Learning Causal Patterns:
Making a transition from data-driven to theory-driven learning

Michael Pazzani
Department of Information and Computer Science
University of California
Irvine, CA 92717
pazzani@ics.uci.edu
Machine Learning, 11, 173-194.

Abstract

We describe an incremental learning algorithm, called theory-driven learning, that creates rules to predict the effect of actions. Theory-driven learning exploits knowledge of regularities among rules to constrain learning. We demonstrate that this knowledge enables the learning system to rapidly converge on accurate predictive rules and to tolerate more complex training data. An algorithm for incrementally learning these regularities is described and we provide evidence that the resulting regularities are sufficiently general to facilitate learning in new domains. The results demonstrate transfer from one domain to another can be achieved by deliberately overgeneralizing rules in one domain and biasing the learning algorithm to create new rules that specialize these overgeneralizations in other domains.

Running Head: Learning Causal Patterns

1.0 Introduction

In order to understand the environment, people must learn to predict the effects of their own actions and the actions of others. This prediction process requires the learner to associate state changes with actions. Many researchers have shown that, even in very young children, this learning process is constrained by general knowledge of the actions that are likely to be responsible for state changes. Bullock, Gelman, and Baillargeon (1982) and Shultz and Kestenbaum (1985) provide excellent overviews of the types of general constraints that are exploited during the learning of specific causal rules.

In Pazzani (1991), we propose a computational learning model that is intended to explain why learning some predictive relationships is more difficult than others for human learners. Here, we report on experiments that show that this learning model provides an advantage over a purely data-driven learner under a variety of circumstances. The learning model relies on an explicit representation of general knowledge of causality. Here, we also propose an algorithm for learning this general knowledge of causality. We computationally explore the hypothesis that much of the general knowledge of causality (and some simple specific causal rules) can be learned by simple correlational processes. Once this general knowledge of causality is acquired, the later acquisition of more complex specific causal rules is constrained and facilitated by the general knowledge of causality that has been acquired. As general knowledge of causality is acquired, the learner shifts learning strategies from a data-driven strategy that attempts to find regularities in a set of examples to a theory-driven strategy that looks for instantiations of its general knowledge.

In particular, a learner will be described that starts with only temporal knowledge of causality (i.e., temporal contiguity and temporal priority). This knowledge is sufficient to induce simple causal rules. We hypothesize that more general causal knowledge is acquired by finding common patterns in the simple causal rules. This more general knowledge represents additional spatial and configural patterns that are present in causal rules.

To make the problem more concrete, consider the sequence of events in Table 1. This is the output of Talespin (Meehan, 1981), a program designed to generate short stories describing actors attempting to achieve goals. We will use Talespin to simulate a world in which the learner must observe actions, and predict their results.

The output is divided into a number of discrete time intervals. In each time interval, some actions may be observed and the state of the world may change. Although this simulated input is necessarily a simplification of the complex world in which a child learns causal rules, we have added a number of complexities to Talespin to make it more realistic while still allowing systematic experimentation by varying parameters such as the amount of noise in the data. For example, two actions may occur in the same time interval (e.g., Time 4). Therefore, the agent must be able to determine which effect is associated with which action. This problem is complicated by the fact that some actions may have more than one effect (e.g., Time 9) and some actions may have no effect. In addition, the same action may have different effects (e.g., dropping an object in Times 4 and 7). Finally, some state changes may be observed, although the action that caused the state change may not be observed (e.g., the phone ringing during Time 10).

This learning problem can be summarized as follows:

Given: A sequence of time interval where each time interval consists of a set of actions and a set of state changes.

Create: A set of rules that predict when a state change will occur.

In this paper, we first discuss the representation of examples, causal rules, and general knowledge of causality. Next, we review a learning method called theory-driven learning (TDL) that makes use of general knowledge of causality to constrain the search for causal rules. Finally, we discuss an extension to theory-driven learning that allows the general knowledge of causality to be learned from initial observations and demonstrate that the induced knowledge facilitates learning in new domains.

Table 1. An example of the output produced by Talespin, divided into 14 time intervals.
(0) Karen was thirsty. (1) She pushed the door away from the cupboard. The cupboard was open. (2) She took a small red plastic cup from the cupboard. Mike pushed the light switch. She had the cup. The cup was not in the cupboard. The light was on. (3) She pushed the door to the cupboard. The cupboard wasn't open. (4) The auburn cat pushed a large clear glass vase to the tile floor. Karen pushed the handle away from the cold faucet. The cold water was flowing. The vase was broken. (5) She moved the cup to the faucet. The cup was filled with the water. (6) She pushed the handle to the faucet. The cold water wasn't flowing. (7) Karen dropped the cup to the tile floor. The cup wasn't filled with the water. (8) She pushed the door away from the cupboard. The cupboard was open. (9) She took a small clear glass cup from the cupboard. She had the cup. The cup was not in the cupboard. (10) Karen pushed the handle away from the cold faucet. The cold water was flowing. The phone was ringing. (11) She moved the cup to the faucet. Lynn picked up the phone receiver. The cup was filled with the water. The phone wasn't ringing. (12) Karen pushed the handle to the faucet. The cold water wasn't flowing. (13) She drank the water. The cup wasn't filled with the water. Karen wasn't thirsty.

Table 2. A rule indicating that picking up a phone receiver results in the phone not ringing.

```
IF  ACT type GRASP
        object PP type RECEIVER
                  component-of ?X
                  (PP type PHONE)
THEN     STATE type RING
                  actor ?X
```

1.1 Representation of examples

To avoid problems of natural language understanding, we will assume that the training examples are in some internal representation. Our learning model will make use of Conceptual Dependency (CD) (Schank & Abelson, 1977), the internal representation used by the Talespin program. A role-filler notation is used to represent CD structures. CD structures have a head and a set of roles and fillers. The filler for a role is a CD structure that must have a head and may have zero or more roles. Common heads are `ACT` (for actions), `STATE` (for unary predicates), `RELATION` (for binary predicates), and `PP` (for physical objects). For `ACT`s, the roles used are `type`, `actor`, `object`, `from`, and `to`. `STATE`s have `type`, `actor`, and `mode` roles. `RELATION`s have `type`, `actor`, `val`, and `mode` roles.[1] `PP`s have a wide variety of roles, such as `type`, `subtype`, `size`, `color`, etc. Every individual `PP` has a special role, `unique-id`, used to indicate the referent in Talespin's world.

In Talespin, each change of the world is indicated by the assertion of a new `STATE` or `RELATION` (indicated by a `mode` of `POS`) or the retraction of a `STATE` or `RELATION` (indicated by a `mode` of `NEG`). `STATES` and `RELATIONS` are assumed to hold for all future time intervals until explicitly retracted (cf. Elkan, 1990).

1.2 Representation of causal rules

The task required by the learner is to predict the state changes that will occur when an action is observed. In order to achieve this task, a set of causal rules is learned. Each rule contains a general description of a class of actions, and a description of a state change. Table 2 illustrates the representation of a simple rule that indicates that a phone stops ringing after the phone receiver is picked up.[2] A question mark before a symbol (e.g., `?X`) indicates that the symbol is a variable. In this rule, the variable is needed to indicate that the phone whose receiver is lifted is the phone that stops ringing. The CD structure after a variable in the

---

[1] The `actor` role of `STATE`s and `actor` and `val` roles of `RELATION`s may be misnomers. This is terminology used by Talespin. A better name for these roles might be `argument1` and `argument2`.

