Theory-Driven Learning: Using intra-example relationships to constrain learning

Michael J. Pazzani
pazzani@ics.uci.edu

Technical Report 91-07

February 6, 1991

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.

Theory-Driven Learning:
Using intra-example relationships to constrain learning

Michael Pazzani
Department of Information and Computer Science
University of California
Irvine, CA 92717
pazzani@ics.uci.edu

Running head: Theory-Driven Learning

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 the learning problem. 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.

1.0 Introduction

Consider the following learning problem: An intelligent agent is in a complex environment. The agent observes actions (i.e., operators) and must learn the effects of these actions and the conditions under which the actions have these effects. In this paper, we consider how intra-example constraints (e.g., knowledge of common patterns of predictive relationships) can be learned and exploited to focus future learning.

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. 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. There are a number of complications in this world. 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 discuss a learning method and a particular type of knowledge that can be exploited to constrain the hypothesis space searched by the learning method. This knowledge consists of intra-example constraints (i.e., constraints between an action and the state changes caused by the actions). We

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.

discuss theory-driven learning (TDL) that makes use of this knowledge. When this knowledge is available, the learning program will be able to create accurate rules from fewer training examples. Finally, we discuss a method of inducing these constraints from the training data and demonstrate that the induced constraints facilitate learning in new domains.

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 system will make use of Conceptual Dependency (CD) (Schank & Abelson, 1977), the internal representation used by the Talespin program.[1]

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 ACTs, the roles used are `type`, `actor`, `object`, `from`, and `to`. STATEs have `type`, `actor`, and `mode` roles. RELATIONS have `type`, `actor`, `val`, and `mode` roles.[2] PPs 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 of PP in Talespin's world.

For ACTs, the `type` role can be filled by one of the Conceptual Dependency actions: ATRANS (transfer of possession), PTRANS (transfer of location), MTRANS (transfer of information), PROPEL (application of force), INGEST (taking something into the body), etc. The `type` roles of STATES are more varied, and include: OPEN, BROKEN, DIRTY, and FLOWING. The `type` roles of RELATIONS include LOCATION and POSSESSION.

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). Table 2 displays the representation used in Time 11 of Table 1 for "Lynn picked up the phone. The phone was not ringing." Note that all role names are in lowercase and the heads of CD structures are in uppercase. All of the roles of a CD structure are indented to the same column.

1.2 Representation of rules

The performance task of the system is to predict the state changes that will occur when an action is observed. In order to achieve this task, a set of rules is learned. Each rule contains a general description

1. Since the learning system and Talespin were designed independently, they use a slightly different syntax for Conceptual Dependency. There is a one-to-one mapping between Talespin's representation and that of the learning system.
2. The `actor` role of STATEs and `actor` and `val` roles of RELATIONs may be misnomers. This is terminology used by Talespin. A better name for these roles might be `argument1` and `argument2`.

Table 2. CD representation for "Lynn picked up the phone. The phone was not ringing."

```
ACT type GRASP
    actor PP type PERSON
              unique-id LYNN
              age 6
              gender FEMALE
              hair BLOND
              complexion FRECKLED
    object PP type RECEIVER
               unique-id PHONE-RECEIVER1
               component-of PP type PHONE
                            unique-id PHONE1
                            color WHITE

STATE type RING
      actor PP type PHONE
                unique-id PHONE1
                color WHITE
      mode NEG
```

of a class of actions, and a description of a state change. Table 3 illustrates a rule that indicates that a phone stops ringing after the phone receiver is picked up.[3] 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 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 3. 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 omission will be reported as percentage of observed states that were not predicted. Errors of commission occur when a state change is predicted but is not observed. Errors of commission will be reported as percentage of predicted states that were not observed.

