Emergence of Meaning
from a Stream of Experience –
Grounding Symbols
by Consequences and Intentions

Bachelor Thesis in Applied Computer Science

Submitted by: Mark Wernsdorfer
Supervisor: Prof. Ute Schmid
Faculty: Information Systems and Applied Computer Sciences
Department: Cognitive Systems
Commenced/completed: 15.07.2010/15.12.2010
Abstract

At the heart of understanding the world is the ability to represent its relevant aspects in a meaningful manner. Trying to model this crucial power in an autonomous agent can contribute essentially to core problems of AI research. Some of these unanswered questions are: How is relevance to be integrated in a per se unambitious machine? What makes meaning from a systematics point of view? Can one single mechanism produce the plethora of mental phenomena we experience? Is high order cognition emergent from simple but yet numerous elementary processes?

In this thesis we argue that relevancy and meaning can be represented by agents in direct relation to their past experiences and present intentions. From this base assumption a supervised multiple prototype learning algorithm is derived that receives examples and labels as feedback from its environment. Random optimisation allows for gradual increase in motor efficiency concerning local maxima of extrapolated sensor evaluations. A sensorimotor map of the agent’s environment is generated that stores and optimises beneficial motor activations in evaluated sensor space by employing temporal Hebbian learning. Therefore this map is representing relevant, hence evaluated, physical activations that are pragmatically meaningful to the agent.

Creating sensorimotor maps is understood as necessary precondition for forming more complex mental representations such as symbols or signs in general. A mechanism is outlined that generates higher level entities from basic sensorimotor representations. These structures represent domain specific competencies that differentiate contexts of beneficial activations in given states. Arbitrarily complex representations can be constructed by the same basic principles for different levels of abstraction. These representations are indisputably grounded in the most basic interaction between agent and world conceivable: sequential vectors in sensor and motor space.
# Table of Contents

1 INTRODUCTION...........................................................................................................................................6

2 THEORETICAL BACKGROUND.........................................................................................................................7
  2.1 Iconisation, Discrimination and Identification............................................................................................8
  2.2 Hybrid Cognitive Architectures....................................................................................................................8
  2.3 Sensorimotor Maps and the Symbol Grounding Problem...............................................................................9
  2.4 Reactive Architectures..................................................................................................................................10

3 MODEL.............................................................................................................................................................11
  3.1 Simulation Environment...............................................................................................................................12
  3.2 Sensorimotor Equipment................................................................................................................................14
  3.3 Intentions and Expectations within a Data Structure..................................................................................15
    3.3.1 Generalisation........................................................................................................................................16
    3.3.2 Specialisation........................................................................................................................................19
    3.3.3 Sequences of Motor Prototypes for Successful Interaction...................................................................20
    3.3.4 Sensor States as Context for Motor Sequences....................................................................................20
    3.3.5 Sensorimotor Graph.............................................................................................................................21
  3.4 Initiating Momentum....................................................................................................................................22
    3.4.1 Noise as a Means of Exploration...........................................................................................................23
    3.4.2 Extrapolating the Evaluation Gradient....................................................................................................26
    3.4.3 Smoothing Evaluation Gradient...............................................................................................................27
    3.4.4 Optimizing Motor Activities..................................................................................................................28
  3.5 Emergence of Meaning..................................................................................................................................29
    3.5.1 Contexts of Sensor Prototypes...............................................................................................................30
    3.5.2 Conditions for Sensor Prototype Transition and Creation.....................................................................31
  3.6 Algorithm........................................................................................................................................................32

4 DOMAIN AND ADAPTIVITY.............................................................................................................................33
  4.1 Topology.......................................................................................................................................................34
  4.2 Discreteness versus Continuity.....................................................................................................................35
    4.2.1 Environment..........................................................................................................................................35
    4.2.2 Evaluation............................................................................................................................................36
  4.3 Relevant Parameters......................................................................................................................................36
    4.3.1 The Need................................................................................................................................................36
    4.3.2 Motor Randomisation.............................................................................................................................37
    4.3.3 Connection Strengths.............................................................................................................................37
4.3.4 Evaluation Smoothing
4.3.5 Vector Step Factor

5 DISCUSSION
5.1 Results
5.1.1 Performance
5.1.2 Saturated Motor Map
5.2 Where to Go from Here?
5.2.1 Planning
5.2.2 Grounding Symbols

6 IMPLEMENTATION

7 APPENDICES
7.1 Performance Table
7.2 Saturated Motor Map
7.2.1 Vertices
7.2.2 Edges

8 BIBLIOGRAPHY
1 Introduction

The first systems designed for modelling cognitive processes such as SHRDLU (Winograd 1971) and ELIZA (Weizenbaum 1966) were strongly influenced by the experience of conscious thought. They imitate the human high order cognition like planning or the production and processing of language. These skills have in common that they operate on signs which represent processes or objects outside the model. According to Newell (1980) systems simulating high level cognition in this way are called physical symbol systems; the assumption that these systems are sufficient to describe human cognition in entirety is called physical symbol systems hypothesis. At the time of its appearance, this hypothesis was confirmed by neurological findings, which suggested that human problem solving processes took place in a similar manner (e.g. Lachmann et al. 1979).

The physical symbol systems hypothesis, however, has been exposed to intense criticism over the years. Of particular interest for this thesis is the symbol grounding problem. It raises the question: What processes may be responsible for the emergence of symbol-like mental representations in the first place? In case symbol-like representations are created by symbolic rule sets, where do these rule sets originate from? Either they need to encode some basic and general truth that needs to be accepted axiomatically, or they need to be inflicted upon the system by its architects (Taddeo and Floridi 2005).

McCarthy and Haynes (1969) first pointed out another crucial predicament concerning physical symbol systems hypothesis. They have shown that in symbol systems of dynamic environments, without establishing explicit rules, it cannot be determined which part of the environment is not affected by certain actions. This is called the frame problem.1 In a sufficiently complex domain, such systems need to break down under the weight of what their actions are not relevant for. Over the years, the logical frame problem has been presented and solved in many different ways (Kamermans and Schmits 2007), its philosophical aspects, however, remain unaffected by present practical solutions. That conflicts concerning physical symbol systems are far

---

1 The term frame problem has been adapted by the philosophy of mind, where it characterises the question how the assumptions of a rational agent about his world are affected by his actions in general (Dennett 1981).
from being resolved can be seen by the current defence of Nilssen (2007) against many of the attacks carried out over the years.

This thesis presents a reactive model controlling a simulated agent that explicitly takes those problems into account from an epistemologically and semiotically sound perspective. Resulting is a sub-symbolic optimisation procedure, that enables an agent to reactively learn beneficial sequences of motor states to maximize an evaluation measure extrapolated from its environment using sensory data. Crucial for what the agent is able to interact with are past experiences and present intentions. Experiences are memorised consequences of executed actions and intentions are defined by sensor states and concurrent feedback.

The sub-symbolic structures serve as a sensorimotor basis for more complex representations of the environment. By separating different sequences of activations, different contexts emerge. On the one hand this tackles the frame problem, because the influence of changes is confined to only one such context, on the other hand those contexts are understood as grounded in sensorimotor interaction. Representing this information by referring a) to its absolute or relative desirability (evaluation) and b) to the measures (motor actions) maximizing this evaluation is understood as a meaningful assignment of relevant matters.

2 Theoretical Background

The need for hooking physical agents into their environments more radically and directly than physical symbol systems allow has been famously voiced by Richard Dreyfus (1972, more recently 2007). This type of critique culminated in the demand for embodied and embedded/situated agents; agents that are directly in touch with their environment, unmoderated by interpretative processes of pre-categorised sensory data.

Some of those systems defy accusations concerning the groundedness of their symbolic representations by not using representations at all (e.g. Brooks 1991). Unfortunately the results of these reactive architectures were unsatisfactory in especially those
domains where physical symbol systems could achieve their biggest successes so far: planning and natural language processing.

This led to the design of hybrid systems using non-representational and representational techniques alike: sub-symbolic processing from reactive architectures for structurally coupling agent and world and symbol manipulation from physical symbol systems for modelling high order cognition. Thus, hybrid systems try to combine the unmediated embodiment of a sub-symbolic reactive system with an adaptive situatedness that previous symbol manipulation systems were statically fixated in. This kind of cognitive architecture is often used to simulate human agents in examining human-machine interaction scenarios. These modular systems feature a wide array of behavioural phenomena that can also be observed with humans.

2.1 Iconisation, Discrimination and Identification

The hybrid approach goes back to Harnad (1990). In his proposal, he outlined the symbol grounding process taking place in three stages: from 1) iconisation over 2) discrimination to 3) identification. Iconisation describes a property-preserving mapping from activations in analogue physical space to representations in digital signal space. These low-level representations serve as a basis for categorising and thereby recognising physical activations. Discrimination is a property-dependant classification of examples from signal space into discrete categories. This allows for a moderate generalisation of sensory data exercising e.g. prototype concept learning. Identification eventually equals some of those categories by abstract criteria like functional or logical equivalence.

