A linear-complexity reparameterisation strategy for the hierarchical bootstrapping of capabilities within perception–action architectures

Mikhail Shevchenko, David Windridge*, Josef Kittler

Faculty of Engineering and Physical Sciences, University of Surrey, Guildford, Surrey GU2 7XH, United Kingdom

1. Introduction

Developing systems of cognitive learning capable of sustaining autonomous behaviour in real-world environments is considered an important research endeavour within the field of artificial agency [3,24,31,40]. The task, as formulated from a cognitive science standpoint, is that an agent should acquire internal representations sufficient to produce adequate responses to events occurring in its surroundings [22]. Conventional artificial intelligence approaches have correspondingly adopted a number of different assumptions concerning intelligent system design (cf e.g. [23]).

In the symbolic approach, for instance, a ‘top-down’ methodology is generally adopted [26]. It asserts that the central mechanism of cognition is a symbolic system with a pre-existing combinatorial syntactic and semantic structure of symbolic elements [10]. This approach argues that the ‘systematicity’ of mental representations relies on the fact that cognitive capabilities demonstrate a certain symmetry and productivity.

The majority of such algorithms, however, rely on symbols defined by internal properties of the system and are therefore neutral with regard to the question of how the symbols are attached to the objects and events of the external environment. A symbolic system that operates only within the domain of its internal representations, thus, cannot be considered a viable model for cognitive reasoning unless there exists machinery for symbol anchoring [15,34].

Standard approaches to the symbol grounding problem link sensorimotor information with symbolic entities by means of fixed mechanisms introduced into the system at the design stage [13,19,12,38] (see also Steels [32] who sets out a mechanism for the grounded acquisition of language). However, in the real world, models defined a priori could not be expected cope with unpredictable changes, or with the high complexity of sensorimotor information. Furthermore, manual attachment of symbols to their semantic meanings would restrict system exploratory capabilities and make the agent unable to develop its representational skills in an autonomous way. A truly cognitive agent might therefore be expected to learn novel symbols via generalising sensorimotor information [18,21,27,39]; the learning process should commence at the level of primitive motor responses and lead to the development of an open-ended structure of high-level behavioural capabilities and cognitive concepts [35].

In a seminal instance of the latter category of symbol grounding, Harnad [16] proposed the idea of categorical perception based on

---

* Corresponding author. Tel.: +44 1483 686043.
E-mail address: d.windridge@surrey.ac.uk (D. Windridge).
detected of invariant sensory projections. The author considers three distinctive stages of learning symbols:

- Iconisation: the transformation of the analogue perceptual signals into internal equivalents;
- Discrimination: a process at aimed detecting invariances in the iconic information and to build up perceptual categories;
- Identification: assigning abstract names to the categories.

In demonstrating possibilities for unsupervised grounding of abstract symbols Harnad, however, does not explicitly consider behavioural aspects of learning. The latter becomes a crucial point for artificial cognition because the problem involves not only acquisition of symbolic concepts such as a language ontology but also gaining corresponding motor functionality [2]. Autonomous learning should hence employ active exploration in order to establish a link between changes in the internal environment and motor actions causing those changes. In such a task-based context the mechanism of acquiring novel behavioural capabilities can be considered as learning to manipulate the environment in a predictable fashion, albeit at the symbolic level.

A robotic architecture that emphasises the role of behaviour for developing intellectual capabilities has been introduced by Brooks [3]. This approach attempts to develop a complex system through networks of simple functional behaviours that map sensors to actuators. Hierarchicality is introduced through the notion of "subsumption", wherein goals defined at each progressive level of abstraction are built in terms of conditionally-enacted tasks defined at lower levels. A number of behavioural tasks, including wall-following and obstacle avoidance, have been successfully demonstrated to show significant advantages in comparison to conventional robotic architectures. However, behavioural approaches of these kinds do not in themselves explicitly incorporate symbolic representation, even if capable of learning (see e.g. [17]: the result of training in this case is represented in an implicit form of associative network states – each network links certain percepts to appropriate actions).

As a potential solution to developing a fully functional cognitive system, an architecture that "seamlessly" grafts the symbol system on top of the percept–action learning system is hereby proposed (see also [28,43,36,37]). In order to overcome the problem of transferring potentially meaningless associative states of distributed sensorimotor representations into the symbolic processing domain, the percept–action learning framework has been modified to accommodate learning explicit parametric representations of high-level percept and motor capabilities. The learning task has hence been reformulated so as to generalise percept–action links in such a way as to make action responses context-free, i.e. independent of the perceptual information irrelevant to given salient feature subspaces regarded as goals. For instance, if the goal is to move an object to position B, the agent should be able to accomplish the task for any initial object position A, irrespective of distractors.

The actual process of building up a hierarchical representation of cognitive capabilities involves the following stages. Initially, the agent performs random exploration of the environment in order to detect ‘distinctive’ perceptual states – perceptual goals. Next, the system attempts to learn a motor solution to attain the goal in a generalised way. Generating appropriate motor responses is achieved via a stochastic search within the domain of available motor capabilities. This stage encompasses the indexing of the perceptual parameter space and the consequent transferring of the updated indices into the motor domain. As a result of such a transference, the system gains the possibility of utilising perceptual attributes for parametric control of the motor capabilities.

Simple perceptual abilities are used for initial structuring of the motor domain; they, in turn, generate prerequisites for updating the space of perceptual parameters and so on. The process of modifying the motor domains is aimed at detecting invariances among corresponding perceptual parameters, and assigning new parameter spaces in terms of the observed invariant structures. Therefore, at each subsequent cycle the system gains a more abstract representation of actions. We term this bootstrapping a hierarchy of motor functionalities.

The reciprocal nature of the parametric percept–action interaction means that the novel behavioural capabilities lead to a re specification of the perceptual symbolisation of the environment, exploiting the new motor parameters to replace the existing perceptual representation. This process thus builds up a hierarchy of perceptual capabilities which jointly (with the corresponding motor functionalities) constitute a hierarchical cognitive world model in terms of the environmental affordances [11,7]. The underlying mechanism of bootstrapping can hence be viewed in terms of the self-determining cognitive capabilities of an active agent.

An experimental analysis of the method will confirm the practical efficiency of such cognitive bootstrapping, demonstrating a linear response of computational requirements with the increase of learning task complexity. Respecifying perceptual capabilities leads to the detection of novel behavioural goals that were unaccessible before because of the highly contextualised contents of the scene representation. This, in turn, increases the robustness of the mechanism driving exploration and also expands the range of possible cognitive capabilities that can be addressed by learning.