Table 3. 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
      mode NEG
```

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

Each rule also has two counters associated with it. One is incremented each time the rule makes a prediction, the other is incremented whenever the rule makes an incorrect prediction. In addition, an exceptions list (i.e., the set of actions on which the rule made an incorrect prediction) is also associated with the rule.[4] This information is used only by the learning system, not by the performance system.

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 rules learned can be viewed as a declarative representation of Talespin's simulator.

1.3 An empirical learning algorithm for predicting state changes

A variety of empirical machine learning programs 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. The empirical learning method used by OCCAM also learns predictive rules. All of these algorithms take advantage of correlations between examples to create new rules. However, none of the algorithms make use of intra-example relationships to constrain the learning process. We describe OCCAM's empirical algorithm briefly here, because this algorithm will be compared to theory-driven learning.

Each time interval is represented by a set of actions and a set of state changes. The learning system 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 rules and a set of state changes are predicted. Next, the predicted state changes are compared to the observed state changes. The appropriate counts and exceptions lists are maintained 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 learning system is run on all pairings of observed actions with unpredicted state changes.

Since this is an unsupervised learning task, the learning system must perform two subtasks (Fisher, 1987). First, due to the variety of training examples, the examples must be aggregated into clusters of similar examples. Both TDL and OCCAM's empirical learning algorithm use the same clustering method, based upon UNIMEM (Lebowitz, 1987). In this paper, we 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.

4. A parameter to the system indicates how many exceptions to associate with a rule. In this paper, each rule maintains the five most recent exceptions.

OCCAM's empirical learning algorithm has three learning operators:

Dropping roles: An example can be viewed as a very specific rule (Dietterich, London, Clarkson, & Dromey, 1982). Dropping a role (and the corresponding role filler) makes a rule more general.

Climbing a hierarchy: The representation of PPs explicitly includes hierarchical information through the use of type and subtype roles. Removing these roles can be viewed as climbing a generalization tree. Although the same mechanism implements dropping a role and climbing a hierarchy, it is useful to view them as separate operators since TDL uses them in different ways.

Introducing equality constraints: A variable indicates that two roles must be filled by the same object.

The empirical learning algorithm is run incrementally. 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. Table 4 displays a rule formed from a single example, "Karen took a large clear glass from the cupboard. Karen had the glass." In this example since Karen fills three roles, the variable `?X` is introduced for the `actor` and the `to` of the ACT, and the `val` of the RELATION.

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. Table 5 displays the rule after an additional example is encountered: "Mom gave Lynn a banana. Lynn had the banana." In this example, the `actor` (MOM) differs from the `to` (Lynn). Therefore, the equality constraint is removed between these two roles. However, since the `to` is identical to the `val` of the RELATION, the variable is retained between these two

Table 4. Initial rule formed from "Karen took a large clear glass from the cupboard. Karen had the glass."

```
IF    ACT type ATRANS
          actor ?X: PP type PERSON
                       age 4
                       gender FEMALE
                       hair BLOND
                       complexion DARK
          object ?Y: PP type CUP
                        composition GLASS
                        color CLEAR
                        size LARGE
          to ?X
          from PP type CUPBOARD
THEN    RELATION type POSSESS
                  actor ?Y
                  val ?X
                  mode POS
```

Table 5. A more general rule formed after seeing "Mom gave Lynn a banana. Lynn had the banana."

```
IF    ACT type ATRANS
           actor PP type PERSON
                     gender FEMALE
           object ?Y: PP
           to ?X: PP type PERSON
                      gender FEMALE
                      hair BLOND
           from PP

