Retrospective Self-Adaptation of an Agent's Domain Knowledge: Perceptually-Grounded Semantics for Structural Credit Assignment

Joshua Jones & Ashok K. Goel
Design Intelligence Laboratory, School of Interactive Computing
Georgia Institute of Technology, Atlanta, USA 30332
{jkj, goel}@cc.gatech.edu

Abstract

AI research on meta-reasoning for agent self-adaptation has generally focused on modifying the agent's reasoning processes. In this paper, we describe the use of meta-reasoning for retrospective adaptation of the agent's domain knowledge. In particular, we consider the use of meta-knowledge for structural credit assignment in a classification hierarchy when the classifier makes an incorrect prediction. We present a scheme in which the semantics of the intermediate abstractions in the classification hierarchy are grounded in percepts in the world, and show that this scheme enables self-diagnosis and self-repair of knowledge contents at intermediate nodes in the hierarchy. We also discuss the implications of this scheme for an architecture for meta-reasoning.

Introduction

It is generally agreed in AI that the capability of meta-reasoning is essential for achieving human-level intelligence (Brachman 2002) (Minsky 1995) (Minsky, Singh, and Sloman 2004). Past AI research has shown that meta-reasoning is useful for control of reasoning (Davis 1980) (Stefik 1981) (Hayes-Roth and Larsson 1996) (Hansen and Zilberstein 2001) (Raja and Lesser 2007), bounding of computations (Horvitz, Cooper, and Heckerman 1989) (Russell 1991) (Horvitz 2001), revision of conclusions (Doyle 1979), selection of learning strategies (Cox and Ram 1999), revision of reasoning processes (Stroulia and Goel 1995) (Leake 1996) (Murdock and Goel 2008), refinement of indices (Fox and Leake 2001), self-explanation (Goel and Murdock 1996), self-monitoring (Ganek and Corbi 2003), and guiding of reinforcement learning (Ulam et al. 2005) (Anderson et al. 2006). (Cox 2005) provides a useful review of some AI research on meta-reasoning.

AI research on meta-reasoning for agent self-adaptation has generally focused on modifying the agent's reasoning processes. The need for self-adaptation of course arises because intelligent agents typically operate in dynamic task environments. It is useful to make a few distinctions here. Firstly, adaptations to an agent can be retrospective (i.e., when the agent fails to achieve a goal in its given environment; (Genesereth 1983) (Birnbaum et al. 1990) (Stroulia and Goel 1996) (Stroulia and Goel 1997) (Murdock and Goel 2008) (Leake 1996), or proactive (i.e., when the agent is asked to operate in a new task environment; e.g., (Murdock and Goel 2008), (Murdock and Goel 2003) (Murdock and Goel 2008). Secondly, adaptations can be either to the deliberative element in the agent architecture (Genesereth 1983) (Birnbaum et al. 1990) (Stroulia and Goel 1995) (Leake 1996) (Murdock and Goel 2008), or the reactive element (Stroulia and Goel 1999), or both. Thirdly, adaptations to the deliberative element may be modifications to its reasoning process (i.e., to its task structure, selection of methods, or control of reasoning; e.g., (Birnbaum et al. 1990) (Stroulia and Goel 1995) (Leake 1996) (Murdock and Goel 2008), or to its domain knowledge (i.e., the content, representation and organization of its knowledge), or both.

A core and longstanding problem in self-adaptation is that of credit (or blame) assignment (Samuel 1959) (Minsky 1995). It is useful to distinguish between two kinds of credit assignment problems: temporal and structural. In temporal credit assignment, given a sequence of many actions by an agent that leads to a failure, the task is to identify the actions(s) responsible for the failure. Reinforcement learning is one method for addressing the temporal credit assignment problem (Sutton and Barto 1998). In structural credit assignment, given an agent composed of many knowledge and reasoning elements that fails to achieve a goal, the task is to identify the element(s) responsible for the failure. Meta-reasoning for self-adaptation typically addresses the problem of structural credit assignment, though this can also be used to guide reinforcement learning (Ulam et al. 2005) (Anderson et al. 2006). It is useful to note the close relationship between agent self-adaptation and agent learning: the use of meta-reasoning for self-adaptation views learning as a deliberative, knowledge-based process of self-diagnosis and self-repair. In this sense, research on self-adaptation via meta-reasoning can be viewed as a bridge between knowledge-based AI and machine learning.

Our past work on addressing the structural credit assignment over reasoning processes has investigated the hypothesis that a declarative self-model that captures the teleology of the agent's design (i.e., its functions and the mechanisms that result in the accomplishment of the functions) may enable localization, if not also identification, of the elements in the reasoning process responsible for a given behavior. Put another way, our work has explored teleology as a central or-

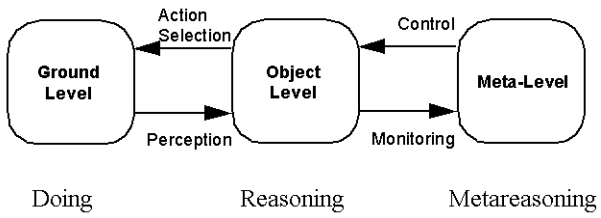

Figure 1: Model of a Metareasoning Agent Adapted From (Cox and Raja 2007)

ganizing principle of self-adaptation of reasoning processes.

