Available online at www.sciencedirect.com
ScienceDirect
Cognitive Systems Research 10 (2009) 270–285
Cognitive Systems RESEARCH
www.elsevier.com/locate/cogsys

# Skill transfer through goal-driven representation mapping

Action editor: Angela Schwering

Tolga Könik *, Paul O'Rorke, Dan Shapiro, Dongkyu Choi, Negin Nejati, Pat Langley [1]

*Computational Learning Laboratory, Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, USA*

Received 19 April 2008; accepted 19 September 2008
Available online 10 January 2009

**Abstract**

In this paper, we present an approach to transfer that involves analogical mapping of symbols across different domains. We relate this mechanism to ICARUS, a theory of the human cognitive architecture. Our system can transfer skills across domains hypothesizing maps between representations, improving performance in novel domains. Unlike previous approaches to analogical transfer, our method uses an explanatory analysis that compares how well a new domain theory explains previous solutions under different mapping hypotheses. We present experimental evidence that the new mechanism improves transfer over ICARUS' basic learning processes. Moreover, we argue that the same features which distinguish ICARUS from other architectures support representation mapping in a natural way and operate synergistically with it. These features enable our analogy system to translate a map among concepts into a map between skills, and to support transfer even if two domains are only partially analogous. We also discuss our system's relation to other work on analogy and outline directions for future research.


*Keywords:* Analogy; Representation mapping; Cognitive architectures; Transfer; Transfer learning; Agent architectures

## 1. Introduction

Many computational learning methods require far more training instances than humans to achieve reasonable performance in a domain. A key reason is that humans often reuse knowledge gained in early settings to aid learning in ones they encounter later. This phenomenon is known as *transfer* in cognitive psychology, where it has received far more attention than in AI and machine learning.

Much of this research has focused on transfer of complex skills for tasks that involve action over time (e.g. Kieras & Bovair, 1986; Singley & Anderson, 1988). This paper reports on a computational approach to transfer that takes a similar perspective. We focus on the acquisition of cognitive skills from experience and on how transfer improves behavior on distinct but related tasks.

We share with many psychologists the idea that transfer is mainly a structural phenomenon. This suggests that transfer is linked closely to how an agent represents knowledge in memory, how its performance methods use these structures, and how its learning elements acquire this knowledge. Theoretical commitments to representation, performance, and learning are often associated with the notion of a cognitive architecture (Newell, 1990). Thus, it seemed natural for us to study transfer in the context of ICARUS (Langley & Choi, 2006), an architecture that takes positions on each of these issues.

In previous research (Choi, Könik, Nejati, Park, & Langley, 2007), we showed that, ICARUS is natively capable of producing *near* transfer due to its commitments to relational, hierarchically composable, and goal-indexed knowledge representations, plus its mechanisms for using and acquiring such structures. This form of transfer is due to

* Corresponding author.
*E-mail addresses:* konik@stanford.edu (T. Könik), paul@ororke.com (P. O'Rorke), dgs@csli.stanford.edu (D. Shapiro), dongkyuc@stanford.edu (D. Choi), negin@stanford.edu (N. Nejati), langley@csli.stanford.edu (P. Langley).
[1] Current address: School of Computing and Informatics, Arizona State University, P.O. Box 87-8809, Tempe, AZ 85287, USA.

1389-0417/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cogsys.2008.09.008

the application of literally similar knowledge in multiple tasks, represented in ICARUS as generalized skills acquired by a flexible learning method.

However, literal similarity does not explain all transfer phenomena. Humans often reason analogically between situations that are very different in the surface but contain structural correspondences apparent in statements such as "an electric battery is like a reservoir". Structural mapping theory (Gentner, 1983) explains this kind of analogy by detecting structural correspondences between two domains. This paper investigates a related approach. Our focus is to explain *far* transfer, defined as transfer between domains that employ different symbols to represent relations among objects. We introduce a new mechanism for goal-directed analogical reasoning that identifies mappings between such domains, and we use that mapping to port both conceptual and procedural knowledge into a new setting. We combine this mechanism with features of ICARUS, resulting in a system that acquires skills from experience in one problem domain and employs them in another, despite significant differences in vocabulary.

More specifically, we develop a novel algorithm for representation mapping, called GAMA (goal-driven analogical mapping) that identifies correspondences between relational representations through explanatory analysis. This process is mainly guided by pragmatic (i.e. related to the purpose of analogy) considerations, using the terminology of Holyoak and Thagard (1989), but it is also biased to satisfy structural and semantic constraints as defined by Gentner (1983). We claim that:

- Goal-directed analogy enables far transfer in a cognitive system.
- The same features that distinguish ICARUS from other cognitive architectures support goal-directed analogy in a natural way.

We support the first claim via computational experiments on far transfer tasks. In particular, we demonstrate a significant increase in problem-solving speed due to mapped knowledge, and then present the results of a lesion study that shows the speedup is due to the GAMA algorithm. We support the second claim by illustrating the dependence of GAMA on core features of the ICARUS architecture:

- two separate long-term memories for procedural (skills) and declarative knowledge (concepts);
- skills indexed by the goals they achieve (thus associating all procedural knowledge elements with declarative knowledge elements); and
- hierarchical and composable representations for both skills and concepts.

These features work synergistically with GAMA by letting it translate a map among concept predicates into a map between skills, and by supporting far transfer even in the case of partial analogy.

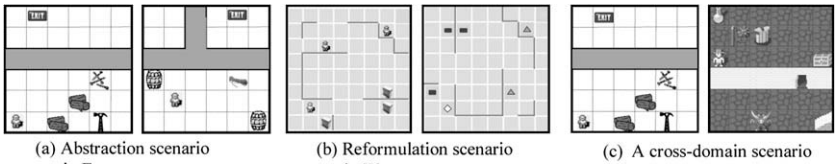
We elaborate on these ideas in the pages that follow. The next section presents an experimental testbed for computer games that illustrates the benefits of reusing learned knowledge structures. In Section 3, we review ICARUS' assumptions about representation, performance, and learning, and how these assumptions support near transfer. In Section 4, we describe our representation mapping algorithm, GAMA, showing how it transfers ICARUS concepts and skills across domains. In that section, we also discuss how ICARUS' architectural commitments help GAMA to support transfer. In Section 5, we evaluate our claims using specific experiments with game playing domains. We conclude by reviewing related efforts on structural transfer (Section 6), describing our priorities for future research on this topic (Section 7), and stating our conclusions (Section 8).

## 2. The transfer setting

Our research has studied transfer via a collection of challenge problems phrased as source-target pairs. Here, the object is to solve the source problem, extract transferable knowledge from that experience, and demonstrate that these structures improve the agent's ability to solve the target problem. We measure transfer by comparing the time required to solve the target problem with and without exposure to the source.

We have employed the general game playing (GGP) framework (Genesereth, Love, & Pell, 2005) to structure this task. GGP encodes tasks (typically games) in a first-order logic language that employs separate theory elements to describe legal moves, state transitions, and the initial state associated with game instances. GGP also enforces a careful evaluation model by presenting game rules at the same time as game instances. This requires agents to employ general mechanisms for performance and learning, while constraining the role of background knowledge.

We have studied three types of far transfer tasks, characterized by the nature of the analogy within each source-target pair. Abstraction tasks admit a one-to-one correspondence between symbols denoting objects and/or relations, and allow elements in the source with no target corollary. Fig. 1a gives an example drawn from a game called Escape, where the agent's goal is to direct the explorer to an exit. The source task requires nailing logs together to create a bridge over the river, while the target requires tying barrels together with rope. The predicates for nailing and tying differ from source to target and while the logs correspond to the barrels, the hammer has no target corollary. This makes transfer difficult because the agent must discover the mapping between source and target symbols and abstract away the parts of the source knowledge (e.g., knowledge about hammer) that is not applicable in the target.



(a) Abstraction scenario in Escape (b) Reformulation scenario in Wargame (c) A cross-domain scenario from Escape to mRogue

Fig. 1. Source-target scenario pairs.

Reformulation scenarios consist of isomorphic problems created by consistently replacing all source symbols to obtain the target task. Fig. 1b gives an example taken from a domain called Wargame, where the goal is to maneuver the soldier to the exit while avoiding/defeating enemies. The enemies actively seek the soldier and move twice as fast, but they can become stuck at walls. Supply points contain weapons and ammunition. These transfer problems are difficult because the agent must discover a deliberately obscured source-target relation.

