Constructivist Learning: A Neural Implementation of the Schema Mechanism

Harold H. Chaput, Benjamin Kuipers and Risto Miikkulainen
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233 USA
{chaput,kuipers,risto}@cs.utexas.edu

Keywords: hierarchical maps, infant development, constructivist learning

In *Proceedings of WSOM '03, The Workshop of Self-Organizing Maps*, Kitakyushu, Japan.

Abstract--- Constructivist learning is a hierarchical part-to-whole learning system used by humans and desirable for use by robots. Current implementations are too resource-intensive to be used for anything but simple environments. In this paper, we reimplement one such system, the Schema Mechanism, using a hierarchy of self-organizing maps. The result is an efficient system for learning perceptual and action schemas that can be used in real-world applications like robotics.

1 Introduction

Constructivist learning describes a hierarchical system that can assemble pieces small-scale, low-level knowledge into higher level, more powerful abstractions. Humans use constructivist learning starting at infancy and continuing throughout adulthood [10]. In artificial intelligence, constructivism promises great potential to extract meaningful knowledge from the simplest initial representations. However, it is a challenge to implement; few have written full-scale models of constructivist learning, and fewer still have succeeded in learning anything appreciably complex.

Perhaps the most promising and best-known implementation of constructivist learning is Drescher's Schema Mechanism [8]. Schemas are learned predictions about the world that, given an initial state, if a particular action is taken, a certain change in the environment will occur. Using the Schema Mechanism, Dresher was able have an agent learn how to identify and manipulate objects in a simulated environment. While successful, the Schema Mechanism's original implementation is unwieldy and resource intensive, limiting its application to only simple environments, and impeding extensive research.

CLASM is a reimplementation of the Schema Mechanism using the Constructivist Learning Architecture, or CLA [1]. CLA is a self-organizing, unsupervised learning system that uses hierarchical Self-Organizing Maps (SOMs), to model infant cognitive development. CLA has successfully replicated infant learning in several domains, such as the acquisition of causal perception [2, 7].

In this paper, we show that CLA can be used to implement the Schema Mechanism more efficiently, making it applicable to more realistic domains. We describe how the Schema Mechanism works, and how we implemented it using CLA. To demonstrate that CLASM is a reimplementation of the Schema Mechanism, we apply CLASM to Drescher's simulated agent and environment, and show that CLASM reproduces the results of the Schema Mechanism. Finally, we discuss the potential for future research in constructivist learning that has been opened up by CLA.

2 Schema Mechanism

Drescher's Schema Mechanism is a constructivist learning system for situated agents based on Piaget's [10] theory of infant cognitive development. It starts by representing the world using atomic sensory and motor primitives and building higher-level knowledge from them. Drescher uses the Schema Mechanism to simulate an infant exploring its environment using vision, touch and taste. Today, the Schema Mechanism stands as one of the best – and one of the few – implementations of constructivist learning. In this section, we describe how the Schema Mechanism works, and discuss some of the efficiency issues it has.

2.1 Implementation Details

The Schema Mechanism starts with a set of primitive sensory *items* that reflect the state of the environment, such as `InFrontOfDoor` or `DoorOpen`. Each item can be on or off. There is also a set of primitive *actions* whereby the agent can manipulate its environment, for example `OpenDoor`. As the system explores the environment by randomly selecting actions and monitoring the item states, the system can start to build *schemas*. A schema is a prediction that given an initial set of item states (*context*), if a particular action is taken, then a certain change in item states can be expected (*result*). A schema is written as *context*/*action*/*result*, where *context* is a set of items that are either on or off (indicated by a prefix of + or -, respectively), *action* is the name of the action, and *result* is a set of items indicated

the change in the context state that occurs (again, prefixed with + or -). As an example:

```
+InFrontOfDoor/OpenDoor/+DoorOpen          (1)
```

This schema says that when I am in front of a door, and I try to open the door, the door will be open.
Schemas are created using a technique called *marginal attribution*. When the system first starts, a schema is created for each action that has an empty context and an empty result, called a *bare schema*. Marginal attribution requires that the system maintain information about correlations between all item transitions and all schemas. When an item transition occurs with enough frequency, a new schema is created with that item in the result.
Schemas can be used to create new *synthetic items* as well. For example, schema (1) may not always be reliable. If, say, the door is sometimes locked, then schema (1) will sometimes fail. This fact may be difficult to learn, particularly if the door's locked state cannot be directly sensed. When a schema is unreliable, it becomes "reified" as a synthetic item, which represents the state of the environment in which the schema is successful. Thus, schema (1) can be reified by a synthetic item:

```
[+InFrontOfDoor/OpenDoor/+DoorOpen]        (2)
```