Current Work

As indicated above, AI research on meta-reasoning for self-adaption, including our own work, has generally focused on modifying the agent's reasoning processes. In this paper, we describe our ongoing work on the equally important problem of using meta-reasoning for modifying the agent's domain knowledge. Since classification is a ubiquitous task in AI, we consider the problem of using meta-knowledge for repairing classification knowledge when the classifier supplies an incorrect class label. This problem arises when the meta-level monitoring process in an agent like that depicted in figure 1 detects an error that is localized to a classifier. More specifically, we consider the subclass of classification problems that can be decomposed into a hierarchical set of smaller classification problems; alternatively, problems in which features describing the world are progressively aggregated and abstracted into higher-level abstractions until a class label is produced at the root node. This subclass of classification problems are recognized as capturing a common pattern of classification (e.g., (Bylander, Johnson, and Goel 1991) (Russell 1988)). We will call this classification task *compositional classification*, and the hierarchy of abstractions an *abstraction network*.

In particular, we consider the problem of retrospective adaptation of the content of the intermediate abstractions in the abstraction network (and *not* its structure) when the classifier makes an incorrect classification. Note that once again structural credit assignment becomes a core problem: given the error at the root node, the structural credit assignment problem now is to identify the intermediate abstractions in the abstraction network responsible for the error. Note also that the hypothesis about using teleological knowledge that works so well for adapting reasoning processes is not useful in this setting because there is no complex reasoning process to model here.

Instead, in this paper we propose and explore an alternative hypothesis for using meta-reasoning for self-adaptation of domain knowledge: if the semantics of domain concepts can be grounded in predictions about percepts in the world, then meta-knowledge in the form of verification procedures associated with the domain concepts is useful for addressing the structural credit assignment problem. Meta-reasoning can then use verification procedures associated with domain concepts to verify the predictions made by those concepts. In the case of compositional classification, this means that intermediate abstractions in the abstraction network are chosen such that each abstraction corresponds to a prediction about percepts in the world, meta-knowledge comes in the form of verification procedures associated with the abstractions, and meta-reasoning invokes the appropriate verification procedures to perform structural credit assignment and then adapt the abstractions. The verification procedures explicitly encode the grounding of intermediate abstractions in percepts from the environment. Below we illustrate, formalize and evaluate these ideas, and briefly discuss the implications of this scheme for a meta-reasoning architecture.

Task Models

To make the problem concrete, we will present an example from the turn-based strategy game called FreeCiv (www.freeciv.org). Figure 2 depicts an example of a partially expanded process model for an agent that plays the game. This model is expressed in a teleological modeling language, Task-Method Knowledge Language (TMKL) (Murdock and Goel 2008). TMKL models of software systems are expressed in terms of tasks, methods, and knowledge. A *task* describes user intent in terms of a computational goal producing a specific result. Tasks encode functional information – the production of the intended result is the function of a computation. It is for this reason that the models specified in TMKL are *teleological* – the purpose of computational units is explicitly represented. A *method* is a unit of computation that produces a result in a specified manner. The *knowledge* portion of the model describes the different concepts and relations that tasks and methods in the model can use and affect as well as logical axioms and other inferencing information involving those concepts and relations. TMKL has been shown to be more expressive than Hierarchical Task Networks (HTNs) (Erol, Hendler, and Nau 1994), as TMKL enables explicit representation of subgoals and multiple plans for achieving a goal. Hoang, Lee-Urban and Munoz-Avila (Hoang, Lee-Urban, and Muñoz-Avila 2005) designed a game-playing agent in both TMKL and HTN and noted that TMKL provided control structures and other features beyond those available in HTN, and that TMKL provides strictly more expressive power than HTNs. Figure 2 displays only the tasks (rectangles) and methods (rounded rectangles) of the FreeCiv playing agent.

On each turn in a game of FreeCiv, the agent depicted in figure 2 must select a compound action that consists of setting various parameters and moving units such as military units and "worker" units, called settlers, that can improve terrain or build new cities. Building new cities on the game map is a crucial action, as each city produces resources on subsequent turns that can then be used by the player to further advance their civilization. The quantity of resources produced by a city on each turn is based on various factors, including the terrain and special resources surrounding the city's location on the map, and the skill with which the city's operations are managed. The agent modeled in figure 2 handles decisions about moving units to build cities in the subtask *Select Build City Action*. Consider what happens when meta-level monitoring detects that the game playing agent