Finally, cross-domain scenarios draw the source and target problems from different games. Fig. 1c shows an Escape to mRogue example (after the ancient text game), where the agent gets points for gathering treasure, defeating monsters, and exiting the playing field. These scenarios do not deliberately support analogies, but all games occur on 2D grids and involve reaching a goal after surmounting obstacles by collecting and employing appropriate tools. Transfer pairs in this class are difficult because the agent must identify both the symbol mapping and the portions of the source solution that are preserved.

## 3. The ICARUS architecture

ICARUS is a cognitive architecture in the tradition of work on unified theories of cognition (Newell, 1990). As such, it provides general-purpose reasoning mechanisms and representation. We discuss these components in the traditional order, beginning with the key representational elements (concepts and skills), followed by the operations that employ them (inference, execution, and learning). We place these components in context at the section's end by describing the capacity for near transfer they naturally support.

### 3.1. Representation of concepts and skills

The ICARUS architecture makes several commitments about knowledge representation. First, it supports two distinct structures to represent tasks; concepts describe situations in the environment while skills describe how to achieve such situations. Both concepts and skills are organized in hierarchies, which consist of primitive structures in the terminal nodes and nonprimitive ones in the internal nodes. This means that ICARUS can employ multiple layers of abstraction to describe the current state and the procedures for manipulating that state.

ICARUS' long-term conceptual knowledge is described by *concept clauses*that resemble traditional Horn clauses in first-order logic (Table 1). An ICARUS agent uses primitive concepts like location to describe symbolic and numeric features of the perceived state. Higher-level concepts such as atExit are defined in terms of other concepts and enable the agent to form beliefs (concept instances) given other concept instances. Since concepts can also represent features of desired states, ICARUS uses them to describe the goals.

ICARUS' skills and concepts correspond to hierarchical task networks like those in SHOP2 (Nau, Cao, Lotem, & Muñoz-Avila, 1999) with one additional constraint; the skills are indexed by the goals they are intended to achieve. All skill clauses (Table 2) have a head, which indicates the skill's goal, and start conditions, which describe the situation when it is applicable. *Primitive skills* like location in Table 2, achieve their goal by executing an action, move in this case, while *nonprimitive skills* specify more complex activities that involve decomposing the skill into subskills (e.g., atExit is decomposed into subskills such as holding).

It is important to note that the skills and concepts contain different kind of information, even though ICARUS uses

Table 1
Example ICARUS concepts.

| *Primitive Concept* | *Nonprimitive Concept* |
|---|---|
| (location explorer ?x ?y)<br>:percepts<br>(entity explorer x-cord ?x<br>y-cord ?y) | (atExit ?agent)<br>:relations<br>(location exit ?x ?y)<br>(location ?agent ?x ?y) |

Table 2
Primitive and non-primitive ICARUS skills.

*Primitive Skills*

```
(location explorer ?x ?y)
 start: (location explorer ?x1 ?y)
        (add ?x ?x1 1)
        (direction east)
        (nextToExplorer east ?x ?y)
        (not (blocked east ?x1 ?y))

 actions: (move explorer east)
```

*Nonprimitive Skills*

```
(atExit ?agent)
  start:     (obstacleType ?Obstacle ?Type)
             (location ?Obstacle ?Xo ?Yo)
             (canCompromise ?Property ?Type)
             (not (destroyed ?Obstacle))
  subgoals:  (property ?Item ?Property)
              % constructs an item
              % that can overcome ?Obstacle
             (holding ?Item)
             (nextToExplorer ?X3 ?Xo ?Yo)
             (atExit ?agent)

(atExit ?agent)
  start:     (location exit ?X ?Y)
  subgoals:  (location ?agent ?X ?Y)
```

the same predicate symbols to refer both. A concept *p* defines what *p* is and when its instances can be inferred, while the skill *p* is a *constructive definition* that describes how instances of *p* can be achieved or created through actions. For example, while the atExit concept (Table 1) defines when an agent is at an exit location, the atExit skill (Table 2) describes how to bring the agent to that exit location.

ICARUS skill clauses refer to concepts in their head, start conditions, and subgoals. The SHOP2 formalism also refers to defined predicates in its methods' preconditions, but their heads and bodies refer to arbitrarily named tasks and subtasks. ICARUS' use of goals (desired concepts) in their place plays important roles in execution and learning.

### 3.2. Execution of hierarchical skills

The ICARUS architecture operates in cognitive cycles, spanning conceptual inference, skill selection, and physical execution (Fig. 2). The system derives its beliefs via a bottom-up inference process, initiated by object descriptions that arrive in the agent's perceptual buffer. After it infers primitive beliefs from these percepts, inference over higher-level concepts generates additional beliefs. In contrast, ICARUS performs skill selection in a top-down manner, starting with the unsatisfied goal having the highest priority. On each cycle, it finds a path from this goal downward through the skill hierarchy; each skill along this path must have an unsatisfied head and be applicable given current beliefs, with the terminal node being a primitive skill that ICARUS executes in the environment. This differs from traditional production-systems (Neches, Langley, & Klahr, 1987), which require multiple cycles and use of working memory to traverse levels of an implicit goal hierarchy.

Since the architecture repeats this procedure on each cycle, agents combine reactivity with goal-driven behaviors. ICARUS is goal-driven because an applicable skill path corresponds to an abstract partial plan for achieving the goal. The system is reactive because it reevaluates skill path on each execution cycle. Moreover, there is a bias towards persistence because the execution module prefers portions of the path selected on the previous cycle as long as they are still applicable.

### 3.3. Learning hierarchical skills

Nejati, Langley, and Könik (2006) reported an analytical method to learn hierarchical skills. This mechanism inputs the goal and a solution trace that achieves it, stated as a sequence of observed states and actions that produce

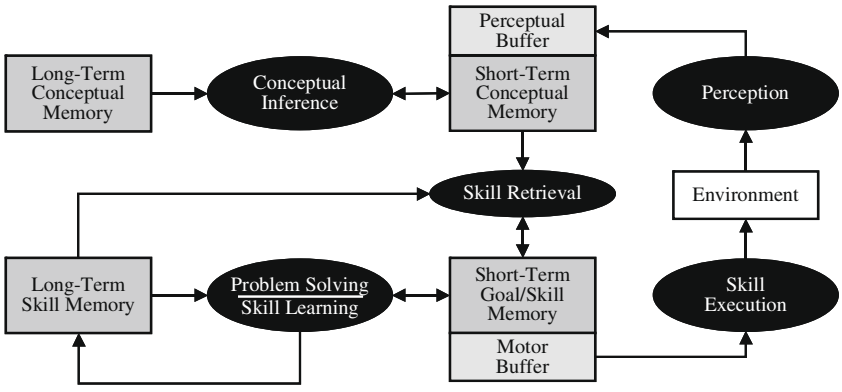



Fig. 2. A schematic of memories and processes in the ICARUS architecture.

them. The mechanism interprets the solution trace in the context of the system's conceptual knowledge and primitive skills and explains how the solution path produced the desired goal. Then, it outputs a skill hierarchy for achieving that goal.

The learning component starts it analysis with the top-level goal and explains it with a recursive procedure, where a set of goals are either decomposed into subgoals using conceptual definitions or explained in terms of the effects of primitive skills. The system then applies the same procedure recursively on the subgoals and the corresponding parts of the solution trace, until it generates all the explanations necessary for the overall task. The architecture transforms the resulting explanation structure into hierarchical skills and adds them to skill memory for future use.

This learning method is related to previous techniques for explanation-based learning (Segre, 1987), but instead of compiling away the explanation structure, it uses this to determine the structure of the skill hierarchy. We reported another approach (Langley & Choi, 2006) for learning skill hierarchies that relies on means-ends problem-solving instead of an explanatory analysis of solution traces, although it requires more search through the problem space.

