**, FP6-511931. We wish to thank Martin V. Butz for his many helpful comments and discussions.

References

1. akira, 2003. http://www.akira-project.org/.
2. M. Arbib. Levels of modelling of mechanisms of visually guided behavior. *Behavioral and Brain Science*, 10:407–465, 1987.

3. M. Arbib. Schema theory. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence, 2nd Edition*, volume 2, pages 1427–1443. Wiley, 1992.
4. M. A. Arbib. *The metaphorical brain 2: Neural networks and beyond*. Wiley, New York, 1989.
5. R. Arkin, K. Ali, A. Weitzenfeld, and F. Cervantes-Prez. Behavioral models of the praying mantis as a basis for robotic behavior. *Robotics and Autonomous Systems.*, 32(1):39–60, 2000.
6. R. C. Arkin. *Behavior-Based Robotics*. The MIT Press, 1998.
7. L. W. Barsalou. Perceptual symbol systems. *Behavioral and Brain Sciences*, 22:577–600, 1999.
8. F. C. Bartlett. *Remembering*. Cambridge University Press, Cambridge, 1932.
9. E. Bates. *The Emergence of Symbols*. Academic Press, 1979.
10. R. D. Beer. *Intelligence as Adaptive Behavior: An Eperiment in Computational Neuroethology*. Academic Press, San Diego, 1990.
11. M. H. Bickhard. Levels of representationality. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(2):179–215, 1998.
12. R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(47):139–159, 1991.
13. M. V. Butz. *Anticipatory learning classifier systems*. Kluwer Academic Publishers, Boston, MA, 2002.
14. F. L. Crabbe. Optimal and non-optimal compromise strategies in action selection. In *Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior*, 2004.
15. A. R. Damasio. *Descartes' Error: Emotion, Reason and the Human Brain*. Grosset/Putnam, New York, 1994. trad. it. L'errore di Cartesio. Adelphi, Milano, 1999.
16. P. Davidsson. A framework for preventive state anticipation. In *Anticipatory Behavior in Adaptive Learning Systems*, number 2684 in LNAI. Springer, 2003.
17. Y. Demiris. Prediction of intent in robotics and multi-agent systems. *to appear in Cognitive Processing*, 2007.
18. Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition (hammer). *Robotics and Autonomous Systems Journal*, 54:361–369, 2005.
19. M. Desmurget and S. Grafton. Forward modeling allows feedback control for fast reaching movements. *Trends Cogn. Sci.*, 4:423–431, 2000.
20. K. Doya. Complementary roles of basal ganglia and cerebellum in learning and motor control. *Curr Opin Neurobiol*, 10(6):732–739, Dec 2000.
21. G. L. Drescher. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, 1991.
22. J. G. Fleischer. Neural correlates of anticipation in cerebellum, basal ganglia, and hippocampus. In *this volume*.
23. C. D. Frith, S. J. Blakemore, and D. M. Wolpert. Abnormalities in the awareness and control of action. *Philos Trans R Soc Lond B Biol Sci*, 355(1404):1771–1788, Dec 2000.
24. V. Gallese and T. Metzinger. Motor ontology: The representational reality of goals, actions, and selves. *Philosophical Psychology*, 13(3):365–388, 2003.
25. H.-M. Gross, S. Volker, and S. Torsten. A neural architecture for sensorimotor anticipation. *Neural Networks*, 12:1101–1129, 1999.
26. R. Grush. The emulation theory of representation: motor control, imagery, and perception. *Behav Brain Sci*, 27(3):377–96, Jun 2004.