THEN    RELATION type POSSESS
                  actor ?Y
                  val ?X
                  mode POS
```

roles. In addition, any roles that differ between the rule and the example are removed from the rule. For example, the `age` of the `actor` and the `type` of the `object` are dropped. Table 5 shows the more general rule.

Rules learned in this manner will result in errors of omission if the rule is overly specific. When this occurs, the rule is generalized to accommodate the example. Rules may also result in errors of commission if more than two actions and two unexpected state changes occur in the same time interval. For example, in Time 11 of Table 1, if both the cup being filled and the phone not ringing are unexpected, a rule may be learned that indicates that moving a small, clear glass cup to the cold water faucet results in the phone not ringing. When errors of commission are detected, a counter is incremented that reduces confidence in the rule (see Section 1.2).

2.0 Theory-driven learning

Theory-driven learning brings additional knowledge to the learning process. Empirical learning programs 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).

Pazzani (in press) discusses how the causal patterns of OCCAM 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. Here, we concentrate on how similar knowledge constrains the learning process and how this knowledge may be induced from examples.

2.1 Representation of causal patterns

The theory-driven learning procedure can only learn rules that conform to one of the causal patterns. There are three types of causal patterns in OCCAM:

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.

Historical: A historical causal pattern is applied when some examples in a cluster are followed by a state change, while others are not, and when no dispositional pattern can account for the difference. It postulates that some prior action is responsible for a state change that enables the current action to result in a state change. Because Talespin does not contain any rules that require historical causal patterns, we will not elaborate on them here. Pazzani (1990) gives examples of historical patterns being run on hand-coded training data.

Table 6 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 3 (picking up a phone results in the phone not ringing) conforms to this pattern. Patterns such as these provide two important constraints for theory-driven learning. First, the learning program can ignore actions and state changes occurring in the same time interval if the action and state change do not conform to one of the causal patterns. For example, in Time 11 of Table 1, moving a small, clear glass cup to the cold water faucet occurs at the same time a phone stops ringing. Since no causal pattern matches this situation, theory-driven learning will not learn a rule predicting that the phone will stop ringing in this situation. Second, the causal patterns warrant theory-driven learning to ignore some regularities in the training data. Since the pattern in Table 6 does not mention the actor of the action, theory-driven learning ignores all similarities between actors when producing a rule to indicate when the phone will stop ringing.

For each exceptionless patterns, there may be a dispositional pattern. A dispositional pattern limits the search for difference between positive and negative examples to one or more roles of the action. Table 7 displays a dispositional causal pattern. In this pattern, there is a single dispositional role: `object`. This

Table 6. 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 7. A dispositional causal pattern

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

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 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.

Appendix I contains a complete list of the causal patterns used by OCCAM in Talespin. This is similar to the list reported in Pazzani (1990). However, the earlier version of OCCAM did not distinguish states from relations and included some historical patterns.

2.2 The theory-driven learning algorithm

