Transfer Learning through Analogy in Games

Thomas R. Hinrichs and Kenneth D. Forbus

■ *We report on a series of transfer learning experiments in game domains, in which we use structural analogy from one learned game to speed learning of another related game. We find that a major benefit of analogy is that it reduces the extent to which the source domain must be generalized before transfer. We describe two techniques in particular, minimal ascension and metamapping, that enable analogies to be drawn even when comparing descriptions using different relational vocabularies. Evidence for the effectiveness of these techniques is provided by a large-scale external evaluation, involving a substantial number of novel distant analogs.*

Effectively transferring previously learned knowledge to a new domain is one of the hallmarks of human intelligence. This is the objective of transfer learning, in which transferred knowledge guides the learning process in a broad range of new situations. In *near transfer,* the source and target domains are very similar and solutions can be transferred almost verbatim. In *far transfer,* the domains may appear quite different and the knowledge to be transferred involves deeper shared abstractions. Thus transfer learning can be viewed as a compromise between the delayed generalization of instance- or case-based reasoning and the early generalization of classical machine learning.

We have explored the use of analogy as a general approach to near and far transfer learning in domains ranging from physics problem solving to strategy games (Klenk and Forbus 2007; Hinrichs and Forbus 2007). Using the same basic analogical mechanism, we have found that the main differences between near and far transfer involve the amount of generalization that must be performed prior to transfer and the way that the matching process treats nonidentical predicates. We present here two extensions of our analogical matcher, *minimal ascension* and *metamapping,* that enable far transfer between representations with different relational vocabulary. Evidence for the effectiveness of these techniques is provided by a large-scale external evaluation, involving a substantial number of novel distant analogs.

This article presents an overview of our transfer learning work in game learning, including both techniques for learning the initial source games and the extensions to analogical matching that enable transfer across increasingly different games. We begin with a description of analogical transfer and the constraints it entails. We then describe the use of near transfer in Freeciv, an open-source strategy game. Next, we present a series of experiments with far transfer in games defined in the general game playing (GGP) framework. We describe metamapping and minimal ascension and measure their impact on far transfer.

Analogical Transfer

We treat transfer learning as fundamentally a problem of finding a good analogy between the source and target and using that correspondence to translate symbolic representations of learned knowledge from the source to the target. The problem of analogical transfer is then to find a good mapping between a source and target that may have very different surface representations and to effectively employ the transferred knowledge to learn more quickly.

We base our approach to mapping on a well-established cognitive theory, *structure mapping* (Gentner 1983). Structure mapping describes how two cases can be aligned on the basis of common structure, a matching process that appears to be at the heart of human analogical processing and similarity judgments. This analogical matching process creates one or two mappings, each consisting of correspondences that specify how statements and entities in the two structured descriptions being compared align, that is, "what goes with what." Mappings also include candidate inferences that represent how information may be projected from one description to the other and a numerical score that estimates the structural quality of the match. The principles of structure mapping are described amply elsewhere, but for this article, a key constraint is tiered identicality. That is, by default, correspondences are only drawn between identical predicates. Nonidentical matches are considered only when they are part of some larger relational structure that could then be transferred. Minimal ascension (Falkenhainer 1988) is one way of relaxing identicality, by allowing matches between two statements if their predicates share a close common superordinate. As described later, we found it necessary to augment minimal ascension with a new technique, metamapping, to handle far transfer, where none of the domain predicates were identical.

Our implementation of structure mapping is the Structure Mapping Engine, or SME (Falkenhainer, Forbus, and Gentner 1989). It is stable, scales well (Forbus, Ferguson, and Gentner 1994), and is integrated with our other planning and reasoning facilities (Forbus, Klenk, and Hinrichs 2009).

Structure mapping depends on symbolic, structured representations. We use as a starting point the Cyc knowledge base,[1] in order to define concepts, individuals, and relationships between them. Although Cyc contains over 3 million assertions, it is only the facilities for defining representations that we use in these experiments, such as the vocabulary for representing a hierarchy of predicates, set relations, and constraints on types of arguments to predicates. We use these to define representations of game concepts and terms to support qualitative and analogical reasoning.

Near Transfer in Freeciv

In our first set of experiments, we examined direct, instance-level transfer across relatively similar situations in Freeciv,[2] an open-source strategy game (see figure 1). We wanted to show that analogical transfer could result in improved learning performance without modification or significant domain generalization. Freeciv is a turn-based strategy game in which players build and grow civilizations with the ultimate goal of taking over the world. Although there are many decisions and strategic aspects to playing the full game, we focused on the subtask of allocating resources for growing cities. What this meant in practice was choosing which tiles in a city to cultivate, mine, or fish to produce needed resources. Such a controlled task made it easier to measure and evaluate transfer.

The transfer task was to apply experience learned in growing one city to more rapidly learn to grow another city with different configurations of resources. Without prior experience, the system learned through trial and error to, for example, avoid farming the desert and to allocate workers to more productive types of tiles. Performance on this task was measured as the total city food production at the end of 50 turns. The transfer learning metric was the score on the initial game with transfer learning versus without.

Figure 2 outlines the direct transfer algorithm for the Freeciv task. The domain learner builds up a case library of resource allocation decisions and resulting city food production. It bootstraps this library using explicit experimentation strategies that guide stochastic resource allocation decisions to ensure variation (step 6). The transfer mechanism uses analogical retrieval (Forbus, Gentner, and Law 1995) to decide which cases to consider for transfer (Step 8). Decisions with poor outcomes are also recorded as nogoods, and retrieved by the same mechanism to avoid reexploring bad decisions during experimentation (step 6).