[2] Note that this rule is true in Talespin's world. In the real world, the situation is more complex.

antecedent (i.e., `PP type PHONE`) indicates that the object bound to the variable is constrained to also match that structure.

A variety of tasks could be achieved using rules such as that in Table 2. The rule can be used for planning (specifying an action to perform to achieve the goal of stopping the telephone from ringing) or for abductive inference (infer what action may have occurred to account for a phone that stopped ringing). Here, we will consider only a prediction task. When an action is observed, all state changes that are the consequent of rules whose antecedents match that action will be predicted. There are two types of prediction errors that can be made. Errors of omission occur when a state change is observed but not predicted. Errors of commission occur when a state change is predicted but is not observed.

The learner needs to monitor the accuracy of the causal rules that are acquired. This is implemented by associating a confidence that is the ratio of two counters with each rule. One counter is incremented each time the rule makes a prediction, the other is incremented whenever the rule makes an incorrect prediction. In addition, any exceptions to a rule (i.e., actions on which the rule made an incorrect prediction) are stored in memory indexed by the rule.[3] The exceptions are used only for learning, and not used during prediction.

In order to generate a story, Talespin contains a simulator that determines the effects of actions. This simulator is essentially a large Lisp conditional expression that asserts state changes when an action occurs. The causal rules learned can be viewed as a declarative representation of Talespin's simulator.

1.3 A data-driven learning process for acquiring causal rules

A variety of data-driven (i.e., associative) learning theories have been proposed that can address the problem. For example, DIDO (Scott & Markovitch, 1989), performs a similar task (i.e., determining the effects of its own operators rather than observed actions). With a suitable change of representation, neural network systems (e.g., Sutton, 1988) could also be applied to this problem. Here, we briefly describe a data-driven learning method based upon UNIMEM (Lebowitz, 1987). In our learning model, data-driven learning will be used to form initial causal rules. More general principles of causality will be derived from these rules (as described in Section 3) and used by the theory-driven learning process (described in Section 2).

---

[3] There is a maximum number of exceptions that can be indexed by each rule. In this paper, each rule maintains the five most recent exceptions.

Each time interval is represented by a set of actions and a set of state changes. The learner must acquire rules to indicate which actions are predictive of each state change. An unsupervised learning task is performed at each time interval. First, the observed actions are compared to existing causal rules and a set of state changes are predicted. Next, the predicted state changes are compared to the observed state changes. The confidence and exceptions are updated for those state changes that were correctly predicted and for those state changes that were predicted but not observed. If there are any unpredicted state changes, the learner is run on all pairings of observed actions with unpredicted state changes.

The learner processes each time interval as it is observed. Since this is an unsupervised learning task, the learner must perform two subtasks (Fisher, 1987). First, due to the variety of training examples, the examples must be aggregated into clusters of similar examples. In this work, we rely on a UNIMEM like approach to clustering and concentrate on the second subtask: creating a general description of the aggregated examples. This general description is the rule used to make predictions about new examples.

The first example in a new cluster becomes the initial rule. The rule is constructed by using the action as the antecedent, and the state as the consequent. If the same object (as indicated by its `unique-id`) fills more than one role in the action or the state, a variable is introduced. Variables represent equality constraints. For example, the variable `?x` in Table 2 indicates the phone that stops ringing must be identical to the phone whose receiver is picked up. Whenever a new example is added to a cluster, the rule is generalized by dropping any roles that differ between the rule and the example. In addition, if the example does not conform to an equality constraint, the equality constraint is dropped.

2.0 Background

Theory-driven learning brings additional knowledge to the learning process. Data-driven learners exploit inter-example relationships (i.e., regularities among several training examples). Theory-driven learning has knowledge of intra-example relationships (i.e., constraints between the role fillers of an action and state change). We call these constraints *causal patterns*. The causal patterns warrant the theory-driven learning program to ignore certain regularities between examples. As a consequence, TDL searches a smaller hypothesis space and would be expected to learn accurate rules from fewer examples (provided that the smaller hypothesis space contains an accurate hypothesis).

At least two learning systems have been developed that make use of general causal knowledge to constrain learning. LUCK (Shultz, 1987) makes use of a general theory of causality to identify the cause responsible for an effect. TDL (Pazzani, 1987) learns causal rules that are consistent with a general theory of causality. Pazzani (1991) discusses how *causal patterns* can be used to encode human's general knowledge of causal relationships. In particular, the causal patterns represent spatial and temporal conditions under which human observers report that an action appears to result in a state change.

2.1 Representation of causal patterns

The theory-driven learning procedure can only learn causal rules that conform to one of the causal patterns. Here, we discuss two types of causal patterns:

Exceptionless: An exceptionless causal pattern applies when all of the examples in a cluster are followed by the same state change (or when there is only one example).

Dispositional: A dispositional causal pattern applies when some examples in a cluster are followed by a state change, while others are not. It postulates that a difference in a particular role filler is responsible for the different results.

Table 3 displays one exceptionless causal pattern: an action performed on a component of an object may result in a state change to that object. The causal rule in Table 2 (picking up a phone results in the phone not ringing) conforms to this pattern.

For each exceptionless pattern, there may be a dispositional pattern. A dispositional pattern limits the search for differences between positive and negative examples to one or more roles of the action. Table 4 displays a dispositional causal pattern. In this pattern, there is a single dispositional role: `object`. This pattern indicates that when similar actions performed on an object have different results, and they are performed on different objects, the differing roles of the object are responsible for the different result. This pattern would be useful in creating a rule that describes the difference between the result of a cat knocking over a large clear glass vase during Time 4 in Table 1 and Karen dropping a small red plastic cup in Time 7. Such a rule would indicate that glass objects break when they fall while plastic objects do not. Note

Table 3. An exceptionless causal pattern: An action performed on a component of an object may result in a state change to that object.

CAUSE	`ACT object PP component-of ?O`
EFFECT	`STATE actor ?O`

Table 4. A dispositional causal pattern

CAUSE	`ACT object ?O`
EFFECT	`STATE actor ?O`
DISPOSITION	`object`

that correlation between examples is needed to determine that the composition rather than the color of the object is important. However, this correlation is constrained to the object role. Therefore, TDL will not entertain a hypothesis that indicates that objects break when dropped by cats, but not humans.

2.2 The process of theory driven learning

Pazzani (1990) describes the theory-driven learning process in detail. Here we only provide an overview that is needed to understand how TDL uses new causal patterns acquired through the process described in Section 3. Both the data-driven and theory-driven learning strategies use the same clustering process. Theory-driven learning occurs whenever a new example is added to a cluster, and the current rule did not make a correct prediction. If a causal pattern matches the situation, TDL will form a causal rule that is consistent with both the data and the causal pattern.

2.3 Experiments with TDL

In this section, experiments with TDL in OCCAM are reported. OCCAM is a multi-strategy learning system that includes an explanation-based, a theory-driven and a data-driven learning component. We compare TDL to the data-driven learning algorithm in OCCAM. For consistency with previous descriptions of OCCAM, we will call this data-driven learning algorithm SBL for “similarity-based learning.” We test the following hypotheses:

1. TDL will tolerate more complex training data than SBL. In this experiment, multiple actions (and state changes) will occur at the same time.

2. The TDL and SBL algorithms will degrade gracefully with noisy training data. Here, we consider one type of noise that was created by making the operators in Talespin's world nondeterministic. With a certain probability, an action may have no effect. For example, glass objects may break only 75% of the time they are knocked off a counter onto the floor.