Theory-driven learning takes as input a set of causal patterns and a sequence of time intervals (consisting of actions and state changes). It produces a collection of rules that can be used to predict state changes. The theory-driven learning routine is invoked whenever a new example is added to a cluster, and the current rule did not make a correct prediction. Each cluster maintains the generalization formed by empirical learning from the training examples. The result of TDL can be more general than this generalization, but not more specific.

Causal patterns are ordered by their type. Exceptionless patterns are tried first, dispositional are tried next, and historical are tried last. The rationale here is the rules produced by dispositional patterns are simpler than those produced by historical patterns (because historical patterns require more than one action to predict a state change). Similarly, exceptionless patterns result in simpler rules than dispositional patterns. If more than one pattern of the same type applies in a situation, one is used randomly.

Causal patterns are applied by matching the cause of a pattern against the generalized action of a cluster and the effect against the generalized state change. If the pattern matches, a new rule is formed by instantiation. The instantiation process depends upon the type of causal pattern:

Exceptionless: The type and mode roles of the action and state change are retained. In addition the typing information of any object bound to a variable is retained. All

other roles of the state change and action are removed. The only equality constraints created are those that are present in the causal pattern.

Dispositional: Two sets of observations are created: the positive examples are those examples in the cluster that are associated with the same type of state change. The negative examples are similar actions that are not associated with the same type of state change.[5] Dispositional causal patterns restrict the search for a condition that differs between the positive and negative examples to an object that plays a specified role (indicated by the disposition facet of the causal pattern in the tables) in the action. The following algorithm is used to find a role filler for the role indicated by the disposition. TDL creates a rule that indicates that this role filler is necessary for the action to result in the state change. To avoid confusion, the term *role* will be used to describe a top-level role of the action. The roles of the role filler of this role will be called *subordinate attributes.*

1. The subordinate attributes common to all objects that play this role in all positive examples are collected. If there are no common subordinate attributes, the pattern does not apply.

2. The number of negative examples that have the conjunction of these common subordinate attributes is calculated. The number of negative examples that have each common subordinate attribute is also calculated. If any individual subordinate attribute is present in the same number of negative examples as the conjunction of all subordinate attributes, then that subordinate attribute is deemed responsible. If several subordinate attributes each cover the same number of negative examples, one is selected randomly. If the conjunction of subordinate attributes is found in fewer negative examples than any individual attribute, then the conjunction is used as the role filler.

The rule is created in a manner identical to the exceptionless patterns except it also contains a filler for the dispositional role that was hypothesized to be responsible for the different state change.

Historical: The causal pattern indicates the causal relationship is conditionally dependent on some previous action. These patterns are processed in a manner similar to the dispositional pattern except that a prior action instead of a role is blamed for the different state change.

5. The negative examples are stored as the exceptions to the rule (see Section 1.2).