2.2 Hybrid Cognitive Architectures

Hybrid architectures combine the advantages of sub-symbolic and symbol processing methods and are thereby able to express a great range of skills (Sun 1995). With Clarion Sun (2000) created an architecture that is hybrid from scratch, accessing and synthesising both local (symbolic), and distributed (sub-symbolic) representations of knowledge. The sub-conceptual level uses connectionist methods of knowledge representa-
tion and acquisition, whereas the conceptual level uses symbolic processing methods (Sun 2006).

According to Sun, cognitive architectures are furthermore principally modular.

In relation to building intelligent systems, a cognitive architecture specifies the underlying infrastructure for intelligent systems, which includes a variety of capabilities, modules, and subsystems. (Sun 2006)

Therefore, Clarion is also modular. It has sub-systems for decision making, for memory access, for motivation and meta-cognition processes. These modules are 1) the action-focused subsystem (AS), 2) the non-action-focused subsystem (NAS), 3) the motivational subsystem (MS) and 4) the meta-cognitive subsystem (MCS). The AS executes mental or motor actions, the NAS represents knowledge, the MS provides state evaluations and the MCS coordinates the interaction between other subsystems.

Each module has a connectionist and symbolic processing level. The two levels of each module communicate in both directions, each one top-down and bottom-up. The connectionist level in each module allows for unsupervised classifications without a priori knowledge of the domain. Categories can therefore continuously be adjusted to current circumstances. Various connectionist learning methods are employed, “capturing statistically significant features of the environments” (Sun 2006).

However, Sun asks “Are there too much specialized mechanisms?” Because of the variety of cognitive phenomena Clarion expresses, his answer is in favour of his system. Sun's motivation behind introducing the action-focused sub-system is modelling “everyday” actions. Therefore the AS carries out actions reactively without planned nor reflected regulation from the meta-cognitive subsystem. Rather than regarding this reactive aspect as a functional part of cognition, like Sun, in the following we concentrate on the reactive level as possible grounding for higher level cognition.

2.3 Sensorimotor Maps and the Symbol Grounding Problem

The symbol grounding problem formulated by Harnad (1990) shows that symbols used by a system to represent its environment can eventually not be grounded by other
symbols inside the system (as is the case in physical symbol systems). If human cognition is to be modelled adequately, symbols should instead be grounded in interaction with the world to which they refer.

Thus Harnad confronted representatives of the physical symbol systems hypothesis with a serious accusation: if the system's representations (its symbols) evolve only from production rules consisting themselves of representations from the same type, how should they be dependant from the world they ought to represent? According to Harnad, for exclusively symbolic systems, the answer must be the following. Because the symbol's function, and thus within a symbol system also their structure, is predefined by the system's developers. Harnad compares this approach to produce meaning with the attempt, to learn Chinese from a Chinese-Chinese dictionary.

2.4 Reactive Architectures

An alternative approach to symbolic processing (within purely symbolic systems or hybrid architectures) is to enable agents to react to environmental changes exclusively by structurally coupling actions with specific sensor activations. Hybrid architectures do this by supplying high level cognition (symbolic manipulation processes like searches in problem spaces) with the categories produced by low level cognition (sub-symbolic classification measures like artificial neural nets). Intending to prevent this functional fragmentation of cognition into two levels, reactive models like Brooks’ (1986) subsumption architecture, Rosenschein and Kaelbling’s (1995) situated automata or Arkin's (1998) behaviour-based robotics present functionally unified approaches for explaining intelligent behaviour. Reactive architectures also represent only a few aspects of the surrounding environment. No explicit knowledge of how to identify an object or what to do in specific situations is stored in the systems’ states.

Sensorimotor maps are a way of providing an agent with sub-symbolic conditional representations of his environment for successful interaction. Although several scenarios engulfing the use of cognitive maps have already been conducted (especially in psychologically motivated AI), the processes behind the generation of such maps are

2 The most popular examples are the vehicles by Valentino Braitenberg (1986).
subject to most recent research (e.g. Butz et al. 2010). We approach the modelling of a reactive agent from a similar perspective. Yet, to our knowledge, the methods we use are unique in modelling reactive behaviour. The most crucial difference from other approaches is that we do not predefine a sensorimotor map to model graph-based planning. Instead we regard planning (among others) as an emergent phenomenon that arises as soon as a sufficient number and complexity of different contexts can be differentiated – this will be detailed in 5.2.1. In doing so we follow Brook’s (1987) statement: planning is just a way of avoiding figuring out what to do next.

Yet, planning itself is not within the scope of this thesis. Instead the goal is to provide a simulated agent with a sensorimotor map representing relevant aspects of his environment and thereby allowing him to optimise an estimated reward function. Concerning the assessment of the agent’s behaviour thus it has always to be kept in mind that it has no explicit memory and has to behave exclusively reactive.

3 Model

The functional separation of cognition into modules employing low level processes and high level processes by hybrid architectures is regarded with scepticism as Sun (2006) points out.

One argument against CLARION would be that in general, more constrained architectures provide deeper explanations, rather than those with a larger pool of specific mechanisms for specific classes of phenomena. (Sun 2006)

Unfortunately, modelling high order cognition seems to come with the need to use methods that are principally different from those available to connectionism.

This thesis explores the possibility of high order cognition – and its functional specialisations (Sun’s modules) – being emergent from simple yet numerous basic processes taking place in parallel at different levels of abstraction. Those layers are not to be confused with Clarion’s sub-conceptual and conceptual layers in each one of its modules. Although their functional part within the architecture may in the end very well be quite
similar, speaking of layers in the context of this work, they are understood to be
grounded in the agent’s interaction with its world rather than inflicted by the designer.

3.1 Simulation Environment

For simulating the agent and its environment we used Simbad 1.4\(^3\). Simbad is a GNU-
licensed 3d robot simulator that targets explicitly academic users searching for a rather
simple way of exploring and evaluating AI algorithms. Illustration 1 shows the standard
graphical user interface and the 20 times 20 square in which the agent's movements can
be observed.

\(^3\) This seems to be both, the most recent and the last version of this GNU-licensed Java 3d robot simulator. For
more details visit http://simbad.sourceforge.net/
To the left of the main window the agent's current state is displayed in the inspector window. The controls below enable changing perspective and controlling the time flow for the simulated environment.\textsuperscript{4}

Simbad also features a \textit{background mode}. In this mode, Simbad runs approximately 16,000 simulation steps per second – a time factor of roughly 1,500 times faster compared to standard mode. This turned out to be very useful when critical parameters had to be adjusted and the agent’s long time performance had to be evaluated. The graphical user interface while running in background mode can be seen in Illustration 2.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Illustration2.png}
\caption{Simbad background mode graphical user interface}
\end{figure}

\textsuperscript{4} Because of relatively slow movements and frequent periods of orientations, it is recommended to set the time factor to 20 at all times.
In order to be able to use textures, Simbad’s source code had to be adapted. Integrating the option to transform the environment from a planar topology to a cylinder or doughnut also made modifications necessary. Furthermore the agent's light sensors and mounted camera (both utilizing the `Eye-class`) could only be confirmed to work on Microsoft Windows XP and the deprecated Java3D version 1.3.1\(^5\). Therefore we decided to use mainly the agent's position as sensory input. Aware of the problems concerning availability of exact positions within an absolute spacial reference system in real world situations\(^6\), the continuity property of changing position values, however, served our purposes well as will be expressed in more detail in 4.2.

### 3.2 Sensorimotor Equipment

Simbad offers a multitude of sensors that can be used to change the agent's internal state according to environmental changes. Proximity sensors comprise for example bumpers, sonar, laser, infra-red and supersonic devices. Unfortunately, in background mode only bumpers and sonar is available. By extracting the agent’s location using its Java3D-vector, it could be arranged to use positioning as another two-dimensional sensory input.

Unfortunately, the light sensors turned out to be quite unsatisfactory concerning measurement accuracy. Determining the agent’s current position could therefore not be implemented by referring to ground markings as was initially intended. Unpredictable delays and tolerances for changes in brightness made clear readings impossible. Instead the agent’s position was directly input as sensory data.