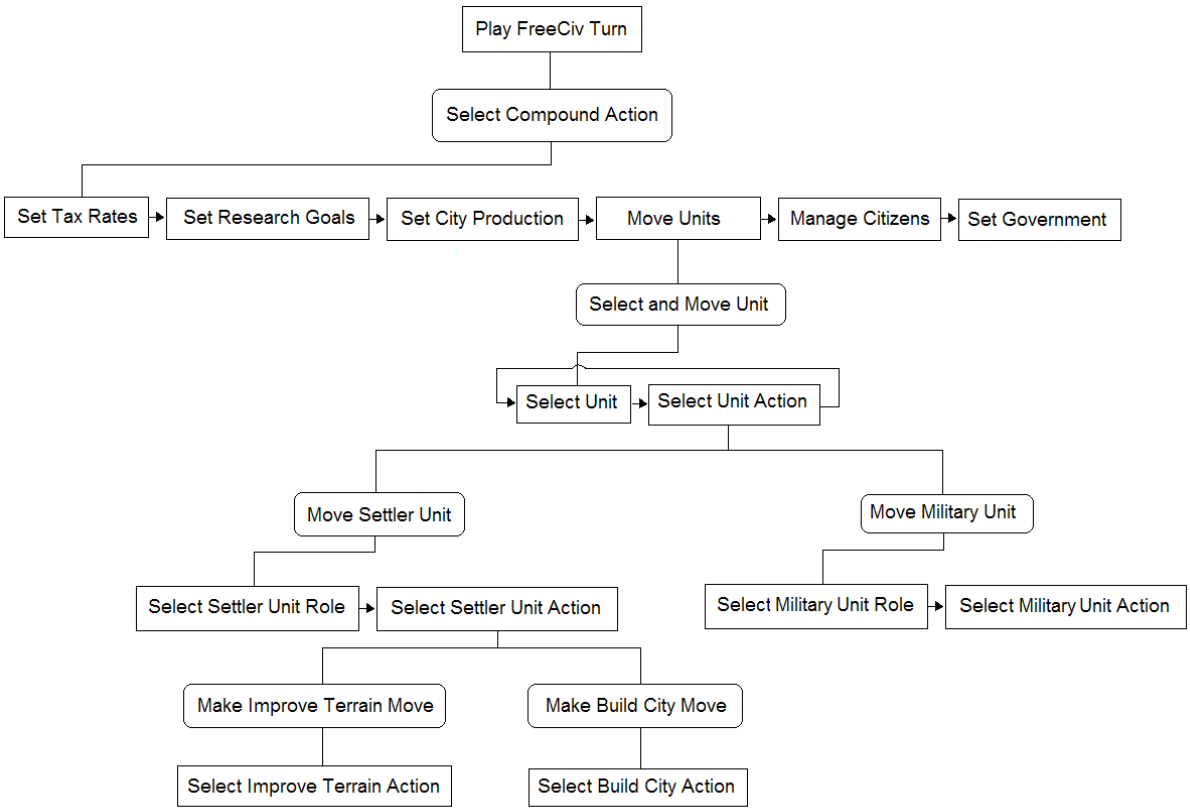

Figure 2: FreeCiv Agent Process Model

has made some error, perhaps failing in its overall goal of winning a game. At this point, a diagnostic procedure like that implemented in REM (Murdock and Goel 2008) is engaged, and the agent reasons over its self-model of object level processing in order to localize the cause for failure. In some situations, this process of self-diagnosis may lead to the identification of some primitive task in the process model as a cause for failure. Primitive tasks are those that are directly achievable by applying some knowledge and/or taking some action in the world. Frequently, these primitive tasks may fundamentally be compositional classification tasks. In this paper, we consider the self-diagnosis and self-repair problem that arises when the agent identifies a task such as the *Select Build City Action* primitive task as the cause of a failure. We address this problem by providing the agent not only with a model of its own *processing* but also of the *knowledge* used in that processing. Then, when a primitive task is identified as responsible for some failure at the process level, self-diagnosis can operate over the model of the knowledge used by that primitive task and enable the repair of that knowledge. In the work described here, we have not actually implemented the integration of the self-diagnosis and repair of knowledge with the self-diagnosis of process provided by REM; rather, we experiment with a compositional classifier operating independently and interacting directly with the environment. However, our intent is that this procedure could be integrated with a system such as REM to form a unified metareasoning-based approach to self-diagnosis and repair over both object-level process and knowledge.

To return to our running example, when our agent selects the action for a unit that is to build a city, a crucial decision is whether the location on the game map currently occupied by the unit is suitable for the placement of the new city. We will judge the quality of a potential city location based upon the quantity of resources that we expect a city built in that location to produce over time. This decision is an example of a compositional classification task. Figure 3 illustrates a knowledge hierarchy for this task used by our FreeCiv game-playing agent.