### 3.4. Near transfer in ICARUS through generalized skills

In our previous research (Choi et al., 2007), we demonstrated ICARUS' near transfer capabilities on Urban Combat, a real-time, first-person perspective game.[2] We claimed that the architecture achieves this effect due to its use of *relational*, *hierarchically composable*, and *goal-indexed* skills, along with mechanisms for using and acquiring them. These representational commitments mean that skills exist as independent knowledge elements that can be composed in novel ways. The learning algorithm just described utilizes this capability by acquiring many small skills that will be composed at run-time as opposed to creating a single macro with a long sequence of actions (Mooney, 1990). ICARUS' flexible execution mechanism supports this ability by making skill decomposition decisions at run-time based on observed situations.

ICARUS' relational representation facilitates transfer by increasing generality of the encoded skills such that they apply in circumstances that are only qualitatively similar to the situations that led to their acquisition. Learning supports this generality by employing relational background knowledge. As a result, the learned skills reference the system's conceptual vocabulary, letting ICARUS retrieve them more flexibly during execution. For example, suppose an agent has acquired a skill that has the goal of reaching a target location. Its start conditions will indicate that this location is collectively-blocked (a background concept that matches when multiple objects collectively impede a path), and it will have subgoals for reaching the closest blocking object and trying to overcome it (e.g., by jumping over it). The generality of the collectively-blocked concept lets the agent apply this learned skill in novel situations where multiple objects collectively block a path in very different configurations.

ICARUS' goal-indexed skill representation facilitates transfer by enabling the skills learned on different occasions to be composed in new situations in novel ways. In cases where learned skills are partially incorrect or inaccurate given the current situation, the execution module can dynamically replace the unusable subskills with others or it can return control to alternative higher-level skills. For example, suppose that ICARUS is executing a navigation skill for reaching a target location in a partial information game. When the agent encounters an unexpected impasse such as a destroyed bridge, its source navigation skill for traversing that region may become inapplicable. Given alternate learned skills, the agent could try to achieve the same subgoal by constructing a raft, or by falling back to default exploration skills that search for an alternative path to a goal location.

## 4. Far transfer through representation mapping

Although the architectural mechanisms described in the previous section account for interesting transfer phenomena, ICARUS does not exhibit far transfer in cases where the source and target domains are represented with different symbols. This section describes a new method for representation mapping and transfer that enables better reuse of knowledge in such settings.

Analogical mechanisms are often described in terms of distinct steps. Here, we extend Thagard, Holyoak, Nelson, and Gochfeld's (1990) decomposition by including an additional first and fifth step:

1. Performance and learning in the source domain.
2. Retrieving or selecting plausibly useful source knowledge.
3. Mapping between the source and the target domains.
4. Translating source information for appropriate use with the target.
5. Performing with the transferred knowledge.
6. Subsequent learning in target domain.

We have developed a new representational transfer algorithm called GAMA (Goal-driven Analogical Mapping) to support far transfer by handling the mapping and translation tasks identified above (steps 3 and 4). We use GAMA to transfer executable knowledge, and demonstrate it in a setting (see Section 5) that includes performance and learning in the source domain as well as performance in the target domain (steps 1 and 5, above). GAMA consists of two parts: the representation mapper finds the correspondences between source and target symbols, while the representation translator uses those correspondences to translate source skills and concepts into skills and concepts in the

[2] Details of this game are available at http://www.urban-combat.net.

target domain. The following subsections describe these components of GAMA in more detail. Note that we have not fully integrated GAMA into the ICARUS architecture, so we make no commitments about how and when the retrieval of relevant past experiences (step 2 above), and transfer should occur in general.

The input of GAMA consists of source and target domain theories[3], a solution for a source task, and source skills learned from that solution. The output comprises target skills and concepts constructed from transferred source knowledge. The new transfer mechanism uses skills learned in the source domain to improve performance in the target domain. To this end, the *representation mapper* finds the correspondences between source and target symbols, while the *representation translator* uses those correspondences to translate source skills and concepts into skills and concepts in the target domain. In the following subsections, we describe these components of GAMA in more detail.

### 4.1. Representation mapping

The intuition behind the representation mapper is that there is an opportunity for analogy if one can explain the solution for a source problem in target terms. In overview, the algorithm employs a source problem and an explanation of how the source problem is solved as input. Next, it generates an analogous explanation from the perspective of the target domain theory by matching target theory clauses against the source explanation. This first forges links between the source and target theories, then returns as output correspondences between concept predicates (and objects) of the source and target theories.

In more detail, the input consists of *source and target domain theories*, *source solution*, and *learned source knowledge*. The domain theories describe the initial state, rules and goal using ICARUS concepts and primitive skills.[4] The source solution consists of a goal instance and a sequence of action-state pairs achieving that goal. Finally, learned source knowledge contains an *explanation* of the source solution and *learned source skills* automatically generated by the learning system described in Section 3.3. The source explanation is an intermediate structure produced by this learning method consisting of a directed graph in which the nodes are facts that were relevant for achieving the source goal and the links represent their justification in terms of source theory clauses. The output is *a representation map* – a set of source-target predicate (or object) correspondences that can convert source skills and concepts to the target representation. GAMA hypothesizes correspondences between source and target predicates by aligning target theory clauses against the source explanation. The system starts this process by forming a correspondence between the source and target goals. Next, GAMA expands the source goal using the explanation for how it was achieved, and the target goal using the target theory for how it might be achieved. GAMA then searches for mappings among predicates and objects that let the target theory match onto the source explanation.

GAMA's search is guided by constraints and heuristics described in the next section. Whereas hard constraints cause GAMA to backtrack, heuristics and soft constraints determine the order of search through the space of correspondences and the preference among alternative mappings. Once GAMA finds a consistent alignment, it returns as output a *representation map M*, which is a hypothesized set of correspondences between the source and target concept predicates such that, given $M$, the target theory can explain the source solution in a similar fashion to the given source explanation. GAMA also facilitates transfer between domains that are analogous at an abstract level by allowing partial mappings that do not perfectly align the source explanation with the target theory. However, it tries to minimize the amount of abstraction with a bias that prefers mappings which cover as much of the source explanation as possible.

Table 3 presents an example from an alignment process. In overview, GAMA expands a given correspondence into antecedents using the explanation on the source side and the domain theory on the target side, creating an opportunity to establish new mappings. The alignment starts with the correspondence (atExit explorer) ↔ (atGoal hero) between the top-level goals. On the target side, GAMA unifies the goal instance (atGoal hero) with the target concept clause (atGoal ?agent), which it will expand later to continue the mapping process. It matches those subconcepts against the antecedents of (atExit explorer) in the source explanation. Consequently, GAMA generates a new set of correspondences between the antecedent pairs such as (location explorer 5 0) ↔ (place hero ?X ?Y), which it will expand later. Note that, in this Table, the antecedent correspondence (not (blocked east 4 0)) ↔ true exemplifies a case when GAMA could not construct a perfect alignment and leaves the source explanation node without a target correspondence.

Table 4 presents the pseudo-code for the GAMA algorithm. Given a source explanation $E$, target theory $T$, the source and target goals $G_S$ and $G_T$, GAMA starts the alignment process by forming a correspondence between $G_S$ and $G_T$. At any given time, it keeps track of a partially expanded correspondence between $E$ and $T$ using two data structures. The *set of cndidate correspondences C* contains correspondences that are not examined yet, whereas the *correspondence support set S* accumulates the correspondences committed to so far. The algorithm uses $S$ to construct its output and to ensure that the collected correspondences are globally consistent. $C$ and $S$ are initialized with the top-level goal correspondence $G_S \leftrightarrow G_T$. On each iteration, GAMA picks the most promising corre-

[3] In the GGP setting, we assume that an agent knows the rules of both source and target games but initially does not possess strategic knowledge in either domain.

[4] We use a preprocessor that automatically translates GGP game rules to ICARUS' representation.

Table 3
Example alignment process of a source explanation against a target theory.