Simbad enables two different ways of controlling the agent's motor activation. The **standard model** sets translational velocity and rotational velocity separately. The **differential kinematic model** on the other hand directly controls the two wheels moving the robot. Having a physical model in mind, we decided to use the differential kinematic


\(^6\) We intended to rather take biological agents as example than artificial ones that possibly feature GPS sensors.
model because specifications of a physical agent (like wheel distance or wheelbase) should remain generic.\footnote{As test runs at the end of the development phase showed, with the presented learning algorithms, the agent is able to learn using both standard and differential kinematic model.}

3.3 Intentions and Expectations within a Data Structure

Intentions and expectations have been symbolically modelled as belief desire intention architectures (e.g. Georgeff et al. (1999), Wooldridge (2000)) since the nineties. More recently this paradigm of human problem solving is entering sub-symbolic domains as anticipatory classifier systems. Although, according to Butz (2002), the methods employed go way back to 1978, the emphasis on the anticipatory aspect has occurred quite recently.

Intentions presuppose some motivational force driving the agent to desire or to fear a certain state. The analogue to biological desire is defined as: satisfying certain needs by acquiring the physical correlate of what has earlier been experienced as concurrent with satisfaction. Accordingly, fear is defined as: circumventing harm by avoiding sensor activations similar to what has earlier been experienced as concurrent with harm. The evaluation that the agent experiences clearly become subjective by using this definitions.

Some sort of representation for expectation from previous experience has to be already established to be able to speak of intentions in that way. These expectations can be modelled using a activation state automaton with probabilistic transitions. Successive states are connected and confirmed edges get stronger while non-successive get weaker. Thus, we use a modification of Hebbian learning where nodes get connected not when firing together but when firing successively. This modification breaks temporal symmetry in conditions for “wiring” by replacing concurrency with sequentiality. Those nodes activated sequentially are connected, rather than those activated concurrently.

Illustration 3 shows the reinforcement of edge $a$ connecting prototype $A$ with prototype $B$ following a successive activation. In our case “firing” refers to the activation of one node that is closest (by euclidean distance) to the sensorimotor state the agent is in. The
result is a dynamic Voronoi tessellation of sensorimotor space by those states prototypically represented as nodes within the map.

![Illustration 3: Successive motor states: from A to B](image)

Next to expectations, intentions require a representation of the desirability of the expected state. Therefore each motor map refers to a unique vector in sensor space and its according evaluation – this will be detailed in 3.3.4.

For implementing the graph data structures we used the Java Universal Network/Graph Framework (JUNG) 2.0.1. It provides one of the most comprehensive libraries concerning processing and evaluation of graphs. Many of the standard graph theory algorithms are already implemented (like centrality, PageRank or A*).

### 3.3.1 Generalisation

By weakening the connection strength between two nodes, eventually a certain threshold will be undercut. As soon as this happens, the edge is removed from the graph. How long an unconfirmed edge endures is directly dependant from three factors.

1. **Weakening value**: Each time a connection \( c \) is disconfirmed, an internal value \( s_c \) is reduced according to formula I. For the present version \( \text{weak} = 0.05 \) and the initial value of \( s_c = 10 \).

---

2. **Confirmation value**: Confirmation takes place in much the same way as can be seen in formula II \( \text{conf} = 1 \).

3. **Removal threshold**: We used a value of \( \text{thres} = 0.01 \). The removal threshold, however, applies not to \( s_c \) but to a reliability value \( \text{rel}(c) \) calculated according to formula III and plotted in Illustration 4.

\[
\begin{align*}
\text{s}_c &= \text{s}_c - \text{weak} \quad \text{(I)} \\
\text{s}_c &= \text{s}_c + \text{conf} \quad \text{(II)} \\
\text{rel}(c) &= (0.5 \times \tanh ((\text{s}_c - 10) \times 0.5) + 0.5) \quad \text{(III)}
\end{align*}
\]

**Illustration 4**: Reliability function

**Illustration 5**: Local reduction of alternative connections (dashed)
The reliability of an edge is also indirectly determined by the amount of activation in its neighbourhood. Illustration 5 shows, we imposed *local pressure* on alternative edges. Confirming the transition $A \rightarrow B$ comes with weakening outgoing edges from $A$ and incoming edges to $B$ – all the other edges remain unchanged. This avoids losing edges (and therefore prototypes) that are not visited frequently but are reliable nonetheless.

*Illustration 6: Generalisation by removing prototypes (from left to right)*

However those connections introduced in an overly specific situation that cannot be confirmed later on get removed in favour of more reliable transitions. This advocates monocausality\(^9\) and models situations as having one definite outcome. Because usually less confirmations occur than disconfirmations, they are employed stronger. Local pressure furthermore enables clean splits of maps as will be outlined in 3.5.1.

Generalisation within a map of prototypes takes place by removing nodes that have lost all incoming and outgoing connections. Although this will later be explained in more detail, let us now assume those nodes are simply dropped. Illustration 6 shows the consequences of removing the node holding the sensorimotor vector within the bluely shaded area.

Illustration 6 visualises the borders of the *1-nearest-neighbour-queries* that lead to the activation of all nodes. Within the Voronoi diagram it can be seen that the vector’s adjacent prototypes’ area is increased after removal. It is important to note that the adjacent

\(^9\) Monocausality is the tendency to ascribe single causes to events.
prototypes within the Voronoi visualisation are not (and should not) be consistent with adjacent prototypes within the sensorimotor map.

The pathological case of the set of adjacent nodes in the Voronoi diagram completely covering the set of adjacent nodes in the sensorimotor map should be avoided because of redundancy. Using a nearest-neighbour-query, nodes that are close within the sensorimotor map do not need to be close by in sensorimotor space. Specialisation premises certain criteria for introducing a new node that tend to avoid this case.

### 3.3.2 Specialisation

In case the same node is activated successively, a consequence might be to connect this node to itself. Assuming, however, transitions only take place if activations change, makes reflexive edges impossible. Instead, motor nodes stay active until further activation leads to a decline in evaluational feedback. In case this happens, a nearest-neighbour-query containing the momentary motor activation is initiated to find a suitable successor. If there is no such successor – the node returned by the query is the same node initiating the query – a new motor node is introduced (see Illustration 7).

![Illustration 7: Creating new motor node B from initially ambiguous node A/B](image)

Each motor node holds a) its momentary activation that is subject to the optimisation processes outlined in 3.4.4 and b) the activation of the motor node that initiated its
creation. By maintaining b), a possibly suitable successor to the momentarily active motor node can always be found, because, either the node querying for a successor is in fact the node that created the successor, or it is at least similar to it.

The newly introduced edge between the creating node and the created node is initially weighted with \( \text{rel}(c) = 0.5 \) but because of the sigmoid function plotted in Illustration 4 it quickly converges to values close to zero or one. The initial activation of a new motor node is zero on all dimensions. It also converges to a local evaluational maximum through random optimisation. As soon as this maximum is reached, a decline in evaluational feedback leads to the activation of a new node and the process starts anew. Either evaluation keeps continuously increasing or the number of adequately specialized successive motor nodes is increased.

### 3.3.3 Sequences of Motor Prototypes for Successful Interaction

Within a saturated motor map (when no new node needs to be introduced any more), momentary declines in evaluational feedback can be countered by activating a new motor prototype. Usually it cannot be expected for the evaluation to continuously increase, therefore more than one prototype needs to be optimised towards its local maximum.

By moving about in its world, the agent changes the conditions for successfully executing previously optimised motor activations.\(^\text{10}\) Using a motor map that allows to infer beneficial sequences of motor s, the agent is able to reactively optimise each prototype individually and to exploit evaluational outcome without the need to explicitly represent its environment through sensory data.

### 3.3.4 Sensor States as Context for Motor Sequences

Sensory data can, however, serve as a helpful indicator for the need to change the momentarily active motor map. Acquired motor sequences (saturated motor maps) can therefore be contextualised using the sensory information available (see Illustration 8). Upon perceiving a change in sensor data the agent is capable of proactively activating the

\(^{10}\) In case the agent turns by 180° it usually cannot employ the same actions for reaching the same goal as before.
Grounding Symbols by Consequences and Intentions

according motor map. So sensory information serves as a hint to what set of actions is adequate.

Illustration 8: Contextualised motor prototypes in one sensor prototype

To be able to organise different sensor states the same map structure is used that has been introduced to keep track of effective motor activations. Each sensor prototype contained in this map holds a vector in sensor space and a map of motor states. The motor prototypes within that map are optimised while that according sensor prototype is active. Although it is only sketched within this thesis, generalisation and specialisation of the sensor map can be conducted very similar to the motor map described in 3.3.1 and 3.3.2. For now the sensor map is static, still the intended mechanisms shall be outlined briefly.