3. A combination of TDL and SBL will tolerate incomplete and incorrect sets of causal patterns. Here we will compare the performance of TDL and SBL combined to the performance of SBL alone. In this combination SBL is run only if TDL fails to find a rule (i.e., no causal pattern matches the experiences).

We will use Talespin to generate test and training examples for these learning algorithms. In this world, actors may have one of three goals (thirsty, hungry or bored) and perform actions to achieve these goals. The thirsty goal is satisfied by drinking a glass of milk or water from the refrigerator, or water from the faucet. Table 1 shows a typical thirsty story. The hungry goal is satisfied by eating a piece of fruit from either the counter or the refrigerator. The bored goal is satisfied by playing catch with either a ball or a balloon. Random actions may occur during any story (cats knocking objects over, phones or door bells ringing, etc.) and actors may have accidents (dropping objects, throwing balls into windows, throwing a balloon into a rose bush). A total of approximately 60 rules are needed to predict the state changes in all of these domains. The fact that 10 causal patterns can facilitate learning indicates that there is a structure in Talespin's world that can be exploited to constrain the learning process.

2.3.1 Simultaneous actions

In most realistic situations, more than one action is occurring at the same time. In order to run this experiment, we modified Talespin so that with a certain probability, P, two contiguous time intervals are merged into one. This merging is done recursively so that, for example, there is a P^2 probability that three contiguous time intervals are merged. Ten runs were made of SBL and TDL on the same randomly generated sequences of Talespin output of 500 time intervals (corresponding to roughly 25 stories). The errors of omission and commission of each algorithm are measured by testing on 600 action-state change pairs after every 20 time intervals for the first 100 examples and every 50 time intervals for the remaining 400 training time intervals.

SBL takes a conservative approach and learns one rule for each action that occurs in the same time interval as a state change. When an error of commission later occurs, the “confidence” in each rule is lowered (by incrementing an exception counter), but the rule is not deleted. Such rules are not deleted, since some actions do not always have the same effect (e.g., sometimes a glass cup does not break when it is dropped). In this experiment, to be more fair to SBL, we will measure errors of commission in two manners. With the first measure, errors of commission will be reported as the percent of total predicted state changes that were not observed. A second measure takes the confidence in the prediction into account by multiplying the total number of predictions and the total number of unsubstantiated predictions by the confidence of the rule that made the prediction.

TDL will not learn a rule if a training example does not conform to a known causal pattern. As a consequence, TDL is predicted to be less sensitive to a complex learning environment in which multiple actions occur simultaneously. Figure 1 shows the mean percentage of errors for SBL and TDL as a function of the number of training examples when P=0.35. The graph in Figure 1 demonstrates that TDL converges on an accurate set of rules more rapidly than SBL.

Insert Figure 1 about here

With this complex training data, TDL has fewer errors of omission and commission than SBL. Taking the confidence of the prediction into account (the line labeled SBL .35 Com' in the figure), lessens the difference between the SBL and TDL. This indicates that the predictions of SBL made with greater confidence are more likely to be correct. To simplify the graph, TDL errors of commission weighted by the confidence is not displayed. TDL typically has higher confidence values, so the two measures of errors of commission are very close in value. The results of this experiment confirm the prediction that TDL will be less sensitive than SBL when multiple actions occur at the same time.

2.3.2 Learning from noisy data

In learning causal rules, we consider one type of noise. With a certain probability, an action will not have any effect. In Talespin's world, this is a natural type of noise. For example, glass objects do not always break when struck and balloons don't always pop when they fall onto a grass lawn. The TDL and SBL algorithms were designed to tolerate training data with this type of noise. In particular, SBL only finds commonalities between examples in which there is a state change. When there is noise in the training data, there are fewer positive examples, so more errors of omission would be expected as the amount of noise increases. Exceptionless causal patterns in TDL operate similarly. However, dispositional patterns attempt to find a role filler (or conjunction of role fillers) that differentiate the actions accompanied by state changes from those that are not. In this case, noise in the training data may cause TDL to blame an incorrect role filler. Since the algorithm attempts to find a role filler that appears in the fewest stored negative examples (rather than a role filler that appears in no negative examples), it is expected to tolerate some amount of noise. Nonetheless, one would predict a greater percentage of errors of commission with increased noise.

Insert Figure 2 about here

In order to test these predictions, an experiment was run in which 20% and 40% of the state changes were deleted randomly from the training data. We measured the accuracy (on noise-free test data) in the same manner as the first experiment. Figure 2 displays the results of these experiments. To avoid clutter, SBL and TDL with 20% noise are not shown for errors of omission. The values typically fell between the algorithm with no noise, and the algorithm with 40% noise.

The graphs show that, as expected, the algorithms degrade gracefully with this type of noise. The algorithms were not intended to deal with other forms of noise (e.g., noise in the role fillers, or state changes sometimes occurring without a cause). We expect that the algorithms would require modification before tolerating other types of noise. However, this noise does not arise naturally in this domain.

2.3.3 Incorrect causal patterns

TDL is one of the learning components of OCCAM (Pazzani, 1990). When a new time interval is observed, if the state change was already expected (i.e., the observed outcome could be predicted by an existing rule), then no learning is necessary. Otherwise, OCCAM first tries EBL, then TDL, and finally SBL. The rationale here is that EBL is expected to produce accurate rules from fewer training examples than TDL, and TDL produces more accurate rules than SBL. Thus, this particular combination of learning methods is intended to maximize the learning rate. In addition, the learning system is intended to be widely applicable, since it can fall back on a data-driven learning algorithm in novel domains. Furthermore, the data-driven and theory-driven learning methods create rules that can be used by the explanation-based methods. Therefore, as OCCAM learns, it can switch from a knowledge-free to a knowledge-intensive learner. In Section 3, we describe how the empirical learning algorithm can learn causal patterns to be used by TDL. Here, we describe an experiment that demonstrates that a combination of TDL and SBL can be used when the causal patterns are incomplete and incorrect. This is particularly important because newly created causal patterns are likely to be incorrect and the system will initially have an incomplete collection of causal patterns.

There are 10 causal patterns used by OCCAM. Six of these patterns were replaced by incorrect versions. Incorrect versions of three patterns were formed by replacing the dispositional roles with other (i.e., irrelevant) roles. Such a change might cause TDL to search for differences between actors while a correct rule has a difference in the object role. If no such difference exists, then SBL will be used. Three other patterns were modified by changing the location of a variable. For example, a pattern requiring the object of an action to be the actor of a state was changed so that the actor of the action was required to be the actor of a state. Such a change would make the pattern match situations that are not causally related and not match sequences that are causally related. In the latter case, SBL will be tried immediately. In the former case, TDL will initially create a rule that makes inaccurate predictions. When further examples are seen, the pattern will no longer be matched (e.g., the equality constraints that were coincidentally true in the initial few examples do not hold in later examples) and SBL will be used to form a new rule.

We ran 10 trials of TDL with these incorrect causal patterns combined with SBL and measured the percentage errors. All time intervals always had exactly one action, and there was no noise in the training data. Runs were made of SBL and TDL on 10 randomly generated sequences of Talespin output of 500 time intervals (corresponding to roughly 25 stories). The errors of omission and commission of each algorithm are measured by testing on 600 action-state change pairs after every 20 examples for the first 100 examples and every 50 examples for the remaining 400 training examples. Figure 3 shows the mean percentage of errors for SBL alone, TDL alone (with correct causal patterns), and TDL with incorrect and incomplete patterns combined with SBL (labeled Bad TDL in the legend) as a function of the number of training examples. In the combined system, approximately 45% of the final rules were learned with TDL and the remainder were learned with SBL. The fact that the combined method has fewer errors of omission than SBL indicates TDL is able to determine when it is not applicable and allows SBL to learn accurate rules. However, the fact that the percentage of errors of commission of the combined methods is higher than that of TDL (with a correct theory) indicates that it takes TDL several examples to determine that an accurate rule cannot be formed that conforms to a causal pattern. This also indicates that the changes made to create the incorrect patterns were not so random that the patterns did not match any training examples.