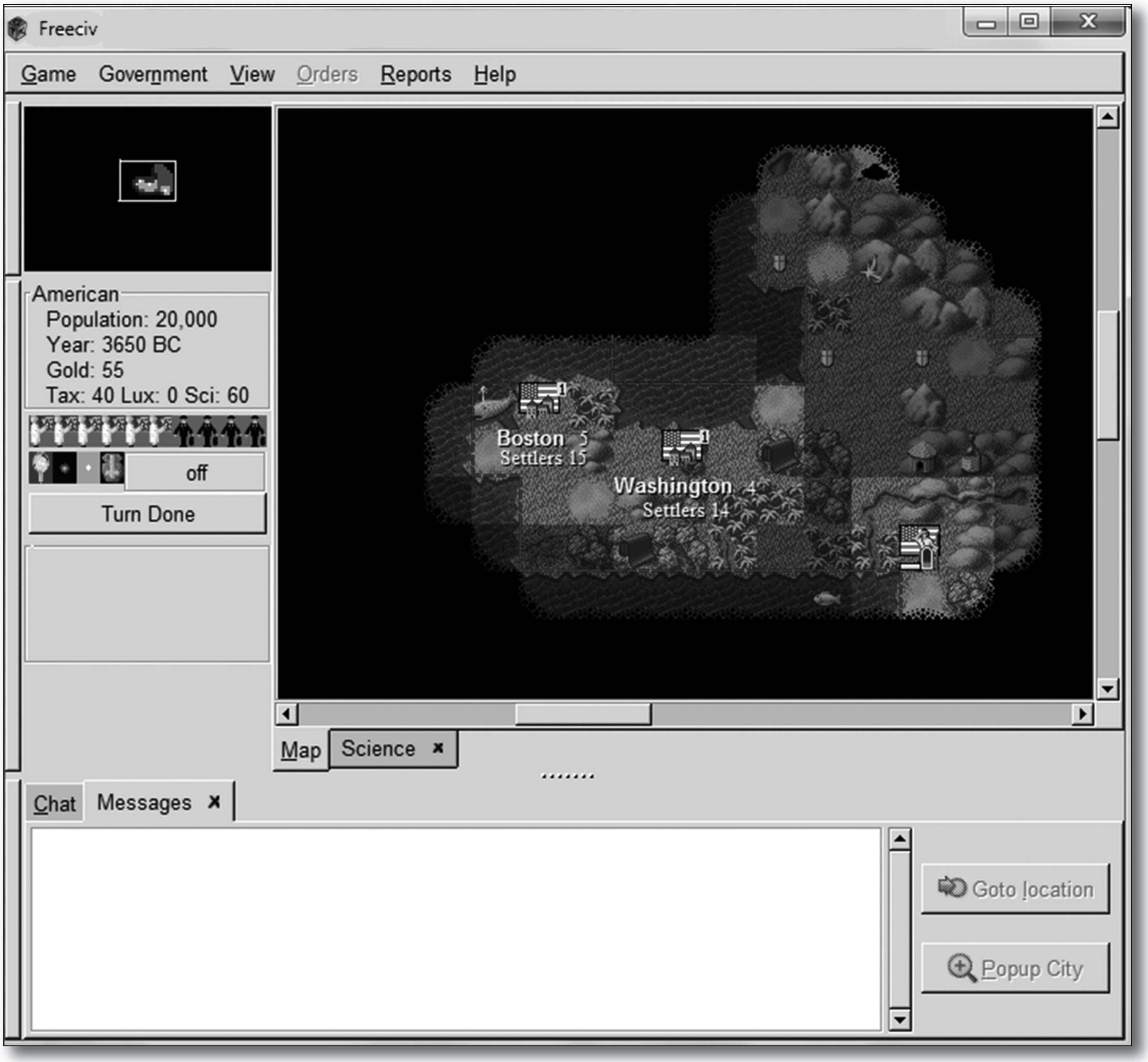


Figure 1. Freeciv.

The initial results showed a pronounced benefit in the initial game, but this was not a robust effect: Improvement was highly sensitive to the particular source and target cities. Learning curves were not stable until we began transferring negative exemplars as well (step 5), which is not common practice in case-based reasoning. With negative exemplars, we achieved a statistically significant 36 percent improvement in performance on the initial game. On subsequent trials, the benefit of transfer diminished as the learning curves with and without transfer converged rapidly. This is due largely to the simplicity of the task, rather than the similarity of the source and target. These Freeciv experiments are reported in more detail in Hinrichs and Forbus (2007). They show that structure mapping's models of analogical mapping and retrieval can be used to do transfer learning "out of the box." Far transfer requires some extensions, as described next.

Far Transfer of GGP Game Strategies

A central goal of transfer learning research is to extend the limits of cross-domain, far transfer, where there is little or no surface similarity between source and target. To better explore this, we needed games for which controlled variants could be easily generated and that could be played to completion in a reasonable amount of time. The general game playing framework (Genesereth, Love, and Pell 2005) satisfied these requirements. The benefits of GGP are that the games are generally simpler, can be played in their entirety, and more easily permit controlled modifications. In GGP, games are encoded in a game definition language (GDL), which is essentially a restricted form of Prolog with a small vocabulary of game-specific terms such as *legal*, *next*, and *goal* (see figure 3). The GDL encodes a relational net, which concisely represents a finite state machine, that is, games are restricted to finite, discrete, synchronous deterministic simulations. We focused on piece-moving games in which there is either no opponent, or simple deterministic agents in a two-dimensional spatial grid, to simplify doing automated experiments. Entire games were played to completion, rather than limiting transfer to a single subtask.

The output of the learning process is a set of strategies for winning the game. We represent learned game strategies as hierarchical task networks, or HTNs (Erol, Hendler, and Nau 1994). HTNs are hierarchical plans that reduce complex tasks to sequences of simpler subtasks, terminating at ground primitive actions to be performed by the agent. HTNs provide two important benefits: They are an explicit symbolic representation of plans that can be composed or concatenated, and their hierarchical nature makes it possible to transfer partial knowledge. Missing or incorrectly transferred subtasks can be relearned without rejecting the entire network. By transferring HTN tasks to a new game, that game can be learned with less trial and error.

```
Input: initial saved game state with one city
Output: case library of execution traces
1:  while city population > 0 and # turns < 50 do
2:  for task ∈ learnable tasks do
3:        m_d ← choose decision method (task)
4:        if m_d = decide experimentally then
5:           Transfer nogoods from failed similar cases
6:           Decide randomly, avoiding nogoods
7:        else if m_d = decide analogically then
8:           precedent = retrieve similar successful case
9:           Adapt and replay precedent plan
9:        else if m_d = decide strategically then
10:          Execute canned plan
11:       Add decision case to case library
```

Figure 2. Direct Transfer in Freeciv.

```
(item weapon1)
(weapon weapon1)
(init (location weapon1 weapon))

; next rules compute the next state:
(<=    (next (wielding ?weapon))
       (weapon ?weapon ?weapon1)
       (pickedUp ?weapon))

; legal rules enable actions:
(<=    (legal hero (move ?dir))
       (not (currentlyAsleep hero))
       (true (location hero ?xOld ?yOld))
       (direction ?dir)
       (nextCell ?dir ?xOld ?yOld ?xNew ?yNew)
       (traversable ?xNew ?yNew))

; terminal rules end the game:
(<=    terminal
       (true (carrying amulet))
       (atExit))
```

Figure 3. Example GDL Encoding of a Portion of a Micro-Rogue Game.

Terms in bold are GDL primitives.

Test Games

We employed three families of games defined in the GGP game definition language[3] (see figures 4–6). Variations of these prototypes were generated by systematically modifying initial configurations, number of elements, identities of objects, and behavioral rules.

In Escape, an explorer must cross some number of obstacles to reach an exit. To win, the player must combine resources to make bridges and rafts, or break through walls. Falling into the water is lethal. In Wargame, a soldier must kill terrorists in order to leave the maze. To win, the player must learn to acquire and shoot various weapons and avoid coming into contact with the terrorists. In Micro-Rogue, a hero must evade or destroy various monsters to acquire a magic amulet and escape the dungeon. To win, the player must apply various offensive and defensive weapons and magical artifacts such as potions and scrolls, whose effects are not initially known.

Although these games have a number of similarities, they differ in some important ways: Escape focuses on combining resources and recognizing adjacency relationships, Wargame emphasizes action at a distance and the directional nature of shooting, while Micro-Rogue distinguishes offensive from defensive weapons, and target-specific effects of actions.

The remainder of this article describes our meth-

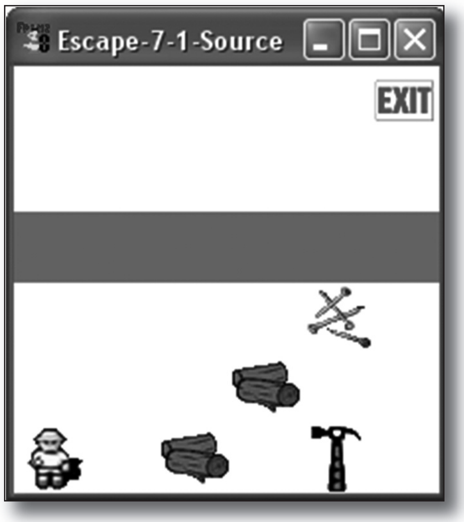


Figure 4. Escape.