3.3.5 Sensorimotor Graph

Sensor nodes consist of a) the environmental evaluation (in contrast to the extrapolated evaluation presented in 3.4.2) at b) one specific vector in sensor space and c) a motor map. Illustration 9 shows sensor prototypes being connected using the same edges as motor prototypes. These connections are also exposed to local pressure:
connections get established or reinforced for successive nodes, alternative connections are locally reduced. Cutting under a minimal bound, these connections are removed, also removing completely disconnected prototypes \textit{from the active map} (potentially forming a new map, see 3.5.1 for details).

\begin{center}
\includegraphics[width=0.5\textwidth]{sensor_map.png}
\end{center}

\textit{Illustration 9: Sensor map showing transition from A to B}

A sensor prototype is activated if it is returned as result from a 1-nearest-neighbour query using the sensor vector momentarily perceived by the agent. So whereas new motor nodes are activated if environmental feedback \textit{by evaluation} occurs, sensor nodes are activated if environmental feedback \textit{by sensors} occurs. In terms of supervised learning the former represents \textit{labels} while the latter represents \textit{examples}.

The sensor space is thereby generalised according to concurrent environmental evaluations resulting in a Voronoi tessellation like in Illustration 6. A new sensor prototype is introduced if expected evaluation and factual evaluation differ significantly. Even if the prototype does never get activated – yet maintained –, the agent benefits from its existence, because it helps specifying evaluational feedback at certain sensor states and sensor states close by.

3.4 Initiating Momentum

This section will describe how the agent receives the impetus necessary for initiating interaction with its environment. The reasons for interaction are mainly twofold: intentional (although not goal-directed) and exploratory. \textit{Intentional actions} are motivated by the agent's representation of its environment and the need to acquire positive evalu-
ational feedback. *Exploration* on the other hand is a randomisation factor inflicted on actions especially during phases of low evaluation.\textsuperscript{11}

Intentional interaction is the result of successful exploration beforehand. A learning process is initiated that is *supervised* through feedback from the environment, yet free from the semantic innatism described by Taddeo and Floridi (2005). Although the “supervised” part might raise suspicion, no reasoning intelligence is necessary to generate the evaluations the agent makes use of. Instead, it forms its sensor and motor categories independently and autonomously.

In case feedback is negative the agent tries to make use of uncertainty. If feedback is positive it tries to repeat actions experienced as beneficial as exact as possible. The uncertainty which originates in imposed noise is used for a random optimisation process that is executed in each motor prototype. In case these optimisations turn out to be successful, the noise is being gradually reduced, relative to positive environmental feedback.

Therefore, without any representations, the agent has to learn by trial and error. The average randomisation imposed for exploration reduces continuously as the agent is able to resort to previous experiences that prevent badly evaluated situations.

### 3.4.1 Noise as a Means of Exploration

Each activation is modified by parametrised Gaussian noise. Therefore the executed activation is never irrelevant; in the worst case it is exploring possibilities. No matter how strong randomisation is, most of the randomised activations will still be close to the stored value. Given enough activations, however, the distribution will gradually shift towards a fixed position in n-dimensional motor space. During optimisation, the distribution’s radius is reduced relative to increasing evaluations.

\textsuperscript{11} It might sound improbable that exploration takes place in phases of *low* evaluation instead of phases of *high* evaluation. It must be emphasised, however, that our goal is to model *low level interaction*. Therefore speaking of exploration within the context of this thesis is different from the exploration e.g. of infants or primates that is primarily motivated by curiosity. (More on curiosity as motivation in Butz et al. (2009).) When speaking of exploration in our context, it may be rephrased by “figuring out how to do it right”.

23
Emergence of Meaning from a Stream of Experience

In an evaluational neutral situation the randomisation is applied according to Illustration 10 with \( \text{radius}_{\text{gau}} = 0.2 \). Each value \( v_m \) within the n-dimensional motor vector \( \vec{v}_m \) is modified according to formula VII by the randomisation function IV. In case \( v_m - r_{\text{Neg}} < \text{min} \), \( r_{\text{Neg}} \) is restricted to the processable motor interval \([\text{min}, \text{max}]\) by applying formula VI. If \( v_m + r_{\text{Pos}} \geq \text{max} \), formula V accordingly.

\[
\begin{align*}
    r_{\text{Pos}}, r_{\text{Neg}} &= \frac{\text{radius}_{\text{gau}} \ast (\text{rnd}_{0,1}^2 + \text{rnd}_{0,1}^2)}{2} \quad (\text{IV}) \\
    r_{\text{Pos}} &= r_{\text{Pos}} \ast (\text{max} - v_m) \quad (\text{V}) \\
    r_{\text{Neg}} &= r_{\text{Neg}} \ast (v_m - \text{min}) \quad (\text{VI}) \\
    v_{m}' &= v_m + r_{\text{Pos}} - r_{\text{Neg}} \quad (\text{VII})
\end{align*}
\]

By making \( \text{radius}_{\text{gau}} \) evaluation dependant, the agent acts exploratory while receiving negative feedback and thus his sensorimotor state changes. In desirable states, however, randomisation is minimised. Illustration 12 visualises \( \text{radius}_{\text{gau}} \) in relation to momentary evaluation starting from \( \text{radius}_{\text{gau}} = 0.3 \).

Formula VIII shows the according equation. In the project and in Illustration 12 we used \( \text{radius}_{\text{max}} = 0.6 \) and \( \text{steep} = 2 \). Computation of \( \text{eval}(\vec{v}_3) \) is explained in formula XI.
radius\_gau = \frac{-radius\_{max}}{2} \ast \tanh\left(\text{steep} \ast \text{eval}\left(\frac{\text{vs}}{\text{eval}}\right)\right) + \frac{radius\_{max}}{2}

Illustration 12: Randomisation function

Illustration 11: Discrete evaluation points
3.4.2 Extrapolating the Evaluation Gradient

Evaluations in a natural environment are assumed to be subjectively close to discrete. Sophisticated agents, however, are able to anticipate these discrete changes in evaluational feedback. Illustration 11 shows the five discrete evaluations from our environment, green representing satisfaction ($eval_{env}=1$), red dissatisfaction ($eval_{env}=-1$).

The best an unprejudiced agent can do in situations receiving unpredicted evaluations is to memorise the concurrent sensor state and to extrapolate a continuous function relating similar sensor states with an evaluation gradient for future use. Although the presented model is theoretically capable of introducing extrapolated evaluations on its own, the need for reproducibility made it necessary to supply the agent with evaluational relevant sensor states beforehand.

Illustration 13: Extrapolated evaluation
The extrapolated evaluation $\text{eval}(\vec{v}_S)$ of the momentary sensor vector $\vec{v}_S$ is calculated by summation over the products of this sensor vector’s similarity to the set $VR$ of all relevant vectors and their evaluation by environment.\(^{12}\) The Result from applying formulae IX – XI to the discrete evaluations from Illustration 11 is Illustration 13.

$$
distance(\vec{v}_S, \vec{v}_R) = \sqrt{\sum_{n=0}^{d} (\vec{v}_{S,n} - \vec{v}_{R,n})^2} \tag{IX}
$$

$$
similarity(\vec{v}_S, \vec{v}_R) = \frac{1}{1 + distance(\vec{v}_S, \vec{v}_R)} \tag{X}
$$

$$
\text{eval}(\vec{v}_S) = \sum_{VR} \text{similarity}(\vec{v}_S, \vec{v}_R) \ast \text{eval}_{env}(\vec{v}_R) \tag{XI}
$$

Note that because of the similarity function’s convergence properties, the average evaluation in the indicated sensor space in Illustration 13 is not zero but approximately +0.5. This seems intuitively correct, because three points are evaluated positively, while only two are evaluated negatively. Therefore the whole environment is rather desirable. This is useful for an average evaluation of an arbitrary set of prototypes, which will be detailed in 5.2.1.

### 3.4.3 Smoothing Evaluation Gradient

As has been outlined in 3.3.2, new motor prototypes are introduced if evaluation declines. More precisely, prototypes are introduced if the weighted two point average gradient of the extrapolated evaluation function falls below zero. To increase the agent’s motivation, we set $need = 0.1$. As detailed in function XIII, $\text{gradient}_{smooth,t}$ averages the difference from this cycle’s and last cycle’s evaluation $\text{gradient}_{t}$ with its result from last cycle $\text{gradient}_{smooth,t-1}$. Therefore the change in evaluation is not overly sensitive to sudden changes but instead takes the $\text{gradient}_{smooth,t}$ of all previous cycles into account by a fraction of $(s_{sc} - 1)/s_{sc}$. In our case $s_{sc} = 20$.

