Learning and Transferring Action Schemas

Paul R. Cohen, Yu-Han Chang, Clayton T. Morrison
USC Information Sciences Institute
Marina del Rey, California

Abstract

Jean is a model of early cognitive development based loosely on Piaget's theory of sensori-motor and pre-operational thought. Like an infant, Jean repeatedly executes schemas, gradually transferring them to new situations and extending them as necessary to accommodate new experiences. We model this process of accommodation with the Experimental State Splitting (ESS) algorithm. ESS learns elementary action schemas, which comprise controllers and maps of the expected dynamics of executing controllers in different conditions. ESS also learns compositions of action schemas called gists. We present tests of the ESS algorithm in three transfer learning experiments, in which Jean transfers learned gists to new situations in a real time strategy military simulator.

1 Introduction

One goal of the Jean project is to develop representations and learning methods that are based on theories of human cognitive development, particularly Piaget's theories of sensori-motor and pre-operational thought. Piaget argued that infants acquire knowledge of the world by repeatedly executing action-producing schemas [Piaget, 1954]. This activity was assumed to be innately rewarding. Piaget introduced *assimilation* of new experience into extant schemas and *accommodation* of schemas to experiences that don't quite "fit" as the principal learning methods for infants. This paper gives a single computational account of both assimilation and accommodation.

Although it seems to be a relatively recent focus in machine learning, *transfer* of learned knowledge from one situation or scenario to another is an old idea in psychology and is fundamental to Piaget's account of cognitive development. This paper demonstrates that the schemas learned by Jean can be transferred between situations, as any Piagetian schema should be.

Jean learns action schemas and gists. An action schema comprises a controller, a representation of the dynamics of executing the controller, and one or more criteria for stopping executing the controller. Gists are compositions of action schemas for common tasks; for instance, `push` involves the sequence `move-to`, `contact` and `apply-force`. Jean's learning method is a kind of state splitting in which state descriptions are iteratively refined (split) to make the transitions between states as predictable as possible, giving Jean progressively more control over the outcomes of actions.

The following sections introduce Jean's action schemas (Sec. 2), and then establish correspondences between action schemas and the more familiar states and finite-state machines (Sec. 3). The state splitting algorithm is described in Section 4.1. Its application to transfer learning and empirical results with a small-unit tactical military simulator are presented in Sections 5 and 6, respectively.

2 Action Schema Components

Action schemas have three components: controllers, maps, and decision regions. Controllers control Jean's behavior. For example, the `(move-to Jean Obj)` controller moves Jean from its current location to the specified object. Jean has very few innate controllers — move-to, turn, rest, apply force. It learns to assemble controllers into larger plan-like structures called *gists* as described in Section 4.1.

As Jean moves (or executes any other controller) certain variables change their values; for instance, the distance between Jean's current location and `Obj` usually decreases when Jean executes the `move-to` controller. Similarly, Jean's velocity will typically ramp up, remain at a roughly constant level, then ramp down as Jean moves to a location. The values of these variables ground out in Jean's sensors, although some variables correspond to processed rather than raw sensory information.

These variables serve as the dimensions of *maps*. Each execution of a particular schema produces a trajectory through a map — a point that moves, moment by moment, through a space defined by distance and velocity, or other bundles of variables. Each map is defined by the variables it tracks, and different maps will be relevant to different controllers.

Each invocation of a controller creates one trajectory through the corresponding map, so multiple invocations will create multiple trajectories. Figure 1 shows several such trajectories for a map of distance for the `move-to` controller. One can see that `move-to` has a "typical" trajectory, which might be obtained by taking the mean of the trajectories in the map. Also, when operating in Jean's environment, `move-to` produces trajectories that lie within some distance of the mean trajectory. In a manner reminiscent of Quality Control Jean can assess whether a particular trajectory is "going out of bounds."

The idea that sensori-motor and pre-operational development should rely on building and splitting maps or related dynamical representations was anticipated by Thelen and Smith [Thelen and Smith, 1994] and has been explored by other researchers in developmental robotics and psychology (e.g., REMOVED-FOR-BLIND-REVIEW [Siskind, 2001; Barsalou, 1999; Mandler, 1992]).