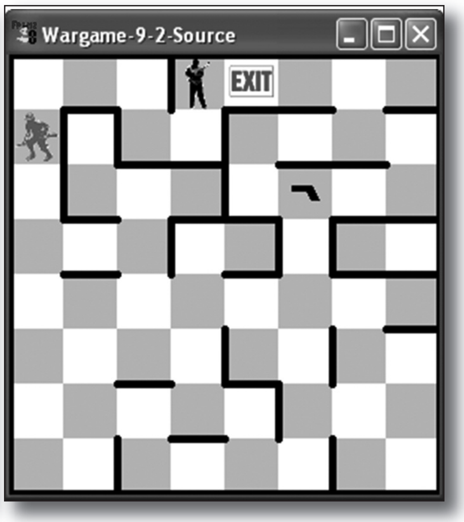


Figure 5. Wargame.

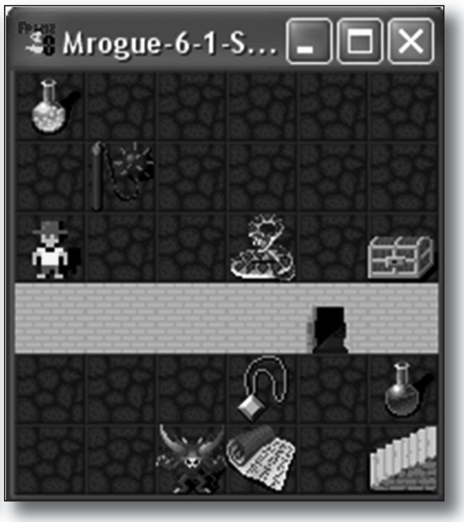


Figure 6. Micro-Rogue.

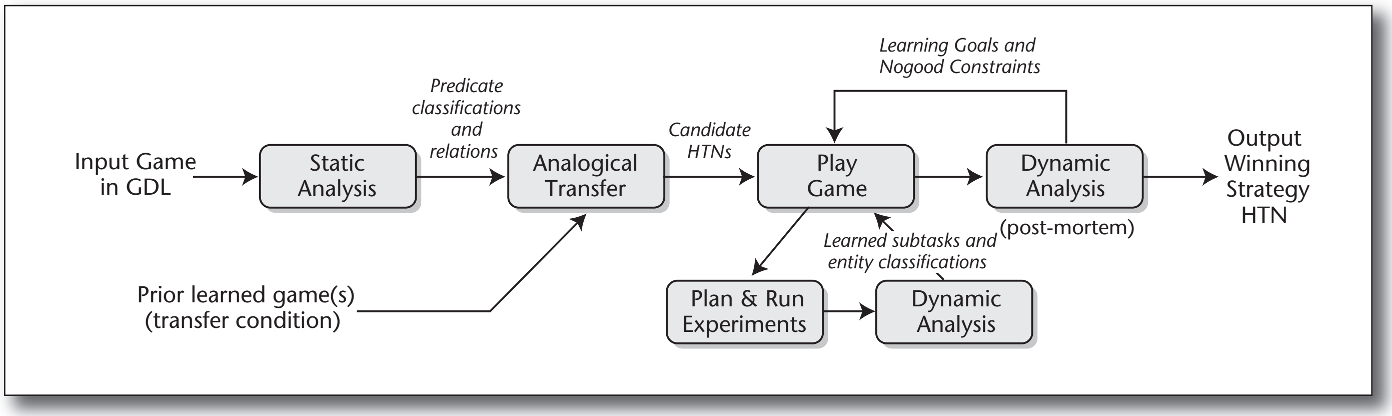

Input Game in GDL
Static Analysis
Predicate classifications and relations
Analogical Transfer
Candidate HTNs
Play Game
Learning Goals and Nogood Constraints
Dynamic Analysis
(post-mortem)
Output Winning Strategy HTN
Prior learned game(s) (transfer condition)
Learned subtasks and entity classifications
Plan & Run Experiments
Dynamic Analysis

Figure 7. Source Game Learning in GGP.

ods for performing and measuring far transfer in GGP games. Our approach involves stages of game analysis, experimentation, and solution analysis, where learning source and target games differ only by the knowledge provided through analogical transfer. Figure 7 provides a roadmap for this process, which we discuss in the following sections.

Source Domain Learning in GGP

Before describing the transfer experiments, we first need to describe the mechanism for learning the source games. Far transfer implies transferring more abstract knowledge, such as game strategies that are likely to help win sooner. For GGP, this required learning explicit and general compositional strategies rather than just accumulating cases. As with the Freeciv experiments, we again adopted an experimental learning approach, but with a greatly expanded repertoire of experimental learning strategies.

While the underlying GGP representation can be viewed formally as a finite state machine, people — the most efficient transfer learners we know of — do not generally reason at that level. Abstract concepts that people apply to such situations, including identifying spatial relations and continuous parameters like health, provide a more efficient search space than working with the underlying state machine. Consequently, our domain learner starts with a static analysis of the game, creating a declarative understanding of it that allows experimentation at the level of actions and effects, threats and hazards, and progress towards goals. This initial understanding of the game is extended by experimentation to pursue learning goals and dynamic analysis of game traces to perform credit assignment (see figure 8). We discuss each in turn.

Static Analysis

The static analysis phase recognizes important concepts implicit in the GDL encoding. This has two purposes. First, the experimentation strategies rely on background knowledge about games in general, for example, path planning and quantity planning. Applying this knowledge requires the static analysis to extract the representations of coordinates, movement operators, directions, quantities, and potential influences on quantities. The second purpose is that these more abstract representations provide a more effective language for analogical mapping and for experimentation.

Aside from eight predefined predicates, such as *next, legal, goal,* and so on, the GDL description of a game is arbitrary and, in some experiments, even the initial inputs are completely obfuscated (for example, "xyblurp" might mean "pickup" or "hammer"). To avoid unintended interactions with our knowledge base contents, the analysis routine renames every predicate with a game-specific prefix, and the relationship between the original names and the modified versions are not available to the rest of the learner. Using different predicates in the source and target this way is a major challenge for analogy, making every pair of GGP games an example of far transfer.

Elaborating the representation involves classifying the GDL game predicates to determine which correspond to actions, which are types, quantities, spatial coordinates, ordinal relations, and so forth. Figure 9 shows some example elaborated representations produced by static analysis. By looking at the collection of rules, static domain analysis first establishes some of the simple algebraic properties of predicates, such as whether or not they are functional, take ordinal or nominal values, are transitive, and/or cyclical. Then, given the assumption of a spatial game, it identifies the most likely candidate predicates for a coordinate system and determines whether the game is a piece-moving or marking-type game based on whether tokens are conserved. These classifications of predicates greatly constrain the search space for learning strategies through experimentation.

Static analysis then proceeds to examine the goal and terminal conditions of the game. This top-down analysis attempts to separate out terminal conditions that lead to a loss from those achievable conditions that define a win. If the top-level goal can be rewritten as a conjunction of subgoals, the achievement of those goals can be a measure of progress in a simple hill-climbing strategy. Of course, game goals are often nonmonotonic and the learner falls back on more blind search when the subgoals cannot be achieved independently. In retrospect, the goal analysis is probably the weakest part of the source game learner and could be strengthened by considering alternative heuristics for progress, such as planning graph reachability heuristics (Bryce and Kambhampati 2007).

```
Input: Game rules and initial state encoded in GDL
Output: Elaborated game representation with winning
strategies encoded as HTN methods
1:  Translate GDL ⟶ Cyc representation
2:  Perform static analysis to elaborate representation
3:  while game not mastered and # trials < 10 do
4:      until terminal state(game) do
5:          if winning strategy known then
6:              Execute strategy
7:          else if applicable learning goal exists then
8:              plan and execute learning experiment
9:          else
10:             Decide randomly while avoiding nogood states
11:         Execute planned turn, update game state
12:         Learn action models from completed experiments
13:     Perform post-mortem analysis
14:         Post learning goals & nogoods (loss)
15:         Extract winning strategy (win)
```

Figure 8. Source Game Learning in GGP.

```
(typePredicate weapon)
(parameterPredicate weaponStrength)
(statePredicate carrying)
(pieceMovingGame move)
(spatialLocationPred location)
(locationAttributeRepresentation location)
(entity weapon1)
(ggpQuantityPred health)
(actionPrimitive hero quaff)
(influencedBy
 ((MeasurableQuantityFn health) hero)
  (GameDistanceFn hero hobgoblin1)))
```

Figure 9. A Portion of the Elaborated Structural Representation.