We could call this item "DoorOpenable." It is on when the world is in a state where the schema will succeed.
Finally, schemas also support the creation of *compound actions*. When a new schema is created, and its result is unique, a new compound action is created, with unique result as the "goal." The schemas are then chained backwards from the goal by 1) finding all schemas whose result matches the goal, 2) finding the next set of schemas whose results match the contexts of the first set of schemas, and so on. Thus a compound action has both a set of contexts from which the goal can be reached, and the actions needed to get from any context to the goal. For example, we could create a compound action with the goal DoorOpen:

```
<+DoorOpen>                                (3)
```

When this action is selected, it selects a sequence of schemas that lead to the door being open.
To summarize, the Schema Mechanism starts with a set of primitive items and primitive actions. It then explores the environment to create a set of schemas. These schemas form the basis of new synthetic items. They also are used in the creation of goal-directed compound actions. This process can continue until the agent has learned a sufficient set of items and actions.

2.2 Efficiency Issues

There are three serious efficiency issues regarding the marginal attribution technique. First, when a new schema is created from a source schema, all items must now be correlated with *both* the source schema and the new schema. Second, as new synthetic items are created, they must also be correlated with schemas. Likewise, each new compound action created produces yet another bare schema to serve as another starting point for schema production. Finally, there is no process in the Schema Mechanism to prune items, schemas or actions. Because of marginal attribution, the system will continually grow until its resources have been exhausted. Consequentially, the Schema Mechanism has been limited to simple domains like Drescher's "Microworld," a seven-by-seven grid that holds two objects that the agent can touch, see and taste. In order for the Schema Mechanism to be usable in realistic environments, a more efficient implementation is necessary, such as the Constructivist Learning Architecture, described next.

3 CLA

The Constructivist Learning Architecture (CLA) [1] shares a common ancestry with the Schema Mechanism: both originate from Piaget's constructivism. CLA is an implementation of the *Information Processing* approach to constructivism, proposed by Cohen to account for a number of experiments in cognitive development [3]. CLA has been used to reproduce the results of studies examining the acquisition of causal perception in infants [2]. Because of its shared link with Piagetian constructivism, CLA fits well with the Schema Mechanism, while addressing its efficiency issues.
In this section, we explain CLA's origins in cognitive development, and the details of CLA's implementation.

3.1 Roots in Cognitive Development

The Constructivist Learning Architecture is based on six *Information Processing Principles* of infant cognitive development, first put forth by Cohen [3, 7]:

(1) Perceptual/cognitive development follows a set of domain-general Information Processing Principles.
(2) Information in the environment can be processed at a number of different levels of organization.
(3) Higher (more holistic) levels can be defined in terms of the types of relations among lower (parts) levels.
(4) Development involves progressing to higher and higher levels.
(5) There is a bias to initiate processing at the highest level available.
(6) If an information overload occurs (such as when movement is added or when the task involves forming a category), the optimal strategy is to fall back to

a lower level of processing.

These principles are an attempt defining the mechanism at work in constructivist learning in infants. They assert that much of cognitive development is not the maturation of hardwired, specialized modules but the result of a ubiquitous domain-general learning mechanism. The Information Processing Principles have been used to explain a wide variety of developmental, such as causality [4], face recognition [6] and language learning [5]. (See [7] for an overview.)

3.2 CLA Implementation Details

The Information Processing approach leads to an unsupervised, self-organizing and multi-level process. Thus it lends itself to an implementation using a hierarchy of self-organizing maps, which are the basis of CLA. At CLA's lowest level, the SOMs are trained on the raw input from the environment. As they are trained, the nodes come to represent what features (or combinations of features) exist in the environment. How well a given node captures the current stimulus can be represented by an *activation value* that is proportional to how close the node's weight vector is to the incoming stimulus.