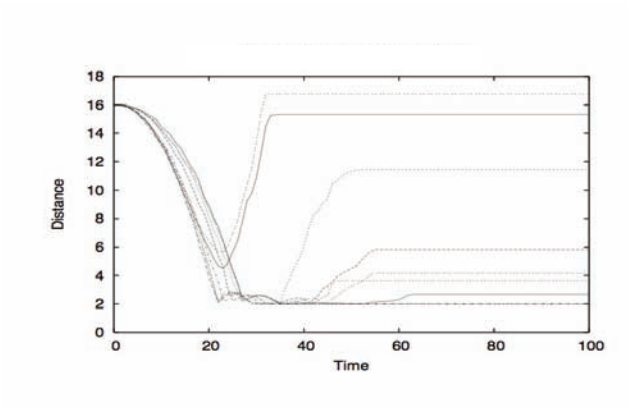

Figure 1: Trajectories through a map of distance between Jean and a simulated cat. Sometimes, when Jean gets close to the cat, the cat moves away and the distance between them increases, in which case a trajectory may go "out of bounds."

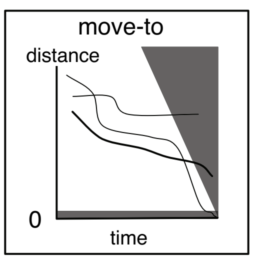

Figure 2: This schematic of a map, for the `(move-to Jean Obj)` controller, has one decision region associated with the distance between Jean and the object `Obj` being zero. The other decision region bounds the area in which Jean cannot reach `loc` even moving at maximum speed.

Every map has one or more *decision regions* within which Jean may decide to switch from one controller to another. One kind of decision region corresponds with achieving a goal; for example, there is a decision region of the `move-to` map in which distance to the desired location is effectively zero (e.g., the thin, horizontal grey region in Fig. 2). Another kind of decision region corresponds to being unable to achieve a goal; for instance, there is a region of a time-distance map from which Jean cannot move to a desired location by a desired time without exceeding some maximum velocity (e..g., the inverted wedge-shaped region in Fig. 2). These regions are sometimes called *envelopes*, [Gardiol and Kaelbling, 2003] and REMOVED-FOR-BLIND-REVIEW

Jean is not the only agent in its environment and some maps describe how relationships between Jean and other agents change. The upper two panels of Figure 4 illustrates how distance, relative velocity, heading, and contact change in an environment that includes Jean and another agent, called the "cat," an automaton that moves away from Jean if Jean moves too close, too quickly.

3 Action Schemas as Finite State Machines

It will help to draw parallels between Jean's maps and the more familiar elements of finite state machines (FSMs). Conventionally, states in FSMs represent static configurations (e.g., the cat is asleep in the corner) and arcs between states represent actions (e.g., `(move-to Jean cat)`). For us, arcs correspond to the intervals during which Jean executes controllers (i.e., actions), and states correspond to decision regions of maps. That is, the elements of action schemas are divided into the intervals during which a controller is "in bounds" (the arcs in FSMs) and the intervals during which Jean is thinking about what to do next (the states). Both take time, and so require a rethink of the conventional view that states in FSMs persist and transitions over arcs are instantaneous. However, the probabilistic semantics of FSMs are retained: A controller invoked from a decision region (i.e, an action invoked in a state) will generally take Jean into one of several decision regions (i.e., states), each with some probability. Figure 3 redraws Figure 2 as a finite state machine. Starting from a decision region of some action schema A, the `move-to` controller will with some probability (say .7) drop Jean in the decision region associated with achieving its goal, and, with the complementary probability, in the region associated with being unable to achieve the goal in time.

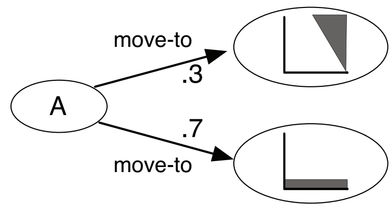

Figure 3: The action schema from Figure 2 redrawn as a finite state machine in which arcs correspond to the execution of controllers and states correspond to decision regions.

