Toward a Perceptual Symbol System

Giovanni Pezzulo
ISTC - CNR
Via S. Martino della Battaglia, 44
00185 Roma, Italy
giovanni.pezzulo@istc.cnr.it

Gianguglielmo Calvi
Noze s.r.l.
Via Giuntini, 25 int.29
56023 Navacchio, Cascina (PI), Italy
gianguglielmo.calvi@noze.it

Abstract

We explore the possibility for a situated system to evolve what Barsalou calls a *perceptual symbol system* (PSS). We describe the peculiarities of perceptual symbols and point out the main capabilities of organized, multimodal frames of perceptual symbols called *simulators*. We present a case study in which perceptual symbols and simulators are evolved and exploited for categorization, prediction and abstraction.

1. Introduction

Our aim is to investigate the emergence of *perceptual symbols* (Barsalou, 1999), "records of the neural activation that arises during perception" and *simulators*, multimodal frames which afford representation and motor control.

According to (Barsalou, 1999), during activity the brain records sensorimotor aspects of the experience in the format of perceptual symbols. These representations are not analogical but schematic: only some features are selectively recorded such as colors, edges, spatial relations, movement, etc. In this way multimodal frames are stored in long term memory; they are not just aggregates of features, but have a distinctive *productive* capability, giving rise to *simulators*. Simulators are organized systems of perceptual symbols: they can generate a simulation of categories of objects or events stored in the frame by rehearsing the associated perceptual symbols, and typically do it in absence of physical exemplars. Specific runs of a simulator reenact the multimodal experience of a category; simulations are always schematic and never complete, and represent types and not tokens.

Simulators play many roles. They intervene in *categorization*: objects or events are simulated and matched with actual perception; typically many simulators run a simulation providing a good fit to the entity, and the best fitting simulator is selected. They thus embed the *principium individuationis* of a concept. Simulators provide a basis for *prediction*, too: once an object or event is categorized, the simulator permits to predict its behavior, since the representation is expressed in an interaction-oriented format. Simulating also produces *visual and motor imagery*. Simulators offer a *semantic theory*, too, being roughly equivalent to concepts: if we have an appropriate simulator of something, then it can be said that we understand the concept. Simulations permit to model the two main peculiarities of concepts, i.e. to be *detached* and to afford *abstraction*. By simulation, subsets of perceptual states are extracted to function symbolically and support the higher cognitive functions. For example, they can be used off-line for providing inferences about likely properties of entities even in their absence. Or, they can represent abstract categories such as friends, enemies, etc. Off-line and on-line activity can also be coupled: off-line simulations can be mapped later to perceived entities. Simulators also permit to avoid the wrong consequences of having amodal concepts, i.e. how to translate modal and amodal symbols and the grounding problem: simulations are modality-specific or even multi-modal, never amodal.

In the rest of the paper we illustrate a computational roadmap from the emergence of perceptual symbols to the formation of simulators, responding to questions such as: *how are they formed? which features are stored?* We also illustrate their role in categorization, prediction and abstraction.

2. Schemas and Simulators

In our schema-based architecture many *perceptual* and *motor* schemas store interactive information about the entities to deal with and permits to engage a reliable interaction with them. In our simulations we will focus only on tracking (with a fovea controlled by perceptual schemas) and following (with a motor controlled by motor schemas), but our methodology can be applied to other forms of interaction.

Perceptual and motor schemas are not analogical representations of the entities, but embed reliable ways of interacting with them in their presence and even in their absence. The core element of the schemas is their *forward model*, permitting to anticipate the consequences of the possible interactions with an entity and to "stay attuned" to it. In the

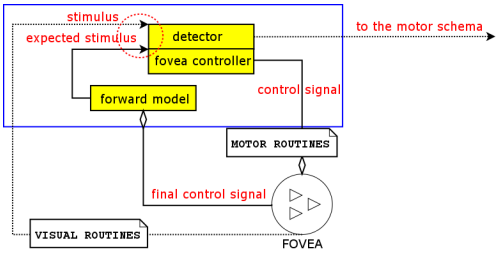

Figure 1: The Model of a Perceptual Schema

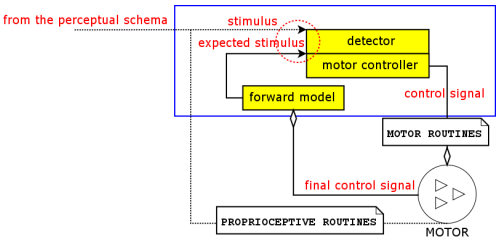

Figure 2: The Model of a Motor Schema

case of perceptual schemas, this does not mean passive visual data processing, but active visual exploration, *tracking* the entity with a fovea. In the case of motor schemas, this means *following* the entity.