Compositional Classification

To more formally describe compositional classification, let T be a discrete random variable representing the class label. Let $S = \{s : s \text{ is empirically determinable and } h[T] > h[T|s]\}$, where $h[x]$ denotes the entropy of x. S is a set of discrete random variables that have nonzero mutual information with the class label and are *empirically determinable*, meaning that there is some way to interact with the environment to determine which value has been taken by each member of S. Each member s of S represents a related set of equivalence classes, where each value taken by

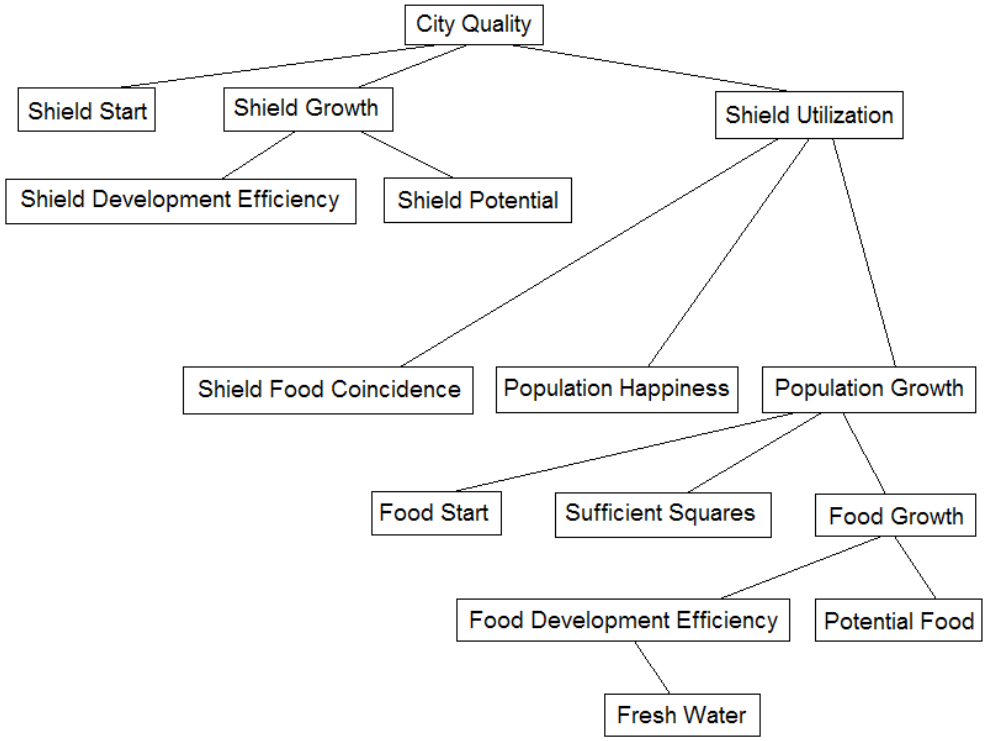

Figure 3: City Estimate Knowledge Hierarchy

s is a unique equivalence class. In the case of our running FreeCiv example, things like the future population growth of the potential city and the amount of food provided by terrain squares around the city location constitute S. A task instance is generated by jointly sampling the variables in $S \cup T$. In FreeCiv, the game engine handles this for us by randomly generating a game map and managing game dynamics that govern the relationships among the variables in S.

Empirical determinability captures the notion of perceptual grounding of concepts, indicating that each equivalence class represents some verifiable statement about the world. In the simplest case, empirical determinability means that the value taken by the variable in a given task instance is directly observable at some later time after classification has occurred. In general, some experiment may need to be performed in order to observe the value of some $s \in S$. In FreeCiv, all of the values can be directly observed, though some only after classification has occurred. This is because in order to be useful, the prediction of city resource production must be made before the city is actually constructed and its resource production rate and the values of the intermediate nodes in the hierarchy can be observed. However, we can obtain the true values later in order to perform self-diagnosis over the knowledge structure used for the classification. We call the problem of predicting T in such a setting *compositional classification*. In order to make such predictions, our agent will make use of a structured knowledge representation called an *abstraction network*, defined in the next section. This representation will capture knowledge about the relationships between variables in S. Knowledge repair will be required if the distributions $\mathcal{P}(s|K), s \in S \cup T, K \subseteq S$ are not always accurately known by the agent, but must instead be inferred from experience.

Abstraction Networks

Representation

Here we formally define the knowledge representation used at the object level for the compositional classification task. This representation is annotated with meta-knowledge used by meta-level reasoning process for self-diagnosis. We call this diagnostic self-knowledge *empirical verification procedures*, described in more detail below.

The knowledge structure contains a node for each $s \in S \cup T$. These nodes are connected in a hierarchy reflecting direct dependence relationships organized according to background knowledge. Each node will handle the subproblem of predicting the value of the variable with which it is associated given the values of its children.