27. M. Haruno, D. Wolpert, and M. Kawato. Hierarchical mosaic for movement generation. In T. Ono, G. Matsumoto, R. Llinas, A. Berthoz, H. Norgren, and R. Tamura, editors, *Excepta Medica International Coungress Series*. Elsevier Science, Amsterdam, 2003.
28. G. Hesslow. Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, 6:242–247, 2002.
29. H. Hoffmann. Perception through visuomotor anticipation in a mobile robot. *Neural Networks*, 20:22–33, 2007.
30. J. Hoffmann. Anticipatory behavioral control. In M. V. Butz, O. Sigaud, and P. Gerard, editors, *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems*, pages 44–65. Springer-Verlag, Berlin Heidelberg, 2003.
31. J. Holland, K. Holyoak, R. Nisbett, and P. Thagard. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, Massachussets, 1986.
32. ikaros, 2002. http://www.lucs.lu.se/IKAROS.
33. irrlicht, 2003. http://irrlicht.sourceforge.net/.
34. W. James. *The Principles of Psychology*. Dover Publications, New York, 1890.
35. B. Johansson and C. Balkenius. An experimental study of anticipation in robot navigation. In *this volume*.
36. R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.
37. D. M. Lyons and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Journal of Robotics and Automation*, 5(3):280–293, 1989.
38. P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.
39. L. Mccauley. Neural schemas: A mechanism for autonomous action selection and dynamic motivation. In *the 3rd WSES Neural Networks and Applications Conference*, 2002.
40. R. C. Miall and D. M. Wolpert. Forward models for physiological motor control. *Neural Networks*, 9(8):1265–1279, 1996.
41. F. A. Middleton and P. L. Strick. Basal ganglia output and cognition: evidence from anatomical, behavioral, and clinical studies. *Brain Cogn*, 42(2):183–200, 2000.
42. G. A. Miller, E. Galanter, and K. H. Pribram. *Plans and the Structure of Behavior*. Holt, Rinehart and Winston, New York, 1960.
43. M. Minsky. *The Society of Mind*. Simon & Schuster, 1988.
44. U. Neisser. *Cognition and reality*. San Francisco, CA: Freeman, 1976.
45. A. Newell and H. A. Simon. *Human problem solving*. Prentice-Hall, Englewood Cliffs, NJ, 1972.
46. D. A. Norman and T. Shallice. Attention to action: Willed and automatic control of behaviour. In R. J. Davidson, G. E. Schwartz, and D. Shapiro, editors, *Consciousness and Self-Regulation: Advances in Research and Theory*. Plenum Press, 1986.
47. G. Pezzulo and G. Calvi. A schema based model of the praying mantis. In S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, O. Miglino, J.-A. Meyer, and D. Parisi, editors, *From animals to animats 9: Proceedings of the Ninth International Conference on Simulation of Adaptive Behaviour*, volume LNAI 4095, pages 211–223, Berlin, Germany, 2006. Springer Verlag.
48. G. Pezzulo and G. Calvi. Toward a perceptual symbol system. In *Proceedings of the Sixth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Lund University Cognitive Science Studies 118, 2006.

49. G. Pezzulo and G. Calvi. Designing modular architectures in the framework akira. *To appear in Multiagent and Grid Systems*, 3(1), 2007.
50. G. Pezzulo, G. Calvi, and C. Castelfranchi. Dipra: Distributed practical reasoning architecture. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1458–1464, 2007.
51. G. Pezzulo, G. Calvi, D. Ognibene, and D. Lalia. Fuzzy-based schema mechanisms in akira. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2*, pages 146–152, Washington, DC, USA, 2005. IEEE Computer Society.
52. G. Pezzulo and C. Castelfranchi. The symbol detachment problem. *to appear in Cognitive Processing*, 2007.
53. J. Piaget. *The Construction of Reality in the Child*. Ballentine, 1954.
54. R. P. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat Neurosci*, 2(1):79–87, January 1999.
55. D. Roy. Semiotic schemas: a framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205, 2005.
56. A. Saffiotti. Handling uncertainty in control of autonomous robots. In *Artificial Intelligence Today*, pages 381–407. 1999.
57. R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ, 1977.
58. W. Schultz. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80:1–27, 1998.
59. O. Selfridge. *The Mechanisation of Thought Processes*, volume 10, chapter Pandemonium: A paradigm for learning, pages 511–529. National Physical Laboratory Symposia. Her Majesty's Stationary Office, London, 1959.
60. D. Shapiro and R. Schmidt. The schema theory: Recent evidence & developmental implications. In J. K. . J. Clark, editor, *The development of movement control and co-ordination.* New York: Wiley., 1982.
61. O. J. M. Smith. A controller to overcome dead time. *ISA Journal*, 6(2):28–33, 1959.
62. J. Tani. Learning to generate articulated behavior through the bottom-up and the top-down interaction processes. *Neural Netw*, 16(1):11–23, Jan 2003.
63. J. Tani, M. Ito, and Y. Sugita. Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using rnnpb. *Neural Netw*, 17(8-9):1273–1289, 2004.
64. J. Tani and S. Nolfi. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Netw*, 12(7-8):1131–1141, Oct 1999.
65. A. Weitzenfeld, O. Peguero, and S. Gutiérrez. NSL/ASL: Distributed simulation of modular neural networks. In *MICAI*, pages 326–337, 2000.
66. D. M. Wolpert, Z. Gharamani, and M. Jordan. An internal model for sensorimotor integration. *Science*, 269:1179–1182, 1995.
67. D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.
68. T. Ziemke, D.-A. Jirenhed, and G. Hesslow. Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68:85–104, 2005.