Whether TDL should always be preferred to SBL in a simple noise-free learning environment depends upon the relative cost of errors of omission and errors of commission. However, in more complex learning environments, TDL exhibits both fewer errors of omission and commission. Furthermore, using the methods in combination, but preferring TDL, yields faster learning rates than just SBL. This combination can be applied even when the causal patterns are incomplete and incorrect.

3.0 Learning causal patterns

The previous section has shown how causal patterns can be represented and used to constrain the hypothesis space. Here, we discuss how causal patterns can be learned from examples. The motivation for this work is to enable TDL to be more easily adapted to new domains. An initial subset of the data is used to acquire general information to apply to the acquisition of specific knowledge in the remainder of the data.

SBL implicitly makes use of a single causal pattern, “if a state change occurs in the same time interval as an action, then the action caused the state change.” This simple causal pattern doesn't contain any constraints between causes and effects. This allows the system to create rules without any more specific causal patterns. By noticing common patterns in established rules, new causal patterns can be created. In the future, hypotheses that conform to one of these patterns are preferred.

3.1 An algorithm for creating causal patterns

We have explored a variety of algorithms for inducing new causal patterns. A number of approaches based upon aggregating rules into clusters of similar rules and forming generalizing among the rules failed. The weak link here was the aggregation algorithm. Although UNIMEM's clustering algorithm produces reasonable clusters of events to generalize to form rules, it did not seem to produce reasonable clusters of rules to generalize to form causal patterns. The rules in such clusters were typically so dissimilar that there was not a generalization in the hypothesis language (conjunctions of role fillers with equality constraints). We also experimented with adapting other clustering algorithms (e.g., COBWEB (Fisher, 1987)) to this task without success (since COBWEB cannot represent equality constraints, and since COBWEB deals with data represented as attribute-value pairs).

The goal of creating causal patterns is to form a general pattern that can be instantiated (by the TDL algorithm) to form a new rule. Here, we describe an “eager” approach to creating new causal patterns. In effect, whenever a rule is created with SBL, the approach determines the most specific causal pattern that could be instantiated to create the same rule with TDL. If the rule learned by SBL is revised, then the pattern is revised.

After forming a rule, an exceptionless causal pattern can be formed by retaining only the equality constraints of the rule (and the heads of the CD structures to make a syntactically correct causal pattern).[4] In addition, a dispositional pattern can be created by adding dispositional roles for every role filler in the rule. Roles explicitly encoding a type hierarchy are not used as dispositional roles, because these hierarchical roles are treated specially by the TDL algorithm. Causal patterns formed in this manner are a deliberate overgeneralization of the training data. Future causal rules that have the same equality constraints as a causal pattern are preferred to those that do not.

An incremental algorithm was implemented to create new causal patterns whenever a rule is created by SBL. If a pattern with the same equality constraints already exists, a new pattern is formed by finding the union of the dispositional roles. A dependency is recorded between each causal pattern and the rule (or rules) that resulted in the formation of the pattern. Whenever a rule is revised, the causal pattern that was formed from the rule is also revised. For example, if a role is dropped from a rule (and no other rule with the same causal pattern uses that role), then a dispositional role is removed from the causal pattern. If an equality constraint is dropped from a rule, and the rule is the only support of a causal pattern, the causal pattern is deleted, and a new causal pattern is formed from the rule (or merged into an existing pattern with the same equality constraints).

Table 5. The initial rule and causal pattern formed from a single example. Those parts that are underlined are deleted when the second example is seen.

Rule

```
IF    ACT type ATRANS
          actor ?A: PP type PERSON
                        age 6
                        gender FEMALE
                        hair BLOND
          object ?B: PP type FRUIT
                        subtype BANANA
                        color YELLOW
          to ?A: PP type PERSON
                        age 6
                        gender FEMALE
                        hair BLOND
          from PP type COUNTER
                  color WHITE
                  composition TILE
THEN  RELATION type POSSES
                  actor ?B
                  val ?A
```

Pattern

```
CAUSE         ACT actor ?a
                  object ?b
                  to ?a
EFFECT        RELATION actor ?b
                           val ?a
DISPOSITION:  actor age
              actor gender
              actor hair
              object color
              to age
              to gender
              to hair
              from color
              from composition
```

---

[4] If there are no equality constraints between the action and a state change, no causal pattern is created. In effect, this algorithm assumes that some object must be involved somehow in both an action and a state change. The algorithm learns additional constraints in terms of which roles of the action can be associated with roles of the state change. TDL takes advantage of these constraints to search a reduced hypothesis space.

An example of creating and revising a causal pattern will help to illustrate the algorithm. Assume the first example seen is “Lynn (age 6, blond hair) took a banana from the counter. Lynn has the banana.” The rule and causal pattern in Table 5 is created. The equality constraints of the rule are preserved in the causal pattern. Every role that has a filler in the rule becomes a dispositional role. For example, the actor's age is present in the rule. Therefore, a disposition role is created to indicate that the age of the actor may be used to distinguish actions that result in a state change from those that do not.

A second similar example, “Mom (brown hair, age 29) gave Karen (age 4, blond hair) a balloon. Karen has the balloon.” causes the rule and pattern to be revised. In the rule, the constraint that the `actor` of the `act`, be the same as the `to` of the `act` and the `val` of the `relation` is dropped. This occurs because in the first example, Lynn is both the actor and the destination, while in the second example, the actor and destination of the action differ. In addition, a number of role-filler constraints are dropped such as the hair color of the `actor` (but not the `to`) and the age of the actor. The revisions to the rule also force revisions to the causal pattern. For example, the variable in the actor role of the cause is removed. In addition, dispositional roles corresponding to constants in the original rule (e.g., the actor's age) are dropped.

The goal of the above algorithm is to create causal patterns by data-driven means from an initial subset of the training data so that theory-driven learning may be used on later parts of the training data. Due to the combination of TDL and SBL, the combined system gradually shifts from using SBL to using TDL to learn new rules. As a result, when the system is trained on one domain, it learns a new domain more quickly, provided that the rules in the new domain conform to the same general patterns as the rules in the old domain.

3.2 Experimental results

We ran an experiment to test whether causal patterns applied in one domain can facilitate learning in a new domain. We ran 10 trials of training the system of 15 randomly generated stories of actor's satisfying the bored goal, followed by 15 randomly generated stories about the hungry goal. Appendix I contains a typical hungry and a typical bored story. The combined system was using TDL, SBL, and creating new causal patterns. Next, all of the rules (but not the causal patterns) were deleted. We then measured the percentage error on 400 training examples of TDL using the induced causal patterns, TDL using the hand-coded causal patterns, and SBL from stories about achieving thirsty goals. This particular training sequence was selected because the thirsty stories are the most complex and varied. The error was measured by testing on 200 randomly generated examples taken from thirsty stories.