In the next section we will present the method for hierarchical bootstrapping behavioural capabilities. We will propose the bootstrapping architecture, the mechanisms for detecting salient perceptual states and generalising corresponding motor solutions. We will also consider the hierarchical aspects of the cognitive bootstrapping mechanism. Section 3 will present an implementation of the approach within the particular learning scenario of solving a ‘shape-sorter’ puzzle game. We will discuss behavioural goals that emerge from the interaction with the shape-sorter environment. A theoretical estimate of the computational profit gained by bootstrapping will be carried out. Section 4 will then present numerical results for the convergence-rate of the bootstrap learning algorithm. It will be demonstrated that the experimental performance of the method correlates with that of the theoretical forecast; we will also show that the hierarchical bootstrapping overcomes the problem of the quasi-exponential increase in task complexity in unconstrained environments. Finally, we will summarise results and discuss future work in Section 5.

2. The hierarchical bootstrapping approach

2.1. Overview of the bootstrapping architecture

The original perception–action architectures [14,35] exploit the idea of directly coupling sensory information to actions, thereby avoiding the traditional recognition phase as an intermediate step. The connectionist approach has been proposed for performing such sensorimotor linking by considering neural network solutions that map certain percepts onto desired actions. Sensorimotor information in this case is implicit and, therefore, syntactically meaningless, i.e. not suitable for symbolic modelling. This sort of representation is referred as the signal system level. Attempts to fill the gap between the signal level and the symbol system mostly rely on manually labelling various stationary patterns that emerge in the sensorimotor state space [5,25]. Developing a solution that
transforms low-level knowledge into symbolic structures in a generic fashion, however, faces problems related with high dimensionality of sensorimotor states.

In our approach the mechanism of perception–action coupling operates directly within the domain of explicitly parametric models – those of high-level percept–action competences. Thus, the bootstrap learning incrementally develops a hierarchy of competences wherein each novel behaviour or perceptual concept emerges through exploitation of already existing capabilities, parametrically redefining the environment in these terms. We will show below that the proposed method of bootstrapping brings with it the possibility of establishing syntactic rules within the hierarchy, such that the percept–action capabilities can be manipulated in a symbolic manner.

The proposed bootstrapping architecture involves two principal stages: learning individual percept–action capabilities and developing the hierarchical world model (Fig. 1).

The hierarchical percept–action part of the system thus proceeds through the following key stages:

- Obtaining primitive action models
- Removing redundant elements of the primitive solutions
- Eliminating constant motor parameters
- Updating parameter vector spaces

The entire mechanism of learning motor solutions for detected perceptual goals thus can be represented as a processing sequence where each cascade is a separate perception–action cycle. The first procedure generates action sequences via random instantiation of already existing elements in the motor domain. It analyses the perceptual feedback and chooses a motor sequence that satisfies the goal (i.e. such sequence instantiation minimises the distance between the current perceptual state and the goal).

A random search within the domain of motor capabilities rarely leads to the optimal solution, especially in cases of complex learning tasks. This process can involve instantiations of actions that are not relevant to the goal or actions that even interfere with the solution. The second stage of processing is thus aimed at eliminating redundant action elements from the motor sequence involved in achieving the goal in order to make the current model closer to optimal.

An internal representation of the action capabilities is then introduced as a function of motor parameters that instantiate particular motor states. As we will see below, a ‘successful’ solution of a behavioural task may correspond to an action sequence containing parameters whose values are constant for any configuration of the environment. In order to detect those constant parameters the system proceeds through a series of exploratory experiments: it attains the same goal state from various initial configurations of the environment.

The objective of learning novel actions is thus to acquire independent motor capabilities that are not simply the combinations of the existing functionalities. To obtain such representations we propose an algorithm for generalising motor parameters. Initially, all non-constant parameters of the action sequence are represented as a novel generalised parameter vector. After that the system performs a non-trivial transformation that replaces the existing vector components by new variables. Those new variables index all possible states of the perceptual goal in the given environmental configuration. Therefore, the algorithm leads to a novel action model that has an independent parameter vector capable of fulfilling all the necessary manipulations in the environment to attain the goal. As we will see later such reparameterisation frequently leads to significant reduction of parameter search space that considerably improves convergence of the learning mechanism.

In our approach we adopt an assumption that the parameterised space of perceptual features can be considered as a perceptual model of the external world. The bootstrapping hence progressively updates the system’s abilities to perceive scenes. For example, the initial capability to perceive object positions and orientations can be replaced by representations where the scene is perceived as labels naming different objects (hence the ‘old’ perceptual parameters defining positions and orientations have here been replaced by a single index that is the object label). Such purely symbolic representations, however, are grounded with semantic, physical meaning through the corresponding motor capabilities to grasp and move objects. The proposed bootstrapping mechanism hence corresponds to aspects of learning affordances [30,33] in an autonomous hierarchical manner [8] where each percept–action capability represents a possibility of interaction with the external physical world.

---

**Fig. 1.** The bootstrapping architecture. Perception–action learning involves four stages: acquisition of a primitive model, removing redundant action chains, elimination of constant motor parameters, updating the motor parameter vectors via re-indexing of the perceptual subspaces.
2.2. Representation of action models

The configuration of the domain of behavioural competences is a structure of elements describing functions of motor parameters $p, C = f(p)$. In defining values of motor parameters we correspondingly instantiate a particular state of the motors. However, the parameters are not the motor states, instead, vector $p$ ranges over the perceptual space. If, for instance, the goal is to move an object then $p$ involves three parameters, namely, the new perceptual position (two coordinates) of the manipulator and the index of the object to be moved.

The starting arrangement of the motor domain is thus a set of elements that directly control different physical motors ($C_0, \ldots, C_n$), called innate motor capabilities. These actions are used to achieve perceptual goals detected in the initial stages of exploration and as a basis to learn corresponding novel actions. This leads to expansion of the motor domain by a set of elements ($C_{no+1}, \ldots, C_{n1}$), where $K_1$ is the number of perceptual goals detected at the current stage. In a typical learning scenario, the system is expected to produce a hierarchy of actions with novel action capabilities subsuming previous action capabilities on lower levels of the hierarchy: (see Fig. 1).

\[
\{C_0, \ldots, C_n\}
\]
\[
\{C_{no+1}, \ldots, C_{n1}\}
\]
\[
\ldots
\]
\[
\{C_{x_n,1+1}, \ldots, C_{x_n}\}
\]

Each of these actions at the various hierarchical levels will have associated with it a corresponding perceptual space, giving rise to a hierarchical parameterisation of the percept domain. The key to achieving this percept–action equivalence at each stage of the hierarchy is hence the definition of new action capabilities in terms of the currently highest level of perceptual hierarchy (rather than the intrinsic motor space), followed by the reparameterisation of the perceived environment in terms of this newly learned action competence, in a potentially unending iterative bootstrap cycle.