*Source Explanation*

```
state28: (atExit explorer) ←
   state0 : (location exit 5 0)
   state28: (location explorer 5 0)

state28: (location explorer 5 0) ←
   state0 :  (direction east)
             (not (blocked east 4 0))
             (add 5 4 1)
   state27:  (location explorer 4 0)
             (nextToExplorer east 5 0)
   state28:  (move explorer east)
```

*Target Theory*

```
% concept clause
(atGoal ?agent)
   :relations  (place goal ?X ?Y)
               (place ?agent ?X ?Y)

% primitive skill clause
(place hero ?X1 ?Y1)
   :start      (direction east)
               (add ?X1 ?X2 1)
               (place hero ?X2 ?Y1)
               (nextToHero east ?X2 ?Y1)
   :action     (move hero east)
```

*Alignment Process*

*Expand goals :* (atExit explorer) ↔ (atGoal hero)

*Hypothesize correspondences:*
(atExit ?X) ↔ (atGoal ?X), explorer ↔ hero

*Unify on target side:*
(atGoal hero) = (atGoal ?agent)

*Align antecedents:*
(location exit 5 0) ↔ place(goal ?X ?Y)
(location explorer 5 0)↔ place(hero ?X ?Y)

*Hypothesize correspondences:*
(location ?A ?B ?C) ↔ (place ?A ?B ?C)
exit ↔ goal, 5 ↔ ?X, 0 ↔ ?Y

*Expand:* (location explorer 5 0) ↔ (place hero ?X ?Y)

*Unify on target side:*
(place hero ?X ?Y) = (place hero ?X1 ?Y1)

*Align antecedents:*
(direction east) ↔ (direction east)
(not (blocked east 4 0)) ↔ true
(add 5 4 1) ↔ add(?X ?X2 1)
(location explorer 4 0) ↔ (place hero ?X2 ?Y)
(nextToExplorer east 5 0) ↔ (nextToHero east ?X2 ?Y)
(move explorer east) ↔ (move hero east)

spondence $s_0 \leftrightarrow t_0 \in C$ based on a heuristic function $h$ (Section 4.2). Next, it applies a *correspondence expansion* on $s_0 \leftrightarrow t_0$, where correspondences are formed between the*antecedents* of $s_0$ and $t_0$. During correspondence expansion, it generates source antecedents $s_1, \ldots, s_n$ that justify $s_0$ in the source explanation, and target antecedents $t_1, \ldots, t_m$ that regress the goal, $t_0$, through a single concept or primitive skill clause. These antecedents are concepts in the source and target domains, which GAMA seeks to analogize. A correspondence expansion returns an antecedent alignment $A = \{\ldots, (s_i \leftrightarrow t_j), \ldots\}$ that is a pairwise correspondence between source and target antecedents. If the number of source and target antecedents are not the same, the system allows a partial antecedent alignment with correspondences such as $true \leftrightarrow t_j$ or $s_i \leftrightarrow true$, which is the basis for learning mappings between domains that are similar only at an abstract level. However, the method assigns a high heuristic cost to such abstract correspondences.

After each correspondence expansion, GAMA updates its data structures. It removes the expanded correspondences from $C$, and adds the newly created antecedent correspondences to $C$ as well as to $S$. The algorithm backtracks if the newly added correspondences create inconsistency in $S$ based on constraints we describe in the next subsection. This process continues recursively until $C$ is empty, when it returns a representation mapping constructed using the information in $S$.

This pseudo-code defines a search process generated by the nondeterministic *correspondence expansion* function. This function calculates all possible antecedent alignments for a given correspondence, but returns them one by one (per a heuristic order) each time the algorithm backtracks to a given choice point. For example, in Table 3 the first correspondence expansion results in (location explorer 5 0) ↔ (place hero ?X ?Y), but GAMA also considers an incorrect alternative correspondence (location exit 5 0) ↔ (place hero ?X ?Y)during its search. The algorithm sorts its alternatives at choice points using a heuristic function $h(\,)$ that we describe shortly. When new additions to the correspondence support set $S$ violate constraints, GAMA backtracks.

Table 4
The GAMA algorithm, with line numbers for nondeterministic steps in bold.

| | |
|---|---|
| gama($G_S$: *source goal*, $G_T$: *target goal*, *T: target theory*, *E: source explanation* ) :- | (1) |
| *heuristic function h() returns a confidence value for a set of correspondences* | |
| initial correspondence $c$ : = $(G_S \leftrightarrow G_T)$ | (2) |
| correspondence support set $S$ : = $\{c\}$ | (3) |
| candidate correspondences $C$ : = $\{c\}$ | (4) |
| garma-loop($S$, $C$, $T$, $E$) | **(5)** |
| gama-loop($S$, $C$ $T$, $E$) :- | (6) |
| if $C$=$\{\}$ terminate with a mapping determined by $S$ | (7) |
| select candidate correspondence $c \in C$ ( using $h(c)$ for ordering candidates) | **(8)** |
| $C$ := $C - \{c\}$ | (9) |
| antecedent correspondences $A$ : = correspondence-expansion($c$, $S$) | **(10)** |
| new candidate correspondences $C_{new}$ : = $C \cup A$ | (11) |
| new correspondence support set $S_{new}$ : = $S \cup A$ | (12) |
| backtrack if $S_{new}$ violates constraints or $h(S_{new}) <$ *min-output-value* | (13) |
| gama-loop($S_{new}$, $C_{new}$, $T$, $E$) | (14) |
| correspondence-expansion($s_0 \leftrightarrow t_0$, $S$) :- | (15) |
| $A$ : = select an antecedent alignment $\{\ldots(s_i \leftrightarrow t_j) \ldots\}$ consistent with $S$ such that | **(16)** |
| a source explanation $s_0 \leftarrow s_1, \ldots, s_n$ is in $E$, | |
| a target clause $t \leftarrow t_1, \ldots, t_m$ exists in $T$, where $t$ unifies with $t_0$, and | |
| the selection is ordered by the heuristic function $h(A)$. | |
| *A is assigned empty set if $s_0$ or $t_0$ does not have antecedents.* | |

It also backtracks to an earlier choice point if the heuristic value of the collected correspondences in $S$ is too low (e.g., if there are too many abstractions).

### 4.2. Constraints and heuristics used in representation mapping

If not controlled, the space of all possible representation mappings is very large. Like other techniques that find analogies, ours is guided by several constraints and heuristics. These fall into three groups: *pragmatic, structural*, and *semantic* constraints, using the terminology of Holyoak and Thagard (1989).

*Pragmatic constraints* concern the purpose of analogy and play a central role in GAMA. In our case, the purpose of analogy is to transfer source skills that can achieve a target goal and the search process is guided through formation of an explanation of that target goal. Moreover, GAMA considers only correspondences that are necessary to transfer relevant source skills. In particular, it forms correspondences for the symbols that occur in the explanation of the source goal that corresponds to the target goal, and, consequently, for the symbols in the source skills learned from that explanation.

*Structural constraints* are based on structural similarity between the source and target theories. GAMA uses structural constraints motivated by Gentner's (1983) structure-mapping theory and its implementation SME (Falkenhainer, Forbus, & Gentner, 1989). One major structural constraint is the assumption that there is a one-to-one relation between objects; for example, the system treats the correspondences hero ↔ ?x and exit ↔ ?x as inconsistent. On the other hand, the correspondences hero ↔ ?x and hero ↔ ?y are consistent, and if both are in the correspondence support set, the system unifies the variables ?x and ?y. GAMA also employs a second structural constraint called *parallel connectivity* by Klenk and Forbus (2007), which assumes that correspondences between predicates imply correspondences between arguments. For example, correspondences (location ?a ?b ?c)↔(place ?a ?b ?c) and (location explorer 1 2) ↔ (place hero ?x ?y) imply the correspondences explorer ↔ hero, 1 ↔ ?x, and 2 ↔ ?y.

GAMA is mainly guided by the pragmatic and structural constraints, but to order alternatives at choice points it also uses *semantic constraints*, which involve literal similarity in the meaning of source and target concepts. Before the algorithm starts its explanatory analysis, it calculates a similarity measure for each source and target predicate (and object) pair, then inserts these into a *conceptual similarity matrix*. At present, we use a naive similarity measure based on shared symbols between source and target domains. If source and target predicates have the same symbols, the

correspondence is assigned the maximum similarity value. If they do not have the same symbol, their similarity is derived from the average similarity of the predicates that occur in their concept definitions. GAMA uses the values from the conceptual similarity matrix in the heuristic function that orders options at choice points.