Figure 4 shows the percentage error of the three learning systems. TDL with learned rules is called LTDL in this graph. LTDL does not perform as well as TDL. An examination of the patterns learned indicates that some patterns were more specific than hand-coded patterns. In particular, some learned patterns included more equality constraints than the hand-coded ones. If these spurious equality constraints were deleted, several specific rules could be merged into a more general pattern identical to the hand-coded ones. As a consequence, fewer situations will match these specific patterns, resulting in more errors of omission for LTDL. In addition, the examination of the learned causal patterns indicates that they contain more dispositions than the hand-coded rules. As a result, the LTDL relies more on correlation and less on the knowledge encoded in the causal pattern to find differences between positive and negative examples.

The graph does indicate that LTDL has fewer errors of omission than SBL. This demonstrates that the induced rules, like the hand-coded rules, capture regularities between rules that can be used to constrain future learning. Learning causal patterns in one domain accelerates learning in a new domain by enabling OCCAM to use TDL rather than SBL in the new domain.

Insert Figure 4 about here

3.3 The role of Conceptual Dependency in TDL

Theory-driven learning is intended to be independent of the representation of causal patterns and training data. For example, the causal pattern “If an action on an object is accompanied by a state change for the object, then the action results in the state change” might be represented as follows:

$$
\begin{gathered}
\forall a,s,t_a,t_s,l_a,l_s\ (Act(a)\&State(s)\&Time(a,t_a)\\
\&Time(s,t_s)\&(t_s\text{-}\varepsilon < t_a < t_s)\&Loc(a,l_a)\\
\&Loc(s,l_s)\&Near(l_a,l_s))\\
\Rightarrow Result\ (a,s)
\end{gathered}
$$

However, it is well known that the representation of the training data can have a major impact on the speed and accuracy of learning programs. OCCAM makes use of Conceptual Dependency to represent its training examples. Conceptual Dependency was designed primarily to facilitate inference for natural language understanding. The Talespin program was implemented without any thought that a learning program would attempt to learn rules that describe the effects of actions in its world. Therefore, in this research, we have not engineered the representation to suit the needs of the learning program. Nonetheless, there are several important properties of CD that simplify the learning task. Here, we identify some of these properties.

First, CD attempts to be explicit and canonical. For example, the representation of “punch” would be to apply a force, and the object applying the force is the hand of the actor. The representation of kick is to apply a force with the foot. Therefore, if one sees an example in which someone punches something and breaks his hand, and an example where someone kicks something, and breaks his foot, a regularity can be detected easily. However, if instead the training examples were not explicit (e.g., kick did not refer to foot), and canonical (kick and punch were not represented in terms of applying a force), then detecting this regularity is greatly complicated. For example, a poor choice of representation for learning might be:

```
1.kick(john1,wall) & twisted(foot4) & foot_of(john1,foot4,left)
2.punch(bob7,wall)  &  sprained(hand3)  &  hand(hand3)  &  right(hand3)
  part_of(bob7,hand3)
```

Note that the learning task is not impossible. With suitable axioms, sprained and twisted could be related, and the relationship between the actors and the body parts could be identified. However, these tasks will require inference. Furthermore, the clustering process would be complicated greatly. By attempting to be explicit and canonical, CD allows these regularities to be detected and generalized by a straightforward matching process. Learning with such a representation succeeds only when the designer of the representation has foreseen which items need to be explicitly represented so that meaningful generalizations can be made by deleting role fillers and introducing equality constraints.

A second useful property of CD is that a syntactic regularity between two representations implies that there is a semantic relationship between the objects being represented. The roles in CD are intended to have a constrained meaning. In representations without explicit role names, the same effect might be achieved by having a systematic interpretation of the position of arguments to predicates. Without such a system interpretation, there is no reason to believe that there would be regularities between rules, and TDL would have one pattern for each rule. For example, a poor choice of representation for learning would be:

```
1.kick(johnl,wall,foot4) &
  injured(foot4)
2.punch(wall,hand3,bill1) &
  injured(hand3)
3.butt(head6,goat3,wall) &
  injured(head6)
```

Note that, once again, additional knowledge could be supplied to relate these actions via an inference process (e.g., `instrument(head6)`, `instrument(hand3)`, etc.). However, CD allows regularities between rules to be discovered by a simple matching process that finds equality constraints between roles.

3.4 Limitations

Theory-driven learning is intended to be applicable to learning causal rules between simple, overt, physical actions and their immediate effects. In these case, there are constraints on the possible sorts of predictive relationships between actions and state changes. Indeed, we deal with only a limited form of causality in which the effect of an action is immediately apparent (cf. Shoham, 1990). In this class of relationships, the conceptual dependency link, `result` (Schank & Abelson, 1977) is appropriate. More complex causal relationships, involving feedback or many intermediate states (e.g., the relationships between the tax rate and the rate of unemployment) are beyond the scope of TDL. In such a domain, a suitable system of representation for monetary actions has not been worked out, so that it is not possible to express constraints on the relationships between possible monetary actions and their effect. Furthermore, TDL does not capture the sense of "cause" that means "make more likely to occur" as in snow causes automobile accidents.

The algorithm that we have implemented for learning causal patterns has some additional limitations. In particular, it only creates exceptionless and dispositional causal patterns. OCCAM also contains "historical" patterns[5] in which a sequence of events is necessary to achieve a state change (e.g., shaking a can of soda, followed by opening the can results in the soda squirting out). The reason for this limitation is that the data-driven learning algorithm that creates causal rules does not attempt correlations between pairs of actions in different time interval and state changes. A more general instance of this same problem is that TDL is not capable of making predictions about objects with unobservable internal states[6]. In such a case, there may be an arbitrary time interval between an action that changes the internal state of an object and a subsequent action whose effect is dependent on the internal state. Unfortunately, this implies that TDL is limited in its ability to reason about the goals and plans of agents.

4.0 Related work

Theory-driven learning is in some ways similar to a variety of previous work. In particular, the exceptionless causal patterns of TDL are similar to determinations (Davies & Russell, 1987) and rule models (Davis, 1978). The theory-driven learning procedure is similar to SPARC (Dietterich, 1980) in that both procedures create rules by instantiation of skeletal rules (called causal patterns in TDL). Finally, TDL is related to other systems that learn with background knowledge such as META-DENDRAL (Buchanan & Feigenbaum, 1978) and explanation-based learning (EBL) (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986) with overly general domain theories (e.g., Cohen, 1990; Mooney & Ourston, 1989).

---

[5] This is not a problem in our experiments, because historical patterns are not needed to make predictions in Talespin.

[6] Pearl and Verma (1991) discuss an approach to discovering hidden variables. However, it only takes advantage of temporal constraints on causal relationships.

4.1 Determinations

Determination rules have been proposed as a form of knowledge that supports analogical reasoning and justifies why one generalization may be given a great deal of credence and another generalization may be viewed suspiciously although both generalizations may have the same number of positive examples and negative examples. For example, one determination rule states that nationality determines language. This determination allows the generalization that all Americans speak English to be created after encountering a single example of an American speaking English. The generalization that all Americans smoke cigarettes would not be created after encountering a single example of an American smoking a cigarette because there is no determination rule that states that nationality determines smoking behavior. For creating new rules from a single training example, the following form of a determination rule is most useful (Russell & Grosof, 1989):

$$\begin{gathered}\forall yz.\{\exists x.P(x,y)\wedge Q(x,y)\}\Rightarrow\\ \{\forall w.P(w,y)\Rightarrow Q(w,z)\}\end{gathered}$$

For example, the determination rule that nationality determines language would be represented as:

$$\begin{gathered}\forall yz.\{\exists x.Nationality(x,y)\wedge Language(x,y)\}\Rightarrow\\ \{\forall w.Nationality(w,y)\Rightarrow Language(w,z)\}\end{gathered}$$

Causal patterns, like determination rules, are a weaker form of background knowledge than the domain theory of EBL. In particular, the rules learned by EBL follow deductively from the domain theory. With causal patterns and determinations, the rules learned follow from the background knowledge *and* the training examples. However, unlike learning from determinations, TDL does not require that the new rule *deductively* follow from the causal patterns and the examples. Rather, the causal patterns are heuristics that suggest rules subject to empirical validation and refinement. Furthermore, a rule learned by TDL may tolerate exceptions, provided that no refinement of the rule has fewer exceptions.