2.3. Detecting perceptual goals

The proposed algorithm of goal capturing is based on identification of dominant modes in Feature Frequency Histograms (FFHs) compiled over the initial perceptual space within a supervised learning context. FFHs are calculated by integrating features perceived by the system over a sequence of exploratory movements. Suppose that the perceptual scene can be represented as a set of features $f_i$ then, for any $f_i$, its FFH $I(f_i)$ is:

\[
I(f_i) = \sum_{j \neq i} \left\{ \frac{1}{f_j = f_i} \right\} j = 1, \ldots, n^0, \quad i = 1, \ldots, N
\]

where $n^0$ is the number of features detected on ith step, $N$ is the number of iterations.

A perceptual feature with the corresponding FFH having a peak exceeding the feature saliency threshold is called a goal feature and the argument of the peak is a goal feature value. Perceptual goals can contain one or more goal features. The number of features associated with the perceptual goal can be defined by the following principle: if the collection of goal features has FFH peaks growing synchronously through the percept–action interaction then all the elements of the collection belong to the same goal. Those elements are also known as goal parameters. Arguments of the perceptual features detected on the current scene excluding the goal parameters constitute the perceptual context of the current scene. To diversify the range of possible perceptual goals we consider FFHs not only for absolute values of perceptual features but also for feature differences calculated between the current and previous scenes. For instance, let us examine a learning scenario where a system learns to play a shape-sorter. That is, to accomplish the task the agent should insert ‘wooden’ pieces (objects) scattered upon the scene into holes made in a board. An object can be inserted into a hole only if their shapes match.

Observing a user playing the game, the system that has an access to relative changes of the perceptual features is able to detect goals that correspond to aligning gripper with an object or inserting objects into holes. This occurs because, while completing the task, the user several times aligns his/her arm with the objects (in order to perform grasping) and inserts the objects into the corresponding holes. Therefore, after a series of observation experiments the histogram modes associated with relative object–arm and object–hole positions become dominant and are treated as perceptual goals. Thus, learning the motor functionalities of aligning the arm with an object (followed by grasping) and filling holes required to accomplish this example game have been made possible only by analysing histograms for feature variations.

To summarise this subsection we make a formal definition of the perceptual goal: a perceptual goal is a vector (single- or multi-component) of perceptual features for which the corresponding FFH modes exceed the threshold of saliency.

\[
g = (f^1, \ldots, f^v), \quad I(f^i) > \tau, \quad \forall i = 1, \ldots, v
\]

Those components of the feature vector that do not participate in any goal we refer to as contextual parameters. They represent the current configuration of the scene and change their values from one scene to another. The goal parameters in contrast demonstrate quasi-stability, i.e. their values remain constant for a number of scene configurations.

Once the system has detected a perceptual goal it is compelled to reattain this goal state by means of its existing motor capabilities and thus, potentially, to generalise the motor solution in a form of a novel action model. As we indicated in Section 2.1, the first stage of the process is generating random action sequences:

2.4. Instantiating action sequences

The system generates random action sequences by arbitrary ordering of the existing action models with randomly instantiated motor parameters. The purpose of this procedure is to find any (not necessarily optimal) motor solution of the given perceptual goal. The algorithm is based on variation of the goal distance function.

Suppose that the perceptual goal is the vector $g = (f^1, \ldots, f^v)$ and the current scene contains both the goal parameters and the contextual parameters $f = (f^1, \ldots, f^n)$. Thereby, distance $D(g, f)$ between the goal $g$ and the current scene $f$ is the following measure:

\[
D(g, f) = \sqrt{\sum_{i=1}^v (f^i - f^i)^2}
\]

The introduced goal distance has two principal properties:

- $D$ is a Euclidean metric measure. Such a property can be validated by the fact that gaining percept–action skills is aimed towards learning spatial knowledge [20], hence the corresponding internal representations should be developed in terms of metric descriptions.
- The goal-specific features used to evaluate $D$ makes this measure (and the whole process of learning novel action capabilities) invariant to the perceptual context. However, the goal distance does depend on the stage of learning.
Let us denote $f^{(0)}$ as a perceptual state at ithe sequence instantiation. If after executing the corresponding motor sequence

$$C = C_{i1}(p_{i1})C_{i2}(p_{i2})\ldots C_{ir}(p_{ir})$$

(4)

where $r_i$ is the random index of action order, the goal distance $D = D(g, f^{(0)})$ remains unchanged, then another random motor sequence is generated. Otherwise, if $D \neq D_{i1}$, then the tested sequence is considered to be relevant to the goal.

Further processing is carried out to determine the values of the motor parameters that make the goal distance equal to zero. The process sequentially takes the actions that form C and performs optimisation of the corresponding parameters by gradient descent. To utilise the stochastic gradient descent procedure[29] in order optimisation of the corresponding parameters by gradient descent.

First of all we wish to clarify the mechanism that performs per-
cept–action parameter linking. As we mentioned earlier the components of motor parameter vector $p$ are the variables with perceptual origins. However, they are not exactly the goal parameters. We define motor parameters as characteristics that index all possible states of the goal parameter vector $g$. For instance, the motor parameter associated with the horizontal position $x_{\text{motor}}$ of an object on an empty scene ranges over the horizontal size of the scene, say, between 0 and $h$. Next, suppose, we placed a permanent obstacle on the scene such that the object cannot be located on top of it. The number of possible states for a potential goal parameter has hence been reduced, although, the perceptual feature $x_{\text{perception}}$ itself still belongs to $[0,h]$, namely, it can be used to determine the position of the introduced obstacle. Thus, $x_{\text{perception}}$ reflects restrictions of the current physical configuration in the environment, however, the abilities to perceive $x_{\text{perception}}$ have not been affected. The initial assumption concerning the motor parameters is that they can take any value (in our experiments any integer number), however, this hypothesis is to be falsified in the course of exploration of the environment.

Within certain configurations of learning tasks it is possible that for a series of random action instantiations one or more motor parameters can remain constant. Such parameters are invariant to the perceptual context and hence do not influence the result of executing the action sequence. We term them as action constants. The following example illustrates the emergence of action constants.

Suppose that the agent’s environment is a 2D scene with a number of different objects. The motor domain contains a set of already known motor capabilities:

- Moving the manipulator from the current position to $(x, y)$; denoted as $M(x,y)$.
- Aligning the manipulator with object $n$: $O(n)$.
- Grasping movement: $G(s)$, where $s = 0.1$ is the state of the gripper.

The learning task is to acquire a capability that moves object $n$ to position $(x, y)$. A successful instantiation achieving the goal is the sequence $O(n'), G(s'), M(x', y'), G(s')$

(5)

This sequence is determined by values of five parameters $(n', s', x', y', s')$, however, for the given goal only three of them, namely $(n, x, y)$, can change values, for example to $(n', x', y')$, within a certain range such that the obtained sequence still achieves the goal. Variation of parameters $(n, x, y)$ will lead to moving different objects on the scene to different positions. The parameters $s'$ and $s$, however, should be constant: $s_1 = 1 – gripping the object after aligning, s_2 = 0 – releasing the object after moving. Assigning any other values to $s_1, s_2$ will not achieve the goal.