These activation values are then collected for each SOM into an *activation matrix* that represents the SOM's output. The matrices of all SOMs at one level are concatenated and become the input to the SOMs in the level directly above. This process then repeats for as many layers as are contained in the system (Figure 1). A layer can receive input from more than one sub-layer. Such is the case in [2], in which CLA replicated the results of studies demonstrating infant acquisition of causality.

To summarize, CLA was built as a computation model of cognitive development based on the Information Processing Principles, an elaboration of constructivism based on recent studies with infants. CLA has been used to model infant cognitive development, and it does so using a hierarchy of self-organizing maps.

4 CLA and Marginal Attribution

CLA uses hierarchical SOMs to generate knowledge of primitives at one level, and use that knowledge as a new set of primitives for a higher level. This is the same process as marginal attribution in the Schema Mechanism. However, CLA's use of SOMs makes generating this knowledge more efficient. Rather than maintaining an ever-growing table of correlation data, the SOM provides a finite data space and a process for different representations to compete for that space. Once a level of SOMs has been trained, they can be harvested for candidate higher-level representations – schemas in this case. The original SOMs can then be retired and their resources could be used for the next level of SOMs.

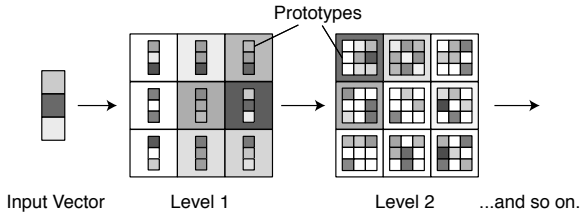

Figure 1: CLA Schematic. The state of the environment, represented by a feature vector (far left) is used to train the first SOM. SOM nodes are more active (darker) when they are closer to the input. Activations are then used as input to the next SOM. And so on.

Using CLA as the constructivist learning system makes the Schema Mechanism more efficient. Consequently, it becomes possible to apply it to more sophisticated domains. This new version of the Schema Mechanism is called the Constructivist Learning Architecture Schema Mechanism, or CLASM.

In this section, we describe how we implemented CLASM using CLA. We then apply CLASM to Drescher's "Microworld," a simple simulated agent and environment, to verify that CLASM learns everything that the Schema Mechanism learns. We then examine the results of this experiment and compare them to Drescher's results.

4.1 CLASM Implementation Details

CLASM (Constructivist Learning Architecture Schema Mechanism) is an instance of CLA that implements the Schema Mechanism using hierarchical SOMs for the marginal attribution process. Each action has an associated *Action SOM*. The weight vector for each Action SOM is twice the number of items that exists at the time of the SOM's creation. The first half of the weight vector represents the *context*, or the state of each item before the action is performed, represented by 1.0 (for on) or 0.0 (for off). The second half represents the *result*, or the change in item status that took place during the action's execution, which can range from –1.0 (an item has turned off) to +1.0 (an item has turned on). A SOM is only trained if the action associated with the SOM was just completed. Initially, an Action SOM is created for each primitive action. The initial input is the context and result of all primitive items (Figure 2a).

Once the Action SOMs have been trained, they are harvested for schemas. When a node becomes a schema, the weight vectors are converted into context and result item lists. Context items with weights less than 0.1 become negative context items, and those with weights greater than 0.9 become positive context items. Result items with weights less than –0.9 become negative result items, and those with weights greater than 0.9 become positive result items. A

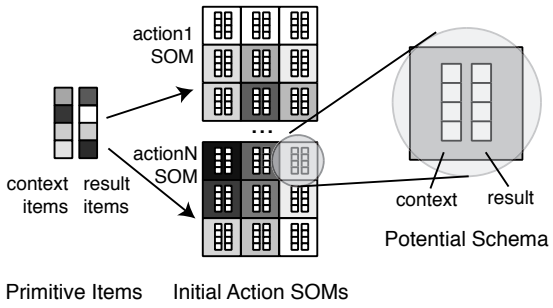

Figure 2a: Initial State of CLASM.

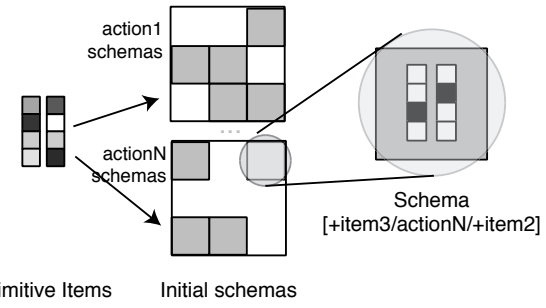

Figure 2b: Harvesting Schemas.