---

\(^{12}\) Speaking of evaluation we always mean the extrapolated, continuous version, in case we refer to the discrete evaluation inflicted directly by the environment, we indicate this explicitly.


\[
\text{gradient}_t = \frac{\text{eval}(\overrightarrow{v}_t) - \text{eval}(\overrightarrow{v}_{t-1})}{2} - \text{need} \quad (\text{XII})
\]

\[
\text{gradient}_{\text{smooth}, t} = \frac{\text{gradient}_t + \text{gradient}_{\text{smooth}, t-1} \ast (s_{sc} - 1)}{s_{sc}} \quad (\text{XIII})
\]

The main reason for smoothing the evaluation gradient is to be able to identify mainly negative situations and therefore not to create new prototypes every time evaluation is a bit worse than before. With \( s_{sc} = 20 \) it takes approximately 20 cycles until \( \text{gradient}_{\text{smooth}, t} \) has adapted to a constant new evaluation. These 20 cycles can be used by the agent to optimise those motor activations that result in a worse evaluation for only a short time but are potentially adequate.

Furthermore, with each newly introduced motor prototype we set \( \text{gradient}_{\text{smooth}, t} = 1 \). This allows the optimisation process to work a few cycles on the motor state before another prototype is activated. Without smoothing, new motor prototypes would most likely be left instantly.

3.4.4 Optimizing Motor Activities

The motor optimisation process mentioned before is a bidirectional random optimisation algorithm. According to Matyas (1965) random optimisation finds local maxima within multidimensional spaces by randomly selecting a point within the Gauss distribution around starting position. In case evaluation at the new point is higher, the process starts anew by selecting the new point as start point. The randomisation is conducted by the processes outlined in 3.4.1, the evaluation by those in 3.4.2 and 3.4.3. Motor prototypes are optimised by setting their vector to the randomised vector within the parameterised Gaussian noise we applied. Break condition is \( \text{gradient}_{\text{smooth}, t} < 0 \), which means it reached a local maximum – either because this motor state is maximally exploited or rather useless.

Because of the way we extrapolate evaluations, no evaluation plateaus can occur (see Illustration 13). Reaching plateaus could render the optimisation algorithm useless and the agent helpless. Nonetheless the agent can be trapped in local maxima in case \( \text{radius}_{\text{gau}} \) is set to small.
We modified the random optimisation algorithm slightly by multiplying the step vector (the difference between starting point and probing point) by a multiple $f_{ac}$ of $\frac{\text{gradient}}{\text{smooth},t}$. In case motor activation has been evaluated positively, the amount of desirability determines the step width towards the new point. If, however, the new activation has been evaluated negatively, the agent reverses direction of the step vector – hence “bidirectional”. The factor $\frac{\text{gradient}}{\text{smooth},t}$ is multiplied with $f_{ac}=10$ in our case.

Because a) there is not one single optimal motor activation for multiple goals and b) the adequacy of motor activations changes on activation (due to the agent’s movement), several prototypes containing conditional motor activations for reaching local evaluational maxima need to be maintained. On having exploited one node, the agent needs to know how to go on in order to maintain a sufficiently high evaluation. This purpose is served by keeping the activation vector of creating node $A$ in created node $B$ as described in 3.3.2. Thus, an optimised sequence of motor activations allows the agent to successfully navigate in a continuously changing sensor space.

3.5 Emergence of Meaning

Within the described model, where could meaning be found? The motor prototypes do not denote anything within the agent’s sensor space. How could they stand for anything perceivable to the agent? Not only do they not represent anything perceivable, they are also themselves not represented sensory. Therefore they can hardly be called meaningful; what would it be to the agent, that means something?

The relevant sensor states, however, directly originate in the agent's environment. The agent can recognise and even compare them to the momentary sensor state he is in. So those are sensory accessible, potential carriers of meaning. What could they denote and how could it be that by denoting they bear meaning? Following a pragmatics approach we equal “$A$ means something to the agent” with “the agent knows how to successfully interact with $A$”. In our case, successful interaction is what a saturated motor
map allows the agent to do. Consequentially we can say that the meaning of the agent’s sensory representation is *what he knows to do with them*.

Beyond the scope of this thesis is an implementation showing how those sensor prototypes may themselves be constituents of high order meaning. In the following, however, one possibility shall be outlined briefly. Executing these still theoretical processes could be circumvented at this stage of development by predefining the agent’s sensory states as well as those structures made up of sensor prototypes, in the following generically called *contexts*.

### 3.5.1 Contexts of Sensor Prototypes

Sensor prototypes are organised in much the same way as motor prototypes: within a map. Therefore connections between successive sensor prototypes are confirmed while all the other outgoing connections from the last prototype and all the other incoming connections to the new prototype are reduced and eventually removed. As has been noted in 3.3.1, this kind of local pressure allows for “clean splits”. That means that by removing alternative edges, eventually alternative nodes will be separated in isolated graphs.

*Illustration 14: Separation of a new contextual map*
Grounding Symbols by Consequences and Intentions

Illustration 14 shows this process of isolating alternatives. In case transition $A \rightarrow B$ occurs more often than $A \rightarrow B'$ and $B' \rightarrow A$, eventually those alternatives get removed. The transition $A' \rightarrow B'$, however, remains unchanged because no information concerning $A' \rightarrow B'$ can be inferred from $A \rightarrow B$ occurring. Two new contexts (1 and 2) are resulting from the removal of edges and the division of the original sensor map.

3.5.2 Conditions for Sensor Prototype Transition and Creation

Although local pressure is more selective than global pressure, it can still be guaranteed that subgraphs that are not visited frequently get isolated at some point in time. Imposing local pressure on edges within motor maps mainly served the purpose of exploring this form of generalisation and to infer plausible values for the parameters $\text{thresh}$, $\text{weak}$ and $\text{conf}$. Yet its main purpose – to divide a given graph in two subgraphs being alternatives to each other – was not employed, because within motor maps, prototypes not visited often enough can just be dropped.

Applied to the sensor map, however, local pressure on connections can theoretically create different interpretations of sensory identical situations. Once the main sensor map is split, those different contexts need to be maintained within a high order data structure referring to them and initiating transitions between them.

Therefore transitioning conditions between sensor prototypes need to be introduced. Concerning sensor prototype selection, the only condition momentarily employed is maximal similarity as described in formula $X$ and neither reduction nor confirmation of sensor connections takes place. A plausible additional condition would be reaching a certain maximum/minimum in extrapolated evaluation.

Furthermore creation conditions for sensor prototypes need to be established. Following the scheme from creating motor prototypes, first thing would be to determine whether the momentarily active prototype is ambiguous. This is the case if circumstances enforce a transition, but the only prototype available is the one that is active. “Circumstances enforcing transitions” can be defined quite clearly by a strong environ-
mental evaluation not represented internally through an according extrapolated evaluation.

Employing these conditions will eventually lead to the separation of sensor maps. Frequent sequences remain in one graph, while less frequent transitions between the states of those graphs are removed. This can be thought of as clustering of prototypes by sequentiality.

3.6 Algorithm

In the following the implemented algorithm is outlined in pseudocode. In case the calculation of values has already been explained, we refer to the according formulae. The optimisation process explained in 3.4.4 is shaded blue, specialisation explained in 3.3.2 is shaded orange and the generalisation and removal of nodes and edges explained in 3.3.1 is yellow. sensorvector\textsubscript{t} and motorvector\textsubscript{t} describe the activation at time \( t \).

<table>
<thead>
<tr>
<th>Procedure optimemotor (sensorvector\textsubscript{t}, motorvector\textsubscript{t}) returns motorvector\textsubscript{t+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 eval\textsubscript{t} ← according to formula XI</td>
</tr>
<tr>
<td>2 gradient\textsubscript{t} ← according to formula XII</td>
</tr>
<tr>
<td>3 gradient\textsubscript{smooth, t} ← according to formula XIII</td>
</tr>
<tr>
<td>4 if gradient\textsubscript{t} &lt; 0 and gradient\textsubscript{smooth, t} &lt; 0 then</td>
</tr>
<tr>
<td>5 activemotornode\textsubscript{t} ← get closest node in map to motorvector\textsubscript{t}</td>
</tr>
<tr>
<td>6 if activemotornode\textsubscript{t} == activemotornode\textsubscript{t-1} then</td>
</tr>
<tr>
<td>7 activemotornode\textsubscript{t} ← new node</td>
</tr>
<tr>
<td>8 add activemotornode\textsubscript{t} to map</td>
</tr>
<tr>
<td>9 gradient\textsubscript{smooth, t} ← 1</td>
</tr>
<tr>
<td>10 else</td>
</tr>
<tr>
<td>11 edge\textsubscript{t} ← edge from activemotornode\textsubscript{t-1} to activemotornode\textsubscript{t}</td>
</tr>
<tr>
<td>12 if edge\textsubscript{t} == nil then</td>
</tr>
<tr>
<td>13 edge\textsubscript{t} ← new edge</td>
</tr>
<tr>
<td>14 connect activemotornode\textsubscript{t-1} to activemotornode\textsubscript{t}</td>
</tr>
<tr>
<td>15 else</td>
</tr>
<tr>
<td>16 confirm edge\textsubscript{t}</td>
</tr>
<tr>
<td>17 else</td>
</tr>
<tr>
<td>18 for outedge ← all outgoing edges from activemotornode\textsubscript{t-1} do</td>
</tr>
<tr>
<td>19 if outedge != edge then</td>
</tr>
<tr>
<td>20 reduce outedge</td>
</tr>
<tr>
<td>21</td>
</tr>
</tbody>
</table>

32
Table 1: Pseudocode of motor optimisation process

4 Domain and Adaptivity

The algorithm has been implemented controlling an agent within a simulated environment. The goals were intended to a) make the way the algorithm works at least in principle traceable from outside observation and b) allow the agent to increase performance significantly. The agent solves simple localisation tasks by manoeuvring in a two-dimensional environment (see Illustration 15).