Schemas Fig. 1 and fig. 2 show the pseudo-closed loop between controllers and forward models in perceptual and motor schemas. The controllers send a control signal to the actuators, which integrate them and act accordingly; on the same time, an efference copy of the (final) command signal is sent to the forward models of all the schemas, which compute the next expected input. The dashed lines indicate that a feedback signal is received; in the case of perceptual routines, this is the stimulus from the fovea (as we will see, the input is represented by the activity level of a set of visual routines, such as *detect_grey is very_active*); in the case of motor routines, the stimulus is the activity level of perceptual schemas and of proprioceptive routines. Some perceptual and motor schemas are thus functionally related, because the motor schemas use as input the activity level of the perceptual schema; we call these schemas *coupled perceptual-motor schemas*.

The dashed circles indicate that there is a comparison between the actual input stimulus and the expected stimulus. The degree of (mis)match between them is used for two main functions: (1) *Adjustment of Control*: the motor commands are adjusted thanks to the feedback signal and can for example compensate time delays, unreliable of absent sensors. (2) *Action Selection*: schemas have a variable *activity level*, which means more or less control of action; more active schemas, in fact, process more input, send more commands to the actuators and spread more activation to other schemas (see later). *The activity level of the schema represents its relevance*: for perceptual schemas it represents a confidence level that a certain entity, encoded in the schema, is or is expected to be present; for motor schemas it represents a confidence level that the behavior encoded in the schema is both applicable and useful in the current situation. We argue that *relevance depends on anticipation*; schemas anticipating well get more activation; the rationale is that they are well attuned to the current situation. This is obtained by matching the actual and expected stimulus and assigning activation proportionally to the degree of match, as in (Wolpert and Kawato, 1998).

Perceptual Symbols and Simulators In this architecture, *schemas are roughly the equivalent of perceptual symbols in (Barsalou, 1999)*, storing information about the related entity in an interaction-oriented format. There is not a clear cut distinction of functioning between perceptual and motor schemas as we intend them; the former ones are specialized for visual exploration of the object, while the latter ones are specialized for other forms of engagement such as following (as in the case of this paper), escaping or manipulating. *A simulator is a coordinated collection of coupled perceptual-motor schemas*, providing an unitary, multimodal categorical principle which unifies the diversity of many modality-specific schemas and a set of patterns of possible interactions. Each simulator is characterized by a specific pattern of activation of the involved schemas, which is also induced by the energetic links evolved by the schemas; the problem of evolving a simulator is also close to the *binding problem* (von der Malsburg, 1981).

Our aim is to investigate how competing schemas, specialized for *tracking* and *following* the same or different features (such as color, size, shape, motion; see (Wolfe, 1996) for a review of the features used in visual tasks) of the same entity (1) arise as perceptual symbols and (2) are organized in simulators.

Perceptual symbols and simulators are evolved in two steps. In the former a "free exploration" of the environment gives rise to many schemas coupling motor commands and expected inputs. In the latter the regularities in the environment (such as the stability of the entity over time and the coherence of its possible modifications, due e.g. to its movement) permit a synthetic representation of the entity since perceptual and motor schemas engaging the same entity evolve reliable patterns of interaction.

