IN-63-TM
91520
P. 8

The Blind Leading the Blind: Mutual Refinement of Approximate Theories

Smadar T. Kedar
John L. Bresina
Sterling Federal Systems
AI Research Branch, Mail Stop 244-17
NASA Ames Research Center
Moffett Field, CA 94035

C. Lisa Dent
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA-91-09

April, 1991

The Blind Leading the Blind: Mutual Refinement of Approximate Theories

Smadar T. Kedar
Sterling Federal Systems
NASA Ames, M/S: 244-17
Moffett Field, CA 94035

John L. Bresina
Sterling Federal Systems
NASA Ames, M/S: 244-17
Moffett Field, CA 94035

C. Lisa Dent
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

April, 1991

Abstract

We describe *mutual theory refinement*, a method for refining world models in a reactive system. The method detects failures, explains their causes, and repairs the approximate models which caused the failures. Our approach focuses on using one approximate model to refine another.

This paper appears in *The Proceedings of the Eighth International Workshop on Machine Learning* (held at Northwestern University, Evanston, IL on June 27, 28, 29).

The Blind Leading the Blind: Mutual Refinement of Approximate Theories

Smadar T. Kedar
Sterling Federal Systems
NASA Ames, M/S: 244-17
Moffett Field, CA 94035

John L. Bresina*
Sterling Federal Systems
NASA Ames, M/S: 244-17
Moffett Field, CA 94035

C. Lisa Dent
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We describe *mutual theory refinement*, a method for refining world models in a reactive system. The method detects failures, explains their causes, and repairs the approximate models which caused the failures. Our approach focuses on using one approximate model to refine another.

1. Introduction

The world model guiding a reactive system is always approximate. Thus even the most carefully coded system will occasionally fail. Our long-term objective is to enable a reactive system to learn – from its failures – refinements of its world model. In this paper, we describe interim results on an incremental learning method, *mutual theory refinement*. The method detects failures, explains their causes, and repairs the approximate models which caused the failures. The learning is incorporated into a reactive system, the Entropy Reduction Engine (Bresina & Drummond, 1990; Drummond, *et al.*, 1991). Our approach focuses on using one approximate model to refine another approximate model. It can also determine when the approximate model is not sufficient to explain the failure, and degrade gracefully, resorting to inductive or rote learning. This is accomplished by exploiting two common features of knowledge-based reactive systems: (*i*) multiple related approximate models whose underlying principles overlap, and (*ii*) multiple sources of experience (*e.g.*, planning and reaction) which, when compared, provide a strong basis for failure detection and explanation.

A knowledge-based reactive system initially has models of the world and actions, however approximate. If that knowledge exists, why not exploit it for learning? Our work follows the key idea: use knowledge when you can, yet recognize when you cannot use it. We are therefore exploring an analytic end of the learning spectrum, while addressing the problem of how to detect the limits of the knowledge and fall back on inductive methods. Our method builds on earlier work in explanation-based learning (EBL) from failure for refining approximate theories (Mostow & Bhatnagar, 1987; Tadepalli, 1989; Chien, 1989; Gupta, 1987). Most of these methods assume a complete and correct theory is available to fix the approximate one. In contrast, we refine one approximate theory with another, and recognize the limitations of either.

We first present some features of our performance system, the Entropy Reduction Engine. Next, we describe the learning method and illustrate it using the NASA TileWorld experimental domain. We then compare and contrast our approach with other work, closing with a discussion of our future plans.

2. Background

Reactive systems are situated in an environment in which they sense and act. Our work is cast within one such system, the Entropy Reduction Engine (ERE). Unlike systems which consist of hand-coded reactions tailored to a particular task (notably Brooks' (1986) *subsumption architecture*), the ERE architecture uses planning and scheduling to *automatically* synthesize reactions appropriate to the given goals and environment. Briefly, this synthesis is accomplished as follows (Drummond & Bresina, 1990). First, a planning component, called the *projector*, performs chronological search through the space of possible world model states to select an operator sequence that satisfies the given goal. Then, using goal regression, the operator sequence is compiled into *Situated Control Rules (SCRs)*; each rule specifies the appropriate action to take in a given situation to satisfy the desired goal. These SCRs are used as advice by the execution component, called the *reactor*.

The projector uses two approximate models: *operators* and *domain constraints* (Drummond, 1986). The operators model the agent's actions in terms of preconditions and effects. The effects can be specified as

*Also affiliated with Rutgers University.

a set of nondeterministic *variant outcomes*, each associated with the probability that the variant outcome will result from execution of the operator. Domain constraints model physical laws as sets of facts which can never co-occur in a world state; *e.g.*, "the agent cannot be in two locations at once". These two models capture some of the same underlying principles from different perspectives; hence, each can serve as a basis to refine the other.