Definition 1 *A* supervised classification learner *(SCL) is a tuple* $< I, O, F, U >$*, where* I *is a set of input strings (input space),* O *is a set of output symbols (output space),* F *is a function from* I *to* O*, and* U *is a function from* $(i, o) : i \in I, o \in O$ *to the set of SCLs that share the same input & output spaces* I *&* O*.*

Definition 2 *An* empirical verification procedure *(EVP) is a tuple* $< E, O >$ *where* O *is a set of output symbols (output*

space) and E is an arbitrary, possibly branching sequence of actions in the environment and observations from the environment concluding with the selection of an $o \in O$.

Any output space O of an empirical verification procedure is an empirically determinable set of equivalence classes. So, a set of equivalence classes is empirically determinable if an empirical verification procedure can be defined with an output space equal to that set of classes.

Definition 3 *An* Abstraction Network *(AN) is a tuple $< N, O, L, P >$, where N is a (possibly empty) set of ANs, O is a set of output symbols, L is an SCL, and P is an empirical verification procedure. Let I be the set of strings formable by imposing a fixed order on the members of N and choosing exactly one output symbol from each $n \in N$ according to this order. The SCL L has input space I and output space O, and the empirical verification procedure P has output space O.*

When N is empty, L is trivial and has no use as the input space is empty. In these cases (the leaves of the AN), a value determination must always be made by invoking P. Thus, EVP execution must be possible before classification in the case of AN leaves, though it is never possible until some time after classification for non-leaf nodes.

Object-level Reasoning

In a given task instance, the values of the leaf nodes are fixed by observation. Each node with fixed inputs then produces its prediction. This is repeated until the value of the class label is predicted by the root of the hierarchy.

begin AN-reasoning(a)

1. If $a.N = \emptyset$, execute $a.P$ and return the result.
2. Else, recursively execute this procedure for each $n \in N$ to generate an input string i for $a.L$, then return $a.L.F(i)$ and store this value and i for the purpose of the self-diagnosis procedure (called $a.last\_value$ and $a.last\_input$ below).

end

Meta-level Diagnosis and Repair

At some time after classification, the true value of the class label is obtained by the monitoring process. If the value produced by object-level reasoning was correct, no further action is taken. If the value is found to be incorrect, the following self-diagnosis and repair procedure is followed:

begin AN-diagnose-and-repair(a)

1. If $a.P == a.last\_value$ then return $true$.
2. $\forall n \in a.N$, call AN-diagnose-and-repair(n). If $\exists n \in a.N$ s.t. AN-diagnose-and-repair(n) $== false$ then return $false$.
3. $a.L < -a.L.U((a.last\_input, a.P))$, return $false$.

end

This procedure has a base case when the leaves are reached, as their true values were obtained before classification, and thus cannot be found to be incorrect.

Notice that an AN abstracts in two ways. One is apparent in the object-level reasoning procedure; information is progressively lost at each node in the hierarchy during reasoning as information is aggregated into equivalence classes, so abstraction takes place during inference. The second source of abstraction becomes clear in the self-diagnosis and repair procedure. The EVPs explicitly encode a process of abstraction from raw state to the equivalence classes produced at nodes in the AN.

Formal Justification of Self-Diagnosis Technique

In this section, we sketch a proof that the self-diagnosis technique used for AN self-diagnosis is optimal with respect to maximizing expected decrease in *diagnostic* search space entropy with each probe (where a probe is an EVP execution), under assumptions outlined below. Here, the diagnostic search space consists of all possible error conditions that lead to an error observed at the root of the AN, under the following provisions:

- A value produced at a node is wrong only if it is both *objectively* wrong (wrong according to the associated EVP) and *subjectively* wrong (recursively, leads to the production of an erroneous value at the parent). In the language of diagnosis, this amounts to ignoring compensating faults.
- Without loss of generality, we assume that each node produces values in a way that is actually dependent on the values of child nodes.

A diagnostic search space for a given failure thus consists of all possible contiguous sets of nodes that include the root. A given diagnosis is correct for a given failure situation if it is the maximal such set for which all nodes in the set produced incorrect values during the related inference. The set represents the nodes that produced erroneous values during the failure, and adaptation within nodes will occur at the fringe of the set, as per the self-diagnosis and repair procedure described earlier. Note that this diagnostic search space is distinct from the hypothesis space searched by the self-diagnosis and repair procedure operating across a sequence of examples; *this* search space is not concerned with any specific knowledge stored at the nodes, but only with explaining the fault location(s) that lead to a particular failure instance.

For any given node in the AN x, define X to be the entropy left in the diagnostic search space if x is probed (associated EVP executed) and found to have produced a good value, and $\neg X$ the entropy left if x is found to have produced a bad value. Then, the expected remaining entropy if x is probed is given by $X \cdot P(X) + \neg X \cdot P(\neg X)$, where $P(X)$ and $P(\neg X)$ represent the a priori probabilities that x will be found to be either good or bad, respectively, if probed.