Fig. 3 illustrates a simulator: set of schemas which "work together" for tracking and following an insect. There are many schemas specialized for tracking and following many kinds of colors, shapes, trajectories,

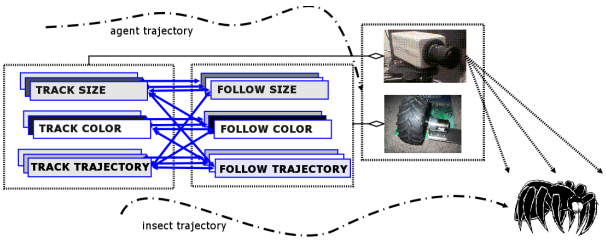

Figure 3: A simulator involving many schemas.

etc. The different colors of the schemas in the figure represent different levels of activation; for example, since in the picture the prey is dark, it is very likely that schemas related to dark colors will be very active, while schemas related to light colors will be not. The same is true for other features. Schemas having coherent patterns of activation evolve links (horizontal edges). This is for example the case of schemas specialized for different features of the same entity. Functionally *coupled* schemas (motor schemas using as input the activity level of perceptual schemas) often become linked, too. Since links afford spreading activation, linked schemas tend to synchronize, resulting in "clusters" of active schemas which we interpret as simulators.

Before describing the architecture and the simulations, two caveats: (1) There is not a one-to-one mapping between a feature and a schema. For example, many competing tracking schemas can be specialized for slow or quick, dark of grey entities. In the case of a "quite slow" entity, it is likely that a simulator will involve both slow-tracking and quick-tracking schemas and, for the sake of tracking, both are needed (but the most fitting ones, the "prototypes", evolve stronger links). (2) Simulators share some schemas, which can be active (at different levels and with different patterns) in different simulations.

3. The Architecture

Fig. 4 shows the components of the architecture: the perceptual and motor schemas (i.e. the components to be evolved); the routines; the actuators.

3.1 The Perceptual Schemas

As shown in Fig. 1, each perceptual schema has three components: a *detector*, a *controller* and a *forward model*. The detector acquires relevant input (preconditions) from the the fovea. The controller sends motor commands to the fovea. The forward model predicts the next stimulus, i.e. the activity level of one or more visual routines after the agent's action.

In addition to the mechanism assigning more activation on the basis of anticipation, perceptual schemas get also activation if the stimuli they are specialized for are indeed present in the environment: again, the preconditions in the detector are matched against the activity level of the corresponding visual routines and activation is assigned proportionally to the degree of match.

Active perceptual schemas influence the rest of the architecture in three ways. Firstly, they send motor commands to the fovea, orienting it toward relevant entities; more active schemas send commands with higher fire rate. By orienting the fovea, the schemas are able to partially determine their next input (they have an active vision). In an anticipatory framework, this functionality is mainly used to test the predictions of the forward models: for example, tracking a moving object is a way to acquire new stimuli in order to test the expectations. Secondly, they spread activation to the related visual routines, priming them and realizing visual imagery (Kosslyn and Sussman, 1994). For example, *track grey* schema primes the *gray-detector* visual routine, even in absence of real stimuli; this functionality can be used to select only relevant stimuli from the fovea and to complete fragmented perceptual inputs. Thirdly, more active perceptual schemas activate more their coupled motor schemas; as above discussed, this leads to re-enacting whole simulators.

3.2 The Motor Schemas

As shown in Fig. 2, each motor schema is similar to a perceptual schema and has the same three components. In the detector the preconditions are matched against the activity level of one or more perceptual schemas. For example, the motor routine *follow grey* has as a precondition the perceptual routine *track grey*; this means that if the activity level of the latter is high, the former gains activation, too. The controller sends commands to the motor. The forward model produces expectations about perceptual stimuli to be matched with sensed stimuli.

3.3 The Routines

The perceptual schemas do not receive raw input from the fovea: a number of preprocessing units, the *visual routines*, filter fovea information (although with different priority). We have included several feature-specific visual routines specialized for colors, sizes, shapes and motion. The activity level of the visual routine directly encodes the presence of absence of associate entities; for example, an active red-detector encodes the presence of red entities. A similar mediating role is played by the *motor routines* (such as *move_right* and *move_left*), commanding the fovea and the motors; in this case, the activity level of *move_right* encodes the turning angle. There are also *proprioceptive routines* such as *move_left* providing feedback information from the motors.