The heuristic function $h(x)$ (Table 4) returns a confidence value for a set of correspondences $x$. The value of a correspondence $h(s \leftrightarrow t)$ is a measure of how well the source predicate $s$ matches against the target theory $t$, and it is determined from a mixture of structural and semantic considerations. If a correspondence $s \leftrightarrow t$ is justified by previous elements in the correspondence support set $S$, GAMA assigns a high positive value to $h(s \leftrightarrow t)$ based on structural consistency. For example, in Table 3, the alignment (location explorer 4 0) ↔ (place hero ?X2 ?Y) is justified by the previously recorded correspondence (location ?A ?B ?C) ↔ (place ?A ?B ?C) and therefore has a high confidence value. On the other hand, if $s \leftrightarrow t$ cannot be justified by structural evidence, GAMA determines its confidence value using semantic considerations. In that case, our implementation assigns $h(s \leftrightarrow t)$ a value proportional to the conceptual similarity metric[5] between $s$ and $t$. When $h$ must determine the value of a set of correspondences, $A = \{s_1 \leftrightarrow t_1, s_2 \leftrightarrow t_2, \ldots, s_n \leftrightarrow t_n\}$, as when $A$ is an alignment of antecedents (Table 4, line 16), the heuristic function simply returns the average of confidence values assigned to each correspondence in that set such that

$$h(\{s_1 \leftrightarrow t_1, s_2 \leftrightarrow t_2, \ldots, s_n \leftrightarrow t_n\}) = \underset{i}{\text{average}}(h(s_i \leftrightarrow t_i)).$$

If the number of source and target antecedents are not the same in a comparison expansion, GAMA can still find a correspondence by abstracting away some of the antecedents. In particular, if GAMA does not find a correspondence for a source explanation predicate $s$, it can choose to construct an abstraction mapping $s \leftrightarrow true$, in which case it will also skip aligning the decedents justifying $s$ in the source explanation. If abstraction were allowed without additional bias, the system would generate too many alternative mappings and would not have sufficient information to prefer among them. For example, the trivial mapping: "abstract all antecedents of top level goals" is always valid so GAMA needs criteria to prefer better mappings if they exist.

GAMA has the structural preference to prioritize mappings that have less abstraction and that explain more of the source solution with the target theory. Abstraction correspondences have a high negative value, so expansions that generate abstractions are considered only as a last resort. GAMA also prefers smaller abstractions over larger ones. If the system forms a source abstraction $s \leftrightarrow true$, which means that the source predicate does not have any target correspondence, the cost of abstraction is proportional to the size of the explanation structure not aligned due to the abstraction. The algorithm estimates the magnitude of that value proportional to the number of all descendants of $s$ in the explanation structure. On the other hand, if no correspondence can be found for a target predicate $t$, the system forms an abstraction $true \leftrightarrow t$. This abstraction has a constant cost, since abstraction of a theory clause does not leave any part of the source explanation uncovered.

### 4.3. Translating skills and concepts

GAMA passes a hypothesized representation map to its *representation translator* component, which is responsible for translating the source skills and concepts to the skills and concepts of the target theory.

Translation of concepts is a simple recursive process. As the base case, given a source concept $s$ for which GAMA has a predicate correspondence $s \leftrightarrow t$, all references to $s$ are replaced with $t$. A concept that does not have a representation translation can still be imported to the target domain by translating its definition. In that case, we say that $s$ is *translated analogically* and a new concept definition $analogical - s$ is created in the target domain by translating the definition of $s$ into target symbols. Moreover, GAMA adds the correspondence $s \leftrightarrow analogical - s$ to the representation map to be used in translation of other concepts and skills.

The ability to translate conceptual knowledge given a mapping between concept predicates is not very surprising. However, the ability to transfer procedural knowledge using a conceptual mapping is less obvious. One of the main representational commitments of ICARUS, associating all skills with the goals they achieve, facilitates transfer of skills given mappings between concept predicates because the skills are referenced with those conceptual predicates.

The representation translator imports skills from source to target by converting predicates in the goal, subgoals, and start conditions of these skills using hypothesized representation correspondences. The translator matches the left hand-side of a correspondence to each conceptual predicate that occurs in a source skill definition and replaces it with the right-hand side. Table 5 presents a representation map, a source skill, and the transferred target skill, including three different kinds of correspondences. While the predicate and object mappings such as (holding ?x1) ↔ (holding ?x1) and doesnail ↔ doestie translate source skill symbols into target symbols, abstraction correspondences like (property ?Item3 hammer) ↔ true and (holding ?Item3) ↔ true mean that the predicates on the left-hand side do not have a target correspondence and must be dropped in start conditions or subgoals of the transferred skills.

As the representation translator converts skills using abstraction maps, it uses additional information the repre-

[5] In our current implementation, conceptual similarity is merely determined by the static similarity matrix generated during the initial phase and does not use information in the correspondence support set. Evaluating conceptual similarity given a correspondence support set may yield a more accurate heuristic function, but only at the expense of more computational cost.

Table 5
Importing source skills into the target domain.

*Learned Mappings*

```
% predicate mappings

  (joined ?x1 ?x2 ?x3) ↔ (tied ?x1 ?x2 ?x3)
  (do-nail ?x1 ?x2 ?x3 ?x4) ↔ (do-tie ?x1 ?x2 ?x3 ?x4)
  (property ?x1 ?x2) ↔ (property ?x1 ?x2)
  (holding ?x1) ↔ (holding ?x1)

% object mapping

  doesnail ↔ doestie
  nailable ↔ tieable

% subskill abstraction

  (property ?Item3 hammer) ↔ true
            if source skill clause is ((joined ?Item1 ?Item2 ?NewItem) start:… )
  (holding ?Item3) ↔ true
            if source skill clause is ((joined ?Item1 ?Item2 ?NewItem) start:… )
```

*Source Skill* → *Target Skill*

```
(joined? Item1 ?Item2 ?NewItem)
  start:
     (property ?Item3 hammer)
     (property ?Nail  doesnail)
     (property ?Item1 nailable)
     (property ?Item2 nailable)
  subgoals:
     (holding ?Item2)
     (holding ?Nail)
     (holding ?Item3)
     (holding ?Item1)
     (do-nail ?Item1 ?Item2
              ?Nail ?NewItem)
```

```
(tied ?Item1 ?Item2 ?NewItem)←
 start:

   (property ?Rope  doestie)
   (property ?Item1 tieable)
   (property ?Item2 tieable)
 subgoals:
    (holding ?Item2)

    (holding ?Rope)
    (holding ?Item1)
    (do-tie  ?Item1 ?Item2
             ?Rope ?NewItem)
```

sentation mapper provides. For example, the translator does not apply an abstraction correspondence like (holding ?Item3) ↔ true everywhere it matches on a source skill, since that would generate incorrect results, such as removing all holding subgoals in Table 5. Instead, the system keeps track of the point in the explanation structure at which the abstraction occurred and only abstracts the corresponding part of the source skill.

### 4.4. Transfer in partial analogy

When the source and target domains have a one-to-one correspondence, it is easy to see how GAMA can transfer knowledge. In this subsection, we describe how it can transfer between source-target domains that are quite different, as in the case of cross-domain scenarios. To that end, we highlight four features that equip our system to handle partial analogies.

First, GAMA's basic explanatory analysis naturally supports abstract correspondences between domains, because explanations refer to only relevant parts of a source domain that play role in a solution. All other features of a source domain are automatically abstracted during transfer.

We have also described a second mechanism GAMA uses to find abstract similarities. It can heuristically ignore portions of the source explanation that do not align against the target theory, or portions of the target theory that do not match against the explanation structure. As a result of this process, given a source predicate *s* that does not have a target correspondence, GAMA acquires a source abstraction map of the form $s \leftrightarrow true$, and applies it to remove portions of the transferred source skills that are not meaningful in the target domain.

A third method for transferring abstract similarities is related to *untranslatable source symbols* – source conceptual predicates and objects that cannot be translated into the target domain through correspondences or translations of their concept definitions. If the translator were to leave such symbols in the start conditions of a skill, the skill would never be applicable in the target domain. Therefore, the representation translator generalizes those skills by removing predicates in the start condition that cannot be translated. For example, the heavy-to-carry concept in Table 4 does not exist in the target game, since that game does not include a weight limit. As a result, GAMA generalizes the transferred skill during translation by removing that condition from the start field.