A procedure for inducing determination rules for binary predicates from training data is described in Russell (1989). In effect, it operates by instantiating *P* and *Q* to binary predicates, finding pairs from the joint domain of *P* and *Q* and calculating how often the determination rule holds. A determination factor, from 0 to 1 is computed, rather than requiring that the determination rule be universally true. This algorithm has been used to demonstrate that interesting and potentially useful determinations exist. However, because the learning algorithm is not incremental, it has not been demonstrated that the acquisition of such determinations from an initial subset of the data facilitate acquiring accurate rules in the remaining data.

4.2 TIERESIAS

TIERESIAS (Davis, 1978) is a system designed to help an expert formulate rules for a rule based expert system. One way it assists an expert is by having rule models. A rule model encodes the type of preconditions typically associated with a rule that makes a particular type of conclusion. For example, rules that identify the category of an organism typically have preconditions describing the site of a culture, an infection, and a portal of entry of an organism. When a new rule is entered, TIERESIAS suggests that it mentions preconditions typically associated with the rule's conclusion. The rule models in TIERESIAS can be created by finding commonalities among rules with similar conclusions. Although it has not been demonstrated, these rule models should be able to provide constraints that facilitate automated learning of new rules.

4.3 SPARC

In some respects, TDL is similar to SPARC (Dietterich, 1980; Dietterich & Michalski, 1986), a system that learns rules that describe patterns in sequential data. SPARC approaches this problem by having abstract, parametrized skeletal rules that can be instantiated to form specific rules. For example, one skeletal rule represents periodic sequences of length *N*. This schema can be instantiated if commonalities are found between examples that are *N* items apart in a sequence. Once a rule is created, it is tested to determine how well it fits the data. TDL also can be viewed as creating new rules by instantiating skeletal rules (i.e., causal patterns). In addition, like SPARC, TDL instantiates a template with values obtained by constrained correlation among training instances. A primary difference between TDL and SPARC is that TDL is accompanied by an algorithm that induces rule templates from training data.

4.4 META-DENDRAL

META-DENDRAL (Buchanan & Feigenbaum, 1978; Buchanan & Mitchell, 1978) is a program that learns cleavage rules to predict which bonds in a molecule will be broken in a mass spectrometer. It starts with a half-order theory that is overly general (i.e., it predicts more bonds will break than actually occur). A program called RULEGEN uses the half-order theory to propose rules that are then tested to see if they are true in many positive examples. Next, a program called RULEMOD refines and revises the rules to insure that few negative examples are covered by a rule. In addition, RULEMOD removes redundant rules.

In TDL, like SPARC, the prior knowledge is abstract knowledge that can be instantiated to form specific rules. In contrast, RULEGEN uses its knowledge in a generate-and-test fashion. It would be possible to use the causal patterns of TDL in a generate-and-test manner. The patterns could generate rules for all combinations of action and state types. These rules would then be tested against the data and incorrect rules deleted. However, by making use of at least one example, the number of rules generated and then tested is considerably reduced.

4.5 EBL with overly general theories

The causal patterns may be viewed as an overly general domain theory. In fact, the algorithm for creating causal patterns deliberately overgeneralizes the data by only including equality constraints. It might be possible to use the overly general domain theory to explain why a particular action resulted in a state change. Then some explanation-based algorithm designed to deal with overly general domain theories could be used to create rules. Here, we discuss how IOU (Mooney & Ourston, 1989) and A-EBL (Cohen, 1990) might approach this problem. IOU (Mooney & Ourston, 1989) operates by first forming a definition via m-EBG (Flann & Dietterich, 1989) for the positive examples. Next, IOU removes any negative examples from the training set that are correctly classified by the results of m-EBG. Finally, IOU deletes those features from the remaining negative and all positive examples, and runs an induction algorithm on the features. The final concept is formed by conjoining the result of induction over the unexplained features with the result of m-EBG. The explanations produced by causal patterns would be overly general explanations. Therefore, the result of m-EBG would typically result in errors of commission.. This result is specialized by an induction process that would eliminate (most of) the errors of commission The primary difference between IOU and TDL is that TDL uses dispositional causal patterns to focus the search for a difference between the positive and negative examples. Since TDL searches a more restricted hypothesis space, one would expect that it would converge on an accurate rule from fewer examples than IOU.

The A-EBL system (Cohen, 1990) is also designed to handle overly general domain theories. It operates by finding all proofs of all positive examples, and uses a greedy set covering algorithm to find a set of operational definitions that cover all positive examples and no negative examples. Unlike IOU, A-EBL will not specialize an operationalized proof to avoid covering any negative examples. A-EBL would not be able to address the problem of learning accurate rules from causal patterns. A-EBL is best suited to those theories that are overly general because the theory has superfluous, incorrect disjunctions. In contrast, causal patterns are overly general because they contain too few preconditions. As a result, no disjunction of the operationalized proofs will exclude the negative examples. Instead, the operationalized proofs need to be specialized by some induction process.

4.6 Multi-strategy learning

There are a variety of approaches to combining multiple learning strategies in an integrated learning system. For example, in Gemini (Danyluk, 1991), an empirical and an analytical learning method have predefined, specific tasks. The result is an integrated strategy in which each learning method has a separate and distinct role. Other systems, such as Meta-AQUA (Cox & Ram, 1991) treat the selection of learning strategies as an additional problem to be solved by the learner. That is, the system reasons about what learning strategy is appropriate for each learning problem.

In OCCAM, there are three learning strategies and each learning strategy can perform the same task (acquiring a predictive relationship). The strategies differ according to the amount of knowledge that they acquire. EBL requires the most detailed, specific knowledge (i.e., a set of causal rules capable of explaining how an action produces a state change). TDL requires more general knowledge of causality and finds causal rules that are instantiations of this more general knowledge. SBL places no such restrictions on the causal rules that are learned. The control strategy of OCCAM is quite simple. It uses the most knowledge-intensive learning strategy that is capable of finding a causal rule to account for an unexpected state change. However, when the less knowledge-intensive strategies are successful, they acquire knowledge that can be used in the future by the more knowledge-intensive strategies. As a consequence, as OCCAM learns, it shifts from data-driven learning methods to knowledge-intensive methods.

In more recent work (Pazzani & Kibler, in press), a tighter integration of learning methods is proposed in which an explanation-based and a data-driven learning algorithm both attempt to produce rules. An information-based evaluation function (Quinlan, 1990), uniformly applied to the hypotheses produced by each method determines which hypothesis to accept. Furthermore, the hypothesis produced by one method may be further refined by either method.

5.0 Conclusions

We have shown how intra-example constraints may be represented and used to constrain the problem of learning a collection of predictive rules. The resulting system converges on accurate concepts more rapidly than a similar system that does not make use of these constraints. Finally, we have shown how these constraints may be discovered in an initial subset of the data, and used to facilitate later learning.