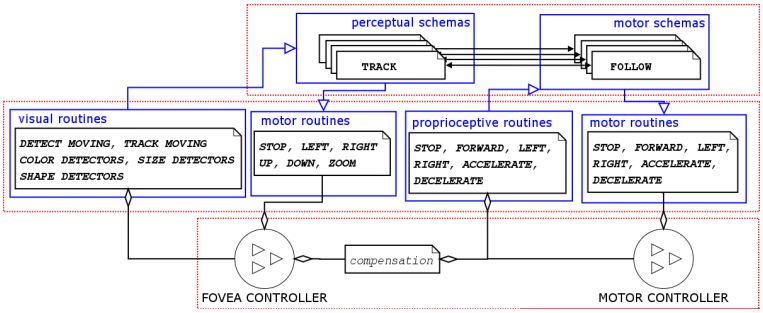

Figure 4: The Components of the Architecture: Schemas, Routines, Actuators

3.4 The Actuators

The actuators (motor and fovea controllers) receive as input commands from all the active motor routines and perform fuzzy based command fusion (Kosko, 1992). Since routines have different priorities, commands are sent asynchronously and with different fire rates. Again, fire rate encodes priority: more active routines send more commands to the actuators and influences it more. The actuators produce in output two vectors of coordinates $< x, y, z >$ and $< x_1, y_1, z_1 >$, representing the next position of the agent and the next fixation of its fovea (that can zoom), and send them to the physical engine.

4. The Three Learning Phases

Our aim is to develop a simulated robotic system which is able to track and follow many entities (say insects) having different shapes, dimensions, velocities, colors and trajectories. In order to do so, the system builds up and exploits a number of schemas for the feature to be tracked and followed, and assemble them into simulators. Simulators, via their forward models, produce predictive information which permit to *categorize* and *predict* the entities' behavior and to "stay attuned" to them in an active exploration. We also discuss how to learn simulators for more *abstract* concepts with the aid of some drives.

The architecture is evolved in three phases. In the first phase, schemas are learned via "free exploration". In the second phase, simulators arise. In the third phase more abstract concepts such as "preys" and "predators" are learned thanks to the introduction of internal drives.

4.1 First Phase: Perceptual Symbols

In the **first phase**, many feature-specific schemas are learned by means of free sensorimotor interaction with the environment only involving few insects per time. Schemas learn the consequences of the agent's actions via the forward model in order to orient the fovea or the motors towards their expected positions. Schemas can thus be seen as predicting rules which permit to be attuned with the transformations of some features of the environment.

As discussed in (Piaget, 1985), there is not jet an "ontology" of the objects of interaction, but only actions and percepts. The input of a perceptual schemas is the value of one or more visual routines; the forward model learns to predict their next values depending on the actions of the fovea and the controller exploits that information for calculating where to move the fovea for centering the target. The input of the motor schemas is the value of one or more perceptual schemas and of one or more proprioceptive routines; the forward models learn to predict their next values depending on the actions of the motor, and the controller exploits that information for calculating where to move the motor in order to reach or follow the target. In both cases the controller exploits the forward model: the former orients the center of the fovea or the motor in the position in which the latter anticipates the presence of the tracked/followed object in the next step. Each schema is learned independently. We argue that *success of prediction* is the main functional criterium for the formation of perceptual and motor schemas. Firstly, schemas learn until their accuracy in prediction is satisfactory. In this way the agent builds up a reliable repertoire of actions affording anticipatory capabilities. Secondly, since motor schemas learn to exploit as inputs the activity level of perceptual schemas, they become *functionally coupled*.

Design Firstly we have designed a set of feature-specific visual routines (for colors, shapes, trajectories, size; see (Wolfe, 1996)) and two sets of motor routines (such as *turn_left* or *turn_right*), one for the fovea and one for the motor. Secondly, we have introduced a set of perceptual schemas specialized for one or more visual routines, and a set of motor schemas specialized for the corresponding perceptual schemas. The detector component was thus specific for a given input (e.g. a given set of visual routines), while the controller and forward models learned to associate visual and motor information.

In both cases we used feed-forward neural networks having three input, three output and twelve hidden unities, using sigmoid functions and learning with error backpropagation. In the perceptual schemas, the input of the controller is the activity level of the perceptual routines related to a single feature (color, size, etc.), while its output is a motor command for the fovea (encoded as a high or low activity level of appropriate motor routines); the inputs and outputs of the forward model are the converse. In the motor schemas, the input of the controller is the activity level of one or more perceptual schemas, while its output is a motor command for the motor; the inputs and outputs of the forward model are the converse.