Note that like SBL, TDL climbs the type hierarchy to find the most specific class of objects involved in a state change. However, TDL does not retain all role fillers of the positive examples. Rather, it only includes a role filler if it is useful in distinguishing positive from negative examples. The goal of the dispositional learning algorithm is to find a role that is true in all of the positive examples, and not true in many (or all) of the negative examples. The search for this difference is constrained to a particular role, indicated by the causal pattern. The algorithm prefers single attribute discriminations to conjunctive descriptions, provided the single attribute discrimination is not less accurate.

An example will help to clarify this algorithm. Assume that there are two positive examples seen so far. "Lynn dropped a small clear glass cup and it broke" and "the female cat knocked over a small red glass vase and it broke." TDL would use an exceptionless pattern to create a rule that states "applying a force to a physical object results in the object breaking." In addition, a general description of all positive examples is retained to summarize the positive examples of the cluster: "a female applying a force to a small glass object results in the object breaking". When a new example is seen, "Dad dropped a large red plastic container (and it didn't break),"[6] a new rule is learned using the dispositional causal pattern from Table 7. To find a relevant difference between the prior positive examples and the negative one, the `object` of the generalized positive examples (i.e., small glass object) is found in Step 1 of this algorithm. In Step 2, the frequency that this conjunction is present in the negative examples (`size = small & composition = glass:` 0) is calculated, as well as the frequency of the individual subordinate attributes (`size = small:` 0; `composition = glass` 0). One of the single subordinate attributes of the object is selected at random (e.g., `composition = glass`) to form a rule that indicates "applying a force to a glass object results in the object breaking." Note that in this example, the hypothesis "a female applying a force to any object results in the object breaking" is consistent with the training data. However, TDL doesn't consider such a hypothesis since the causal pattern indicates that the features of the actor are not important.

2.3 Experiments with theory-driven learning

In this section, experiments with TDL in OCCAM are reported. We compare TDL to the empirical learning algorithm in OCCAM. We will call this empirical learning algorithm SBL. We test the following hypotheses:

1. TDL will converge to an accurate rule from fewer training examples than SBL. The rationale for this prediction is that TDL searches a smaller hypothesis space than SBL and this restricted hypothesis space includes the correct hypothesis.
2. TDL will learn more rapidly than SBL when there are many irrelevant roles.
3. TDL will tolerate more complex training data than SBL. In this experiment, multiple actions (and state changes) will occur at the same time.

6. The glass not breaking is not explicitly part of the representation. Rather the action occurred followed by no state change. Note that the processing with respect to this rule would be identical if the representation also included "... and the liquid in the glass spilled." In addition, a rule for spilling might be learned or revised.

4. 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.
5. 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).

In the second experiment, we will measure the accuracy of TDL and SBL learning a single rule. In the remaining tests, 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. In addition, 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). Appendix II contains the English representation of typical bored and hungry stories. 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 Comparing the accuracy of TDL and SBL

The first experiment is the simplest. All time intervals always have exactly one action, and there is no noise in the training data. Runs are 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 1 shows the mean percentage of errors for these two algorithms as a function of the number of training examples.

The graph in Figure 1 demonstrates that TDL converges on an accurate set of rules more rapidly than SBL. The rules that TDL learns are usually more general than the rules learned by SBL from this training data. However, these rules are occasionally too general. As a consequence, TDL results in some errors of commission from this training data. In contrast, SBL is a "one-sided" algorithm that creates the maximally specific hypothesis warranted by the training data. Therefore, no errors of commission occur for SBL.