Lemma 1 *For any pair of unprobed nodes a, b in an AN such that a is a (possibly indirect) ancestor of b:*

1. $B \geq A$
2. $\neg B \geq A$
3. $\neg A \leq B + \neg B - 2 \cdot A$

To see that (1) & (2) are correct, notice that if a is probed and found to have produced a correct value, all diagnoses consistent with this observation are consistent with *either* result of probing b; that is, the set of consistent diagnoses remaining if b is probed and found to have produced a correct value is a subset of the diagnoses consistent with a having been found to be correct, and likewise for b being found incorrect. The intuition behind this proof lies here. Since we do not know whether node b's status is important until we probe node a, it is more fruitful to probe at a first. (3) is a related inequality, and is based on the observation that if b is a direct descendant of a, the diagnoses consistent with b being shown correct are those diagnoses consistent with a being shown correct, plus some fraction of those consistent with a being shown incorrect. Likewise for b being shown incorrect. Since there is no diagnosis consistent with neither b being shown correct nor b being shown incorrect, and there is no diagnosis consistent with a being shown incorrect, b being shown correct and b being shown incorrect, we arrive at a variant of (3) with an equality rather than an inequality. Now notice that if b is an indirect descendant of a, B and $\neg B$ will not only be duplicating coverage of A, but will also be duplicating coverage of some portions of $\neg A$ that are consistent with correct values having been produced by some nodes on the path from b to a. This leaves us with (3) as presented above, since the RHS may exceed the LHS due to this duplication.

Theorem 1 *For any two unprobed nodes a and b within an AN, where a is a (potentially indirect) ancestor of b, the expected remaining entropy in the diagnostic search space is less at a, making a a more desirable probe point.*

To briefly sketch the proof, let us assume that b constitutes a better next probe point than a, on the basis of expected remaining entropy in the diagnostic search space. Then $B{\cdot}P(B){+}\neg B{\cdot}P(\neg B) < A{\cdot}P(A){+}\neg A{\cdot}P(\neg A)$. Substituting using our lemma and simplifying yields $P(A) > 1$, a contradiction.

Moving from this result to our AN self-diagnosis procedure is straightforward; it is preferable to probe nodes with ancestors that have been probed (and found incorrect, of course). Further, the order of probing among such nodes is arbitrary since the probing of any node with this characteristic cannot remove the characteristic from any other node, and no such nodes can remain when a diagnosis is uniquely selected.

FreeCiv Experiment

We have experimented in the FreeCiv domain using the AN depicted in figure 3. This AN was used to produce outputs from a set containing three values, corresponding to poor, moderate and good city resource production. These values indicate predictions about the resource production expected from a city built on a considered map location. Specifically, the values correspond to an expected degree and direction of deviation from a logarithmic baseline resource production function that was manually tuned to reflect roughly average city resource production. Each of the intermediate nodes in the AN has an output set consisting of 5 values in this experiment. The empirical verification procedures simply discretize observed game features. We placed a very simple rote learner within each node in the AN. These simple learners offer no generalization power of their own, so this experiment relies on the power of the AN representation itself rather than on powerful learners within nodes. The content of each rote learner was initialized arbitrarily in a way that was known to be incorrect in some cases for each of the learners. Because we expect resource production from cities built on various kinds of map locations to potentially differ qualitatively as games progress, we trained 3 separate AN-based learners, with one of each learning to make predictions about resource production in the early, middle or late stages of the game. Results reported are cumulative across all three learners of the appropriate type.

To test our self-diagnosis and self-repair procedure, we ran 60 independent trials, each consisting of a sequence of 49 games played by an agent using the AN of figure 3. Results reported in this section are an average across these trials. Each game played used a separate randomly generated map, with no opponents. The agent always builds a city on the first occupied square, after making an estimate of the square's quality. Building in the first randomly generated occupied square ensures that the agent will have opportunities to test its knowledge in a variety of states. We evaluated the result of our self-diagnosis and self-repair procedure by comparing the average performance of the agent during the first 7 games to the average performance during the last 7 games. Making this comparison, we observed an average 52% decrease in the error rate of the learner. This improvement in performance is evidence that the meta-level process has been successful in repairing faulty knowledge at the object level. We have also experimented with the AN self-diagnosis and self-repair procedure in other domains such as prediction of the direction of the Dow Jones Index (Jones and Goel 2007) and in synthetic settings, with similarly positive results.

Related Research

As we mentioned in the introduction, the use of meta-reasoning for self-adaptation in intelligent agents is related to learning: it views learning as deliberative, knowledge-based self-diagnosis and self-repair. In particular, our work on use of meta-reasoning for structural credit assignment in compositional classification is related to past work on tree-structured bias (TSB) (Russell 1988)(Tadepalli and Russell 1998). In TSB, a concept hierarchy like those represented by ANs is used to limit the hypothesis space that must be searched by a learner. However, there are several *fundamental* differences between our work and past work on tree-structured bias. First, TSB has dealt only with binary classifications at all nodes in the hierarchy, while ANs can deal with multivalue classifications. Next, TSB research does not have the concept of EVPs, which encode the meta-knowledge used in our self-diagnostic procedure, instead relying on carefully constructed queries to the environment to learn the functions at internal nodes. Thus, rather than using explicitly represented meta-knowledge to perform self-diagnosis, TSB has a fixed training procedure that implicitly