In this phase the schemas reliability depends on the stability and predictability of the environment. As we will illustrate in the *categorization* and *prediction* experiments, since in our simulations the features of the insects to track and follow are stable and predictable, the evolved schemas are reliable.

4.2 Second Phase: Simulators

In the **second phase**, simulators emerge as multimodal frames providing an unitary categorical principle for assembling many of the previously learned schemas. For example, in the first phase the schemas permit to anticipate the direction of an entity or the persistence of its color, but they do not associate these two features to the same entity. Simulators can do that. The simulators will be assembled according to the most distinctive features of the entities to interact with (in this case, the entity to track).

We argue, again, that *success of prediction* is the main functional criterium for their formation. Perceptual and motor schemas having *coordinated patterns of prediction*, in fact, evolve energetic links. Since in our model successful prediction means more activation, many *coupled perceptual-motor schema*, having stable patterns of activation, are thus assembled in a multimodal frame, i.e. a simulator. As argued before, embodiment and regularities in the environment permit an unitary account of the entities; all the features of the entities to track change (e.g. move) in a coherent way; for this reason, schemas related to features of the same entity evolve coordinated patterns of prediction. Moreover, only schemas representing features which discriminate well the entity are recorded in the simulator; the other ones will not evolve stable links and will not be a proper part of the simulator. Schemas are assumed to discriminate well if their pattern of prediction is coherent with the other schemas of the simulator and if their *informativeness* is positive (i.e. if they are significantly more active when the entity is indeed present). As an effect of the diversity of the entities to be tracked, the number of simulators (and thus of categories) will be equal to the number of discriminable entities for the sake of tracking/following; if two insects can be tracked/followed in the same way, only one simulator will be stored.

Design In this phase we adopted Differential Hebbian Learning (Kosko, 1992) for evolving links between schemas which afford spreading activation. Lately, by analyzing the topology of the connections it is possible to observe that clusters have emerged, i.e. simulators. This method realizes both our desiderata: (1) simulators arise thanks to the coherence in predictions (producing coherence in the patterns of activation); in fact, schemas which predict well gain activation and fire more often; and those who fire together wire together. (2) only informative schemas become part of a simulator; schemas which can deal with many entities typically fire in synchrony with many other schemas, not necessarily related to a single simulator, and evolve weak links with all them. Since simulators are distributed they give many advantages over single schemas: each schema in a simulator can prime the coupled ones and the whole simulator is re-enacted in presence of a discriminable entity, modeling input reconstruction. Moreover, even if some schemas in a simulator fail (e.g. a shape detector failing to track a partially occluded insect) the whole simulator is more robust. We will test this capability by comparing the performance of perceptual symbols and simulators.

4.3 Third Phase: Drives and Abstraction

Until now the system has evolved a simulator for each discriminable entity in the environment. However, the agent has only one drive, which is to track and follow all the entities. Since all the simulators have the same functional role (to allow tracking and following), it is not advantageous to cluster them in alternative ways and to have abstract representations.

(Barsalou, 2003) discusses many senses of abstractions; here we only investigate the emergence of abstract concepts such as *preys* and *predators*, which are not entities but *roles played by the entities*. We argue here that these abstract concepts are ways of clustering base level concepts such as entities. Again, according to the idea of simulators, clustering does not mean simply collecting all the features (or, in this case, the base level concepts): a simulator for prey or predator is a schematic representation, thus containing only the most distinctive features and concepts.

Design In this phase half the insects of the previous phases played the role of preys, and half of predators. In order to make this distinction meaningful for the system we introduced two drives, *hungriness* and *fear*, that can be seen as two inner states which are sensed by the agent. Drives are modeled as two nodes

in a Fuzzy Cognitive Map (Kosko, 1992) and have inhibitory links: in this way, when hungriness is active it inhibits fear, and vice versa. Hungriness is raised routinely with a biological clock, increases when a prey is close and decreases if a prey is reached; fear increases when a predator is close and decreases otherwise. We also let the system learn *avoiding* motor schemas in which the detector and the forward model are the same of *following* schemas, but the controller sends as motor command the opposite one.