Note that, unlike STRIPS operators, ERE operators specify only what is added, not what is deleted. The facts in the pre-state that should be deleted from the post-state are those that contradict the operator's effects. These contradictions are detected using the domain constraints. For example, when the agent moves to a new cell, the above domain constraint indicates that the agent cannot also be in the old cell, so that (old) fact is deleted.

The current testbed used in building and testing ERE is the NASA TileWorld experimental domain. It consists of a simulator of a single agent in a two-dimensional grid of cells, able to move and grasp tiles.

3. Mutual Theory Refinement

Given the hand-coded operators and domain constraints, the system may occasionally experience a failure: the expected outcomes do not match the observed outcomes. Let WM be a *world model* representing aspects of the world. WM is *approximate* if it is incomplete and/or incorrect in its representation. A *prediction failure* in WM for a reactive system is a discrepancy between the *predicted state* resulting from planning in WM and the *observed post-state* resulting from reaction in the world. If there is no discrepancy, it is a *correct prediction*. WM′ is a *refinement* of WM with respect to a prediction failure if planning with WM′ now results in a correct prediction. The *mutual theory refinement problem* is: given a set $\{WM_i\}$ of approximate models, and a prediction failure, find a refinement $\{WM_i'\}$ of $\{WM_i\}$. The *mutual theory refinement method* follows three steps. It **detects** a prediction failure, then uses some models to **explain** the cause of the failure (due to approximations in other models). It then **repairs** the failure producing a refinement $\{WM_i'\}$, resulting in a correct prediction. A model WM becomes *increasingly correct* (Mitchell, 1990) with respect to the world over time if there is a decrease in prediction failures in WM. The goal of the learning is to produce increasingly correct models.

Given a prediction failure, how does the method determine which of the approximate models are causing the failure, which models can be used to make the refinement, and which refinement can be made? Prediction failures for ERE may result from one or more of the following: missing or extra (over-general or over-specific) preconditions; missing or extra (over-specific or over-general) variant outcomes; incorrect preconditions or outcomes; missing or incorrect domain constraints. We have focused thus far only on refinements of incomplete, rather than incorrect, models (missing preconditions, missing variant outcomes, or missing domain constraints). Our method draws on a catalog of heuristics which determine, for each type of incompleteness, what models can be used to make the refinement, and what refinement can be made. We envision the refinement process as incremental and plausible, not necessarily correct. All refinements are annotated and may be revised if further relevant information becomes available. The degree of confidence in a particular refinement depends both on the degree of confidence in the knowledge used in the refinement, and on whether the refinement is inductive or analytic.

4. Operator Refinement

In this section we describe methods for detecting and repairing missing preconditions and missing variant outcomes, illustrated with a simple TileWorld example. For these types of incompletenesses in the approximate operator model, the method attempts to use the approximate model of domain constraints as a basis for refinement. The initial recommended repair is to use an explanation of the failure derived from the domain constraints to add missing preconditions. If the domain constraints are insufficient to explain the failure, then the recommended repair is to use inductive methods to add a missing variant outcome.

4.1 Missing Preconditions

Consider a MOVE operator which describes the agent moving in some direction while grasping a tile in some other direction. Suppose that the preconditions test whether the destination cell of the agent is empty, but *not* whether the destination cell of the grasped tile is empty (*i.e.*, a "cell-empty" precondition is missing).[1] The operator's single variant outcome specifies that both the agent and the grasped tile will end up in their destination cells. The initial *faulty* operator definition is the following:

```
(defop :name MOVE(?dir)
  :preconditions
    (agent-location (?x ?y))
    (grasping ?dir2 ?t)
    (tile-location ?t (?x2 ?y2))
    (cell-adjacent (?x ?y) ?dir (?x1 ?y1))
    (cell-adjacent (?x2 ?y2) ?dir2 (?x3 ?y3))
    (cell-empty (?x1 ?y1))
  :variant outcome :name straight :prob 1.0
    (agent-location (?x1 ?y1))
    (tile-location ?t (?x3 ?y3))
    (cell-empty (?x ?y))
    (cell-empty (?x2 ?y2))
```

[1] Similar errors of omission have actually occured.