Static analysis classifies game-specific predicates such as weapon and augments the case description. Predicate names have been simplified here for clarity.

Learning by Experiment

Based on the results of static analysis, the domain learner conducts experiments to satisfy explicit learning goals for knowledge acquisition (step 8). The primary learning goals are to learn the affordances of different entities in the game, the effects of an action, the applicability conditions of an action, and to learn how to decompose a goal into subgoals. The experiments are specified by plans that use high-level actions, such as going to an entity to find out what happens, trying a primitive action to see what it does, or achieving conjunctive subgoals in different orders. This constitutes a

```
(preconditionForMethod (true)
  (methodForAction (winGame hero)
  (actionSequence
    (TheList
      (runSequence
        (achieve (true (carrying weapon1)))
        (achieve (true (carrying armor1)))
        (achieve (true (health snake1 0)))
        (achieve (true (carrying amulet)))
        (gotoLocationOf hero exit)))))
```

Figure 10. A Typical Learned HTN Method.

search through a higher-level space than the raw state machine representation and is more efficient at driving exploration.

For example, the hero in Micro-Rogue might go to the snake to find out what it does. This is invariably fatal, but does teach a valuable lesson. It might try reading the scroll when that becomes legal, in order to discover the effect on quantities and relations in the game.

Initially, experiments are performed bottom-up to learn action effects and preconditions of actions. When those action-level learning goals are satisfied, it then focuses on learning goals that decompose the performance goal of the game and tries to develop a winning strategy. In addition to driving exploration this way, the experiments also serve to focus explanation and credit assignment, as described later.

Dynamic Analysis

Dynamic analysis is the ex post facto review of an execution trace to empirically assign credit or blame. This happens both during gameplay (step 12) and in a postmortem analysis after the game is won or lost (steps 13–15). It regresses through the game execution trace to explain an effect and construct a plan to achieve the effect or posts a nogood constraint to avoid it. Preference heuristics seek to explain as much as possible in terms of the player's actions or other agents' actions. In the absence of immediately preceding actions, it seeks explanations in terms of simple spatial configurations, such as colocation, adjacency, or cardinal directions. As part of this process, some entities may be classified as, for example, obstacles, threats, or hazards.

The execution trace contains not only the explicit low-level actions and state changes, but also the experiments being pursued. Knowing the experiment that was pursued provides an operational task for achieving the effect. For example, everything else being equal, if it becomes legal to shoot the gun after going to the gun, then that becomes a task for achieving the precondition of shooting.

After winning a game, the learner walks back through the execution trace to construct a high-level HTN plan for winning the game in terms of previously learned subtasks. It applies the preference heuristics and replaces particular coordinates and directions with general spatial relations and relative directions. It simplifies the strategy by removing tasks whose outcome is never used, such as achieving preconditions of actions that are never taken.

Figure 10 shows the representation of a typical learned HTN method from a Micro-Rogue game. In this example, the method expands the winGame task to a sequence of subtasks with an empty (true) precondition. The structural vocabulary of such plans is derived from Cyc (that is, preconditionForMethod, methodForAction, actionSequence). Other predicates, such as achieve, winGame, runSequence, and gotoLocationOf are specific to our learner. True is a built-in predicate from GDL. The remaining predicates and entities are specific to the particular game and are simplified here for clarity.

Execution Flexibility

The learned HTNs need to be flexible enough that they are not overly sensitive to initial conditions or other specifics of the source game and don't need to be planned in detail from start to finish. We achieve this in two ways: by interleaving planning and execution on each turn and by learning high-level tasks with default conditional expansions.

When the game player executes a plan, it runs code associated with the primitive actions at the leaves of the HTN. Usually, these are game actions that perform a turn in the game. However, some primitives may store information in the knowledge base or even reinvoke the planner after executing a turn. This last operator, called *doPlan,* is key to supporting flexible execution. It allows the agent to plan at a high level and defer detailed planning of later turns until more information is available.

The domain learner incorporates this deferred planning in the plans it produces. This is straightforward because it really only learns three kinds of high-level HTN tasks: tasks to achieve a state, tasks to achieve the preconditions of an action, and top-level tasks to win an entire game. These achievement tasks have default methods that do nothing if the state or precondition is already satisfied (essentially a conditional plan). Learned sequences of these "achieve" tasks are themselves wrapped in a task called *runSequence* that encapsulates each subtask in *doPlan.* This makes the sequence execute incrementally by replanning after each step, in order to accommodate unforeseen effects of actions and adversarial responses.

The "achieve" tasks also guide the process by which an overly specific sequence of actions is turned into a hierarchical network of subtasks. When constructing a new HTN from an execution trace, if a subsequence is recognized as achieving the precondition of a primitive, that sequence is replaced with the (achieve (PreconditionFn <*primitive*>)) task. If a sequence is recognized as achieving one of the conjuncts of the overall game performance goal, it is replaced with the (achieve <*state*>) task. This makes the task hierarchical. One benefit of this is that if some action method does not transfer correctly, it can be rejected and relearned in the new game without necessarily rejecting the entire network.

Flexibility also stems from replacing some constants with variables, a process known as *lifting*. Specific coordinates are replaced with descriptions relative to entities, so that, for instance, an action to go to location 5,5 might be lifted to (LocationFn exit). *Exit* is an entity that will be analogically mapped and translated to the appropriate target entity, whereas specific coordinates cannot be. When an expression cannot be lifted without ambiguity, it is left unchanged until new experiments can resolve the ambiguity.

The learned HTNs are not guaranteed to be optimal. They may execute subtasks in an inefficient order or include unnecessary detours, such as picking up the gold in Micro-Rogue. The strategy learner uses heuristics to minimize this by omitting tasks that achieve preconditions that are never executed and by preferring strategies learned after at least three trials. By the third trial, it will have discharged some action-learning goals and a winning sequence will be less likely to have unnecessary actions in it. We define mastery as being able to win by following the learned strategy, not by simply repeating the exact sequence of actions.

This doesn't guarantee that the strategy will be lifted to its fullest generality, nor that the transfer process will correctly map all the entities, or that the strategy is even applicable in the target domain. Nevertheless, if enough constituent subtasks can be successfully transferred, it will result in learning the target domain faster than starting from scratch, if the underlying domains are in fact similar in how they operate.

To summarize, the static analysis of the rules and focused experimentation strategies typically allows the agent to master games within ten trials. It produces compositional strategies, which is important because source-target pairs are not isomorphs: Only some of the strategies might be applicable to a new game. For more detail on the domain learning, see Hinrichs and Forbus (2009).

Target Domain Learning in GGP

Target domain learning uses the same basic process as source domain learning except that in the transfer condition it takes as input one or more previously learned source games (see figure 7). After performing static analysis on the target game, it performs analogical mapping and transfer to propose HTN strategies from the source. When given multiple source domains, it attempts to compose the top-level strategies by concatenating their subtasks and trying them in both orders.

Because transfer is a defeasible inference, it records the provenance of transferred strategies. Transferred strategies that fail in the new domain are removed and replaced by learning goals. We discuss minimal ascension and metamapping next, and then describe the mechanics of the transfer algorithm.