Drives are also carriers of energy and can spread activation to schemas satisfying them. The system learns to associate drives satisfaction to schemas responsible for satisfaction: the drives fire more when they are being satisfied (i.e. when their value lowers) and thus they evolve links with related schemas. After learning *hungriness* sends activation to the schemas which successfully *track* and *follow* preys; of course, when hungriness is high (i.e. when a prey is in sight or the agent is hungry) the drive is able to provide more energy and the agent is more oriented toward tracking and following preys. *Fear* sends activation to the schemas which *track* or *avoid* predators; again, when fear is high (i.e. when a predator is close) fear sends more energy. Since the energetic resources are limited, schemas for dealing with preys and predators inhibit each other, too. As in the second phase, Differential Hebbian Learning was used for learning new links (or modifying already learned ones) between the schemas.

In the *abstraction* experiment we show how this process produces two more simulators, for preys and predators, which embed a schematic representation of the entities to follow or avoid.

Related Models As usual in schema-based design (Arbib, 1992) our model exploits concurrent unities whose activity level encodes relevance. While many schema-based architectures are reactive, our approach emphasises the role of anticipatory mechanisms, as in simulation theory (Barsalou, 1999), emulation theory (Grush, 2004) and interactivism (Bickhard, 2001). In (Pezzulo and Calvi, 2006) a related model exploiting pre-designed schemas is used for comparing anticipatory and reactive capabilities, showing a significant advantage of anticipatory ones. Related anticipatory schema system are described in (Drescher, 1991) and (Stojanov, 2001): both are able to learn context-action-result triplets and assemble them into higher-level schemas organized hierarchically. With respect to those systems, in our model schemas are more complex, involving both an inverse and a forward model. Other related models are MOSAIC (Wolpert and Kawato, 1998) and HAMMER (Demiris and Khadhouri, 2005), also exploiting inverse and forward models for motor control. Differently from these systems, in our model: (1) there

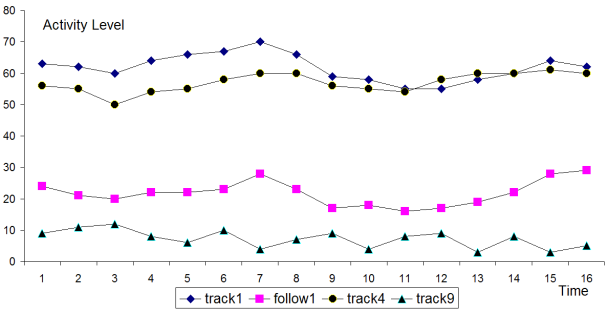

Figure 5: Activity timeline of four schemas

is parallelism and the activity level of the components, including schemas, is mapped into the priority of their threads; (2) schemas can spread activation via evolved links; (3) there is a limited amount of resources which is shared by all the components, thus schemas compete for resources and only few of them can be active at once. As we will see these features permit to evolve simulators for concrete and abstract entities and to use them as "competing hypotheses" for categorizing and guiding the behavior.

5. The Experiments

We implemented the above described architecture by using the framework AKIRA (akira, 2003) and the 3D engine IRRLICHT (irrlicht, 2003). The set-up was a 3D surface with hills (offering partial cover) and involved twenty insects having variable size, colors, shapes and trajectories. Two agents, both having three schemas for each feature, were compared: the former (*PS*) only learned in the first phase; the latter (*SIM*) also learned in the second phase. During the first learning phase up to five insects were present together in the environment; the agent learned the forward models of its schemas by interacting with them. One example was sampled every twelve, with a total of thirty-six; the learning stopped when the error (the euclidean distance between the actual and predicted position in 3D, $0.1 * 10^{-6}$) was less than 0,0000001 (positions vary between -10000 and +10000 in the three axes).