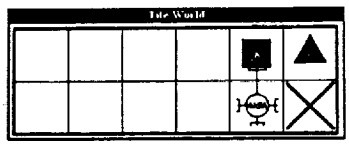

Figure 1: Failure while moving a tile

The first step **detects** failure when the observed post-state differs from all predicted states (one for each variant outcome in the operator). For example, suppose the agent is attempting to move to the right while grasping a square above, but there is a triangle next to the square (see Figure 1). Since MOVE's preconditions do not require the destination cell of the grasped tile to be empty, projection predicts that the agent and square will move right. However, when the reactor attempts to execute the move, it is prevented from doing so by the physics of the TileWorld simulator. Thus, the agent and square remain where they were in the previous state. Hence, the predicted post-state of the MOVE operator differs from the observed post-state.

The next step **explains** the difference between the observed and predicted states. A possible cause for this discrepancy is that the predicted state is inconsistent and, hence, can never be observed. Therefore, for each variant outcome of the operator, the resulting predicted state is tested for inconsistencies using the domain constraints. In our example, the square is predicted to move to the cell that is occupied by the triangle (see Figure 2). Hence, the predicted state violates the constraint that "no two distinct tiles can be in the same cell". This constitutes a single-step explanation of the failure.

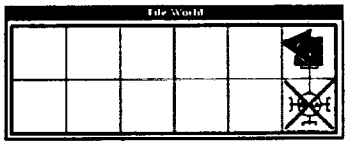

Figure 2: Inconsistent Projected State

The final step **repairs** the operator by adding missing precondition(s) that will prevent predicting this and similar inconsistent states; although the projected outcomes of the revised operator might still violate other domain constraints. This assumes that the domain constraint theory is sufficient to explain why a predicted state is inconsistent and, therefore, why the action outcome during reaction was unexpected.

That is, given a constraint C which the predicted outcome state violates, the faulty operator's preconditions are restricted to prevent the projection of *not only* this particular outcome state, but of *any* outcome state that could violate C. Goal regression (Nilsson, 1980) is used to accomplish this general repair. Regressing a goal over a single operator produces the weakest (*i.e.*, most general) preconditions that must hold in a state such that executing the operator satisfies the goal. In this case, we want to ensure that the execution of the operator results in a state that satisfies the operator's effects *and* does not violate C. Hence, the "goal" to regress is the operator's outcome restricted to prevent violation of C. The conditions resulting from regression are the new preconditions, which are a superset of the original preconditions.

In our example, the goal to regress is determined by restricting the outcomes of the MOVE operator to prevent violation of "no two distinct tiles can be in the same cell". Since one of the effects of MOVE is that "tile t is in cell (x_3, y_3)", preventing violation of the domain constraint requires that for all tiles t_2, either "tile t_2 is *not* in cell (x_3, y_3) *or* t and t_2 are *not* distinct tiles". This restriction can be expressed as "no tile other than t is in cell (x_3, y_3)" (this transformation is currently hand-coded). Hence, the goal to regress consists of the effects of MOVE plus this additional restriction. The result of regression consists of this restriction plus the original preconditions (with appropriate variable bindings). Thus, the repair step "compiles" aspects of the domain constraints into the operators, introducing new terms such as "no other tile". The operator definition is, thus, repaired by adding the new precondition: "no tile other than t is in cell (x_3, y_3)".[2]

4.2 Missing Variant Outcome

In the MOVE operator, the predicted outcome is that the agent will move straight in the intended direction to the adjacent cell. However, the TileWorld simulator will occasionally cause the agent to "veer" so that it ends up in a cell to the right or left of the intended destination. This variant of the outcome is missing from the initial operator definition.

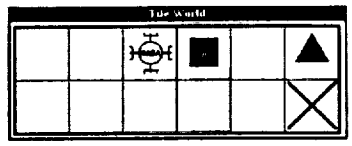

Figure 3: Missing Variant Outcome: Veer Left

As above, the first step **detects** when the observed post-state differs from all predicted states. That is, the operator has a different effect on the world than is expected. In our example, when the agent was in cell $(1, 0)$, a 'move east' resulted in the agent veering left and ending up in cell $(2, 1)$ instead of moving straight

[2]Note, this added precondition is slightly more specific than the desired "cell-empty" precondition.

to cell $(2, 0)$ as predicted (see Figure 3). For this type of prediction failure, the domain constraint theory cannot **explain** the discrepancy between prediction and observation as an inconsistency in projection. In our example, there is nothing inconsistent about predicting the agent will move straight.