Finally, GAMA can sometimes transfer a skill, even if it contains untranslatable symbols in its head and subgoals. Just like in the cases of transferring concepts by translating their definitions, our system can transfer skills with untranslatable goals by translating the skill definitions of those goals. A source skill *s* is translatable to a target skill *analogical-s*, if all of its subgoals are translatable. For example, the source skill carrying in Table 6 refers to a sub-

Table 6
Transfer of an mRogue source skill into Wargame. GAMA removes untranslatable start condition concepts such as heavy-to-carry and imports skills with untranslatable goals such as pickedup by translating their definitions.

*Learned Mapping*

```
GAMA correspondences:
    hero ↔ soldier, (carrying ?x)↔(carrying ?x)
   (item ?x)↔(item ?x)
   (nextHeroLocation ?a ?b)↔(nextSoldierLocation ?a ?b)

Untranslatable source concepts:
   heavy-to-carry, pickedup

Concepts translated through their definition:  sameLocation
```

*Source Skill*

```
(carrying ?item)
 start:
   (not (heavy-to-carry ?item))
 subgoals:
   (pickedup ?item)


(pickedup ?item)
 start:
   (location  hero  ?X  Y)
   (location ?item  ?Xi  ?Yi)
   (not (sameLocation ?Xi  ?Yi ?X  ?Y))
   (heavy-to-carry ?item)
 subgoals:
   (nextHeroLocation ?Xi ?Yi)
```

*Target Skill*

```
(carrying ?item)
 start: <empty>

 subgoals:
   (analogical_pickedup ?item)


(analogical-pickedup ?item)
 start:
   (location soldier  ?X ?Y)
   (location ?item  ?Xi  ?Yi)
   (not(analogical-sameLocation
      ?Xi  ?Yi ?X  ?Y))
 subgoals:
  (intendedSoldierLocation ?Xi ?Yi)
```

goal pickedup for which the translator could not find a conceptual translation. Nevertheless, this skill can still be translated since its subgoals can be translated. In this case, GAMA can create a new target skill analogical-pickedup[6] because pickedup's only subgoal nextHeroLocation has a translation, namely, intendedSoldierLocation. This shows that the system can transfer skills even if there is a partial mapping that cannot translate all skill symbols. ICARUS' commitment to hierarchically composable skills plays a major role in this ability, because it lets the agent interleave target skills with analogically translated source skills.

## 5. Experimental evaluation

We have claimed that goal-directed analogy supports far transfer. This section provides support for that claim in the form of two computational experiments. The first examines the magnitude of the transfer effect produced on a battery of far transfer tasks designed by an independent agency.[7] We expect a substantial increase in problem-solving speed due to transferred knowledge. The second experiment involves a lesion study that determines the source of power behind the observed transfer, specifically if it is due to reuse of generalized skills without representation mapping (the lesion case) or transfer of generalized skills with representation mapping (the non-lesion case). We hypothesize that the ability to map skills across problem representations will qualitatively improve transfer in the non-lesion case relative to the lesion case.

### 5.1. Experimental methodology

Details of the general game playing framework led us to conduct these experiments with two important changes from the ICARUS architecture. In particular, we replaced the bottom-up inference module with a top-down mechanism to handle the number of entities perceived in the environment. Because GGP provides fully modeled, deterministic domains, we also replaced the system's reactive execution module with one that mentally simulated execution with N-step look ahead, using technology adapted from Nau et al.'s (2003) SHOP2 planner. We employed this execution module to generate solution traces for input to the learning algorithm described earlier and to evaluate learned skills in the target domain.

The system begins by translating the GGP game specification into concepts and primitive skills, then invoking automatically generated exploration skills to search for a solution to the source problem. Upon finding one, the system acquires new hierarchical skills (as discussed in Section 3.3) that both generalize the solution and become the object of transfer to the target problem. After porting these concepts and skills via representation mapping, the performance system employs them in the target domain, falling back on exploration behavior when guidance from source knowledge is exhausted.

[6] Since an analogically translated skill *analogical-s* does not have a corresponding concept definition, the interpreter never tests to see if such goals are achieved. This limits reactivity but enables flexible transfer.

[7] This system was one of three cognitive architectures tested in the DARPA Transfer Learning Program, via a formal evaluation process controlled by the Navy Center for Applied Research.

In more detail, our experiments measure transfer as the difference in agent performance (solution speed) on a given target problem with and without exposure to the corresponding source (normalized to facilitate comparison). The non-transfer (N) agent sees the target problem alone and must solve it by a base set of exploration skills. In contrast, the transfer (T) agent has the opportunity to solve the source problem, acquire knowledge from that experience, and make it available for transfer. The T protocol involves several steps that are consistent with the general game playing format:

1. download the source game definition and initial state;
2. solve the source task via exploration;
3. learn hierarchical skills and concepts from the solution;
4. download the target game definition;
5. select a representation map;
6. instantiate the mapped skills and concepts in the target;
7. if learned skills fail to improve performance on the source task, then set mapped skills = {};
8. download initial state data for the target problem; and
9. solve the target problem using the mapped knowledge.

Since this experiment concerns the potential benefits of transferred knowledge, not the cost of acquiring that knowledge, the protocol only reflects the time required for the last two steps in the transfer agent's performance score (although steps 4–6 were short by comparison). The transfer and non-transfer agents both had access to the same exploration skills and supporting background knowledge to solve the target task. They act as a fallback to the transfer agent if mapped knowledge does not suffice. Step is intended to avoid negative transfer. It ensures that the system will only attempt to transfer high-quality learned knowledge from the source to the target task and will rely on exploration if that test fails.

### 5.2. Experimental results

Fig. 3 presents the results of the experiment to measure far transfer. It examines the magnitude of the effect obtained in 17 scenarios, where the first five are abstraction tasks (A), the next five are reformulation examples (R), and the last six are cross-domain transfer tasks. Each data point averages across ten trials of the given target problem for the transfer agent and the non-transfer agent, both. Values above zero indicate positive transfer (the transfer case solution is faster than the non-transfer case), values near-zero indicate no transfer, and values below zero indicate negative transfer.

The results show that the system produces positive transfer in 10 of 17 cases. These include several instances of very positive transfer. For example, the 0.93 score in CrossDomain-4 indicates that the agent solved the problem more than ten times faster with transferred knowledge (and five to six times faster in Escape A-1 and Wargame R-2). The system obtains positive transfer from examples in each of the abstraction, reformulation, and cross-domain classes. The cross-domain case is somewhat surprising, since these scenarios were not in any way designed to afford positive transfer.

The four instances of near-zero transfer in Escape A-3, Wargame A-1, Escape R-1, and CrossDomain-5 correspond to cases where the source–source filter failed (step above). These scenarios should produce near-zero transfer, given that both the transfer and non-transfer agents employ exploration to solve the target task. Escape R-2 also falls into this class, suggesting its moderate positive transfer is due to random variation.

The instances of negative transfer in Fig. 3 are also easy to explain. The result for Wargame-R-1 is due to an incorrect representation map (found in eight of ten transfer trials) applied to correct skills, while the effect in CrossDomain-1 is due to a partial map that creates actionable but misleading skills. The effect in CrossDomain-2 is due to random fluctuation, as that scenario also failed the source–source filter.

In summary, we can explain the instances of zero transfer as the (automatically diagnosed) failure of the learning system to acquire useful skills, and the one case of significant negative transfer as the product of an incorrect mapping. Interestingly, the instances of positive transfer are

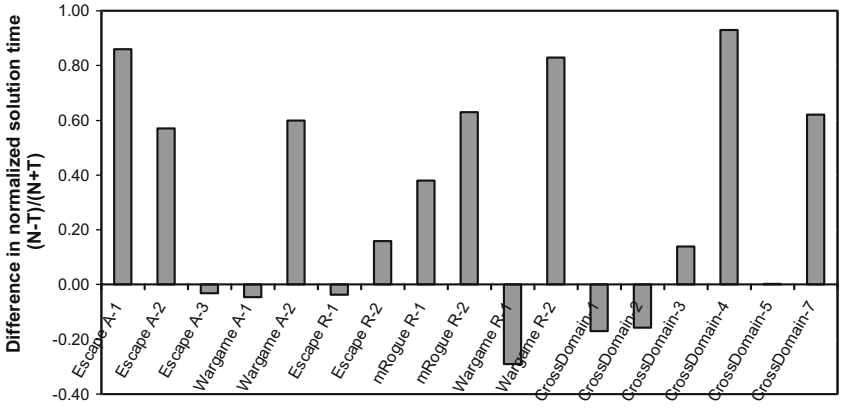



Fig. 3. Far transfer in 17 scenarios measured as the normalized difference of the non-transfer and transfer cases.