During the second learning phase (one session lasting three minutes) links between schemas were evolved, too. K-means cluster analysis (using euclidean distance of the activity level of the schemas in time) was used for investigating how many simulators evolved; since the result was sixteen, we removed from the set-up four "aliased" insects. As an example, fig. 5 shows a sample timeline involving four schemas during an interaction with one insect. K-means analysis (number of classes = 2, euclidean distance k-means = 0,53) shows that the first three have coordinated activity patterns and form a cluster (which we interpret as a simulator), while the fourth

Agent	Recall	Gen. 75%	Gen. 50%
PS	81%	64%	48%
SIM	92%	78%	65%

Table 1: Categorization: percentage of success

Ag.	Schema	Rec.	G. 75%	G. 50%
PS	Track	0,29%	0,4%	0,5%
PS	Follow	0,31%	0,33%	0,61%
SIM	Track	0,14%	0,24%	0,37%
SIM	Follow	0,17%	0,25%	0,43%

Table 2: Prediction: mean error in %

is unrelated; it has a low activity level, too, since it is not very relevant.

In order to test the efficacy of simulators we designed two tasks: *categorization* and *prediction* with three levels of complexity: *recall* (involving insects used for learning); *generalization 75% and 50%* (involving insects sharing 75% and 50% of the features with the ones used for learning). Our hypothesis is that *SIM* performs better in all the conditions.

Categorization One insect per time is in the scenario and the tasks consists for the agent in categorizing it by activating the relevant cluster of schemas. As above described, the sixteen clusters of *SIM* were determined with cluster analysis and interpreted as simulators. We adopted k-means cluster analysis for *PS*, too, also obtaining sixteen clusters. We thus evaluated the reliability of the clusters by considering how many times the same ones were active when the same insects were in play. The cluster maximizing $\frac{active\_schemas}{total\_schemas}$ was considered the selected category. Since simulators compete for limited resources and an active simulator inhibits the other ones there was often an unambiguous way to determine the category in *SIM*, while in *PS* the most active clusters often involved different schemas. The percentage of correctly categorized insects in 100 simulations is shown in Tab. 1; SIM categorizes significantly better than PS ($p < 0{,}001$ in all the conditions).

Prediction The task consists in successfully tracking the insects (presented singularly) from a start to an end point. Since all the active schemas send commands to the actuators, the task is an evaluation of the "coherence" of the simulators. We collected data about the mean error in prediction (in %) throughout the trajectory of the final position calculated by the controllers. Results in 100 simulations are shown in Tab. 2; SIM predicts significantly better than PS ($p < 0{,}001$ both for track and follow).

Abstraction In the same setup we also designed an *abstraction* task for testing an agent after the third learning phase. The task consists in surviving in an environment including preys and predators. We compared two agents, both having the same schemas and drives described in the third learning phase. The former (*NO-ABS*) only learned in the first two phases; the latter (*ABS*) learned as described in the third phase, too. *Drives satisfaction* was used as success metric: each agent had to satisfy its drives, *fear* and *hungriness*, i.e. keep their values close to zero. Since in order to satisfy the drives the agents have to abstract appropriately, distinguishing preys from predators, this task permits to evaluate the appropriateness of the simulators for abstract concepts. Our hypothesis is that, with the introduction of two drives, two more simulators arise for *predators* and *preys*. We collected data about mean satisfaction, calculated as $1 - mean\_drive\_value$, in 100 simulations. Tab. 3 shows the results of our simulations; ABS predicts significantly better than NO-ABS ($p < 0{,}001$ both for hungriness and fear).

Agent	Hungriness	Fear
ABS	0,792	0,921
NO-ABS	0,632	0,756

Table 3: Abstraction: mean drives satisfaction

Discussion Our results indicate that after the first two learning phases an agent is able to categorize and predict even if, not surprisingly, its performance degradates when new insects with few features in common are introduced. After the third phase it is also able to abstract the role of the insects. Our results indicate a significant advantage of using simulators in all the tasks. Simulators categorize and predict better than simple clusters: in many cases one single schema is doomed to fail (e.g., because its forward model is not totally reliable or because inputs are aliased or missing or partially unpredictable), but a coordinated set of schemas can reach a cooperative solution, prime and compensate each other. It is worth noting that in the abstraction task typically both the simulator for the insect (e.g. *insect#3*) and for its role (e.g. *prey*) arise (and they partially overlap). Depending on the task, the former or the latter become more relevant: we have tested *ABS* in the categorization and prediction tasks and its performance does not significantly differ from *SIM*.

6. Conclusions

We have illustrated an architecture for evolving perceptual symbol systems and to assemble them in simulators. We have tested the architecture in three tasks, *categorization*, *prediction* and *abstraction*. Our results indicate the advantage of having perceptual symbols in the form of schemas and sim-

ulators in the form of multimodal frames of schemas having coordinated patterns of activation.