For this case, the recommended **repair** is to just add the observed state as a new expectation for the future, that is, as an additional variant outcome of the operator. The new variant outcome is computed as the difference between the observed post-state and the pre-state. This assumes that all observed changes can be attributed to the previous agent action (rather than to some exogenous event). The new variant is fully instantiated since there is no theory that supports deductive generalization of the observed instance. We intend to use inductive learning to generalize over a set of new variant outcomes. Currently, the instantiated variant outcome is included in the operator definition with an arbitrarily low probability, and pairs of pre- and post- state instances are retained for induction over future observations. In our example, the new variant outcome is simply "agent in cell $(2, 1)$".

5. Learning Domain Constraints

In our first two cases, the approximate operator model was refined using the domain constraint model. To demonstrate the mutuality of the theory refinement, we sketch how the the domain constraint model could in turn be refined using the operator model. In particular, we are working toward a method to address the problem of missing domain constraints which perform 'deletes' in projection. If such a domain constraint is missing, the predicted state could differ from the observed state. Suppose the (previously mentioned) domain constraint "the agent cannot be in two places at once" is missing. Then, during projection of the MOVE operator, the assertion regarding the previous location of the agent will not be deleted. Hence, in the predicted post-state the agent is both in its old and new locations. This differs from the post-state observed during reaction, in which the agent is in its new location only. Since the prediction cannot be explained as inconsistent, yet is a superset of the observed state, this guides the method to check whether the superfluous predictions match preconditions. If so, the failure may be that certain preconditions were not deleted in projection because of a missing domain constraint.

A new domain constraint is plausibly derived based on the approximate operator model. The domain constraint should describe those superfluous predictions which match preconditions, yet are not in the observed outcomes. These need to be deleted during projection. The result is a new domain constraint which specifies that those preconditions and all the outcomes of the operator cannot co-occur. In this example, the new domain constraint, derived from the MOVE operator states that the precondition "agent at old location" and outcomes, including "agent at new location", cannot co-occur. Thus a specialization of the missing domain constraint "the agent cannot be in two locations at once" can be learned using the operators.

6. Related and Future Work

There are many approaches to theory refinement. Traditional approaches to refinement of knowledge-bases in expert systems (*e.g.*, Politakis & Weiss, 1984) typically perform induction over cases. Purely inductive approaches to refining models for reactive systems include reinforcement learning (Lin, 1990) and experimentation (Gil, 1991; Christiansen, *et al.*, 1990).

Analytic, or explanation-based learning approaches to refining approximate theories fall into four broad categories: (*i*) using a complete and correct auxiliary theory to refine the approximate one, as in most EBL from failure (Hammond, 1986; Mostow & Bhatnagar, 1987; Chien, 1989; Gupta, 1987; Tadepalli, 1989); (*ii*) relying on induction and possibly experimentation when the explanation is insufficient (Rajamoney & DeJong, 1988; Ourston & Mooney, 1990; Pazzani, 1988; Danyluk, 1989; Ali, 1989); (*iii*) augmenting the system's knowledge through apprenticeship learning from the user (Wilkins, 1988; Smith, *et al.*, 1985; Laird, *et al.*, 1990); (*iv*) using one approximate theory to refine another. We distinguish our work from the above approaches to refining approximate theories in that we use one approximate theory to refine another (Bennett (1990) is a closely related approach).

In order to circumscribe the problem initially, we have made a number of assumptions, and did not address certain issues. One major assumption underlying the current work is that the approximations are due to incompleteness rather than incorrectness. Another assumption is that there is a single recommended refinement. These need to be removed in the future. Other future issues are: (*i*) utility of the refinement – being selective as to which new information to retain and which to "forget"; (*ii*) consistency maintenance – coordinating the refinements in several models to reflect each other correctly as new refinements are made; (*iii*) dependency maintenance – retaining the appropriate justifications to retract earlier faulty decisions, as in (Smith, *et al.*, 1985); (*iv*) eager versus lazy refinement – trading offline processing of errors versus waiting until they are observed through failure; (*v*) deciding what to sense in the world for failure detection; (*vi*) refinement based on partial observations.

In conclusion, we have discussed three cases of mutual theory refinement (the first two of which have been implemented). In the first, the operator model was refined with the aid of the approximate domain constraint model. In the second, inductive learning

was used because the domain constraint model was deemed insufficient to refine the operators analytically. And in the third, the domain constraint model was refined using the approximate operator model. These methods begin to pave the way for more robust reactive systems, better able to learn from their failures and refine their models with experience.

Acknowledgements

Thanks to the other members of the ERE group: Mark Drummond, Rich Levinson, Andy Philips, and Keith Swanson. Thanks also to Mark Drummond, Rich Keller, Pat Langley, and Steve Minton for comments on earlier drafts. Lisa Dent's collaboration was funded through San Jose State University Summer Program.