The next step of processing, learning an optimal motor solution, proceeds through removing redundant chains in the initial action sequence. The system attempts to attain the goal sequentially taking away the actions in the sequence that do not influence $D$. At each step, if the attempt succeeds (i.e. the goal has been achieved), the action is completely erased from the sequence. Otherwise, the considered action remains exactly at its previous position within the sequence since, not only the correct values of motor parameters, but also the chosen order of the actions is necessary for attaining the goal.

Thus, in order to produce a novel action capability in the form of a function $C = f(p)$ we assign a new motor parameter vector to the action sequence that excludes action constants. $p_{\text{new}} = (n, x, y)$. This sequence also has the redundant chains removed. In general, if the sequence $\hat{C} = \hat{C}_0(p_{q0}), \hat{C}_q(p_{q1}), \ldots, \hat{C}_q(p_{qm})$

(6)

is optimal for the given goal then the new motor parameter vector is given as:

$$p_{\text{new}} = (p_1, p_2, \ldots, p_k)$$

(7)

where $p_i = (p_{q0}, p_{q1}, \ldots, p_{qm})$ are non-constant motor parameters.

In this way a new capability is added to the body of total motor capabilities parameterised in the manner indicated. Hence, in the example given, we add the new capability of moving objects (named $MO$, say, by a sequential token generator) along with its corresponding motor parameter vector to the total set of available motor capabilities:

$$\{MO(n, x, y), O(n), G(s), M(x, y)\}$$

Thus, although there is a specifically hierarchical relationship between $MO(n, x, y)$ and $(O(n), G(s), M(x, y))$, in future iterations of the bootstrapping algorithm, we are free to choose motor capabilities at any level of the hierarchy. The process is consequently completely open-ended.

Another important aspect of generalising is that the algorithm does not simply determine restrictions for the values of motor parameters, it also introduces novel variables that index the observed states in the space of perceptual goals. Let us consider another example where the system learns to insert objects into holes while playing the shape-sorter game. The successful instantiation of action capabilities is one of the following actions

$$MO(n_1, x_1, y_1), MO(n_2, x_2, y_2), \ldots, MO(n_K, x_K, y_K)$$

where $K$ is the number of the objects (holes) in the game.

The corresponding parameter vector of the action model contains three components, all of them non-constant. However, if we calculate the number of possible perceptual states that correspond to the solved task (an object has been inserted into the corresponding hole) we obtain only $K$ cases. Therefore, we can introduce an index $k = 1, \ldots, K$ to replace vector $p = (n, x, y)$ as a new motor...
parameter. The novel action model \( H(p) \) that solves the task of filling a hole by an object is hence parameterised by a single scalar number \( k \): \( H(p) = H(k) \).

### 3. Theoretical analysis of learning characteristics for a specific learning scenario

#### 3.1. The learning scenario

In this section we carry-out a theoretical evaluation of system behaviour in a predefined environment. We analyse different stages of the learning process in order to estimate a performance characteristic that represents convergence of the learning tasks, namely, the number of percept–action cycles necessary for accomplishing the behavioural goal in a generic fashion.

The scenario that has been chosen to implement our system is a variant of the shape-sorter game. The goal of the game is to insert all objects into holes of corresponding shape. Our system is intended to autonomously learn a general solution of the game commencing from only the primitive behavioural skills that involve \( \text{Moving the manipulator } M(x, y) \), \( \text{Rotating the gripper } R(\theta) \) and \( \text{Performing gripping movement } G(s) \).

\[
(C)_{\text{prim}} = \{ M(x, y), R(\theta), G(s) \}
\]  

(8)

The algorithm should gain percept–motor capabilities sufficient to accomplish the shape-sorter solution from any initial layout of the objects and holes and also independently of the irrelevant objects in the scene. The choice of the proposed scenario is motivated by the ability of variable environmental configurations to explicitly demonstrate the processes of gaining novel cognitive capabilities and representing them in an abstract hierarchical fashion.

In the following discussion we provide the reader with detailed description of four distinctive learning stages: \( \text{aligning the manipulator with an object} \), \( \text{moving an object, filling holes and solving the game} \). It is important to clarify that all the presented results can be obtained automatically (demonstrated explicitly later). The reason we have manually chosen the stages of learning here is to enable characterisation of the theoretical and experimental efficiency of bootstrapping.

#### 3.2. The different stages of the learning scenario

##### 3.2.1. Aligning the gripper with objects

At this first stage the perceptual goal is represented by the conditions wherein the manipulator position is equal to location \((x_n, y_n)\) of one of the objects in the scene. Since the manipulator is also perceived as an object (we denote its position as \((x_M, y_M)\)) the goal is:

\[
\begin{align*}
X_M &= x_n, \\
Y_M &= y_n
\end{align*}
\]  

(9)

where \( N \) is the number of the objects in the scene excluding the manipulator.

By finding the relevant action sequence, performing percept–action parameter linking and removing redundant chains we generalise the corresponding new action model, see Section 2.5

\[
O(p) = M(x_n, y_n)
\]  

(10)

where \( O(p) \) is the new action model.

To transfer the goal features \((x_n, y_n)\) into the components of parameter vector \( p \) of model \( O(p) \) we analyse the possible states that can be assigned to the goal. According to (9) the number of possible manipulator positions that achieve the goal is \( N \), therefore, the goal states can be indexed by scalar number \( n = 1, \ldots, N \). The parameter vector \( p \) therefore has a single component that refers to the object with which the manipulator is aligned:

\[
p = n \quad (11)
\]

\[
O(p) = O(n) \quad (12)
\]

##### 3.2.2. Moving objects

A perceptual goal that underlies the motor capability to move objects can be expressed by following conditions

\[
\begin{align*}
\sqrt{\alpha_{x_n}^2 + \alpha_{y_n}^2} & \neq 0, \quad n = 1, \ldots, N \\
\text{fgrip} &= 0
\end{align*}
\]  

(13)

where \( \text{fgrip} \) is the perceptual feature that characterises the state of the actuator: \( \text{fgrip} = 0 – \text{gripper is off}, \text{fgrip} = 1 – \text{gripper is on} \). Eq. (13) indicates that the goal is achieved if the position of the object has been changed and the gripper has been released.

In contrast to the previous stage of learning there is not just a single possibility for generating an action sequence capable of converging on the goal. One way is to use only the ‘innate’ capabilities of manipulator control. In this case the desired sequence is

\[
C(p) = M(x_n, y_n), G(1), M(x', y'), G(0)
\]  

(14)

where \((x', y')\) is the new position of object \( n \).