Matching Nonidentical Predicates. Far transfer requires finding correspondences between nonidentical predicates. The tiered identicality constraint of structure-mapping theory allows nonidentical matches when needed to enable mapping a larger relational structure. That is, consider two statements S_1 and S_2. By structure mapping's structural consistency constraint, they can only align if their arguments can be aligned. Suppose that s_1 and s_2 are corresponding arguments of S_1 and S_2, respectively. It is worth relaxing identicality for s_1 and s_2, because if they can be made to match, a larger structure (that is, S_1 and S_2, and anything above them) might be made to match. The insight is that higher-order predicates, such as logical connectives, argument structure, planning and discourse relationships, and so on, are typically the same across a wide variety of domains. For example, while the contents of plans to win a particular game will involve relationships specific to that game, the relationships that describe the preconditions and consequences of tasks in the planner are the same for all games. Thus overlaps in higher-order structure provide a tight focus on what nonidentical matches involving lower-order structure should be considered. This is far more efficient than exploring the entire space of possible nonidentical predicate matches, given just the two initial descriptions. In essence, it reframes the problem as confirming whether or not a proposed nonidentical predicate correspondence should be allowed, rather than finding such correspondences a priori. We use two techniques: minimal ascension and metamapping. We discuss each in turn.

Minimal ascension (Falkenhainer 1988) operates during SME's initial stage of finding local match hypotheses, sanctioning nonidentical correspondences locally (see figure 11). It exploits hierarchical relationships between predicates. Let S_1 and S_2 be statements that SME has deemed useful to align, due to their being corresponding arguments in

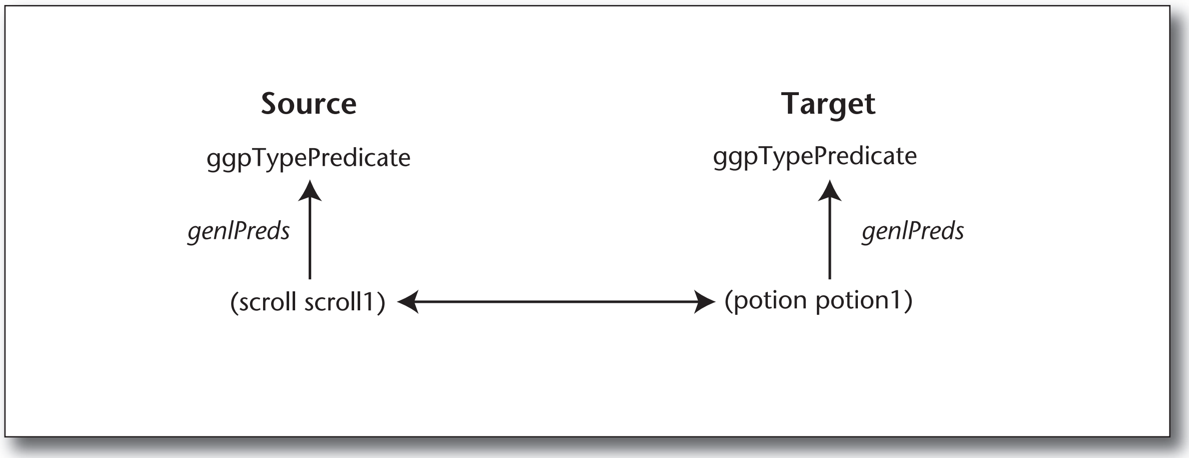


Figure 11. Minimal Ascension.

Scroll and potion are game-specific predicates. Static analysis determines that they define types and adds the genlPreds link to the parent predicate ggpTypePredicate, supporting the mapping between nonidentical predicates.

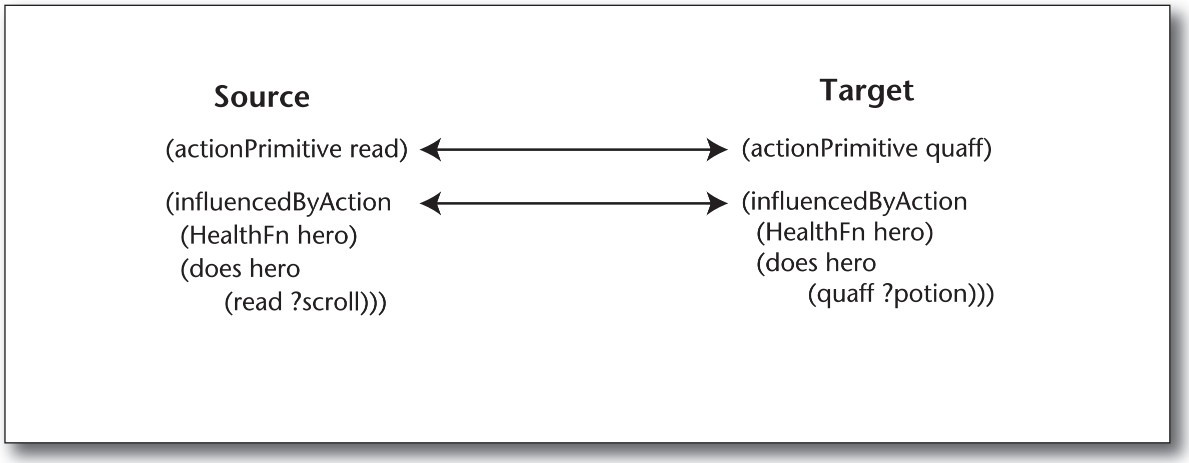

Source
(actionPrimitive read)
(influencedByAction (HealthFn hero) (does hero (read ?scroll)))
Target
(actionPrimitive quaff)
(influencedByAction (HealthFn hero) (does hero (quaff ?potion)))

Figure 12. In Metamapping, the Game-Specific Action Predicates Read and Quaff Match Because They Are Treated as Entities Embedded in a Larger Relational Structure Constructed during Static Analysis.

Here, the structure represents the qualitative relation that the health quantity of the hero entity can be affected in some fashion by reading (or quaffing) something.

some larger potentially mappable structure. Further let P_1 be the predicate of S_1 and P_2 be the predicate of S_2. Minimal ascension allows aligning S_1 and S_2 if P_1 and P_2 have a close common superordinate. This corresponds to their having a common ancestor in the predicate abstraction hierarchy with a limited depth (= 2). For example, (genlPreds scroll ggpTypePredicate) means that the scroll predicate is a specialization of the more general ggpTypePredicate. The static analysis routines automatically classify relationships in a game description as instances of more general classes of predicates. These derived genlPreds relationships partition potential matches, through the advice that they provide to minimal ascension.

Metamapping can, in one sense, be viewed as a generalization of minimal ascension. Minimal ascension only uses one kind of information about predicates, their genlPreds relationships. Metamapping uses all available structural information about predicates to find which predicates are most alike, in terms of their structural properties (see figure 12). By structural properties, we mean what features they have, and what relationships hold between them and other predicates. In essence, a metamapping treats predicates like entities. The correspondences found by comparing the structural information about predicates are additional suggestions about possible nonidentical predicate matches that can then be applied to the original mapping. Most knowledge bases include such information about their predicates as part of their definitions. Here, since the predicates are entirely new, the structural information we rely on is that produced by the static domain analysis, as illustrated in figure 9. Predicate names have been simplified for clarity: for example, location is actually Mrogue-8-1-targetLocation.

Minimal ascension and metamapping are both defeasible heuristics that depend on relational structure to distinguish alternative correspondences. We initially assumed that the horn clauses in the game definitions could provide some of this structure, since they encode much of the meaning of predicates and entities. We found that this does not work well and that SME becomes swamped with irrelevant matches. Instead, the case used for analogy consists solely of the (nonrule) statements in the game description plus the elaborated statements from static analysis.