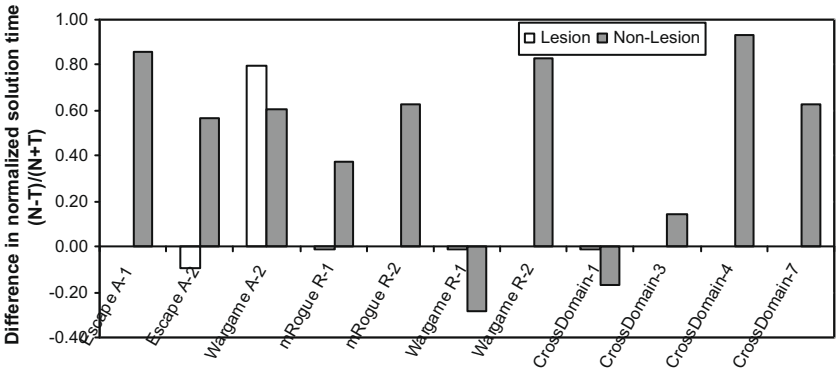



Fig. 4. A lesion study showing the impact of representation mapping on transfer.

more enigmatic. The benefit could be due to either of the two transfer mechanisms: direct application of learned skills or transfer of learned skills via representation mapping. The second experiment disambiguates these effects.

The lesion study examined the source of transfer in the 11 scenarios (out of 17) where the system acquired reliable knowledge from solving the source problem it could then employ as a basis for transfer (i.e., the scenarios that passed step). Fig. 4 shows the results of this study. Here, the lesion case employs generalized skills without representation mapping, while the non-lesion condition duplicates a subset of the cases shown with representation mapping in Fig. 3. As before, each data point averages across ten trials of the given target problem for both the transfer agent and the non-transfer agents.

The results show that the system without representation mapping (the lesion condition) produces circa zero transfer in 9 of 11 cases. This effect is easy to understand. Lacking a representation map, the system has no mechanism for relating source skills to target needs, so the transfer and non-transfer agents both rely on exploratory behavior, which produces no net transfer. The main exception is Wargame A-2, where the terms that differed between source and target were bound to variables in skills. This made source skills directly applicable for the transfer agent, leading to positive transfer.

More broadly, we can draw two conclusions from the lesion study. First, representation mapping generates virtually all of the positive transfer observed in the 11 scenarios. In particular, it provides the system with the capacity to exploit learned knowledge given the representational divide characteristic of far transfer tasks. Second, this effect appears robust across problem classes, as it explains all but one case of positive transfer.

Finally, we note that the scenarios were designed to afford positive transfer. This is least true in the cross-domain cases where the source-target relationship is unconstrained. However, the abstraction and reformulation scenarios obey what might be called a "constant content assumption": given the correct mapping among symbols, solutions for the source can solve the target task. This is evident in the reformulation case, while the transformations defining the abstraction class admit no new information in the target, implying source solutions are preserved. Experiments with even more disparate source and target tasks would clarify the limits of the GAMA representation mapping algorithm.

## 6. Related research

The research reported in this article integrates analogical transfer based on representation mapping into a larger architectural framework that learns to solve problems more effectively. There are large bodies of related work on analogy, learning, and problem-solving. In this section, we describe the work that we believe to be most relevant, concentrating primarily on analogy since it plays a key role in our approach to transfer.

There are many surveys of analogy in cognitive science. Hall (1989) gave a comparative analysis of early computational approaches to analogy, while Gentner and Holyoak (1997) discussed analogy in the context of reasoning and learning. Two brief, relatively recent encyclopedia articles on analogy have emphasized the perspectives of cognitive science and psychology (Gentner, 1999, 2003). French (2002) and Kokinov and French (2003) provide recent summaries of computational models of analogy, while Holyoak (2005) gives a broad and detailed overview of research on analogy in psychology, including computational models. One conclusion drawn in the most recent of these articles is that "models of analogy have not been well integrated with models of problem-solving..." and "integration of analogy models with models of general problem-solving remains an important research goal". Our work integrates analogy with problem-solving and thus makes significant progress on this goal.

In the discussion that follows, we focus on a subset of the research described in these surveys that is particularly relevant to our work. We also discuss some related work not mentioned in the reviews.

Gentner (1983) presented an early analysis of analogy in which she proposed key hypotheses, including the one-to-one constraint and the systematicity principle. The first states that, in most human formed analogies, an object in

one domain will not map to two objects in the other. The systematicity principle states that, in general, analogies map relations that preserve higher-order structure and relations like causes and implies. Psychological studies provided experimental support for these hypotheses (Clement & Gentner, 1988; Krawczyk, Holyoak, & Hummel, 2005). Our approach not only incorporates both the one-to-one constraint and the systematicity principle, but it is also guided by the purpose of analogy (pragmatic constraints).

Falkenhainer et al. (1989) used Gentner's analogy theory as the basis for an early computational model, the structure-mapping engine (SME). In later work, the PHINEAS system (Falkenhainer, 1990) used analogical mapping as part of a larger theory formation process that involved explanation and that included retrieval, transfer, and qualitative simulation. While his system uses qualitative models of dynamic systems as background knowledge and applies them in a qualitative simulation setting, our system uses action models and applies them in a problem-solving setting. Unlike our system that learns how to achieve goals, Falkenhainer's system uses analogy to improve imperfect target domain theories.

Klenk and Forbus (2007) describe another system built upon SME that learns new general domain knowledge in the domain of rotational kinematics by analogy with translational kinematics. This system finds analogies between worked source and target solutions to physics problems, which involve mental activities such as simplification of formulas. In contrast, the solutions our system uses are sequences of action with physical effects, and instead of solutions in the target domain, we use target domain theory to form analogies. Moreover, while our system learns how an agent should behave in the target, Klenk et al.'s system learns how the target domain functions.

Burstein (1988) described CARL, a system that constructed analogies which mapped actions, preconditions, and results. The program also constructed maps between relations and objects; however, unlike our algorithm that maps whole source solutions and explanations against a target theory automatically, CARL incrementally constructs and refines maps by interacting with a teacher that specify partial maps, demonstrate actions one by one, and give feedback on predictions about the expected effects.

Holyoak and Thagard (1989) advocated the use of pragmatic constraints involving goals. They implemented an analogy engine called ACME and applied it to a collection of examples including the atom-solar system analogy. However, their algorithm assumed that pragmatic values for predicate matching were given, leaving open question of how they arise. In contrast, our system's goal-driven explanatory analysis is implicitly based on the pragmaticity principle.

Several researchers have used goals in analogy. For instance, one of the earliest analogy systems (Winston, 1982) used causal networks to infer the motives of an agent based on a similar situation with another agent. However, this work used small examples that involved bits of plots or stories. It did not deal with complex plans and it did not involve cross-domain transfer, as does our approach. Purpose-directed analogy (Kedar-Cabelli, 1985, 1988) used goals, causal relations, and explanations to show how an object satisfies a given purpose. For example, a styrofoam cup lets one drink hot liquids just as does a ceramic mug with a handle. Her system introduced the use of goal regression to guide analogical reasoning. Veloso and Carbonell's (1993) PRODIGY/ANALOGY used a goal-directed form of analogy to improve the performance of a means-ends planner. They conducted experiments in a logistics domain that involved moving objects using trucks and planes. Neither work incorporates representation mapping between relations or involve cross-domain transfer.

Our approach also bears similarities to work on analogical problem-solving. VanLehn and Jones' (1993) CASCADE used a form of analogical search control to guide top-down chaining through an AND–OR tree. Their system stored a semi-persistent mapping that it reused later in a given problem but it did not map across domains or representations. Jones and Langley's (2005) EUREKA also used analogical search control, in this case to direct a means- ends problem solver that spread activation through stored subproblem decompositions. This system exhibited a limited ability for cross-domain transfer, but only when provided with connections between predicates.