Figure 2: First Stages in CLASM. Initially, Action SOMs are created for all primitive actions, and their input is the current and result state of all primitive items. In 2a, there are N primitive actions and four primitive items. Each node in the Action SOMs is a potential schema, with context and result trained in the node, and the action defined by the node's Action SOM. When training completes, schemas are harvested from the Action SOMs, and the remaining structure is disposed. In 2b, the example schema [+item3/action/+item2] states that when item3 is on and actionN is executed, then item2 will activate. This technique allows selecting and representing only the relevant feature correlations, saving computational resources.

node becomes a schema only if there is at least one result item (Figure 2b). If a schema has a novel result, a new action is created with that result as the goal. Once harvested, Action SOMs are deleted.

At the beginning of the following stage, new Action SOMs are created in association with every action, including the new compound actions. The input vector to the new Action SOMs is the context and result of every item, including the new synthetic items (Figure 3). A synthetic item is activated if its schema holds, i.e. if a) the schema's action is performed, b) all positive context items are true and all negative context items are false when the action initiates, and c) all positive result items are on and all negative result items off when the action terminates.

4.2 Experiment: CLASM in Microworld

To verify that CLASM is a reimplementation of the Schema Mechanism, CLASM was run on the "Microworld" described by Drescher [8, pp.113-119]. The resulting learned constructs were then compared to those reported by Drescher [8, pp.120-141].

The "Microworld" is a simulation of an agent in an environment. The agent can see in a five-by-five visual field, and can move the visual field in the four cardinal directions. The agent can see details of items that directly in or adjacent to the center of vision (known as the "fovea"). The agent can also move a "hand" through the environment, which the agent can also see. The agent can feel objects directly under or adjacent to the hand, and can feel detailed information of objects in the same position as the hand. The agent can also close the hand. If the hand is closed while an object is at the hand position, the object is grasped and moves with the hand. If a grasped object is brought directly in front of the agent, the agent can "taste" the object. The agent can only hold one object at a time.

The environment is a seven-by-seven grid and has three objects: a round and soft ball, a hard cube, and the agent's hand. The ball and the cube cannot occupy the same position at the same time.

There are 10 primitive actions: moving the hand in the four cardinal directions (`handf`, `handb`, `handl`, `handr`), moving the eye in the four cardinal directions (`eyef`, `eyeb`, `eyel`, `eyer`), closing the hand (`grasp`) and opening the hand (`ungrasp`).

There are 76 primitive items: hand position (`hp11`…`hp33`), eye position (`vp11`…`vp33`), coarse feeling (`tactf`, `tactb`, `tactl`, `tactr`), detailed feeling (`text0`…`text3`), coarse feeling with body (`bodyf`, `bodyb`, `bodyl`, `bodyr`), taste (`taste0`…`taste3`), hand closed (`hcl`), hand grasping (`hgr`), coarse vision (`vf00`..`vf44`), and detailed vision (`fovf0`…`fovf2`, `fovb0`…`fovb2`, `fovl0`…`fovl2`, `fovr0`…`fovr2`, `fovx0`…`fovx2`).

Training each level took place in three stages with different learning parameters for the SOMs (Table 1). Two levels were trained. At the first level, each stage invoked 10,000 actions (thus, each SOM was trained approximately 1000 times). At the second level, each stage invoked 100,000 actions (to account for the greater number of Action SOMs).