To measure the accuracy of minimal ascension and metamapping, we examined by hand the nonidentical predicate alignments for all of the 41 source/target (S/T) pairs used in the external evaluation scenarios described below. As shown in table 1, overall accuracy for minimal ascension was 75 percent, while overall accuracy for metamapping is 59 percent. This is not too surprising, given that minimal ascension operates over both the relational structure present in the original game descriptions and the structure derived from static analysis, while metamapping only operates on the derived structure. Since the two techniques treat predicates quite differently, they sometimes provide overlapping or conflicting results. When they conflicted, the correspondence provided by minimal ascension was almost always the correct one. Thus the substitution process prefers mappings from minimal ascension when available. While minimal ascension is more accurate, metamapping has better coverage, and so the two techniques are complementary.

	# S/T	Minimal Ascension		Metamapping	
		correct	%	correct	%
Escape	14	76	94%	190	72%
Rogue	10	158	72%	83	43%
Wargame	10	75	82%	28	74%
Differing	7	7	24%	20	42%
Overall	41	316	75%	321	59%

Table 1. Accuracy of Mapping Nonidentical Predicates.

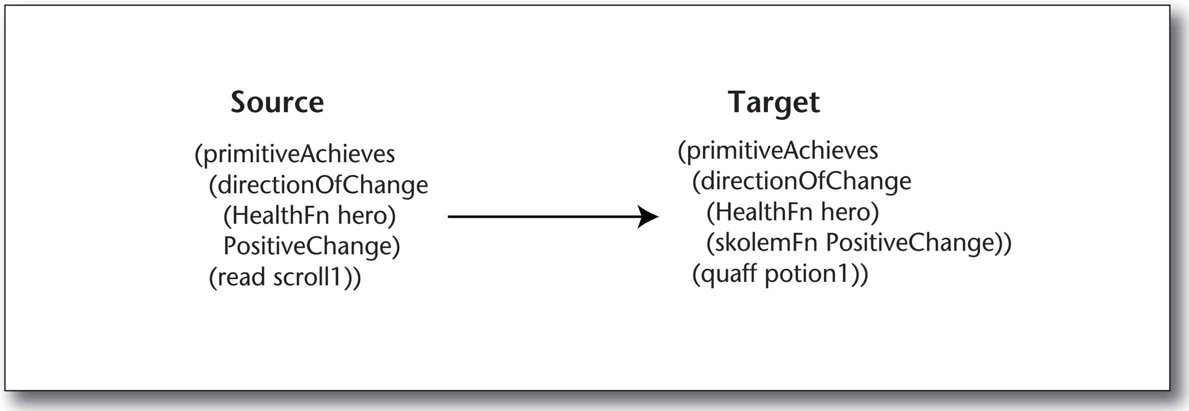
```
Input: base, a representation of the source game
       target, a representation of the target game
       Pr, a set of predicates relevant for transfer
Output: CI, a set of translated candidate inferences
1:  Elaborate base and target with structural information from
    static domain analysis
2:  M ⟵ best mapping of SME applied to base and
    target, using minimal ascension
3:  CI ⟵ {c ∈ CandidateInferences(M) | predicate(c) ∈ Pr}
4:  Translate non-identical predicates in CI into target
    vocabulary, preferring predicate alignments from minimal
    ascension over metamapping where available.
5:  Resolve skolems in CI
```

Figure 13. The Far Transfer Algorithm.

The Far Transfer Algorithm

Figure 13 describes the analogical transfer algorithm. The base and target are the game representations, which typically consist of 500 to 1000 assertions each. Pr is the set of predicates to be transferred. In this case, it is the predicates used to represent the learned types and relationships (for example, *threat,* or *obstacle*) and HTN strategies (for example, *preconditionForMethod*). The static domain analysis, described earlier, reifies the predicates in the domain as entities (for example, figure 9). Including these statements about predicates means that the SME mapping computed in step 2 includes the metamapping as well as the domain mapping itself. (They cannot interfere because the metamapping treats the predicates as reified entities, as opposed to functors.) Step 3 extracts the relevant candidate inferences by selecting only those which involve learned knowledge. Step 4 is required because the knowledge to be transferred must be expressed in the predicate vocabulary of the target domain. Step 5 operationalizes the candidate inferences.

Candidate inferences can include analogy skolems, defined as entities whose existence is hypothesized in the target through inference, but whose actual identify is unknown. Figure 14 provides an example, where the target statement must include something like PositiveChange from the base. The process of skolem resolution figures out what such entities are. The general problem of skolem resolution remains one of the frontiers of analogical reasoning research. Here we use a set of rules that are appropriate for GGP. Specifically, we replace variables with variables, and integer constants with the same integer constants. Global constants, such as PositiveChange or CardinalDirection, resolve to themselves. Finally, skolems of locations are resolved relative to any entities in



Source

```
(primitiveAchieves
  (directionOfChange
    (HealthFn hero)
    PositiveChange)
  (read scroll1))
```

Target

```
(primitiveAchieves
  (directionOfChange
    (HealthFn hero)
    (skolemFn PositiveChange))
  (quaff potion1))
```

Figure 14. A Candidate Inference Is the Plausible Projection of Learned Knowledge from the Source Game to the Target.

The predicates and entities in the source expression are translated to the target vocabulary using the structure mapping.

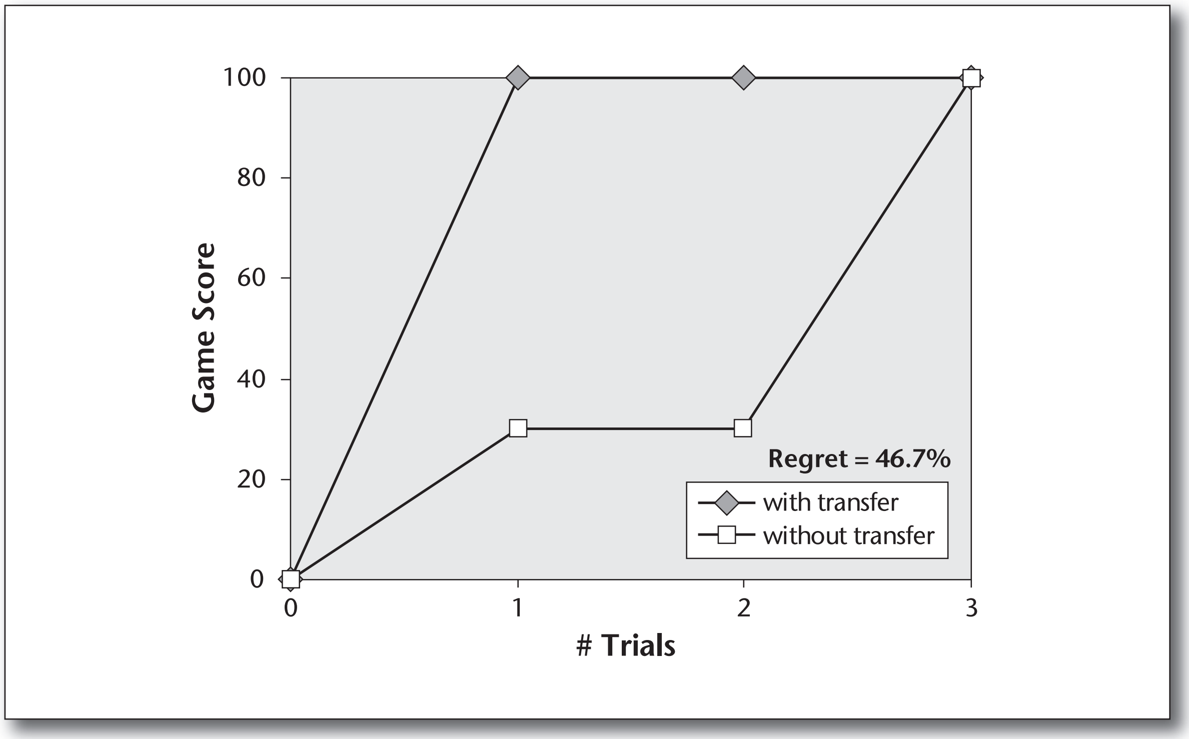


Figure 15. Example Learning Curve.