There is one special case: Sometimes the world changes when Jean is doing nothing. We model this as a schema in which no controller is specified, called a *dynamic schema* to distinguish it from an action schema. Dynamic schemas do have maps, because variables such as distance between Jean and another agent can change even when Jean does nothing; and they have decision regions, because Jean may want to invoke a controller when these variables take particular values (e.g., moving away when another agent gets too close). The FSMs that correspond to dynamic schemas have no controller names associated with arcs but are otherwise as shown in Figure 3.

4 Learning Action Schemas and Gists

Jean has a small set of innate controllers and a few "empty" maps, and it "fills in" these maps with trajectories and learns decision regions. Jean also learns compositions of action schemas called *gists* for their story-like or plan-like structure. One principle underlies how Jean learns decision regions. It is to maximize predictability, or minimize the entropy of "what's next." Within an action schema, what's next is a location in a map. At the boundaries of action schemas (that is, in decision regions, or states) what's next is the next state. We call the entropy of what's next the *boundary entropy*. Jean learns decision regions that minimize boundary entropy between states. Interestingly, this criterion tends to minimize boundary entropy within maps as a side effect.

4.1 Experimental State Splitting Algorithm

We give a formal outline of the Experimental State Splitting (ESS) algorithm in this section. Jean receives a vector of features $F^t = \{f_1, \ldots, f_n\}$ from the environment at every time tick t. Some features will be map variables, others will be inputs that have not yet been associated with maps. Jean is initialized with a goal state s_g and a non-goal state s_0. S_t is the entire state space at time t. A is the set of all controllers, and $A(s) \subseteq A$ are the controllers that are executed in state $s \in S$. Typically $A(s)$ should be much smaller than A. $H(s_i, a_j)$ is the boundary entropy of the state s_i in which controller a_i is executed. A small boundary entropy corresponds to a situation where executing controller a_j from state s_i is highly predictive of the next observed state. Finally, $p(s_i, a_j, s_k)$ is the probability that executing controller a_j in state s_i will lead to state s_k.

For simplicity, we will focus on the version of ESS that only splits states; an alternative version of ESS is also capable of learning specializations of parameterized controllers. The ESS algorithm follows:

- Initialize state space with two states, $S_0 = \{s_0, s_g\}$.
- While ϵ-optimal policy not found:
  - Gather experience for some time interval τ to estimate the transition probabilities $p(s_i, a_j, s_k)$.
  - Find a schema feature $f \in F$, a threshold $\theta \in \Theta$, and a state $s_i \in S$ to split that maximizes the boundary entropy score reduction of the split: $\max_{S,A,F,\Theta} H(s_i, a_i) - \min(H(s_{k_1}, a_i), H(s_{k_2}, a_i))$, where s_{k_1} and s_{k_2} result from splitting s_i using feature f and threshold θ: $s_{k_1} = \{s \in s_i | f < \theta\}$ and $s_{k_2} = \{s \in s_i | f \geq \theta\}$.
  - Split $s_i \in S_t$ into s_{k_1} and s_{k_2}, and replace s_i with new states in S_{t+1}.
  - Re-solve for optimal plan according to p and S_{t+1}

Finding a feature f and the value on which to split states is equivalent to finding a decision region to bound a map.

Without heuristics to reduce the effort, the splitting procedure would iterate through all state-controller pairs, all features $f \in F$, and all possible thresholds in Θ, and test each such potential split by calculating a reduction in boundary entropy. This is clearly an expensive procedure.

ESS uses a simple heuristic to find threshold values for features f and, thus to split a state: States change when several state variables change more or less simultaneously. This heuristic is illustrated in Figure 4. The upper two graphs show time series of five state variables: headings for Jean and the cat (in radians), distance between Jean and the cat, and their respective velocities. The bottom graph shows the number of state variables that change value (by a set amount) at each tick. When the number of state variables that change simultaneously exceeds a threshold, Jean concludes that the state has changed. The value of the schema f at the moment of the state change is likely to be a good threshold for splitting f. For example, between time period 6.2 and 8, Jean is approaching the cat, and the heuristic identifies this period as one state. Then, at time period 8, several indicators change at once, and the heuristic indicates Jean is in a new state, one that corresponds to the cat moving away from Jean.