Two different tasks alternate on success. The first task is called “exploring” and it is achieved by approaching one of the targets up to a distance of one unit. By doing so the agent has to avoid harmful red targets (points 1 and 4) and approach desirable green ones (points 2, 3 and 5). The second task is called “retreat” and is successfully fulfilled as soon as the agent approaches his home position (point 0) up to a distance of one unit.

---

13 The size of the environment is a square with a side length of 20 units.
4.1 Topology

Having experimented with doughnut and ring topologies these turned out to effect some problems. Because of the limited area within a doughnut topology, the agent activated its goals often by accident rather by successful optimisation of motor prototypes. Using a ring topology turned out to yield problems as well, because the extrapolated evaluation function (plotted in Illustration 13) is no longer continuous. Changing the topology turned out to require fundamental changes in the geometry of the environment the agent moves in. Simbad does not support these changes.

![Illustration 15: The agent’s world](image)

Consequently we decided train the agent in a completely planar topology. Being forced to use an infinitely large environment, pathological cases can occur where the agent leaves the 20 by 20 square marked by texture. Although it usually returns to those locations evaluated with a local maximum, there are situations where this takes unac-
ceptably long. Solving tasks using a different topology therefore necessarily requires changing the simulation environment.

4.2 Discreteness versus Continuity

Constraining the accessible space by sensory perceivable obstacles is a sound next step in development. Proximity sensors (e.g. horizontally aligned brightness sensors) for perceiving these obstacles can also substitute the absolute localisation sensor that has been criticised as biologically implausible in 3.2.

It is of utmost importance, however, to supply the agent with enough sensory information to fulfil the given tasks. At this reactive stage the agent is not able to reach either exclusively discrete goals or goals that are sensory indistinguishable from non-goals. The first thing our agent does is to associate unexpected feedback (satisfaction or harm) from the environment with concurrent perceptions, to be able to recognize and autonomously evaluate sensor states afterwards. So the agent needs to be able to anticipate environmental feedback from sensory information.

4.2.1 Environment

Discrete sensorimotor interaction describes scenarios where the agent's movements and perceptions are a subset of a finite set of all possible movements and perceptions (e.g. chess boards). This kind of interaction is used to reduce complexity and maintain traceability of the agent's actions. While this approach is adequate in specific domains, we believe modelling grounded representations clearly requires not only infinity but also continuity of possible sensor and motor states. Sensorimotor categories are exactly what needs to be established grounded through the agent, not predefining by the developer.

The freedom from continuous sensorimotor space enables a lot of “errors”.14 In our case these occur especially when the agent instead of moving straight towards a target,
is circling there in a quite unexpected manner. It indicates, however, that the agent does neither have a concept of “moving straight forward” nor of “forward” or any orientation whatsoever. By restricting its movements e.g. to “move forward”, “turn left by 90 degree” and “turn right by 90 degree” we would impose these categories and inappropriately assume they fit for its purposes as they do for ours.

4.2.2 Evaluation

Continuity of evaluational feedback on the other hand, is understood as result of a cognitive and/or evolutionary process. The continuity the agent experiences does not originate in the environment, it is rather a constructed heuristics allowing optimisation processes to “get a hold” of the right direction. In case this does not seem intuitively correct, one may think of a situation where we suddenly and unexpectedly hurt ourselves. What happens is a discrete evaluation inflicted purely by the environment. Even if we do not remember consciously what happened during such a incident, we can recognise and avoid similar situations just by feeling uncomfortable upon approach. Biological agents learn this not only during ontogenetic but also during phylogenetic development involving evolutionary processes spanning many generations.

4.3 Relevant Parameters

Some of the parameters that have been introduced in 3.3 and 3.4 turned out to be highly sensitive for the agent’s performance. Because the used values only apply to generating motor maps but not yet to sensor or context maps, they can only serve as a first approximation. By generating different maps in different domains they need to be adapted accordingly.

4.3.1 The Need

The value $\text{need}$ inflicts a constant reduction on the extrapolated gradient $\text{gradient}$, and therefore also on $\text{gradient}_{\text{smooth}, t}$. It causes the agent to evaluate changes more negative than determined by the gradient of the extrapolation function.
alone. Therefore the agent creates prototypes that are optimised not only for increasing evaluation but to increasing it *above a certain minimum* defined by *need*.

The higher this value is set, the more selective the agent is concerning prototypes worth optimising. Setting it too high leads to an over-specialisation because new prototypes are introduced before old ones can be optimised sufficiently. In case the value is too low, however, the agent's ambition for reaching goal states suffers.

### 4.3.2 Motor Randomisation

The applied Gaussian motor randomisation can be modified in several ways. We concentrate on the sigmoid function's steepness *steep* and its limit on the negative end *radius*<sub>max</sub>. At the positive end, another limit than zero makes hardly any sense. In formula VIII we also coupled commencing randomisation with maximal randomisation. This need not necessarily be the case.

The maximal Gaussian radius determines the extent of the agent's exploration. Generally speaking, low values tend to leave the agent with suboptimal prototypes, that are active for a period too brief to reach sufficient optimisation – dependant from *c*<sub>sc</sub>, see 4.3.4 on that. Large values, however, bear the risk of randomising successful states. This can be countered by modifying *steep* or shifting the sigmoid function towards its negative end.

### 4.3.3 Connection Strengths

The connection values determine the coherence of prototype maps. Reducing *conf* therefore leads to the same results as increasing *weak*. Concerning motor maps, these values determine mainly how long ambiguous prototypes will be maintained. Applied to higher order maps (sensor or context maps) the coherence of the map also determines how fast new contexts are created by splitting off parts from already existing ones.

Also the removal threshold *thresh* is constrained in relation to the reliability function described in formula III. The sigmoid function can only make sense if *thresh* is
close to its limit towards negative infinity. Therefore formula III and \( \text{thresh} \) may need to be attuned to each other.

4.3.4 Evaluation Smoothing

The main purpose of \( s_{sc} \) is to smooth the extrapolated evaluation gradient \( \text{gradient} \). The number of cycles available for optimising active nodes is directly dependant from \( s_{sc} \). Choosing this value too high reduces specialisation, a value too low reduces generalisation.

The value \( \text{gradient}_{\text{smooth}, t} \) is a weighted two point moving average that recursively takes its own result from last cycle into account. Alternative methods of noise reduction like triangular smoothing might also be applicable.

4.3.5 Vector Step Factor

The step factor \( \text{fac} \) determines the width of the steps taken from the algorithm towards the presumed maximum. Originally random optimisation chooses the random point as new start point, while we are trying to use the directional information and multiply the step width by a constant factor to increase optimisation efficiency and reduce the risk of overshooting.

Consequently changing this value influences convergence properties of the optimisation process. High values cause it to miss the point of maximal evaluation. Therefore prototypes do not get exploited optimally. Values too low give the momentary value too much inertia, meaning the agent's situation (e.g. orientation) changes faster than optimisation can adapt. Therefore new prototypes are created where existing ones should still be exploited.

5 Discussion

We described a basic system capable of successful reactive interaction with its environment. This can be achieved by consequentially referring to experienced causalities
from its past and intentions towards its future. We argue that those representations, once stabilised, can be referred to as meaningful and relevant to the agent.

Concluding from this project we see the need to consider premises for human-like understanding by considering ontogenetic and phylogenetic development processes first. This is important to note, because for an autonomous agent the biases that biological agents feature need to be acquired exclusively during runtime. Although parameters can be modified beforehand, certain affinities (towards warmth or seeking darkness when fearful) emerge biologically over thousands of generations. In this project the agent obtains these basic tendencies during “ontogenesis”. The extrapolation of the evaluation gradient in 3.4.2 is modelling just that. The crux is that these mechanisms are assumed to be similar to those constituting high order cognition.

5.1 Results

In the following, some of the results from the optimising process are outlined. An example of a stabilised motor representation and the performance within the described Markov environment from 4.1 and 4.2 are presented.