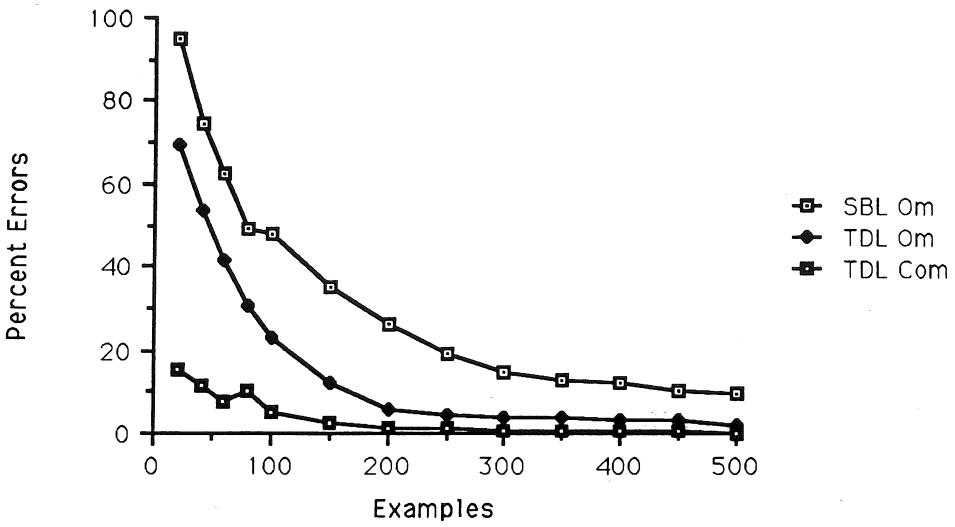

Figure 1. Errors of SBL and TDL as a function of the number of training examples. Errors of omission are shown for both SBL and TDL. Errors of commission are shown only for TDL because SBL does not make any errors of omission on this data.

2.3.2 Sensitivity to irrelevant roles and irrelevant subordinate attributes

One way to quantify the improvement of TDL over SBL is to compare the number of examples each approach requires to find a single rule that makes accurate predictions. The knowledge encoded in the causal patterns warrants TDL to discount irrelevant roles included in the description of examples. The causal patterns of TDL focus the search for similarities and differences on those roles that normally play a part in the causal relationship. SBL is not constrained by the theory of causality, so that it must correlate over additional roles.

In this experiment, it is useful to distinguish between top-level roles and subordinate attributes. Top-level roles describe the roles that objects play in an action or a state change (e.g., actor or object). Subordinate attributes refer to roles of objects (e.g., age or color). We recorded the number of observations required by SBL and TDL to learn the rule "when a glass object is struck, the object breaks." When there were no irrelevant roles, the systems are presented with actions that include only the composition and type of the object. Additional irrelevant top-level roles are descriptions of the actor, location, the time, and some randomly generated role names. We ran trials in which there were 0, 4 and 10 irrelevant roles used to describe each activity. For each of these three conditions, SBL and TDL were run on problems in which each top-level role was described by 3, 5, 10, 20 and 40 irrelevant subordinate attributes. The results of the experiment are illustrated in Figure 2. The value shown for each point is the number of examples (averaged over 25 runs) required to learn an accurate rule (i.e., one that makes at most 5% errors of omission and no errors of commission on a set of 100 randomly generated test examples) as a function of the number of attributes used to describe each top-level attribute.

As expected, the performance of TDL was not dependent on the number of top-level irrelevant roles used to describe the input activities. The graph only lists TDL with 10 irrelevant roles because the result does not differ substantially[7] from TDL with 4 or 0 irrelevant top-level attributes. The graph clearly shows the advantage that TDL has over SBL when the data conform to a known causal pattern. If the consequence of acting upon an incorrect hypothesis are important, or if there are few examples, TDL is a significant improvement over SBL. The causal patterns are an explicit form of bias that focuses the search for hypotheses.

2.3.3 Simultaneous actions

In most realistic situations, more than one action is occurring at the same time. The learning task in this experiment is more complex than that studied in the previous two experiments because the learner must try to determine which action is responsible for each state change. 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.

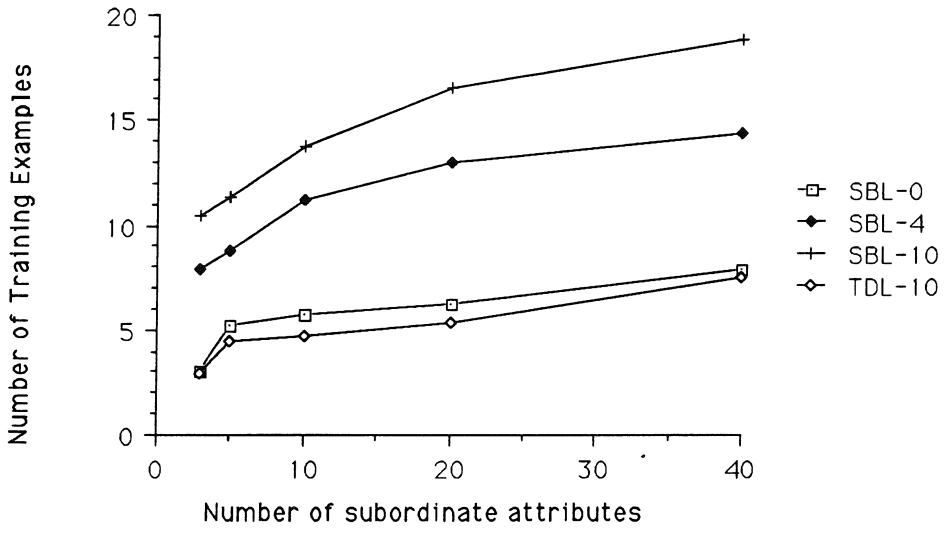

Figure 2. The mean number of examples required for similarity-based learning and theory-driven learning to find a hypothesis that is at least 95% accurate as a function of the number of irrelevant subordinate attributes. SBL is shown with 0, 4 and 10 irrelevant top-level roles; TDL is shown only with 10 irrelevant top-level roles.

7. Increasing the number of top-level attributes does increase the number of causal patterns that may be matched and the expense of matching. However, these effects did not increase the learning rate in this experiment.

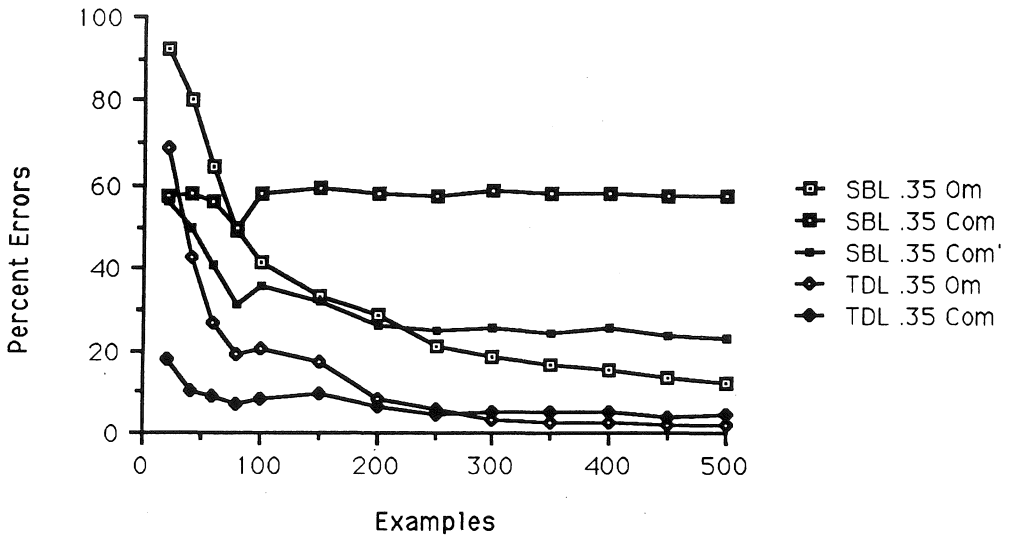