relies upon a given type of query. This procedure can be seen as requiring a very specific kind of empirical verifiability for internal nodes – thus forcing a particular (and rather complex) form on the EVPs that a designer would write if applying TSB procedures within the AN framework. In the work described here, we take the stance that, in general, a broader set of queries to the environment may be possible. If this is the case, it will be more efficient to make use of the observations that most directly allow us to determine the value of an internal node when learning. In fact, the motivating example given by Tadepalli and Russell (Tadepalli and Russell 1998), concerning a credit-card domain, appears clearly to have a strong kind of direct empirical verifiability at internal nodes that could be exploited by an AN using very simple EVPs. The explicit representation of EVPs by ANs is also crucial to a major difference between AN research and past work on TSB. EVPs represent an abstraction from observable quantities to concepts used in an AN hierarchy. Because the grounding of concepts in observable quantities is explicitly represented, it becomes fair game to be operated upon during adaptation. It also means that we are able to adapt intermediate concepts themselves according to their functional roles – recognizing that intermediate concepts are not set in stone by the environment, but that they are constructs that exist in order to allow for correct overall classification.

Relationship to Cox & Raja's Architecture for Meta-Reasoning

Figure 1 illustrates Cox & Raja's (2007) architecture for meta-reasoning. On one hand, our past and current work on self-adaptation of an agent's *reasoning processes* fits well with this architecture: we can imagine the teleological model of the agent's deliberative reasoning element and the meta-reasoner using the teleological knowledge for structural credit assignment over the deliberative reasoning processes as residing in the meta-reasoning element in Cox & Raja's architecture.

On the other hand, our work also departs from Cox & Raja's architecture in several significant ways. Firstly, Cox & Raja's architecture asserts (or at least implies) that an "object-level" reasoning element necessarily mediates between the "ground-level" and the "meta-level" elements. However, our work on structural credit assignment and self-adaptation in reactive control agents (Stroulia and Goel 1999) directly applies meta-reasoning over reactive control without any intermediate object-level element for deliberative reasoning. In principle, we see no particular reason why deliberative reasoning must necessarily mediate between "doing" and meta-reasoning. Secondly, Cox & Raja's architecture asserts (or at least implies) that the ground-, object- and meta-levels are distinct and separate. However, our work on using meta-reasoning for guiding reinforcement learning (Ulam et al. 2005) views the three levels as substantially overlapping. Again, in principle, we see no particular reason for a complete separation between the ground-, object- and meta-levels. Thirdly, our current work described in this paper suggests that some of the actions at the ground level may be in service of verifying predictions made by semantics of the domain knowledge at the object level. Also note that in these cases, meta-knowledge may take the form of an explicit representation of the connection between knowledge used at the object level and predictions about percepts accessible from the ground level. Finally, this work suggests that meta-knowledge useful for adapting domain knowledge may be distributed over the domain concepts in the object level, and not necessarily confined to Cox & Raja's meta-level. In general, we believe that while Cox & Raja's architecture provides a good starting point for identifying a common architecture for all meta-reasoning, meta-reasoning is more dynamic and flexible than their architecture seems to imply.

Conclusions

In this paper, we described a scheme for using meta-reasoning in intelligent agents for self-adaptation of domain knowledge. In particular, we considered retrospective adaptation of the content of intermediate abstractions in an abstraction network used for compositional classification when the classifier makes an incorrect classification. We showed that if the intermediate abstractions in the abstraction network are organized such that each abstraction corresponds to a prediction about a percept in the world, then meta-knowledge comes in the form of verification procedures associated with the abstractions, and meta-reasoning invokes the appropriate verification procedures in order to perform structural credit assignment and then adapt the abstractions. This provides credence to our hypothesis about the use of meta-reasoning for self-adaptation of domain knowledge: if the semantics of domain concepts can be grounded in predictions about percepts in the world, then meta-knowledge in the form of verification procedures associated with the domain concepts is useful for addressing the structural credit assignment problem. We note however that whether this hypothesis holds for tasks other than compositional classification is an open question.

Acknowledgements

This research is supported by an NSF (SoD) Grant (#0613744) on Teleological Reasoning in Adaptive Software Design.

References

Anderson, M. L.; Oates, T.; Chong, W.; and Perlis, D. 2006. The metacognitive loop i: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance. *J. Exp. Theor. Artif. Intell.* 18(3):387–411.

Birnbaum, L.; Collins, G.; Freed, M.; and Krulwich, B. 1990. Model-based diagnosis of planning failures. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, 318–323. Boston, Massachusetts, USA: AAAI Press/MIT Press.

Brachman, R. J. 2002. Systems that know what they're doing. *IEEE Intelligent Systems* 17(6):67–71.