Early work on children's understanding of causality (Piaget, 1930) pointed out differences in causal explanations among various age groups. In spite of more recent evidence (Leslie & Keeble, 1987) that very young infants are able to perceive causal relationships, there is no question that older children are better at attributing causality than younger children. A current research topic in developmental psychology addresses the question of how much of this causal knowledge is innate and how much is learned empirically (Bullock, Gelman, & Baillargeon, 1982; Carey, 1984; Shultz, Fisher, Pratt, & Rulf, 1986; Siegler, 1975). The amount of initial knowledge required to induce causal patterns provides additional evidence in support of the view that much of the general knowledge of causality is learned empirically. In particular, in addition to representational biases, the only initial knowledge of causality required by this computational model is temporal contiguity. From this knowledge, it is possible to learn simple predictive rules. The additional knowledge of causality (e.g., spatial contiguity) is derived from common patterns in the predictive rules. Once learned empirically, this knowledge is available to constrain future learning.

Acknowledgements

This research is supported by a National Science Foundation Grant IRI-8908260 and by the University of California, Irvine through an allocation of computer time. Comments by Kamal Ali and Caroline Ehrlich on an earlier draft of this paper were helpful in improving the presentation.

Appendix I. Typical stories involving hunger and boredom.

Karen was hungry. She asked Mom, “Would you give me the yellow long banana?” Mom picked up it. She had it. The phone was ringing. Dad picked up the receiver. The phone wasn’t ringing. He had the receiver. He pushed the light switch. The light was on. The black cat pushed a large red plastic vase to the tile floor. Mom gave Karen the banana. Karen had it. Mom didn’t have it. Karen peeled it. She ate it. She wasn’t hungry. Karen threw the peel to the basket. She didn’t have the peel. Mom pushed the light switch. The light wasn’t on.

Lynn was bored. Lynn asked Karen, “Would you throw me the balloon?” She asked Mom, “Would you give me the balloon?” Mom pushed the door away from the cupboard. The cupboard was open. The auburn cat pushed a large clear glass vase to the tile floor. The vase was broken. She took the balloon from the cupboard. She had the balloon. The cupboard didn’t have the balloon. She pushed the door to the cupboard. The cupboard wasn’t open. She exhaled into the balloon. It was inflated. Mom picked up the balloon. She had it. She exhaled into it. It was inflated. She let go of it. She didn’t have it. It was flying. It wasn’t inflated. She picked up it. She had it. She exhaled into the balloon. It was inflated. She tied it. It was sealed. She gave Lynn the balloon. Lynn had it. Mom didn’t have it. Lynn went to the outside. Karen went to the outside. Lynn threw Karen the balloon. Karen had it. Lynn didn’t have it. Karen threw Lynn the balloon. Lynn had it. Karen didn’t have it. Lynn threw Karen the balloon. Karen had it. Lynn didn’t have it. Karen dropped the balloon to the green pointed grass. The balloon burst. She didn’t have it.

References

Buchanan B., & Feigenbaum, E. (1978). Dendral and Meta-Dendral: Their applications dimension. *Artificial Intelligence*, *11*, 5-25.

Buchanan. B., & Mitchell, T. (1978). Model-directed learning of production rules. In D. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.

Bullock, M., Gelman, R., & Baillargeon, R. (1982). The development of causal reasoning. In W. Friedman (Ed.), *The developmental psychology of time*. New York: Academic Press.

Carey, S. (1984). *Conceptual change in childhood*. MIT Press.

Cohen, W. (1990). Learning from textbook knowledge: A case study. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 743-748). Boston, MA: Morgan Kaufmann.

Cox, M., & Ram, A. (1991). Using introspective reasoning to select learning strategies. *Multi-strategy Learning Workshop* (pp. 217-230). Harpers Ferry, VA.

Danyluk, A. (1991). Gemini: An integration of analytical and empirical learning. *Multi-strategy Learning Workshop* (pp. 191-206). Harpers Ferry, VA.

Davies, T., & Russell, S. (1987). A logical approach to reasoning by analogy. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 264-270). Milan, Italy: Morgan Kaufmann.

Davis, R. (1978). Knowledge acquisition in rule-based systems: Knowledge about representation as a basis for system construction and maintenance. In D. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternate view. *Machine Learning, 1*, 47-80.

Dietterich, T. (1980). Applying general induction methods to the card game Eleusis. *Proceedings of the First National Conference on Artificial Intelligence* (pp. 218-220). Stanford, CA: Morgan Kaufmann.

Dietterich, T., London, B., Clarkson, K., & Dromey, G. (1982). Learning and inductive inference. In P. Cohen & E. Fiegenbaum (Eds.), *The handbook of artificial intelligence* (Vol. 3). San Mateo, CA: Morgan Kaufmann.

Dietterich, T., & Michalski, R. (1986). Learning to predict sequences. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.

Elkan, C. (1990). Incremental, approximate planning. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 145-150). Boston, MA: Morgan Kaufmann.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2,* 139-172.

Flann, N., & Dietterich, T. (1989). A study of inductive methods for explanation-based learning. *Machine Learning, 4,* 187-226.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning, 2*, 103-138.

Leslie, A., & Keeble, S. (1987). Do six-month-old infants perceive causality? *Cognition, 25*, 265-288.

Meehan. J. (1981). Talespin. In R. Schank & C. Riesbeck (Eds.), *Inside computer understanding: Five programs plus miniatures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. *Machine Learning, 1,* 47-80.

Mooney, R., & Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 5-7). Ithaca, NY: Morgan Kaufmann.

Pazzani, M. (1990). *Creating a memory [illegible]al relationships: An integration of empirical and explanation-based learning metho[illegible]* [illegible]dale, NJ: Lawrence Erlbaum Associates.

Pazzani, M. (1991). A computation[illegible] of learning causal relationships. *Cognitive Science, 15*, 401-424.

Pazzani, M., & Kibler, D. (in pre[illegible]ole of prior knowledge in inductive learning. *Machine Learning*.

Pearl, J., & Verma, T. (1991) A [illegible]nferred causation. *In Proceedings of the Second International Conference on [illegible] of Knowledge Representation and Reasoning*, (pp.

Piaget, J. (1930).

Quinlan, R. (1990).

239–266.

Russell, S. (1989).

Russell, S., & Grosof, B. (1989). Declarative bias: An overview. In P. Benjamin (Ed.),

Schank, R., & Abelson, R. (1977).

Shultz, T. (1987). *Learning and using causal knowledge*. Paper presentation at the Meeting for Research in Child Development. Baltimore, MD.

Siegler, R. S. (1975). Defining the locus of developmental differences in children's causal reasoning. *Journal of Experimental Child Psychology, 20*, 512-525.

Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning 3,* 9-44.