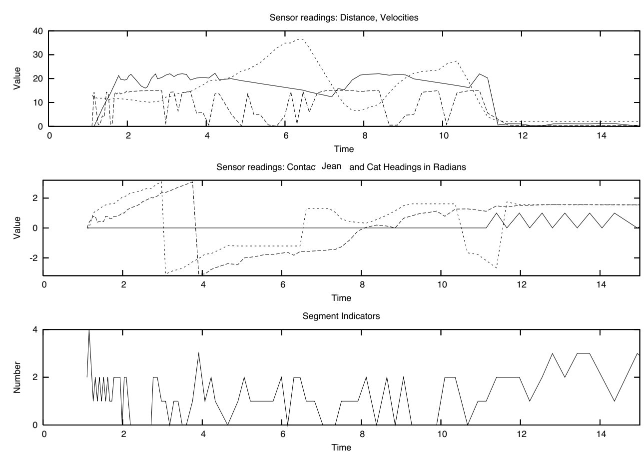

Figure 4: New states are indicated when multiple state variables change simultaneously.

5 Transferring Learning

Although ESS can learn action schemas and gists for new situations from scratch, we are much more interested in how previously learned policies can accommodate or transfer to new situations. Gists capture the most relevant states and actions for accomplishing past goals. It follows that gists may be transferred to situations where Jean has similar goals and the conditions in the situation are similar.

The version of ESS that we described above is easily modified to facilitate one sort of transfer: After each split we remove the transition probabilities on all action transitions between each state. This allows the state machine to accommodate new experience while maintaining much of the structure of the machine (see REMOVED-FOR-BLIND-REVIEW for a previous example of this idea). In the experiments in the next section we explore the effects of transfer using this mechanism in several conditions.

6 Experiments

To measure transfer we adopt a protocol sometimes called **B/AB**: In the **B** condition the learner learns to perform some tasks in situation or scenario B. In the **AB** condition, the learner first learns to perform tasks in situation or context A and then in B. By comparing performance in situation B in the two conditions after different amounts of learning in situation B one can estimate the effect of learning in A and thus the knowledge transferred from situation A to situation B. For instance, A might be tennis and B squash, and the **B/AB** protocol compares learning curves for squash alone (the **B** or control condition) with learning curves for squash after having learned to play tennis (the **AB** or transfer condition). We say positive transfer has occurred between A and B if the learning curves for squash in the transfer condition are better than those in the control condition.

In general "better" learning curves means faster-rising curves (indicating faster learning), or a higher value after some amount of learning (indicating better performance), or higher value after no learning (indicating that knowledge from A helps in performing B even before any learning has occurred in B). In our experiments, better learning performance means less time to learn a gist to perform a task at a criterion level. Thus, a smaller area beneath the learning curve indicates better learning performance, and we compare conditions **B** and **AB** by comparing the area beneath the learning curves in the respective conditions. A bootstrap-randomization procedure, described shortly, is used for significance testing.

We tested Jean's transfer of gists between situations in the 3-D real time strategy game platform **ISIS**. ISIS can be configured to simulate a wide variety of military scenarios with parameters for specifying different terrain types, unit types, a variety of weapon types, and multiple levels of unit control (from individual soldiers to squad-level formations).

In each of three experiments, Jean controlled a single squad at the squad level, with another squad controlled by an automated but non-learning opponent. Jean's squad ranged in size from 7 to 10 units while the opponent force ranged from 1-3 units. Although the opponent was smaller, it could move faster than Jean's forces. In each experiment, Jean's goal is to move its units to engage and kill the opponent force.

Jean is provided four innate action schemas: `run`, `crawl`, `move-lateral`, and `stop-and-fire`. It must learn to compose these into gists that are appropriate for different engagement ranges, possible entrenchment of the opponent, and some terrain features (mountains).