Figure 3. A comparison of TDL and SBL when there is a 0.35 probability that time two actions occur simultaneously. Errors of omission and commission are plotted as a function of the number of training examples. Com' indicates that the errors of commission calculation takes account of the confidence of the prediction.

SBL takes a conservative approach and learns one rule for each action that occurs in the same time interval. 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 3 compares SBL and TDL when P=0.35. The test and training procedure are identical to that of the first experiment.

With this more 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.

2.3.4 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.

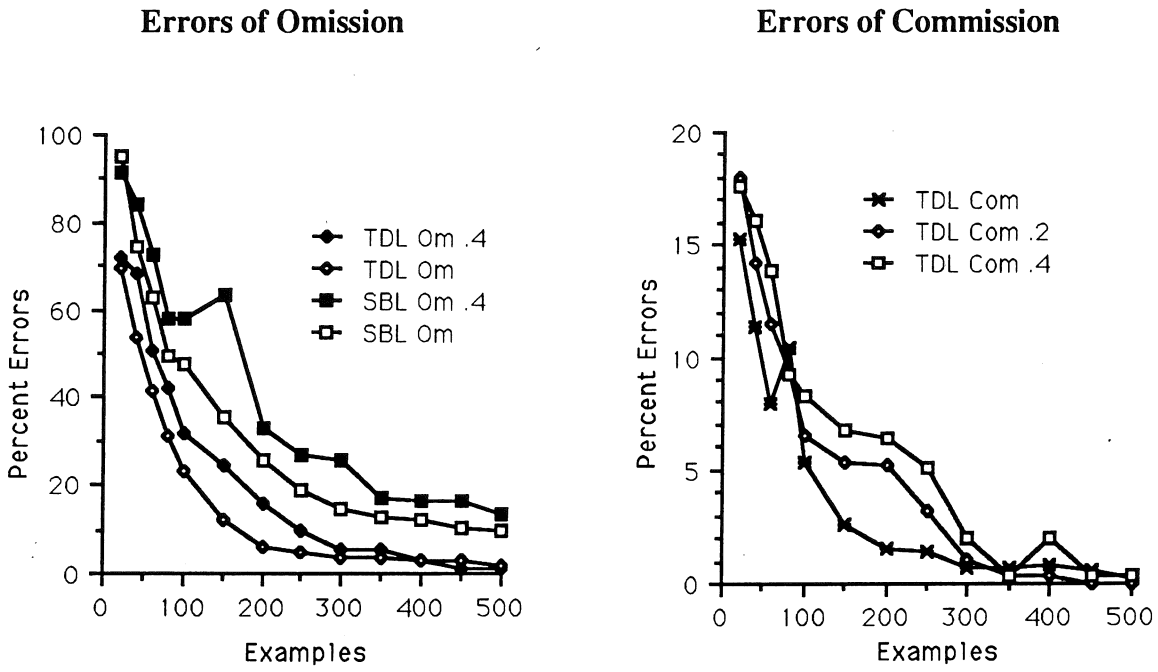

Figure 4. Errors of omission and commission when learning from noisy data as a function of the number of training examples. SBl does not make errors of commission on this data.

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 4 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.

2.3.5 Incorrect causal patterns

TDL is one of the learning components of OCCAM (Pazzani, 1990), a system that combines explanation-based, theory-driven and empirical learning algorithms. 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 an empirical learning algorithm in novel domains. Furthermore, the empirical 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 than demonstrates that a combination of TDL and SBL can be used when the causal patterns are incomplete and incorrect.

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.

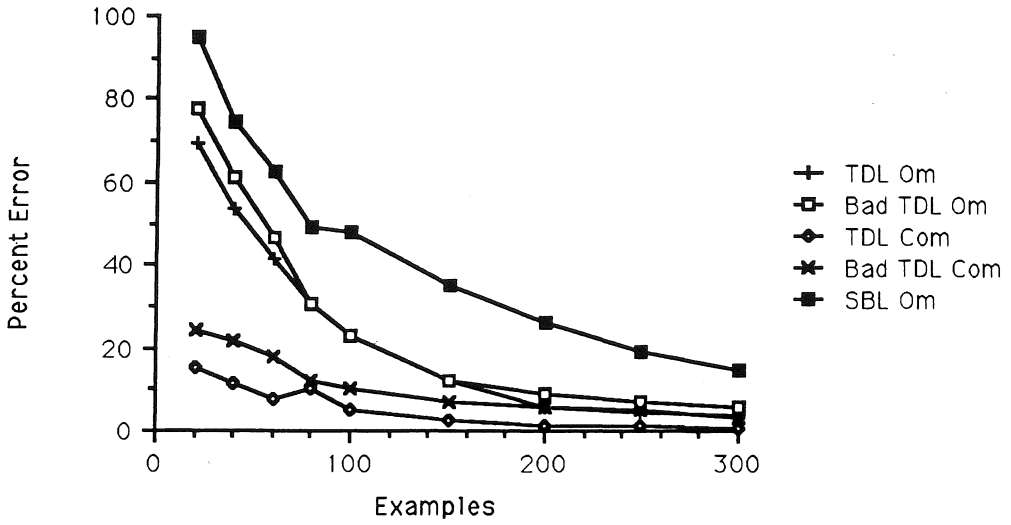

Figure 5. Errors of TDL with incorrect causal patterns (Bad TDL) as a function of the number of training examples. The data from the first experiment for SBL and TDL is also shown to facilitate comparison.

We ran 10 trials of TDL with these incorrect causal patterns combined with SBL and measured the percentage errors in the same manner as the first experiment. In these runs, approximately 45% of the final rules were learned with TDL and the remainder were learned with SBL. Figure 5 plots the errors as a function of the number of training examples. The number of training examples graphed terminates at 300 rather than 500 because the algorithms are so similar as to be indistinguishable past this point. The data from the first experiment is repeated to show the relationship between the combined learning method with incorrect patterns, and the individual methods. 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. The technique that we propose here for creating such general patterns is very straightforward. When a rule is formed by SBL, it is formed by climbing a generalization hierarchy, dropping features and adding equality constraints. 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).[8] 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 over generalization of the training data. Future rules that have the same equality constraints as a causal pattern are preferred to those that do not.

8. 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.

Table 8. 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	Pattern
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	**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
THEN RELATION type POSSES actor ?B val ?A mode POS	

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 pointer is retained from each causal pattern to 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).

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 8 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 patterned 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 empirical 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 demonstrate how 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. 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 6 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.

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:

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

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.

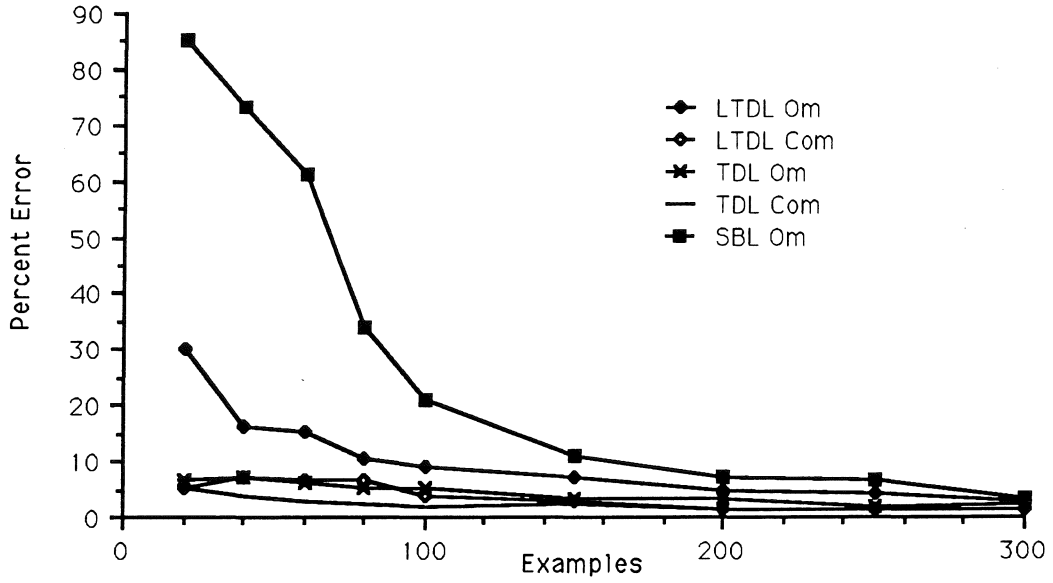

Figure 6. Errors of SBL, TDL and LTDL (TDL with learned causal patterns) as a function of the number of training examples.

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) &
   part_of(bob7,hand3) & right(hand3)
```