References

Ali, K. M. 1989. Augmenting domain Theories for Explanation-Based Generalization. *Proc. of the Sixth ML Conf.*, Ithaca, NY.

Bennett, S. W. 1990. Reducing Real-World Failures of Approximate Explanation-Based Rules. *Proc. of the Seventh ML Conf.*, Austin, TX.

Bresina, J. and Drummond, M. 1990. Integrating Planning and Reaction: A Preliminary Report. *AAAI Spring Symposium Series.*

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2, 14-23.

Chien, S. A. 1989. Using and Refining Simplifications: Explanation-Based Learning of Plans in Intractable Domains. *Proc. of the Eleventh IJCAI*, Detroit, MI.

Christiansen, A. D., Mason, M. T., and Mitchell, T. M. 1990. Learning Reliable Manipulation Strategies without Initial Physical Models. In *Proc. of the IEEE International Conf. on Robotics and Automation.*

Danyluk, A. P. 1989. Finding New Rules for Incomplete Theories: Explicit Biases for Induction with Contextual Information. *Proc. of the Sixth ML Conf.*, Ithaca, NY.

Drummond, M. 1986. A Representation of Action and Belief for Automatic Planning Systems. *Proc. of Reasoning About Actions and Plans*, Timberline, OR.

Drummond, M. and Bresina, J. 1990. Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. *Proc. of the Ninth AAAI Conf.*, Boston, MA.

Drummond, M., Bresina, J., and Kedar, S. 1991. The Entropy Reduction Engine: Integrating Planning, Scheduling, and Control. *AAAI Spring Symposium Series.*

Gil, Y. 1991. A Domain-Independent Method for Effective Experimentation in Planning. *Proc. of the Eighth ML Workshop*, Evanston, IL.

Gupta, A. 1987. Explanation-Based Failure Recovery. *Proc. of the Sixth AAAI Conf.*, Seattle, WA.

Hammond, K. J. 1986. Learning to Anticipate and Avoid Planning Problems through the Explanation of Failures. *Proc. of the Fifth AAAI Conf.*, Philadelphia, PA.

Laird, J. E., Hucka, M., Yager, E. S., and Tuck, C. M. 1990. Correcting and Extending Domain Knowledge Using Outside Guidance. *Proc. of the Seventh ML Conf.*, Austin, TX.

Lin, L. 1990. Self-Improving Reactive Agents: Case Studies of Reinforcement Learning Frameworks. *Proc. of the International Conf. on Simulation of Adaptive Behaviour*, Paris, France.

Mitchell, T. M. 1990. Becoming Increasingly Reactive. *Proc. of the Ninth AAAI Conf.*, Boston, MA.

Mostow, J. and Bhatnagar, N. 1987. Failsafe – A Floor Planner that Uses EBG to Learn from its Failures. *Proc. of the Tenth IJCAI*, Milan, Italy.

Nilsson, N. J. 1980. Principles of Artificial Intelligence. Palo Alto, CA: Tioga Publishers.

Ourston, D. and Mooney, R. 1990. Changing the Rules: A Comprehensive Approach to Theory Refinement. *Proc. of the Ninth AAAI Conf.*, Boston, MA.

Pazzani, M. J. 1988. Integrated Learning with Incorrect and Incomplete Theories. *Proc. of the Fifth ML Conf.*, Ann Arbor, MI.

Philips, A. B. and Bresina, J. L. 1991. NASA Tile-World Manual. Tech. Report NASA TR-FIA-91-04, NASA Ames. February, 1991.

Politakis, P. and Weiss, S. M. 1984. Using Empirical Analysis to Refine Expert System Knowledge Bases. *Artificial Intelligence* 22(1):23-48.

Rajamoney, S. A. and DeJong, G. F. 1988. Active Explanation Reduction: An Approach to the Multiple Explanations Problem, *Proc. of the Fifth ML Conf.*, Ann Arbor, MI.

Smith, R. G., Winston, H. A., Mitchell, T. M., and Buchanan, B. G. 1985. Representation and Ues of Explicit Justifications for Knowledge Base Refinement. *Proc. of the Ninth IJCAI*, Los Angeles, CA.

Tadepalli, P. Lazy Explanation-Based Learning: A Solution to the Intractable Theory Problem. *Proc. of the Eleventh IJCAI*, Detroit, MI.

Wilkins, D. C. 1988. Knowledge Base Refinement Using Apprenticeship Learning Techniques. *Proc. of the Nineth AAAI Conf.*, St. Paul, MN.