Bylander, T.; Johnson, T. R.; and Goel, A. 1991. Structured matching: a task-specific technique for making decisions. *Knowl. Acquis.* 3(1):1–20.

Cox, M. T., and Raja, A. 2007. Metareasoning: A Manifesto, Technical Report, BBN TM-2028, BBN Technologies.

Cox, M. T., and Ram, A. 1999. Introspective multistrategy learning: on the construction of learning strategies. *Artif. Intell.* 112(1-2):1–55.

Cox, M. T. 2005. Metacognition in computation: a selected research review. *Artif. Intell.* 169(2):104–141.

Davis, R. 1980. Meta-rules: Reasoning about control. *Artif. Intell.* 15(3):179–222.

Doyle, J. 1979. A truth maintenance system. *Artif. Intell.* 12(3):231–272.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, 1123–1128. Seattle, Washington, USA: AAAI Press/MIT Press.

Fox, S., and Leake, D. B. 2001. Introspective reasoning for index refinement in case-based reasoning. *J. Exp. Theor. Artif. Intell.* 13(1):63–88.

Ganek, A. G., and Corbi, T. A. 2003. The dawning of the autonomic computing era. *IBM Syst. J.* 42(1):5–18.

Genesereth, M. 1983. An overview of meta-level architecture. *In Proceedings of the Third National Conference on Artificial Intelligence* 119–123.

Goel, A. K., and Murdock, J. W. 1996. Meta-cases: Explaining case-based reasoning. *EWCBR* 150–163.

Hansen, E. A., and Zilberstein, S. 2001. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence* 126(1-2):139–157.

Hayes-Roth, B., and Larsson, J. E. 1996. A domain-specific software architecture for a class of intelligent patient monitoring systems. *Journal of Experimental and Theoretical Artificial Intelligence* 8(2):149–171.

Hoang, H.; Lee-Urban, S.; and Muñoz-Avila, H. 2005. Hierarchical plan representations for encoding strategic game AI. In Young, R. M., and Laird, J. E., eds., *AIIDE*, 63–68. AAAI Press.

Horvitz, E. J.; Cooper, G. F.; and Heckerman, D. E. 1989. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, 1121–1127.

Horvitz, E. 2001. Principles and applications of continual computation. *Artificial Intelligence* 126(1-2):159–196.

Jones, J., and Goel, A. 2007. Structural credit assignment in hierarchical classification. In Arabnia, H. R.; Yang, M. Q.; and Yang, J. Y., eds., *IC-AI*, 378–384. CSREA Press CSREA Press.

Leake, D. B. 1996. Experience, introspection and expertise: Learning to refine the case-based reasoning process. *J. Exp. Theor. Artif. Intell.* 8(3-4):319–339.

Minsky, M.; Singh, P.; and Sloman, A. 2004. The St. Thomas common sense symposium: Designing architectures for human-level intelligence. *AI Magazine* 25(2):113–124.

Minsky, M. 1995. Steps toward artificial intelligence. 406–450.

Murdock, J. W., and Goel, A. K. 2003. Localizing planning with functional process models. In Giunchiglia, E.; Muscettola, N.; and Nau, D. S., eds., *ICAPS*, 73–81. AAAI.

Murdock, J. W., and Goel, A. K. 2008. Meta-case-based reasoning: self-improvement through self-understanding. *J. Exp. Theor. Artif. Intell.* 20(1):1–36.

Raja, A., and Lesser, V. R. 2007. A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 15(2):147–196.

Russell, S. J. 1988. Tree-structured bias. In *AAAI*, 641–645.

Russell, S. 1991. Principles of metareasoning. *Artif. Intell.* 49(1-3):361–395.

Samuel, A. 1959. Some studies in machine learning using the game of checkers. *IBM Journal* 3(3):210–229.

Stefik, M. 1981. Planning and meta-planning (molgen: Part 2). *Artif. Intell.* 16(2):141–170.

Stroulia, E., and Goel, A. 1995. Functional representation and reasoning in reflective systems. *Journal of Applied Intelligence, Special Issue on Functional Reasoning* 9(1):101–124.

Stroulia, E., and Goel, A. K. 1996. A model-based approach to blame assignment: Revising the reasoning steps of problem solvers. *AAAI/IAAI* 2:959–964.

Stroulia, E., and Goel, A. K. 1997. Redesigning a problem-solver's operations to improve solution quality. *IJCAI* 1:562–567.

Stroulia, E., and Goel, A. K. 1999. Evaluating PSMs in evolutionary design: the autognostic experiments. *Int. J. Hum.-Comput. Stud.* 51(4):825–847.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tadepalli, P., and Russell, S. 1998. Learning from examples and membership queries with structured determinations. *Mach. Learn.* 32(3):245–295.

Ulam, P.; Goel, A.; Jones, J.; and Murdock, W. 2005. Model-based guidance for reinforcement learning. *In Proc. IJCAI-05 Workshop on Game Playing* 107–112.