Another choice is to utilise the previously gained motor capability to align the manipulator and the object \( O(n) \) (which is similar to the example discussed in Section 2.5):

\[
C(p) = O(n), G(1), M(x', y'), G(0)
\]  

(16)

To decide which of those two possibilities should be chosen we wish to estimate how rapidly in each case a sequence that achieves the goal can be instantiated via random exploration. In other words we would like to calculate the relative probability of achieving solutions (14) and (16). In the first case, the action domain involves only the primitive actions, hence

\[
P_{\text{prim}} = P_{\text{seq}} \cdot P_M \cdot P_C \cdot P_{M'} \cdot P_C'
\]  

(17)

where \( P_{\text{prim}} \) is the probability of retrieving solution (14); \( P_{\text{seq}} \) is the probability of the correct ordering of the sequence; \( P_M \) and \( P_{M'} \) are probabilities of moving the manipulator to positions \((x_n, y_n)\) and \((x', y')\), respectively; \( P_C \) and \( P_C' \) are probabilities of performing gripping-on and gripping-off actions.

Sequence (14) has four independent action elements. The number of available inbuilt motor capabilities [see Eq. (8)] is \( K = 3 \); hence

\[
P_{\text{seq}} = \left( \frac{1}{K} \right)^4 = \frac{1}{81}
\]  

(18)

Thus,

\[
P_{\text{prim}} = \frac{1}{81} \cdot \frac{1}{(X - N)(Y - N)} \cdot \frac{1}{2} \cdot \frac{(X - 1)(Y - 1)}{XY} \cdot \frac{1}{2}
\]  

(19)

where \( X, Y \) are the scene height and width.\(^4\) The denominator \((X - N)(Y - N)\) in (19) indicates that moving any of \( N \) objects satisfies the goal conditions; multipliers \( P_C \) and \( P_C' \) are equal to \( \frac{1}{2} \) because the gripping state takes only two values \( s = 0, 1 \).

Returning to the second case in which the algorithm exploits the previously learned action model \( O(n) \), the number of available

\(^4\) Eq. (19) and further considerations in this section imply pixel-wise precision in determining object positions. Assuming that in the real world the error of perception is more significant, however, will not affect the comparative results of the bootstrapping method.
actions is $K = 4$, and the corresponding probability of successful instantiation $P_{\text{bootstrap}}$ is:

$$
P_{\text{bootstrap}} = P_{\text{eq}} \cdot P_{\text{O}} \cdot P_{\text{C}} \cdot P_{\text{M}} \cdot P_{\text{G}}$$

(20)

where

$$
P_{\text{eq}} = \left(\frac{1}{K}\right)^4 = \frac{1}{256}$$

(21)

hence,

$$
P_{\text{bootstrap}} = \frac{1}{256} \frac{1}{N} \frac{1}{2} \frac{(X-1)(Y-1)}{XY} \frac{1}{N}$$

(22)

Comparing both results we have

$$
P_{\text{bootstrap}} \approx \frac{81(X - N)(Y - N)}{256N}$$

(23)

Assuming that the number of objects in the scene is much less than the number of possible object positions ($N \ll X \cdot Y$) we obtain

$$
P_{\text{bootstrap}} \approx \frac{81XY}{256N} \gg 1$$

(24)

In our experimental setup (see Section 4) we used $300 \times 300$ pixel visual scenes and, typically, five different objects. Thus, in such environmental configuration, the theoretically estimated computational profit of the hierarchical bootstrapping as compared with the percept–action learning within the dimension of primitive actions is

$$
P_{\text{bootstrap}} \approx 5695$$

(25)

According to the strategy discussed above we designate the novel capability to move objects as a symbolic token $MO(p)$ and add this element into the hierarchy of actions. In order to create the new parameter vector we take into account that the parameters of the gripping actions in sequence (16) always remain constant: $G(s) = G(1), G(1) = G(0)$. Having that, we find that vector $p$ contains only three components: index $n$ of the object to be moved and its new position $(x, y)$

$$
p = (n, x, y)$$

(26)

$$
MO(p) = MO(n, x, y)$$

(27)

3.2.3. Inserting objects into holes

Next, we consider a scene that, along with $N/2$ randomly scattered objects, contains a board with $N/2$ holes. Each hole has the same shape as one of the presented objects, i.e. all the objects can be inserted into the corresponding holes. Holes and objects are represented in the perceptual domain as ‘shapes’ $\mu_i$ with certain positions $(x_i, y_i)$ and orientations $\theta_i$,

$$
f = (x_1, y_1, \theta_1, \ldots, x_N, y_N, \theta_N, \mu_N)$$

(28)

thus, the goal here is that the position and orientation of one shape (an object) should be the same as the position and orientation of the complementary shape (a hole), such that:

$$
\begin{align*}
x_m &= x_n \\
y_m &= y_n \\
\theta_m &= \theta_n \\
H_m &= \mu_n
\end{align*}$$

(29)

The estimated computational benefit of bootstrapping at this stage, calculated in the same way as for Section 3.2.2, gives:

$$
P_{\text{bootstrap}} \approx 2XY \approx 2 \cdot 10^4$$

(30)

The parameter vector for the new capability $H(p)$ responsible for filling holes is as follows. Suppose that $O(n), G(1), M(x_m, y_m), R(\theta_m), G(0)$ is the retrieved action sequence. The motor parameters excluding constants for this sequence is hence

$$
p = (n, x_m, y_m, \theta_m)$$

(31)

however, the allowed number of goal states, according to (29), is $N/2$. Therefore, all the vector components in (32) can be replaced by a scalar, $k, k = 1, \ldots, N/2$, that indexes objects to be inserted into holes:

$$
p = k$$

(33)

$$
H(p) = H(k)$$

(34)

3.2.4. Solving the shape-sorter

The perceptual goal of the final stage of the game differs from the previous one only in the condition that all the objects should be inserted into the holes, i.e. for $\forall \ m, n = 1, \ldots, N$:

$$
\begin{align*}
x_m &= x_n \\
y_m &= y_n \\
\theta_m &= \theta_n \\
\mu_m &= \mu_n
\end{align*}$$

(35)

Obviously, the corresponding action sequence contains the chains of the same actions $H(k)$ with different values of $k$ under any permutation. There is hence only one possibility to attain the goal – namely, insert ALL the objects into the holes. The final action model has no parameters since all of motor vector components become constants

$$
S(p) = H(k_1), \ldots, H(k_{N/2}) \quad p = \emptyset$$

(36)

The estimated computational profit for the whole game is consequently,

$$
P_{\text{bootstrap}} \approx 3.2 \cdot 10^5$$

(37)

4. Experimental results

4.1. Scope of experimental study

Following the preceding theoretical outline of the performance potential of bootstrapping percept–motor capabilities, in this section we consider a series of computational experiments intended to analyse the system behaviour at different stages of bootstrapping and in various configurations of the environment. The experiments were carried out in the virtual environment presented in Section 4.2 that simulates the 2D shape-sorter game. We pursue the following objectives for our computational study:

- Estimate and compare the overall computational requirements for the proposed mechanisms of generalised percept–action learning and hierarchical bootstrapping.
- Perform the convergence test at different stages of the scenario.
- Provide experimental evidence that the acquired hierarchical world model is relevant to the shape-sorter game; demonstrate the contribution of various learning stages to accomplishing the final behavioural goal of the scenario.