Regret measures the difference in area under learning curves normalized by the bounding box.

those locations and the locations of their corresponding entities in the target.

Far Transfer Experiment

To show that far analogical transfer improves learning performance, an external evaluation was conducted. A set of experimental scenarios was developed by an independent evaluator that included one or more source (training) games and a target game. The final games were not provided until the beginning of testing, when our system was frozen.

Each target game was learned twice, first without prior learning on the source game (the no-transfer condition) and then again after learning the source game (the transfer condition). Transfer was measured in these experiments by a normalized *regret* score, which characterizes how much improvement transfer makes. Figure 15 provides

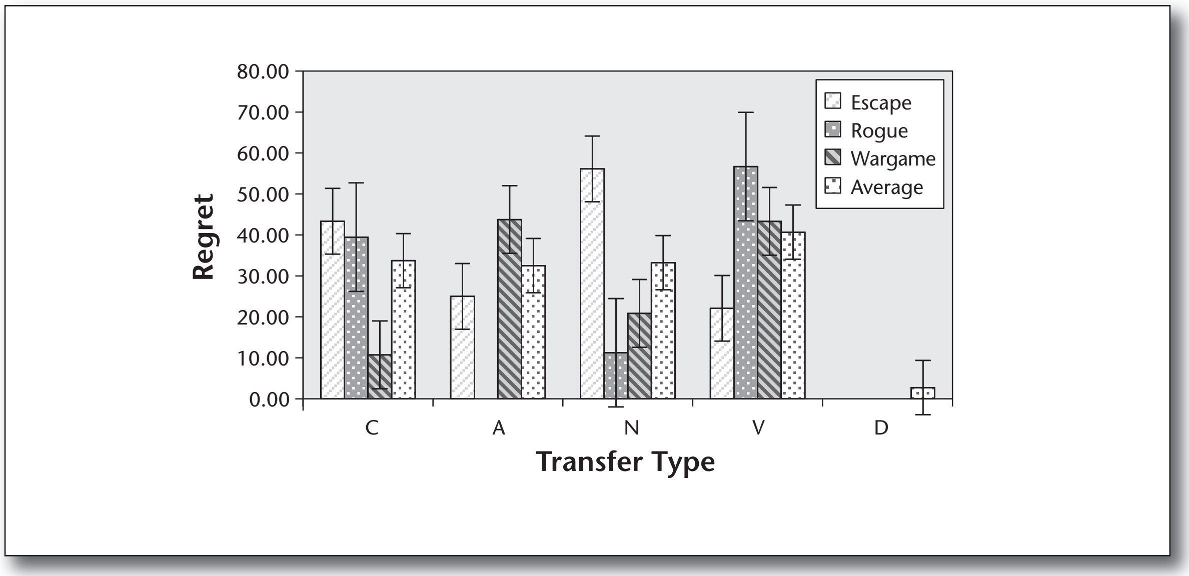


Figure 16. Regret Scores by Transfer Type.

an example. The horizontal axis shows the trial number, stopping when the no-transfer condition first wins a game, or after 10 trials, whichever comes first, while the vertical axis is the maximum game score reported so far at the end of each trial. The regret score is the difference in area under the two learning curves, normalized by the area of the bounding box. A higher regret score means higher performance was attained sooner under the transfer condition, that is, transfer was beneficial. Negative regret scores indicate negative transfer, that is, the prior experience confused or impeded target learning.

The games within a scenario were designed to test how well transfer occurred with particular relationships between the source and target games (DARPA 2005). Transfer condition type C required composing elements of two source domains. Game pairs in type A shared common abstractions. In type N, source and target contained different numbers of elements. The type V condition used games with different vocabularies of predicate and entity names. Finally, transfer condition D drew source and target games from entirely different game domains.

Figure 16 shows results over the entire set of scenarios, broken down by game domain and transfer type. Each bar represents the average regret over 1–3 scenarios. Note that there were no common abstraction (A) scenarios for Micro-Rogue (that is, this is a lack of data, not a zero regret score). The data presented here represents 34 target scenarios, consisting of 329 target game trials. The average regret over all 34 scenarios was 27.38, indicating a clear benefit from transfer learning.

Where did it do well and where did it do poorly? As shown in figure 16, the regret scores were positive for all transfer types tested, though there was a sharp drop off for transfer across different game types (D). The main problem with the differing experiments was that, in an effort to ensure that there was some common strategy that could be transferred, the source games were modified slightly in ways that tended to break source learning. For example, source game 2 introduced a "fireball" scroll to a Micro-Rogue-like scenario that was intended to be similar to a grenade. Because our learning agent didn't have learning goals and experiments for learning about thresholds in time and space, it could not master that source game. In general, the source games for the differing scenarios were much harder than for other transfer types, thus conflating difficulty of learning with transfer distance. Ultimately, only one out of the eight differing source games was mastered, so there was nothing to transfer for seven of them. When the source game could be mastered, the transfer process worked remarkably well, as evidenced by the results in the differing vocabulary condition (V), where games were structurally isomorphic, but predicates and entities were completely renamed.

In addition to limitations of the source learning, the other problem we saw was the occasional cross-mapping of predicates between source and target. For example, in some Micro-Rogue games, *armor* in the source game was mapped to *weapon* in the target. This happened whenever there was no structural reason to prefer one mapping over another, in this case, because there were no *wield* or *putOn* actions. The abstract nature of the games and impoverished perceptual representation made such cross mappings more likely, but did not usually cause problems in execution.

Related Work

Other researchers in transfer learning have approached the problem differently. For example, Peter Stone's group has focused on reinforcement learning and transferring evaluation functions (Banerjee and Stone 2007), including using their own version of SME for near transfer on domain pairs of their own choosing (Liu and Stone 2006). ICARUS also treats transfer as an analogy problem, using a domain-specific matcher incorporating structure-mapping principles (Shapiro, Könik, and O'Rorke 2008). Their matching process walks a solution trace, reexplaining it in terms of target domain concepts in order to build up a table of correspondences. A different approach is to treat transfer as a problem of speedup learning, employing chunking in Soar to reduce search time.[4] Tamar (Mihalkova, Huynh, and Mooney 2007) describes a predicate mapping technique for Markov logic networks, using three source/target pairs, each involving a smaller number of relationships per pair than the games described here. We suspect that metamapping could be used to replace their exhaustive cross-predicate initial step for improved efficiency.

Two cross-domain analogy efforts are also very close to this work. The most similar cross-domain analogy work is that of Falkenhainer's (1988) Phineas system, which used SME to map distant domain theories based on correspondences derived from mapping observed behaviors. Klenk and Forbus (2007) use SME to map distant domain theories based on mapping worked solutions to problems. Like them, we use minimal ascension to help find nonidentical predicate alignments. Our source of overlapping knowledge is different, that is, the automated domain analysis, and our metamapping technique is also novel.

A related approach to learning HTNs is described by Hogg, Munoz-Avila, and Kuter (2008). Their algorithm, HTN-MAKER, takes classical planning operators and successful plan traces to construct hierarchical task networks. It has the benefit of being formally defined and can be shown to be complete and correct. Our source domain learner, however, addresses a somewhat different learning task in which the effects of actions and the behaviors of adversaries are incompletely known. Our approach therefore is necessarily more heuristic and defeasible in nature.