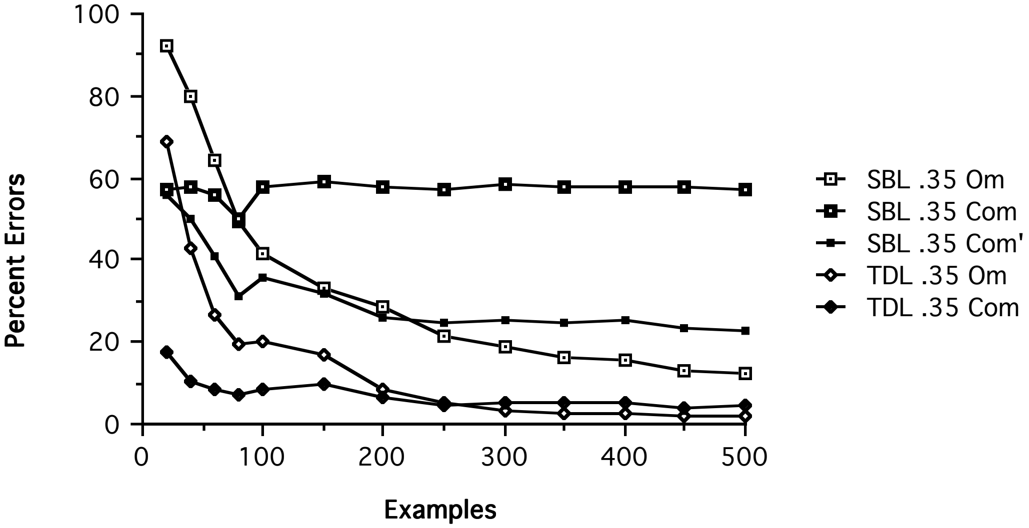

Figure 1. A comparison of TDL and SBL when there is a 0.35 probability that time two actions occur simultaneously. Errors of omission and commission are plotted as a function of the number of training examples. Com' indicates that the errors of commission calculation takes account of the confidence of the prediction.

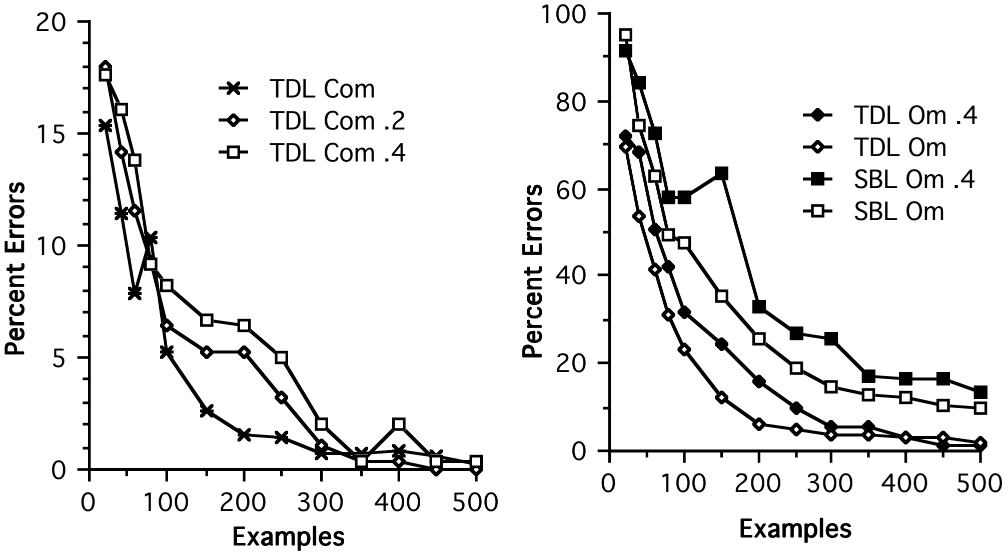

Figure 2. Errors of omission and commission when learning from noisy data as a function of the number of training examples. SBL does not make errors of commission on this data.

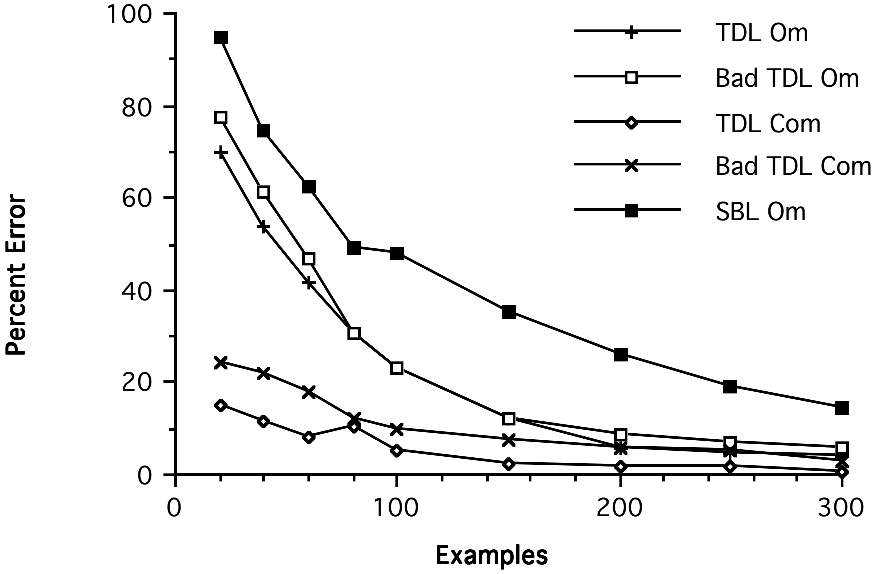

Figure 3. Errors of TDL with incorrect causal patterns (Bad TDL), TDL and SBL as a function of the number of training examples.

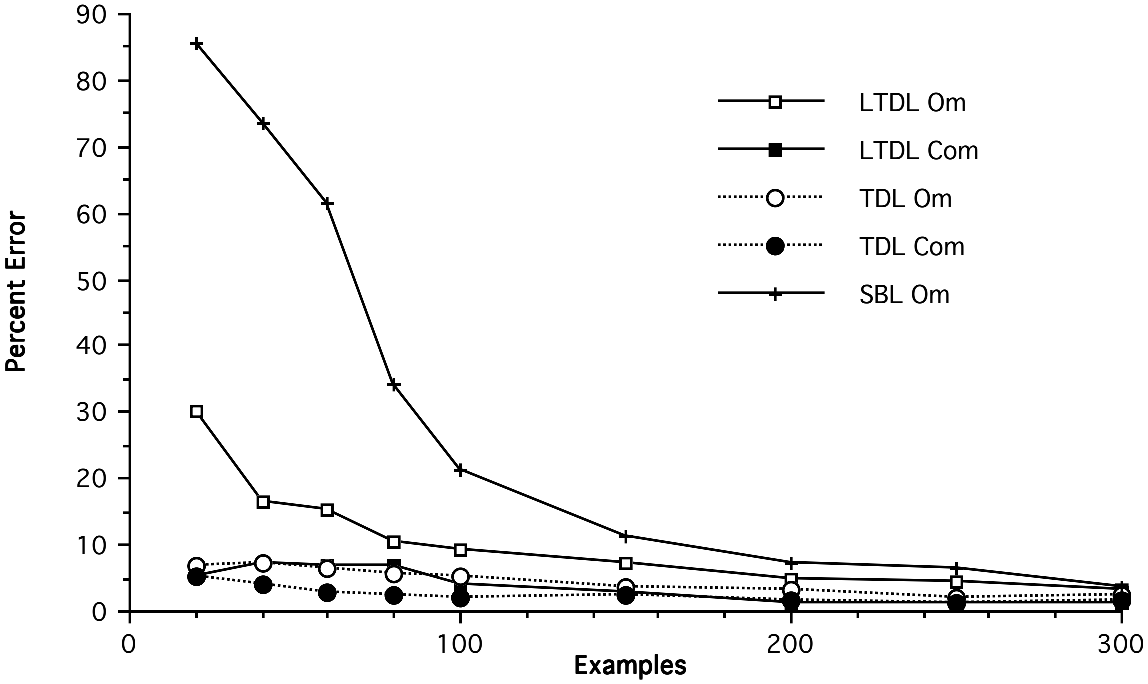

Figure 4: A comparison of SBL, TDL and LTDL (TDL with learned causal patterns).