Note 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(john1,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.

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 over general domain theories (e.g., Cohen, 1990; Mooney & Ourston, 1989).

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):

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

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

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

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), 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 generalized 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 review how IOU (Mooney & Ourston, 1989) and A-EBL (Cohen, 1990) would 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.

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 debate 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.

Bibliography

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, *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.

Davies, T., & Russell, S. (1987). A logical approach to reasoning by analogy. *Proceedings of the 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). Los Altos, 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. J. (1990). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods.* Hillsdale, NJ: Lawrence Erlbaum.

Pazzani, M. (in press). A computational theory of learning causal relationships. *Cognitive Science.*

Piaget, J. (1930). *The child's conception of physical causality.* London: Kegan Paul.

Russell, S. (1989). Analogical and inductive reasoning. London: Pittman Press.

Russell, S., & Grosof, B. (1989). Declarative bias: An overview. In P. Benjamin, (Ed.) *Change of representation and inductive bias.* Norwell, MA: Kluwer Academic Press.

Schank, R., & Abelson, R. (1977). *Scripts, plans, goals, and understanding.* Hillsdale, NJ: Lawrence Erlbaum.

Scott, P., & Markovitch, S. (1989). Learning novel domains through curiosity and conjecture. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 669-674). Detroit, MI: Morgan Kaufmann.

Shultz, T., Fisher, G., Pratt, C., and Rulf, S. (1986). Selection of causal rules. *Child Development, 57,* 143-152.

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.

APPENDIX I: OCCAM's causal patterns

```
CAUSE       ACT object ?O
              to ?T
EFFECT      RELATION actor ?O
                    val ?T
DISPOSITIONS to

CAUSE       ACT actor ?A
                object ?O
EFFECT      RELATION actor ?O
                    val ?A

CAUSE       ACT object ?O
                from ?F
EFFECT      RELATION actor ?O
                    val ?F
DISPOSITIONS to

CAUSE       ACT object ?O
                from ?F
EFFECT      RELATION actor ?F
                    val ?O

CAUSE       ACT to ?A
EFFECT      RELATION actor ?A

CAUSE       ACT object ?A
EFFECT      RELATION actor ?A
DISPOSITIONS to

CAUSE       ACT object ?O
EFFECT      STATE actor ?O
DISPOSITIONS to
                object
                from

CAUSE       ACT object PP component-of ?O
EFFECT      STATE actor ?O
DISPOSITIONS to
                from
                object component-of

CAUSE       ACT from ?F
EFFECT      STATE actor ?F

CAUSE       ACT to ?TO
EFFECT      STATE actor ?TO
DISPOSITIONS object
                to
```

APPENDIX II: Examples of Talespin Output for Hungry and Bored Stories

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 was bursted. She didn't have it.