In summary, our approach is closely related to, and builds upon, earlier research that involved structural analogy and pragmatic constraints. It has been guided by psychological results on structural analogy, but it incorporates pragmatic information about goals and explanations. Moreover, it is distinguished by its use of analogical representation mapping to achieve cross-domain transfer.

## 7. Directions for future research

Our future work will advance our ability to reuse previously acquired knowledge in robust ways. We plan to pursue this goal by improving the representation mapping and translation mechanisms, and by completing their integration into the ICARUS cognitive architecture.

We intend to improve the representation mapping technology by supporting more complex mappings between source and target domains. For example, while our current work allows predicate abstraction in cases where there is no exact match, we can also consider the elimination of and reordering of predicate arguments. This change will increase the number of possible representation maps, and require additional attention to search control.

We can also improve the quality of representation maps by employing feedback from the application of transferred skills. For example, we can test mapped skills and reduce the heuristic score for any map that produces a failed skill in the target domain. This process can be interleaved with map generation. We can also remodel the map evaluation heuristic to reflect an expected utility of transferred knowl-

edge, again updated from experience. This change would focus GAMA on the most useful mappings. In addition, we can construct mechanisms for cumulative transfer that incrementally augment the representation map as the agent accrues experience in the target domain over time. We can expand the impact of representation mapping by employing learning in ICARUS to improve transferred knowledge in the target domain. We are pursuing this avenue now by tuning and patching somewhat incorrect skills via theory refinement.

Our larger goal is to integrate representation mapping into ICARUS at the architectural level. This would require us to address the retrieval problem, and fold representation mapping into the architecture's cognitive processing, such that is invoked under appropriate conditions. These are non-trivial extensions, but they would produce a version of ICARUS that seeks opportunities to retrieve, transfer, and apply knowledge from prior experience to current tasks, in analogy to human behavior.

## 8. Concluding remarks

In this paper, we discussed an approach to transfer that involves analogical mapping of symbols across different domains. We combined this method with representational assumptions from ICARUS, a theory of the human cognitive architecture. Our transfer algorithm, GAMA, conducts representation mapping to generate correspondences between source and target concepts by matching the explanation of a source solution against the target theory. Next, it conducts representation translation by using those correspondences to import source skills and concepts into the target domain. GAMA is a pragmatic system guided by the purpose of analogy, since its search is directed by the target goal that the system aims to achieve using transferred skills. It further constrains search by using structural and semantic constraints. We showed how GAMA finds abstract similarities even when the source and target domains are partially analogous. Its representation mapper contributes to abstraction by omitting parts of an explanation that cannot be aligned to the target theory, whereas its representational translator handles unmapped source symbols by translating their concepts and skills definitions or by omitting them when generalization is appropriate.

We claimed that the same features which distinguish ICARUS from other architectures facilitate representational transfer. Most notably, indexing skills by their goals supports transfer by enabling use of conceptual correspondences in translating skills. Moreover, hierarchically composable skill representation allows transfer even in the cases of partial mappings letting the agent interleave analogically translated skills with mapped skills. We presented experimental evidence that our new representational transfer mechanism improves on more basic learning processes. We also demonstrated far transfer among reformulation tasks, abstraction tasks, and in less constrained cross-domain scenarios. In this research, we have taken steps towards the adaptation of past experience to current goals, which is important for building cognitive architectures that learn continuously and cumulatively.

## Acknowledgements

We thank Ugur Kuter, Nate Waisbrot, Tom Fawcett, and Chunki Park for their contributions to this work. This paper reports research sponsored by DARPA under agreement FA8750-05-2-0283. The US Government may reproduce and distribute reprints for Governmental purposes notwithstanding any copyrights. The authors' views and conclusions should not be interpreted as representing official policies or endorsements, expressed or implied, of DARPA or the Government.

## References

Burstein, M. (1988). Incremental learning from multiple analogies. In A. Prieditis (Ed.), *Analogica* (pp. 37–62). London: Pitman.

Clement, C. A., & Gentner, D. (1988). Systematicity as a selection constraint in analogical mapping. In *Proceedings of the 10th annual conference of the Cognitive Science Society* (pp. 412–418). Montreal: Lawrence Erlbaum.

Choi, D., Könik, T., Nejati, N., Park, C., & Langley, P. (2007). Structural transfer of cognitive skills. In *Proceedings of the eighth international conference on cognitive modeling* (pp. 115–120). Oxford, UK: Taylor & Fransis.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41*, 1–63.

Falkenhainer, B. (1990). A unified approach to explanation and theory formation. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 157–196). San Francisco, CA: Morgan Kaufmann.

French, R. M. (2002). The computational modeling of analogy-making. *Trends in Cognitive Science, 6*, 200–205.

Genesereth, M. R., Love, N., & Pell, B. (2005). General game playing-overview of the AAAI competition. *AI Magazine, 26*, 62–72.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7*, 155–170.

Gentner, D. (1999). Analogy. In R. A. Wilson & F. C. Keil (Eds.), *The MIT encyclopedia of the cognitive sciences* (pp. 17–20). Cambridge, MA: MIT Press.

Gentner, D. (2003). *Analogical reasoning psychology of in encyclopedia of cognitive science*. London: Nature Publishing.

Gentner, D., & Holyoak, K. (1997). Reasoning and learning by analogy. *American Psychologist, 52*, 32–34.

Hall, R. (1989). Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence, 39*, 39–120.

Holyoak, K. J. (2005). Analogy. In K. Holyoak & R. G. Morrison (Eds.), *The cambridge handbook of thinking and reasoning* (pp. 117–142). Cambridge, UK: Cambridge University Press.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13*, 295–355.

Jones, R. M., & Langley, P. (2005). A constrained architecture for learning and problem solving. *Computational Intelligence, 21*, 480–502.

Kedar-Cabelli, S. (1985). Purpose-directed analogy. In *Proceedings of the seventh annual conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum.

Kedar-Cabelli, S. (1988). Toward a computational model of purpose-directed analogy. In A. Prieditis (Ed.), *Analogica* (pp. 89–107). London: Pitman.

Kieras, D. E., & Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. *Journal of Memory and Language, 25*, 507–524.

Neches, R., Langley, P., & Klahr, D. (1987). Learning development and production-systems. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production-system models of learning and development*. Cambridge, MA: MIT Press.

Klenk, M., & Forbus, K.D. (2007). Learning domain theories via analogical transfer. In *Proceedings of the 21st international workshop on qualitative reasoning*, Aberystwyth, U.K.

Kokinov, B., & French, R. M. (2003). Computational models of analogy-making. In *Encyclopedia of cognitive science* (pp. 113–118). London: Nature Publishing.

Krawczyk, D. C., Holyoak, K. J., & Hummel, J. E. (2005). The one-to-one constraint in analogical mapping and inference. *Cognitive Science, 29*, 29–38.

Langley, P., & Choi, D. (2006). Learning recursive control programs from problem solving. *Journal of Machine Learning Research, 7*, 493–518.

Mooney, R. J. (1990). *A general explanation-based learning mechanism and its application to narrative understanding*. San Francisco, CA: Morgan Kaufman.

Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on artificial intelligence* (pp. 968–973). San Francisco, CA: Morgan Kaufmann.

Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., et al. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research, 20*, 379–404.

Nejati, N., Langley, P., & Könik, T. (2006). Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on machine learning* (pp. 665–672). New York: ACM Press.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Segre, A. (1987). A learning apprentice system for mechanical assembly. In *Proceedings of the third IEEE conference on AI for applications* (pp. 112–117).

Singley, M. K., & Anderson, J. R. (1988). A keystroke analysis of learning and transfer in text editing. *Human–Computer Interaction, 3*, 223–274.

Thagard, P., Holyoak, K. J., Nelson, G., & Gochfeld, D. (1990). Analogical retrieval by constraint satisfaction. *Artificial Intelligence, 46*, 259–310.

VanLehn, K., & Jones, R. M. (1993). Integration of analogical search control and explanation-based learning of correctness. In S. Minton (Ed.), *Machine learning methods for planning* (pp. 231–273). San Francisco, CA: Morgan Kaufmann.

Veloso, M. M., & Carbonell, J. (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning, 10*, 249–278.

Winston, P. H. (1982). Learning new principles from precedents and exercises. *Artificial Intelligence, 19*, 321–350.