4.3 Results

The training session resulted the reproduction of all the schemas reported by Drescher. Some generated schemas were functionally equivalent to reported schema, but most were actually identical. Following is a summary of schemas generated by CLASM, and the page number in

Table 1: SOM Learning Parameters

Stage	Learning Rate	Neighborhood
1	0.2	2
2	0.15	1
3	0.1	0

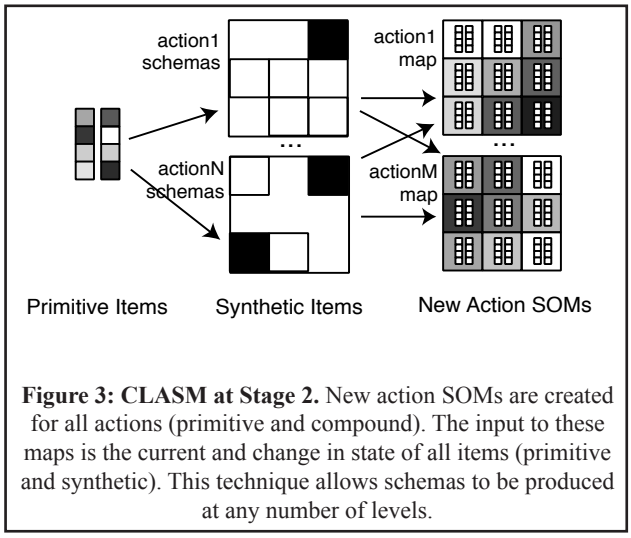

Figure 3: CLASM at Stage 2. New action SOMs are created for all actions (primitive and compound). The input to these maps is the current and change in state of all items (primitive and synthetic). This technique allows schemas to be produced at any number of levels.

Drescher [8] where they are reported.

Grasping (p120)
These are examples of the simplest schemas:

```
/grasp/+hcl,
+text0+text2/grasp/+hcl+hgr.
```

The first one says that when I close my hand, my hand is closed. The second schema says that when I can feel something, and I close my hand, my hand is closed and holding something.

Elaborating the visual field (p122)
These schemas report the change in relative position of seen objects when the eye moves:

```
+vf21/eyer/+vf11-vf21
+vf30/eyer/+vf20-vf30
```

They can be grouped to scan an area:

```
+vf34/eyer/-vf34+vf44,
+vf44/eyel/-vf44+vf34,
+vf44/eyef/+vf43-vf44,
+vf43/eyeb/+vf44-vf43.
```

Foveal relations (p124)
This schema says, if I see something directly behind the center of vision, and I move my eye back, they I will be looking directly at the object:

```
+vp23+vf22+fovb0/eyeb/+vp22-vp23-fovb0+fovx0.
```

All foveal relations of this nature are generated.

Elaborating the proprioceptive fields (p126)
The following is a network of visual schemas:

```
+vp23/eyer/-vp23+vp33,
+vp33/eyel/-vp33+vp23,
+vp33/eyeb/-vp33+vp32,
+vp32/eyef/+vp33-vp32.
```

Also produced is a network of hand schemas:

```
+hp23/handr/-hp23+hp33,
+hp33/handl/+hp23-hp33,
+hp22/handf/-hp22+hp23,
+hp31/handf/-hp31+hp32.
```

These relations exist for every movement action between all positions.

Negative consequences (p127)
In this case, an action will turn an item off. We have already seen an example in the "fovial relations" section above. Here are three more:

```
+vp31+vf10/eyel/+vp21-vp31-vf10,
+vp21/eyel/+vp11-vp21,
+hp13+vf24/handb/+hp12-hp13.
```

Positional actions (p127-129)
Once the first stage is complete, the results of these schemas form the goals of new compound actions. CLASM produces the same actions reported in Drescher:

```
New action: <+hp22>,
New action: <+hp33>,
New action: <+vp22>,
New action: <+vp33>,
New action: <+vf34>,
New action: <+vf12>.
```

Moving the hand to the mouth (p129)
CLASM has also learned how to pick up objects and move them to the mouth:

```
+hp22+vp22+text1+text3+hcl+hgr
+vf22+fovx1+fovx2/handb/
+hp21-hp22+bodyf+taste1+taste3+vf21
-vf22+fovb1+fovb2-fovx1-fovx2.
```

Visual effects of incremental hand motions (p130-131)
CLASM has learned how the visual field is altered when the hand moves:

```
+vf11/handl/+vf01-vf11.
```

The following schema says, if I see my hand at (3,2), and I move my hand right, then I see my hand at (2,2):

```
+hp22+vp32+vf11+fovl2/handr/
-hp22+hp32-vf11+vf21-fovl2+fovx2.
```

Touching what is seen, and vice versa (p132)
CLASM can also learn to touch what it sees with the help of a compound action. A schema using the compound action `<+vf34+hp23>` can be used to move the hand to touch the item at (3,4):

```
+hp13/<+vf34+hp23>/+tactb-hp13+hp23.
```

Similar schemas can move the eye to see what is touched.