Our experiments have implications for the PSS theory (Barsalou, 1999), too.

- *Active sensing vs. topographical maps* Perceptual symbols may or may not be topographically organized. For example, in (Joyce et al., 2003) PSS are modeled with topologically organized connectionist networks. Our model is instead schema-based and closer to (O'Regan and Noe, 2001), in which vision is a mode of exploration of the world by mastering the appropriate schemas and not a "re"-presentation of the world inside the brain.
- *Which Simulators Emerge?* Simulators emerge for entities that are (1) predictable and (2) discriminable either for some of their features, as in the case of different insects, or for the role they play, as in the case of predators and prays, which assumes meaning for the system thanks to drives.
- *Perception and Action* Our interpretation of perceptual symbols emphasizes the coupling of perception and action. Schemas are not only a rehearsal of perceptual traces but include motor programs for interacting with objects, and the recorded perceptual traces are typically the expected consequences of those interactions.

Further Work In our experiments we have exploited the expectations produced by the forward models only in the sensorimotor cycle of the schemas. The further step (the "fourth phase") of this work is to *decouple* the expectations and use them for off-line processing (Barsalou, 1999). Engaging in a simulated interaction with the entities even in their absence is crucial for example in planning, permitting to generate possible alternatives and to reason about them without actually attempting them.

Acknowledgements

This work is supported by the EU project **MindRACES**, FP6-511931. Thanks to Larry Barsalou for helpful discussions.

References

akira (2003). http://www.akira-project.org/.

Arbib, M. (1992). Schema theory. In Shapiro, S., (Ed.), *Encyclopedia of Artificial Intelligence, 2nd Edition*, volume 2, pages 1427–1443. Wiley.

Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences.*, 22:577–600.

Barsalou, L. W. (2003). Abstraction in perceptual symbol systems. *Philos Trans R Soc Lond B Biol Sci*, 358(1435):1177–1187.

Bickhard, M. H. (2001). Function, anticipation and representation. In Dubois, D. M., (Ed.), *Computing Anticipatory Systems. CASYS 2000 - Fourth International Conference*, pages 459–469, Melville, NY. American Institute of Physics.

Demiris, Y. and Khadhouri, B. (2005). Hierarchical attentive multiple models for execution and recognition (hammer). *Robotics and Autonomous Systems Journal*, 54:361–369.

Drescher, G. L. (1991). *Made-Up Minds: A Constructivist Approach to Artificial Intelligence.* MIT Press, Cambridge, MA.

Grush, R. (2004). The emulation theory of representation: motor control, imagery, and perception. *Behav Brain Sci*, 27(3):377–96.

irrlicht (2003). http://irrlicht.sourceforge.net/.

Joyce, D., Richards, L., Cangelosi, A., and Coventry, K. (2003). On the foundations of perceptual symbol systems: Specifying embodied representations via connectionism. In Bamberg, U., (Ed.), *Proceedings of ICCM V*, pages 147–152.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems.* Prentice Hall International, Singapore.

Kosslyn, S. M. and Sussman, A. (1994). Roles of imagery in perception: Or, there is no such thing as immaculate perception. In Gazzaniga, M., (Ed.), *The cognitive neurosciences*, pages 1035–1042. Cambridge, MA: MIT Press.

O'Regan, J. and Noe, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5):883–917.

Pezzulo, G. and Calvi, G. (2006). A schema based model of the praying mantis. In *Proceedings of The Ninth International Conference on the Simulation of Adaptive Behavior (SAB'06).*

Piaget, J. (1985). *Equilibration of cognitive structures.* University of Chicago Press.

Stojanov, G. (2001). Petitage: A case study in developmental robotics. In *Proc. 1st Int. Workshop on Epigenetic Robotics*, Lund, Sweden.

von der Malsburg, C. (1981). The correlation theory of brain functions. Internal Report 81-2, Max-Planck-Institut Biophys. Chem., Gttingen FRG.

Wolfe, J. M. (1996). Visual search. In Pashler, H., (Ed.), *Attention.* London, UK: University College London Press.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329.