4.2. Implementation of the system

The learning scenario in which the agent bootstraps its capabilities to play the shape-sorter has been implemented within a simulated world (Fig. 2). The visual environment contains a 2D surface
(2-1), a manipulator (the zoom tool) (2-2), objects of five different colours and shapes (2-3) and a board with five shaped holes (2-4). Using the interface, the user is able to place the requisite number of objects into the desired positions and insert or remove the board [box (2-5)]. The perceptual goal can be defined manually, using interface window (2-6), or loaded from memory as a visual state of the world [see box (2-7)]. However, there is also a mechanism for automatic goal acquisition via calculating FFHs.

The software consequently operates in three modes:

- 'RUN' (box (2)-8) – a sequence of movements simulates a user solving the shape-sorter. At this stage the system detects perceptual goals.
- 'PLAY' (box (2)-9) – the agent attains the given goal by manipulating the primitive actions using the gradient search algorithm.
- 'PLAY 2' (box (2)-10) – the agent attains the goal by learning novel capabilities for solving the sub-goals detected in the learning scenario.

Box (2-11) demonstrates dynamics of the perceptual goal distance while solving the game in one or another mode. List (2-12) shows the current status of the learning process including the set of known motor capabilities.

The visual scene is perceived as a Cartesian label space (Fig. 3) where each state corresponds to an attractor (an object or a hole) with the following components of visual description:

- Shape \( \mu \)
- Cartesian position \((x,y)\)
- Spatial orientation \(\theta\)
- Position increment after previous percept–action iteration \((dx,dy)\)
- Orientation increment after previous percept–action iteration \(d\theta\).

Thus, the perceptual feature vector \(\mathbf{f}\) has the components

\[
\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_N)
\]

where each \(\mathbf{f}_i\) corresponds to a separate attractor

\[
\mathbf{f}_i = (\mu_i, x_i, y_i, \theta_i, dx_i, dy_i, d\theta_i)
\]

\((i = 1, \ldots, N, N\) is the number of attractors).

The goal distance is a normalised sum of distances among attractors that belong to the current \(\mathbf{f}\) and goal \(\mathbf{g}\) states. Suppose, the goal state is a vector of \(\Gamma\) attractors

\[
\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_\Gamma)
\]

where \(\mathbf{g}_k, k = 1, \ldots, \Gamma,\) have representation (39). If we introduce an auxiliary measure, attractor distance \(d(\mathbf{f}_i, \mathbf{g}_k)\), or \(d(i,k)\) for convenience, then goal distance \(D\) becomes the following value

\[
D(\mathbf{f}, \mathbf{g}) = \sum_{i=1}^{N} \sum_{k=1}^{\Gamma} d(i,k)/(N \cdot \Gamma)
\]

The attractor distance is calculated as a weighted sum of distances between corresponding components of attractor features

\[
d(i,k) = \frac{1}{3} (d'(i,k) + d''(i,k) + d'''(i,k))
\]

Thus, the perceptual feature vector \(\mathbf{f}\) has the components

\[
\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_N)
\]

where each \(\mathbf{f}_i\) corresponds to a separate attractor

\[
\mathbf{f}_i = (\mu_i, x_i, y_i, \theta_i, dx_i, dy_i, d\theta_i)
\]

\((i = 1, \ldots, N, N\) is the number of attractors).

The goal distance is a normalised sum of distances among attractors that belong to the current \(\mathbf{f}\) and goal \(\mathbf{g}\) states. Suppose, the goal state is a vector of \(\Gamma\) attractors

\[
\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_\Gamma)
\]

where \(\mathbf{g}_k, k = 1, \ldots, \Gamma,\) have representation (39). If we introduce an auxiliary measure, attractor distance \(d(\mathbf{f}_i, \mathbf{g}_k)\), or \(d(i,k)\) for convenience, then goal distance \(D\) becomes the following value

\[
D(\mathbf{f}, \mathbf{g}) = \sum_{i=1}^{N} \sum_{k=1}^{\Gamma} d(i,k)/(N \cdot \Gamma)
\]

The attractor distance is calculated as a weighted sum of distances between corresponding components of attractor features

\[
d(i,k) = \frac{1}{3} (d'(i,k) + d''(i,k) + d'''(i,k))
\]
where \( d(i,k) \) is the binary shape difference measure:
\[
\delta M = \begin{cases} 
0 & \mu_i = \mu_k \\
1 & \mu_i \neq \mu_k
\end{cases}
\]
\( d'(i,k) \) is the angular difference of attractor orientations and orientation increments,
\[
d'(i,k) = \frac{\sqrt{(\theta_i - \theta_k)^2 + (d\theta_i - d\theta_k)^2}}{2\sqrt{2\pi}}
\]
and \( d''(i,k) \) is the Cartesian distance between attractor position and position increments:
\[
d''(i,k) = \frac{\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + (dx_i - dx_k)^2 + (dy_i - dy_k)^2}}{2\sqrt{x^2 + y^2}}
\]

4.3. Configurations of the environment

We seek to test system behaviour in a variety of regimes in order to demonstrate the transferability of acquired capabilities into other environmental settings. The experiments were carried out for three distinctive configurations of the simulator.

1. In the first experimental setup the environment contains only the objects and events that are necessary for solving the shape-sorter game, what we call a simple environment. Moreover, the user can guide the process of detecting goals by explicit demonstration of the desired behavioural functionalities. The manipulations are performed in such a way as to introduce goals incrementally with the following increasing scale of complexity: controlled manipulator movements → aligning manipulator with an object → moving an object → filling holes → solving the game.

2. The environment is setup as above; however, learning individual capabilities is not guided by the user, instead, it relies completely on agent-world interaction. Although the final goal – a completed shape-sorter game – has been demonstrated, the system proceeds through detecting and learning the required hierarchy of percept–action capabilities autonomously.

3. Along with the pieces and holes used to play the shape-sorter game, the environment contains other objects and events that do not influence or can even interrupt the solution of the final task. This is the case of a complex distractor-strewn environment typical of real-world scenarios. There is also no user-guided detection of behavioural goals in such environments.