5.1.1 Performance

The performance has been evaluated by using the console output of Eclipse. The number of cycles the agent needed to reach one evaluated point (positive or negative) and return to its original position has been recorded. This equals changing from “explore” context to “retreat” context and back exactly once and will be referred to as one iteration in the following.

Fifty successive iterations have been recorded five times. The amount of cycles necessary for each of the fifty iterations has been averaged over all five runs.\textsuperscript{15} Result is Illustration 16, showing a continuous reduction in cycles necessary to reach a goal. In 7.1 each run can be analysed in detail.

\textsuperscript{15} To simplify matters, the runtime in cycles has been summed up, not divided by five.
Sudden amplitudes indicate that the agent uses motor prototypes that before have been optimised to a different situation and are unintentionally being re-adapted whereas new ones should be introduced. (Illustration 16 shows those incidents only indirectly because of a) the averaging over five test runs and b) several motor prototypes being activated during one single iteration.) To reduce those amplitudes the parameters from 4.3 need to be adapted. Unfortunately a solid heuristics for setting those values could not yet be determined.

5.1.2 Saturated Motor Map

After the second test run a motor map from “retreat” context has been extracted. This was done during execution by setting a breakpoint and manually copying the relevant values from Eclipse’ debug mode into a table. The raw data can be found in 7.2, separated by nodes and edges. We visualised the results in Illustration 17, thicker edges represent more reliable connections.

In this case it can be seen that a circular path within the motor map correlates with good performance (the second run was the best of five). Here this path of continuously reinforced edges is $C \rightarrow A \rightarrow D \rightarrow B \rightarrow C$. This indicates that the agent was able to create a sequence of motor activations that enables successful interaction. In some cases the agent still has to refer to alternatives $E$ or $F$, but local pressure keeps the map quite small nonetheless.
Comparing the optimised activation from one prototype with the fixed activation inherited from each node's creator in Table 2 shows the reason for this circular sequence. The map has been automatically created in such a way that each prototype is superseded by a successor that bears a optimised activation vector which can keep evaluation high even under different circumstances.

Illustration 17: Saturated motor map

5.2 Where to Go from Here?

The next step in this project is figuring out what parameters are most generic. The amplitudes in Illustration 16 should be removable or at least reducible. Therefore ambiguity (represented through nodes F and E in Illustration 17) should also be considerably reduced. It may also be worth considering to adapt the momentarily fixed vectors from the creating prototype to actually preceding prototypes. As soon as reliable reference

---

16 An indicated “opt” refers to the vector being optimised, a “fix” refers to the vector from the creator node as described in 3.3.2.
values for generating a completely stable motor map can be determined, the autonomous creation of sensor maps can be approached.

Coupled with creating sensor maps is handling sensor prototype transitions. Like motor prototype transitions are handled by parenting sensor prototypes, sensor prototype transitions will be handled by parenting context prototypes. Therefore those both may need to be introduced simultaneously. With the introduction of context prototypes emergent phenomena are expected to sprout.

5.2.1 Planning

As has been argued before, planning-like behaviour is expected to be a consequence of, and emergent from certain basic processes. Like the agent learns beneficial sequences of basic motor activations, sensor and context prototype sequences can be learned using the same basic temporal Hebbian learning procedures outlined in this work. By representing those sequences in high order maps and evaluating them as described in the last paragraph of 3.4.2, arbitrarily complex sequences of high order prototypes can be established. Considering high order processes like planning, however, circular paths like in the saturated map from Illustration 17 should be rather exceptional.

5.2.2 Grounding Symbols

The competencies an agent is able to obtain by the described processes allow it to successfully interact with the relatively simple environment we designed. Given planning – and high order cognitive functions in general – do emerge, how can we say, however, that mental symbols are grounded in simple processes like those described?

We argue that a first step towards grounded symbols from a semiotics perspective is grounding mental signs in general, and that signs on their behalf need to be grounded in sensorimotor interaction between agent and environment. Whereas the latter has been widely accepted in cognitive science and has been subsumed under the labels embedded and embodied, the former seems to have remained largely unnoticed. While the dependencies between signs and symbols still need a more thorough examination from
cognitive science perspective, we assume that processes capable of grounding signs are a necessary prerequisite for grounding symbols.

The signs our agent synthesises are contained in a sensorimotor map of meaningful and relevant prototypes. At low levels those representations assign simple motor or sensor activations. At higher levels, however, the obtained sequences can potentially form discrete competencies representing abstract dependencies between world and agent. Among those high level representations also are situations and objects. Intending to unify those representations we chose the term “context” to emphasise their common nature. They serve as permanent background knowledge that is sensitive to changes in the agent’s situation.

Based on the outlined assumptions, sensorimotor representations are at the basis of grounded symbols. We showed a way of generating sensorimotor maps using a temporal Hebbian learning method that synthesises representations by associating sequences of prototypes. While further research is necessary to synthesise more complex representations we could show that an agent is able to learn successful interaction through forming meaningful and relevant representations.

6 Implementation

The implementation of the outlined model can be found as Eclipse project on the attached compact disc or a more recent version at http://code.google.com/p/grepsemag/. Most of the relevant parameters can be changed within the classes in the package src.core.nodes. Alternative environmental evaluations are located in src.core.tools. Changing the environment itself as well as those parameters concerning the simulation physics, sensors and motors can be done in src.simbad.custom. In this package MyEnvironment.java enables manipulation of the agent’s world (e.g. obstacles or textures), Robot.java changes the agent’s properties (e.g. sensors or drive model) and Scene.java defines whether accelerated background mode or standard mode is started. Scene.java is also this project’s main class. Once executed, Simbad is started in standard mode. Clicking “start” executes the simulation.
Illustration Index

Illustration 1: Simbad standard mode graphical user interface.......................................................... 12
Illustration 2: Simbad background mode graphical user interface.................................................. 13
Illustration 3: Successive motor states: from A to B........................................................................ 16
Illustration 4: Reliability function.................................................................................................... 17
Illustration 5: Local reduction of alternative connections (dashed).................................................... 17
Illustration 6: Generalisation by removing prototypes (from left to right)........................................... 18
Illustration 7: Creating new motor node B from initially ambiguous node A/B................................. 19
Illustration 8: Contextualised motor prototypes in one sensor prototype...................................... 21
Illustration 9: Sensor map showing transition from A to B................................................................. 22
Illustration 10: Gaussian distribution with a radius of 0.2................................................................. 24
Illustration 11: Discrete evaluation points........................................................................................ 25
Illustration 12: Randomisation function............................................................................................ 25
Illustration 13: Extrapolated evaluation............................................................................................. 26
Illustration 14: Separation of a new contextual map......................................................................... 30
Illustration 15: The agent’s world.................................................................................................... 34
Illustration 16: Optimisation of motor competency over fifty iterations............................................. 40
Illustration 17: Saturated motor map............................................................................................... 41

Index of Tables

Table 1: Pseudocode of motor optimisation process........................................................................ 33
Table 2: Transitions from saturated motor map................................................................................ 41
Table 3: Performance table................................................................................................................ 46
Table 4: Nodes from motor map....................................................................................................... 46
Table 5: Connections from motor map.............................................................................................. 47
7 Appendices