All experiment scenarios were governed by a model of engagement ranges that determined how the squads interact and how the opponent controller would respond to Jean's actions. Engagement ranges are defined as follows:

iv. *Outer Range* (beyond 250 meters): As long as the opponent is within line of sight (i.e., not obscured by terrain features), no matter what the distance, Jean can locate the opponent. However, beyond 250 meters, the opponent cannot see Jean's forces.

iii. *Visual Contact* (up to 250 meters): At this range, if Jean's forces are standing they will be sighted by the opponent. If Jean's forces are crawling, and they have not begun firing, then Jean's forces won't be sighted (until they are within range *i*).

ii. *Firing Range* (up to 200 meters): Within this range, either force can fire on the other. Once Jean's forces have fired, they are considered sighted, even if crawling, and the opponent can return fire with the same effectiveness as Jean's forces.

i. *Full Contact* (up to 100 meters): At this range, even if Jean's forces are crawling, they will be sighted by the opponent. Direct fire has full effect.

Each experiment had a transfer condition **AB** and a control condition, **B**. In the former, Jean learned gists to accomplish its goal in a scenario designated A and then learned to accomplish its goal in scenario B. In the latter, control condition, Jean tried to learn in scenario B without benefit of learning in A.

The experiments differ in their A and B scenarios:

Experiment 1 : All action takes place in open terrain. A scenarios all have Jean's forces starting near enemy forces. B scenarios are an equal mix of starting near the enemy or far away from the enemy.

Experiment 2 : All action takes place in open terrain. A scenarios all have Jean's forces starting far from the enemy forces. B scenarios are an equal mix of starting near the enemy or far away from the enemy.

Experiment 3 : The terrain for A scenarios is open, whereas the terrain for B scenarios has a mountain that, for some placements of Jean's and the enemy's forces, prevents them seeing each other. (The advantage goes to Jean, however, because Jean knows the location of the enemy forces.) The A scenario is an equal mix of starting near or far from the enemy, the B scenario is an equal mix of starting near and far from the enemy in the mountain terrain.

6.1 Metrics and Analysis

We plot the performance of the Jean system in the various experimental scenarios as learning curves over training trials. Better learning performance is indicated by a smaller number of training instances required by Jean to achieve a criterion level of performance. Thus, a smaller area beneath the learning curve indicates better learning.

Given n learning curves for the **B** and **AB** conditions, we test the null hypothesis of "no transfer" as follows: Let $\mathcal{B}$ and $\mathcal{AB}$ denote the sets of n learning curves in the **B** and **AB** conditions, respectively, $\overline{\mathcal{X}}$ be the mean learning curve for a set of learning curves $\mathcal{X}$, and $Area(\mathcal{X})$ be the area under learning curve $\mathcal{X}$. We measure the benefit of transfer learning with the *transfer ratio*

$$r(\mathcal{B}, \mathcal{AB}) = \frac{Area(\overline{\mathcal{B}})}{Area(\overline{\mathcal{AB}})}.$$

Values greater than one indicate that the area under the learning curves in the control condition **B** is larger than in the transfer condition, or a positive benefit of transfer.

The null hypothesis is $r = 1.0$, that is, learning proceeds at the same rate in the control and transfer conditions. To test whether a particular value of r is significantly different from 1.0 we require a sampling distribution for r under the null hypothesis. This is provided by a randomization-bootstrap procedure [Cohen, 1995]. First, we combine the sets of learning curves, $\mathcal{C} = \mathcal{B} \cup \mathcal{AB}$. Then, we randomly sample, with replacement, n curves from $\mathcal{C}$ and call this a psuedosample $\mathcal{B}^*$; and again randomly sample, with replacement, another n curves from $\mathcal{C}$ to get pseudosample $\mathcal{AB}^*$. Then, we compute $r(\mathcal{B}^*, \mathcal{AB}^*)$ and store its value in the sampling distribution of r. Repeating this process a few hundred times provides an estimate of the sampling distribution. For one-tailed tests, the p-value is simply the proportion of the sampling distribution with values greater than or less than r (depending on the direction of the one-tailed alternative hypothesis). Because we expect learning rates in the transfer condition to be higher, our alternative hypothesis is $r > 1$ and the p value is the proportion of the sampling distribution with values greater than r. (N.B.: When there is a possibility of *negative* transfer, in which the knowledge acquired in A actually impedes learning in B, it might be more appropriate to run two-tailed tests.)