The outlined experimental configurations have been proposed with the purpose of investigating the difference of system performance in the bootstrapping mode and ordinary percept–action learning (characterised by the unimodally-restricted ‘dimensionality’ of behavioural inferences). Furthermore, the indicated scenarios are analogous to children’s cognitive learning at the early stages of development.

In the initial stage the learning is intended to gain simple capabilities via explicit careful demonstrations. A teacher configures the scenario, first, to train primitive skills and then gradually increases the complexity of the goals and behaviours. Once the initial capabilities become available to the agent, the next stage is to address tasks that do not have a direct solution by means of the existing functionalities (or for which the solution is far from optimal). However, this framing environment does not contain elements that distract or interrupt the process of acquiring the solution. At this level the cognitive system initialises a ‘bootstrap’ search such that instead of exploiting the existing capabilities to solve the problem the agent explores the world in order to learn novel skills and thus more quickly accomplishes the final goal. Capturing behavioural goals and learning the corresponding action models at this stage is carried out fully autonomously. Finally, the third scenario, perhaps set in the real world, implies the presence of distractors on the scene that can lead to gaining capabilities irrelevant to the goal of the game. The cognitive agent should be able to detect the redundant behaviours and to correct its strategy of interaction with the world.

4.4. Overall consideration of computational requirements

We firstly give a direct comparison of the two methods of learning: stochastic instantiation of parameters within the set of innate action models without reparameterisation of the percept–action domains, and learning via generalisation of the parameter space and creating a hierarchical structure of capabilities.\(^5\)

Fig. 4 shows the system performance for the first learning strategy. The horizontal axis here is the initial perceptual distance to the goal; the vertical axis is the number of percept–action cycles needed to attain the current goal. Different colours mark the stages of the learning scenario, with the colour map as follows: green corresponds to the task of aligning the manipulator with an object, red is moving an object, blue is inserting an object into a hole, cyan is the final stage of solving the shape-sorter. The space of initial motor capabilities contained three actions: moving the manipulator, rotating the actuator and performing gripping; at each stage the system carried out 200 randomly-initiated runs in attempting to attain the corresponding goals.

There are two main outcomes of these experiments. Firstly, the number of percept–action cycles does not depend on the initial distance to the goal. That reflects the nature of the process that instantiates goal-relevant action sequences as being invariant to the perceptual context. The second result is that the computational requirements grow significantly with the increase of goal complexity.

In contrast, the results obtained under the same experimental conditions for the bootstrapping mode (Fig. 5) demonstrate a very compact distribution of percept–action cycles. Overlapping clusters of measurements for different goals indicate that the complexity of the learning tasks remains within the same range at any stage of the scenario.

More explicit comparison of the methods for each of the learning tasks is presented in Fig. 6. It plots the average numbers of percept–action cycles at each stage of the game; the cyan and green curves are the conventional percept–action learning and bootstrapping modes, respectively. Here we observe a quasi-exponential increase in the computational requirements for the first approach, whereas the hierarchical bootstrapping demonstrates near-constancy in the task complexity. Overcoming this problem of exponentially-increasing complexity thus makes it possible for the hierarchical bootstrapping approach to begin to satisfy the requirements of an open-ended architecture for intelligent agency.

4.5. The convergence test

In this subsection we investigate the system performance when conditioned by the open-ended hierarchical bootstrapping. We will analyse the computational requirements of the method as a function of the various learning stages. This benchmark is characterised as the required number of percept–action cycles to complete the puzzle averaged over a series of runs with respect to different configurations.\(^5\)

\(^5\) Both methods are applied initially to the simple guided environment for the purpose of direct comparison at each of the hierarchical stages. System performance in the fully autonomous experimental configurations is addressed in Section 4.5.
starting configurations. The functions for each of three environmental setups introduced in Section 4.3 will be considered separately.

**4.5.1. The simple guided environment**

The results of the convergence test within the simple guided environment are shown in Fig. 7 (Series (1)). The graph exhibits an approximately linear descent of the average percept–action iterations $n$ on a logarithmic scale. Such an evolution of $n$ suggests that acquiring each new capability exponentially simplifies the convergence on the final goal of the game. The learning environment has been intentionally setup so as to permit the increment of learning complexity to occur smoothly and gradually, progressively providing the system with sufficient behavioural functionalities for solving each subsequent goal. In the same manner as for the overall analysis of the computational requirements (Section 4.4), the variation of the results due to randomly chosen starting layouts is not significant at any stage of learning.

The exponential decrease of $n$ indicates that each subsequent level of the hierarchy does indeed improve the ability to accomplish the final task, i.e. all the detected behavioural goals are relevant to solution of the shape-sorter. Thus, the given graph can be considered as a benchmark to compare with the other fully autonomous scenarios of learning.

**4.5.2. The simple autonomous environment**

It is important to appreciate that in the following configuration the system has been given the overall goal (via demonstrations) of solving the shape-sorter, however, the process of learning the necessary capabilities proceeds autonomously through exploration of the environment.

As has been shown earlier, a system that possesses only primitive motor capabilities requires a very large number of iterations in order to solve the shape-sorter. Moreover, the obtained solution cannot be generic. Hence, in further experiments it would be
necessary to incur significant computational costs to achieve the same goal once again. Our computational experiments demonstrate that the mechanism of bootstrapping is capable of overcoming such issues, producing performance similar to that of the benchmark considered above.

**Fig. 7** (Series 2) presents the system characteristics within a simple autonomous environment. It can be seen that all the stages intrinsic to the benchmark behaviour have been discovered independently in the course of autonomous exploration. The shapes of curves (1) and (2) in Fig. 7 remain almost the same; for each subsequent level of the hierarchy the solution of the scenario becomes more optimal. Analogous results, namely, that the solution becomes less sensitive to the starting layout, can be inferred from the standard deviation.

A slight modification of the system behaviour towards increased learning requirements at the final stage is observable. However, notice that the considered configurations of the environment represent a significantly more generic case than the user-guided scenario. The system is originally placed in the complex environment and proceeds through the autonomous discovering of complex behaviours, whereas for the simple scenario the complexity grows gradually under the control of the external supervisor. This demonstrates the efficiency of the method provided by the powerful generalising abilities of hierarchical bootstrapping.

**4.5.3. The complex autonomous environment**

Finally, we analyse the system performance in the autonomous regime in the case when the environment contains phenomena (objects or events) that do not influence (are neutral with regard to), or even interfere with, the solution of the shape-sorter game. In the course of interaction with the environment the system captures the redundant phenomena in the form of behavioural goals and ‘digresses’ to learning the corresponding motor functionalities that, however, fail to assist in further stages of inference.

The software simulating the virtual environment involves two distracting phenomena:

- Lighting up an indicator when an object is placed on a labelled position in the scene (Fig. 8). The position is represented by a red circle that enables its perceptual detection. This is a neutral event.
- A new object is dropped in the scene if an existing object has been placed onto a particular labelled position. The new object has the same shape and the colour as the object moved onto the indicated position. Thus, the given event complicates the further solution of the game by bringing an additional object in to play (hence, it is an interfering event).