Palpable and visible persistent objects (p136-137)
These are synthetic items that represent the beginning of persistent-object concept. In this example, the compound act `<+hp21>` is used by the synthetic item below to mean "Palpable object at (1,1)" (that is, if I put my hand at (2,1), I feel something to the left):

```
[/<+hp21>/+hp21+tactl].
```

Similarly, the following synthetic item represents "Visible object at (3,3)":

```
[/<+vp22>/+vp22+vf33].
```

These, and others, combine to form groups of synthetic items that correspond to an object's persistent identity.

Cross-modal representations (p140)
Finally, these schemas represent persistent-object information across modalities. The first schema describes how to touch something seen, and the second one describes how to view something felt:

```
+vp12+fovb1/<+hp21>/+hp21+tactl,
+hp21+tactl/<+vp12>/+vp12+fovb1.
```

CLASM has produced all of the same constructs that were produced by Drescher. Thus, CLASM is a true reimplementation of the Schema Mechanism.

5 Discussion and Future Work

CLASM succeeded in reproducing the results of the original Schema Mechanism. It did so using SOMs as a competitive learning system and thus maintained a tight control on the use of resources. CLASM greatly improves the efficiency of the original Schema Mechanism while remaining a faithful to it.
Part of what makes CLASM more efficient is the way it bypasses simple schemas with one or two result items, and jumps directly to more complex, multi-result and multi-context schemas. There is no need to go through the stepwise building process to get to more complex schemas because the entire context and result are being presented to the SOM at once. The resulting complex schemas simply emerge from the data. Simpler schemas are easy to derive from the final product, should they be desired.
A more efficient Schema Mechanism opens the door for further application and experimentation with constructivist learning. Constructivist learning has a wide range of applications, including grounded knowledge, agent-based systems and cognitive science. As the next step, we plan to apply this system to mobile robots. A robot control system that is built from the robot's own experience and experimentation will be more efficient and robust. And, while using something like the Schema Mechanism was once untenable, it is now possible to use constructivist learning in a real-world environment.

6 Conclusion

CLASM is an efficient implementation of the Schema Mechanism based on the Constructivist Learning Architecture (CLA). The experiment presented in this paper demonstrates that CLASM faithfully reproduces the original results. The most prominent schemas automatically emerge as units in SOMs, instead of having to keep all correlations in memory at all times. Such an efficient implementation is the first step towards applying constructivist learning techniques to autonomous robots in real environments.

References

[1] H. H. Chaput, "Post-Piagetian constructivism for grounded knowledge acquisition," *Proceedings from the AAAI Spring Symposium on Grounded Knoweldge* (2001).
[2] H. H. Chaput and L. B. Cohen, " *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, pp. 182-187, 2001.
[3] L. B. Cohen. An information processing approach to infant perception and cognition. In G. Butterworth and F. Simion (Eds.), *Development of Sensory, Motor, and Cognitive Capacities in Early Infancy: From Sensation to Cognition*. Sussex: Erlbaum (UK) Taylor & Francis (1998).
[4] L. B. Cohen & G. Amsel, "Precursors to infants' perception of the causality of a simple event," *Infant Behavior and Development 21 (4)*, 713-732 (1998).
[5] Cohen, L. B., Amsel, G., Redford, M. A. & Casasola, M. (1998). The development of infant causal perception. In A. Slator (Ed.), *Perceptual development: Visual, auditory and speech perception in infancy.* London: UCL Press (Univ. College London) and Taylor and Francis.
[6] L. B. Cohen & C. H. Cashon. Infant object segregation implies information integration, *Journal of Experimental Child Psychology*, (2000).
[7] L. B. Cohen, H. H. Chaput & C. H. Cashon, "A constructivist model of infant cognition," *Cognitive Development 17*, pp.1323-1343 (2002).
[8] G. Drescher, *Made-up minds: a constructivist approach to artificial intelligence*, Cambridge, MA: MIT Press, 1991.
[9] T. Kohonen, *Self-organizing maps*, Berlin: Springer-Verlag, 1987.
[10] J. Piaget, *The origins of intelligence in children*, New York: International Universities Press (1952).