To obtain a confidence interval for r, as opposed to a p value, we again construct a sampling distribution, but this time, instead of first combining the two sets $\mathcal{B}$ and $\mathcal{AB}$, we sample $\mathcal{B}^*$ with replacement from $\mathcal{B}$, and $\mathcal{AB}^*$ similarly from $\mathcal{AB}^*$. Then we calculate $r(\mathcal{B}^*, \mathcal{AB}^*)$. Repeating this process yields a sampling distribution for r. A 95% confidence interval around r is the interval between the 2.5% and 97.5% quantiles of this distribution [Cohen, 1995].

6.2 Results

Let us start with a qualitative assessment of what Jean learned. In Experiment 1, Jean learned in scenario A to run at the enemy and kill them. In scenario B, Jean learned a gist that included a conditional: When starting near the enemy, use the gist from scenario A, but when starting far from the enemy *crawl* — don't run — until one is near the enemy and then use the gist from scenario A. The alternative, running at the enemy from a far starting location, alerts the enemy and causes them to run away. State splitting did what it was supposed to do: Initially, Jean's gist for scenario B was its gist for A, so Jean would always run at the enemy, regardless of starting location. But through the action of state splitting, Jean eventually learned to split the state in which it ran at the enemy into a run-at and a crawl-toward state, and it successfully identified the decision region for each. For instance, the decision region for the crawl-toward state identifies a distance (corresponding to being near the enemy), from which Jean makes a transition to the state in which it runs at and shoots the enemy.

Similar results are obtained in Experiment 3, where Jean learns to run at the enemy from a far starting location as long as the mountain prevents the enemy from sighting Jean, otherwise to crawl.

In Experiment 2, Jean learned nothing in scenario A and was no more successful in scenario B. This is due to the difficulty of the scenario. Jean is always initialized far away from the enemy units, and must learn a policy for killing them by exploring a continuous, high-dimensional feature space using her four available actions. Many of these actions result in the enemy soldiers detecting Jean's presence and running away, thus reducing Jean's chances of ever reaching her goal by simple exploration. Since Jean does not learn anything useful in the A scenario, her performance in the **AB** transfer condition is no better than in the control condition **B**.

The learning curves for the A and B scenarios in Experiment 1 are shown in Figure 5. Note that the vertical axis is "time to achieve the goal," in the scenarios, so a downward-sloping curve corresponds to good learning performance. Jean learned a good policy in the A scenario where the enemy units are initially close to Jean's position. Jean receives a significant benefit when it learns in the B scenario after having learned in the A scenario (i.e., in the transfer condition **AB**). The **AB** learning curve starts out immediately with much better performance than the **B** learning curve. In fact, it takes 200 trials for learning in the **B** condition, in which no transfer happens, to reach the level of performance observed at all levels of training in the transfer condition **AB**. However, the **AB** curve is roughly flat, which means learning in scenario A provides a boost in performance in B but has no impact on the rate of learning in B. We rejected the null hypothesis that **B** and **AB** have the same mean learning curve with a p value of 0.035; the **AB** condition has significantly better learning curves.

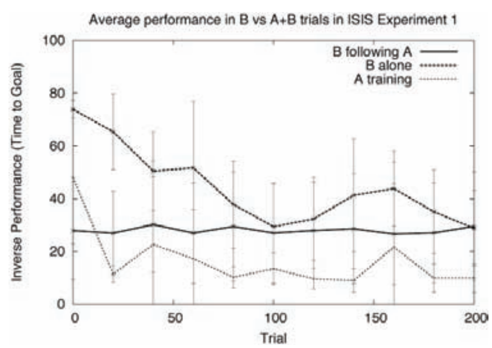

Figure 5: Learning curves for learning in scenario A, in scenario B, and in scenario B after having first learned in scenario A. Each point is averaged over eight replications of the experiment. Error bars are two standard deviations wide. Each point of each curve is the average of ten fixed test trials. Test trials are conducted every 20 training trials. The x-axis plots the number of training trials that the agent has completed.

In contrast, in Experiment 2 Jean does not succeed in achieving any transfer. Jean does not ever learn anything useful in the A condition because, as noted above, when Jean starts far from the enemy the search space of possible gists is too large. The transfer ratio $r = .97$ is nearly equal to the expected value under the null hypothesis of no transfer, and the p value is accordingly high, $p = .557$.

In Experiment 3, Jean transfers learned knowledge from the A scenario, which involves open terrain to "jump-start" performance in the B scenario, which includes a mountain.