7.1 Performance Table

<table>
<thead>
<tr>
<th>iteration</th>
<th>1st run</th>
<th>2nd run</th>
<th>3rd run</th>
<th>4th run</th>
<th>5th run</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1235</td>
<td>1491</td>
<td>5268</td>
<td>13518</td>
<td>9071</td>
<td>30583</td>
</tr>
<tr>
<td>2</td>
<td>2588</td>
<td>2466</td>
<td>2022</td>
<td>711</td>
<td>1425</td>
<td>9212</td>
</tr>
<tr>
<td>3</td>
<td>5219</td>
<td>2779</td>
<td>2851</td>
<td>1200</td>
<td>525</td>
<td>12574</td>
</tr>
<tr>
<td>4</td>
<td>7013</td>
<td>1397</td>
<td>559</td>
<td>636</td>
<td>475</td>
<td>10080</td>
</tr>
<tr>
<td>5</td>
<td>4329</td>
<td>776</td>
<td>1592</td>
<td>611</td>
<td>500</td>
<td>7808</td>
</tr>
<tr>
<td>6</td>
<td>3533</td>
<td>655</td>
<td>1569</td>
<td>214</td>
<td>572</td>
<td>6543</td>
</tr>
<tr>
<td>7</td>
<td>1387</td>
<td>688</td>
<td>1591</td>
<td>410</td>
<td>574</td>
<td>4650</td>
</tr>
<tr>
<td>8</td>
<td>1241</td>
<td>732</td>
<td>2245</td>
<td>253</td>
<td>205</td>
<td>4676</td>
</tr>
<tr>
<td>9</td>
<td>1382</td>
<td>2423</td>
<td>9751</td>
<td>4477</td>
<td>682</td>
<td>18715</td>
</tr>
<tr>
<td>10</td>
<td>1868</td>
<td>6678</td>
<td>8458</td>
<td>509</td>
<td>3409</td>
<td>20922</td>
</tr>
<tr>
<td>11</td>
<td>1286</td>
<td>151</td>
<td>2228</td>
<td>3175</td>
<td>236</td>
<td>7076</td>
</tr>
<tr>
<td>12</td>
<td>1183</td>
<td>155</td>
<td>826</td>
<td>1414</td>
<td>225</td>
<td>3803</td>
</tr>
<tr>
<td>13</td>
<td>3224</td>
<td>195</td>
<td>929</td>
<td>571</td>
<td>219</td>
<td>5138</td>
</tr>
<tr>
<td>14</td>
<td>1073</td>
<td>211</td>
<td>682</td>
<td>1472</td>
<td>206</td>
<td>3644</td>
</tr>
<tr>
<td>15</td>
<td>1207</td>
<td>102</td>
<td>645</td>
<td>2274</td>
<td>227</td>
<td>4455</td>
</tr>
<tr>
<td>16</td>
<td>624</td>
<td>375</td>
<td>807</td>
<td>793</td>
<td>186</td>
<td>2785</td>
</tr>
<tr>
<td>17</td>
<td>3815</td>
<td>160</td>
<td>824</td>
<td>785</td>
<td>235</td>
<td>5819</td>
</tr>
<tr>
<td>18</td>
<td>1355</td>
<td>195</td>
<td>2130</td>
<td>12351</td>
<td>202</td>
<td>15809</td>
</tr>
<tr>
<td>19</td>
<td>2091</td>
<td>232</td>
<td>786</td>
<td>735</td>
<td>218</td>
<td>4062</td>
</tr>
<tr>
<td>20</td>
<td>242</td>
<td>230</td>
<td>633</td>
<td>930</td>
<td>210</td>
<td>2245</td>
</tr>
<tr>
<td>21</td>
<td>1367</td>
<td>304</td>
<td>763</td>
<td>2686</td>
<td>257</td>
<td>5377</td>
</tr>
<tr>
<td>22</td>
<td>1005</td>
<td>338</td>
<td>816</td>
<td>1130</td>
<td>214</td>
<td>3503</td>
</tr>
<tr>
<td>23</td>
<td>854</td>
<td>195</td>
<td>806</td>
<td>733</td>
<td>251</td>
<td>2839</td>
</tr>
<tr>
<td>24</td>
<td>680</td>
<td>175</td>
<td>1004</td>
<td>905</td>
<td>267</td>
<td>3031</td>
</tr>
<tr>
<td>25</td>
<td>615</td>
<td>143</td>
<td>852</td>
<td>515</td>
<td>250</td>
<td>2375</td>
</tr>
<tr>
<td>26</td>
<td>1023</td>
<td>103</td>
<td>2130</td>
<td>12351</td>
<td>202</td>
<td>15809</td>
</tr>
<tr>
<td>27</td>
<td>540</td>
<td>129</td>
<td>858</td>
<td>3715</td>
<td>213</td>
<td>5455</td>
</tr>
<tr>
<td>28</td>
<td>665</td>
<td>151</td>
<td>811</td>
<td>536</td>
<td>207</td>
<td>2370</td>
</tr>
<tr>
<td>29</td>
<td>691</td>
<td>147</td>
<td>934</td>
<td>347</td>
<td>182</td>
<td>2301</td>
</tr>
<tr>
<td>30</td>
<td>540</td>
<td>103</td>
<td>980</td>
<td>384</td>
<td>218</td>
<td>2225</td>
</tr>
<tr>
<td>31</td>
<td>961</td>
<td>72</td>
<td>927</td>
<td>425</td>
<td>205</td>
<td>2590</td>
</tr>
<tr>
<td>32</td>
<td>879</td>
<td>314</td>
<td>924</td>
<td>401</td>
<td>158</td>
<td>2676</td>
</tr>
<tr>
<td>33</td>
<td>6204</td>
<td>100</td>
<td>1020</td>
<td>340</td>
<td>315</td>
<td>7979</td>
</tr>
<tr>
<td>34</td>
<td>731</td>
<td>95</td>
<td>658</td>
<td>238</td>
<td>649</td>
<td>2371</td>
</tr>
<tr>
<td>35</td>
<td>634</td>
<td>951</td>
<td>2416</td>
<td>257</td>
<td>1867</td>
<td>6125</td>
</tr>
<tr>
<td>36</td>
<td>6396</td>
<td>164</td>
<td>1758</td>
<td>428</td>
<td>1035</td>
<td>9781</td>
</tr>
<tr>
<td>37</td>
<td>4271</td>
<td>68</td>
<td>5768</td>
<td>653</td>
<td>358</td>
<td>11118</td>
</tr>
<tr>
<td>38</td>
<td>1331</td>
<td>148</td>
<td>386</td>
<td>842</td>
<td>309</td>
<td>3016</td>
</tr>
</tbody>
</table>
Table 3: Performance table

7.2 Saturated Motor Map

7.2.1 Vertices

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Fixed value from creator</th>
<th>Activation value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>core.nodes.NewMotor@c8cc</code>&lt;br&gt;<code>b1, A</code>&lt;br&gt;[-0.44659680046059, -0.9981690860083847]</td>
<td>[-0.16286251069792737, -0.2714684970838651]</td>
<td></td>
</tr>
<tr>
<td><code>core.nodes.NewMotor@1368&lt;br&gt;c5d, B</code>&lt;br&gt;[-0.008563821311013074, -5.902144587523511E-4]</td>
<td>[0.4185879123218078, -0.2007492648807027]</td>
<td></td>
</tr>
<tr>
<td><code>core.nodes.NewMotor@bb1e&lt;br&gt;ad, C</code>&lt;br&gt;[0.0964035988727802, -0.06110113220034756]</td>
<td>[-0.5523182190580108, -0.66353313191201]</td>
<td></td>
</tr>
<tr>
<td><code>core.nodes.NewMotor@1966070, D</code>&lt;br&gt;[-0.008997204863874907, -0.14926764066993184]</td>
<td>[-0.14298613195033244, 0.0379505551214073]</td>
<td></td>
</tr>
<tr>
<td><code>core.nodes.NewMotor@14828e7, E</code>&lt;br&gt;0.2368798781371538, 0.1326324141963799</td>
<td>[-0.15571374316421574, 0.009597358784380144]</td>
<td></td>
</tr>
<tr>
<td><code>core.nodes.NewMotor@bc36ff, F</code>&lt;br&gt;0.9997001259695346, 0.9998805449942668</td>
<td>[-0.008907535800595308, 0.007740000374342378]</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Nodes from motor map
7.2.2 Edges

<table>
<thead>
<tr>
<th>Edge</th>
<th>State</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>core.Connection@fe135d, B → C</td>
<td>20.449999999999797</td>
<td>0.99960550124619</td>
</tr>
<tr>
<td>core.Connection@90d4f2, F → B</td>
<td>11.199999999999946</td>
<td>0.710949502625</td>
</tr>
<tr>
<td>core.Connection@15cd9a, A → D</td>
<td>20.649999999999853</td>
<td>0.99966043311652</td>
</tr>
<tr>
<td>core.Connection@17e6064, F → D</td>
<td>10.449999999999907</td>
<td>0.58358311551726</td>
</tr>
<tr>
<td>core.Connection@101ea1e, C → F</td>
<td>13.049999999999944</td>
<td>0.90783648973785</td>
</tr>
<tr>
<td>core.Connection@755866, C → E</td>
<td>3.999999999999902</td>
<td>0.010986942630593</td>
</tr>
<tr>
<td>core.Connection@126c5a5, D → B</td>
<td>20.849999999999997</td>
<td>0.99970771825024</td>
</tr>
<tr>
<td>core.Connection@f6852d, C → A</td>
<td>20.599999999999866</td>
<td>0.99964746216213</td>
</tr>
<tr>
<td>core.Connection@1feb3a6, E → B</td>
<td>4.0999999999999999</td>
<td>0.011832526132669</td>
</tr>
</tbody>
</table>

Table 5: Connections from motor map
8 Bibliography


Brooks, R. (1987), 'Planning is just a way of avoiding figuring out what to do next', MIT Artificial Intelligence Laboratory.


Winograd, T. (1971), 'Procedures as a representation for data in a computer program for understanding natural language'.

Affirmation
Ich erkläre hiermit gemäß §17 Abs. 2 APO, dass ich die vorstehende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

_________________________________________________
Bamberg, den 14.12.2010