A series of the experiments under the indicated configurations of the environment has demonstrated the following system characteristics: (Fig. 7 (Series 3)). On the graph we can see two regions of non-monotonicity for function \( n \). They correspond to the distractors introduced to the environment. The part referring to the capability of lighting up the indicator, \((B–C)\), is constant, showing that the given functionality does not improve the solution of the shape-sorter scenario. The second region of the graph, in which the system becomes capable of dropping new objects into the scene, \((D–E)\), slightly impedes the system from achieving the final goal. A system that has learned to bring new objects into the game exploits this capability to acquire solutions for other sub-goals, and thus, it only complicates the problem of searching for desired behaviours.

Nevertheless, the overall performance in the given configuration does not indicate system failure while solving the scenario. The method does indeed require more computations for searching relevant solutions at each stage of learning. However, even in the worst case of dropping new objects into the scene, the function \( n \) increases insignificantly in comparison to the previous phase. The general systemic tendency to achieve the goal of the game is essentially as for the benchmark scenario.

**4.6. The motor parameter space**

Another alternative for comparing the computational requirements for the ordinary percept–action learning and bootstrapping mechanisms is to estimate the number of motor parameters involved in the search of the goal-relevant motor solutions.

It has been shown that the bootstrapping mechanism generalises percept–action capabilities via the update of the motor parameter vector, causing a reduction of possible motor states within each action model. **Fig. 9** demonstrates the evolution of the motor parameter dimensionality, \( n_p \), for the ordinary percept–action and bootstrapping learning approaches within the complex autonomous environment configuration. In the first case (Fig. 9 (Series 1)) function \( n_p \) increases significantly with the complexity of the behavioural capabilities (a particularly distinctive peak of the parameter dimensionality occurs with the final stage of the game, where the same action of filling a hole has to be repeated several times for various values of the motor parameters). The bootstrap-
ping approach, in contrast, exhibits only local variations of $n_2$ that finally tend toward zero (Fig. 9 (Series 2)). These results hence agree with our theoretical prediction when all the motor parameters in the action sequence become constant (such that the corresponding capability to solve the shape-sorter takes no parameters, see Eq. (36)).

The restricted dimensionality of motor parameter space intrinsic to the hierarchical bootstrapping provides us with a robust solution procedure even under the circumstances of distracting events and complex environmental configurations.

5. Discussion and conclusions

We have, in this paper, presented an approach to autonomous learning that involves bootstrapping cognitive capabilities in a hierarchically open-ended way. The learning process is carried out as a continuous perception–action cycle. Cognitive capabilities are represented by pairwise links between acquired behavioural skills and corresponding perceptual concepts. At each level of learning the system performs reparameterisation of the perception and action domains and hence modifies its abilities to perceive and manipulate the external world. Based on generic principles of random exploration the system seeks motor solutions for detected perceptual goals. The goal, as a state of the perceptual space, can be specified by demonstration or else be detected automatically using statistics gathered over the agent's exploratory percept–action experiences. Generalising of individual percept–action capabilities is performed through a series of attempts to solve the goal under different physical configurations of the environment. The mechanism of parameter updating estimates the number of possible states for the current goal and assigns variables that constitute the new motor parameter space. As our experiments have demonstrated, the procedure leads to a significant reduction of parameter dimensionality (for the shape-sorter learning scenario, the final acquired capability of solving the game contains no independent parameter vectors, indicating that the corresponding behavioural capability can solve the game for any initial layout and content of the shape-sorter scene).

We believe that this progressive, iterative parametric reduction has significant implications for the resolution of the symbol grounding problem in so far as it enables the spontaneous bottom-up construction of hierarchies of progressively abstracted percept–motor relations which, at the highest levels of the hierarchy, come to resemble abstract symbol-manipulation systems. These abstractly-manipulated symbols are thus grounded through their subsumptive hierarchical relation to the lowest sensorimotor level (in effect, abstract symbols acquire relevant environmental context as they are transmitted down the hierarchy). At no stage other than the highest level of the hierarchy is there thus a requirement for a globally self-consistent (rather than para-consistent) representation of the environment. A fuller discussion of the philosophical implications of the approach for theories of representation and epistemology is given in [39].

The main practical distinction between the bootstrapping method and other percept–action learners is hence how they address the problem of learning novel skills. In order to achieve desired goals conventional approaches directly exploit available capabilities. For instance, within the shape-sorter scenario, such a system would attempt to solve the puzzle by means of the primitive actions that, as has been shown, require extensive computation. The bootstrap solution, in contrast, proceeds via an implicit nesting of sub-goals that acquire the structure of auxiliary competences for achieving the goal in a much more efficient manner. Thus, the system incurs a certain initial computational cost for exploring novel behavioural possibilities; however, the return of such a strategy is the significant reduction of the search space via the compact representation of percept–action parameter linkages.

Although the notion of an exploration–exploitation trade-off has traditionally been applied in the context of Reinforcement Learning algorithms (e.g. [1,9,19]), this terminology is also appropriate to explain key aspects of bootstrapping. We hence refer to conventional percept–action methods as techniques employing a Horizontal Exploration–Exploitation Trade-off, since such learning proceeds along uniform, one-dimensional path of optimisation for achieving goals, re-using existing capabilities at every stage of the explore-or-exploit decision-sequence. The bootstrapping approach, however, reveals Vertical aspects of the exploration/exploitation trade-off, being compelled to consider the exploration of other, as yet unknown, action possibilities within the space of possible behaviours in order to find the optimal path to the goal state. Hence, we regard bootstrapping as a method embodying a Two-Dimensional exploration–exploitation trade-off.

The bootstrapping solution to modelling artificial cognition raises another important question to be more fully addressed in future research, namely the extent of the intrinsic ability to exploit capabilities acquired within a certain environmental configuration in other, absolutely different, learning scenarios. The problem is known as knowledge transferability and is related to the system's approach to grounding its internal representations in the physical attributes of external world entities. For instance, the competence of moving objects that has been learned for the shape-sorter scenario can be employed not only to manipulate objects within the same but also, in principle, to move chess figures or to collect parts for assembling mechanical units. In some cases the agent might discover that the existing models do not exhibit the optimal performance and, hence, need to be refined. However, the process of improving the available capabilities that we have here described should naturally enable convergence on the novel goal much more rapidly than a system that attempts to tackle similar tasks using only primitive behavioural skills.

Acknowledgments

The work presented here was supported by the European Union, Grant COSPAL (IST-2003-004176), and also from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 215078. However, this paper does not necessarily represent the opinion of the European Community, and the European Community is not responsible for any use which may be made of its contents.

References