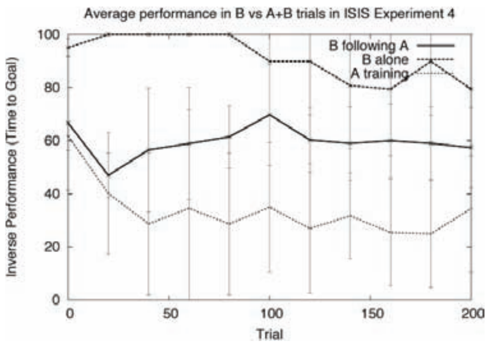

Figure 6: Experiment 3. Learning curves for the A, AB, and B conditions, averaged over eight replications of the experiment. The x-axis plots the number of training trials that Jean has completed.

	r	p	2.5% quantile	97.5% quantile
Expt. 1	1.591	0.035	1.02	2.97
Expt. 2	0.970	0.557	0.63	1.34
Expt. 3	1.531	0.0034	1.24	1.85

Table 1: The transfer ratio r, the p value, and lower and upper bounds of a 95% confidence interval around r in all three experiments.

Figure 6 shows the average learning curves we observe in this experiment. Note the wide error bars. While learning in the transfer condition is significantly more successful than in the control condition ($r = 1.53, p = 0.0034$) the data do not support the conclusion that Jean's learning in the B scenario is *accelerated* by learning in A. As in Experiment 1, it appears that the curve for learning in B after A is quite flat and the benefit of knowledge learned in A is felt immediately in domain B.

Table 1 summarizes the data we observed from the three experiments and provides confidence intervals for the transfer ratios.

7 Discussion

The Experimental State Splitting algorithm splits an undifferentiated gist comprising just a start state and goal state into a sequence of states that follow each other with high predicability. To do this, ESS finds new decision regions for action schemas, that is, it finds values of map variables that indicate Jean should switch from one action schema to another. ESS often reduces the entropy within action schemas, that is, it reduces the entropy of trajectories in maps and "tightens up" its expectations of the dynamics associated with executing a controller. This is because trajectories lead to decision regions and wildly varying trajectories lead to large numbers of decision regions, and, thus, to highly entropic distributions of next states. Splitting states to reduce the entropy of the distributions of next states is equivalent to reducing the number of decision regions associated with a state and thus the variability of trajectories in a map.

Interestingly, this hints at an unexplored aspect of the relationship between the Piagetian learning mechanisms of assimilation and accommodation. Assimilation means incorporating experiences into known schemas, which, for Jean, means adding trajectories to the maps of action schemas. Accommodation means modifying schemas when experiences don't fit. For Jean, this means state splitting. We have learned that the impetus for accommodation, and thus for maintaining narrow bounds on what's assimilated, is a desire to predict the next state.

We also have preliminary evidence that gists can transfer between situations by a relatively simple mechanism, namely, keeping the structure of a gist's FSM but deleting its transition probabilities.

Much work remains to be done. Further transfer experiments are already underway. Better heuristics for proposing states to split and feature values as splitting thresholds are being developed. Perhaps most challenging, we want ESS to be a truly experimental kind of state splitting, an algorithm that proposes and carries out experiments to assess the causal influences of features on map trajectories.

References

[Barsalou, 1999] L. W. Barsalou. Perceptual symbol systems. *Behavior and Brain Sciences*, 22:577–609, 1999.

[Cohen, 1995] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, MA, 1995.

[Gardiol and Kaelbling, 2003] N. H. Gardiol and L. P. Kaelbling. Envelope-based planning in relational mdps. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2003.

[Mandler, 1992] J. Mandler. How to build a baby: Ii. conceptual primitives. *Psychological Review*, 99:597–604, 1992.

[Piaget, 1954] J. Piaget. *The Construction of Reality in the Child*. New York: Basic, 1954.

[Siskind, 2001] J. Siskind. Grounding lexical semantics of verbs in visual perception using force dynamics and even logic. *Journal of AI Research*, 15:31–90, 2001.

[Thelen and Smith, 1994] E. Thelen and L. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action*. The MIT Press, Cambridge, MA, 1994.