Conclusions

A central problem in transfer learning is handling far transfer, where the representational vocabulary used in the domains can differ significantly. We have described a new technique, metamapping, which exploits both minimal ascension and structural information about predicates to attain reasonably accurate cross-domain mappings. The experiment was an important milestone in far transfer research, since it marks one of the first times that an analogical transfer system has successfully operated over novel representations provided by an external organization, and on more than a handful of source/target pairs. Thus it provides strong evidence for the utility of this technique.

There are several future directions that we plan to explore. First, we want to test the transfer algorithm and metamapping over a broader range of types of domains, and over different styles of domain learning. Of particular interest are when substantial background knowledge is involved, which we suspect will greatly improve the traction that metamapping provides. Second, our system currently handles inaccuracy in transfer by completely discarding transferred knowledge that is found to be inaccurate through experimentation. A better approach might be repairing the mapping, so we plan to investigate strategies for diagnosing and repairing mapping failures. Third, for some domain pairs, a mapping may only be possible with a more complex transformation than substitutions (for example, coordinate transformations, reification, etc.) Yan, Forbus, and Gentner's (2003) theory of rerepresentation looks promising for handling such situations, but it has not to our knowledge been embodied in any performance-oriented system. Finally, the problem of retrieval in transfer learning needs to be addressed. Except for Falkenhainer (1988) and Klenk and Forbus (2007), all far transfer learning experiments that we are aware of provide the source domain as an input. Retrieving a relevant source from a broad corpus of experience will be a necessary capability for any realistic robust transfer learning system. Importantly, retrieval in far transfer is known to be extremely difficult for people, much more difficult than mapping once a source has been found. It could be that the reasons people are limited in this way will apply to any intelligent system, or it could be that, given different engineering constraints, better-than-human solutions for retrieval could be found. This is an exciting open question for transfer learning research.

Acknowledgements

This research was performed under the DARPA Transfer Learning Program. We would like to thank David Aha and Matthew Molineaux for their work in evaluating the system and Jeff Usher for programming assistance.

Notes

1. See research.cyc.com.
2. See freeciv.wikia.com/wiki/Main_Page.

3. Prototype versions of these games were developed by transfer learning researchers: Escape by John Laird's group at University of Michigan, Wargame by the ISLE group at Stanford, and Micro-Rogue by Hinrichs at Northwestern and David Aha at NRL.

4. Personal communication with John Laird in 2008.

References

Banerjee, B., and Stone, P. 2007. General Game Learning Using Knowledge Transfer. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence,* 672–677. Menlo Park, CA: AAAI Press.

Bryce, D., and Kambhampati, S. 2007. A Tutorial on Planning Graph-Based Reachability Heuristics. *AI Magazine* 28(1): 47–83.

Defense Advanced Research Projects Agency (DARPA). 2005. Transfer Learning Proposer Information Pamphlet (PIP) for Broad Agency Announcement 05-29. Washington, DC: United States Department of Defense.

Erol, K.; Hendler, J.; and Nau, D. 1994. HTN Planning: Complexity and Expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,* 1123–1128. Menlo Park, CA: AAAI Press.

Falkenhainer, B. 1988. Learning from Physical Analogies: A Study in Analogy and the Explanation Process. Ph.D. diss., Department of Computer Science, University of Illinois at Urbana-Champaign. Technical Report UIUCDCS-R-88-1479.

Falkenhainer, B.; Forbus, K.; and Gentner, D. 1989. The Structure Mapping Engine: Algorithm and Examples. *Artificial Intelligence* 41(1): 1–63.

Forbus, K.; Ferguson, R.; and Gentner. D. 1994. Incremental Structure Mapping. In *Proceedings of the 16th Annual Meeting of the Cognitive Science Society,* 313–318. Wheat Ridge, CO: Cognitive Science Society.

Forbus, K.; Gentner, D.; and Law, K. 1995. MAC/FAC: A Model of Similarity-Based Retrieval. *Cognitive Science* 19(2): 141–205.

Forbus, K.; Klenk, M.; and Hinrichs, T. 2009. Companion Cognitive Systems: Design Goals and Lessons Learned So Far. *IEEE Intelligent Systems* 24(4)(July/August): 36–46.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science* 7(2): 155–170.

Genesereth, M.; Love, N.; and Pell, B. 2005. General Game Playing — Overview of the AAAI Competition. *AI Magazine* 26(2): 62–72.

Hinrichs, T., and Forbus, K. 2007. Analogical Learning in a Turn-Based Strategy Game. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence,* 853–858. Menlo Park, CA: AAAI Press.

Hinrichs, T., and Forbus, K. 2009. Learning Game Strategies by Experimentation. Paper presented at the IJCAI-09 Workshop on Learning Structural Knowledge from Observations. Pasadena, CA, July 12.

Hogg, C.; Munoz-Avila, H.; and Kuter, U. 2008. HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence,* 950-956. Menlo Park, CA: AAAI Press.

Klenk, M., and Forbus, K. 2007. Cross-Domain Analogies for Learning Domain Theories. In *Analogies: Integrating Multiple Cognitive Abilities,* Volume 5-2007, ed. Angela Schwering. Osnabrück, Germany: Institute of Cognitive Science.

Liu, Y., and Stone, P. 2006. Value-Function-Based Transfer for Reinforcement Learning Using Structure-Mapping. In *Proceedings of the 21st National Conference on Artificial Intelligence,* 415–420. Menlo Park, CA: AAAI Press.

Mihalkova, L.; Huynh, T.; and Mooney, R. 2007. Mapping and Revising Markov Logic Networks for Transfer Learning. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence,* 608–614. Menlo Park, CA: AAAI Press.

Shapiro, D.; Könik, T.; and O'Rorke P. 2008. Achieving Far Transfer in an Integrated Cognitive Architecture. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence,* 1325–1330. Menlo Park, CA: AAAI Press.

Yan, J.; Forbus, K.; and Gentner, D. 2003. A Theory of Rerepresentation in Analogical Matching. In *Proceedings of the 25th Annual Meeting of the Cognitive Science Society.* Wheat Ridge, CO: Cognitive Science Society.

Thomas R. Hinrichs is a research associate professor at Northwestern University. He received his B.S. from Cornell University and his Ph.D.from from Georgia Tech in 1991. His research interests include plausible reasoning, machine learning, and cognitive architectures. His email address is t-hinrichs@northwestern.edu.

Kenneth D. Forbus is the Walter P. Murphy Professor of Computer Science and Professor of Education at Northwestern University. He received his degrees from MIT (Ph.D. in 1984) and is a Fellow of the Association for the Advancement of Artificial Intelligence, the Cognitive Science Society, and the Association for Computing Machinery.

2012 Spring Symposium Series Proposals Solicited

AAAI is currently accepting proposals for the 2012 Spring Symposium Series, to be held at Stanford University, California, March 26–28, 2012.

Approximately eight symposia on a broad range of topics within and around AI will be selected for the 2012 Spring Symposium Series. All proposals will be reviewed by the AAAI Symposium Committee, (Chair: Chad Jenkins, Brown University; Cochair: TBD; and Associate Chair: Alan Schultz, Naval Research Laboratory). Symposium proposals should be submitted as soon as possible, but no later than April 15, 2011. The Call for Proposals is available at www.aaai.org/Symposia/Spring/sss12.php.

Please submit your symposium proposals by electronic mail (no postal submissions), and inquiries concerning symposia, to Chad Jenkins, Brown University, cjenkins@cs.brown.edu.