Toward Automatic Ontology Curation with Similarity-Based Reasoning

Scott E. Friedman, J. Benton, & Dan Bryce
SIFT, LLC
319 1st Ave North, Suite 400
Minneapolis, MN 55401
{friedman, jbenton, dbryce}@sift.net

Abstract

Building and maintaining large, reusable ontologies is a prerequisite for building automated systems that reason and plan in a changing world. Unfortunately, our needs, best practices, and mental models of the world *also* change. These factors— in addition to common human mistakes— lead to *model drift* over time in our computational models and ontologies. As a result, ontologies contain direct inconsistencies, anomalies, partial or duplicate descriptions, and erroneous constraints between components. This paper presents ongoing development on our system, Marshal, designed to help to curate ontologies and task models by making suggestions while users write procedures and plans. We present experiments in three ontology curation settings: (1) inducing new categories; (2) detecting anomalous instances; and (3) classification of unlabeled instances into an existing ontology. On each task, we compare multiple similarity-based inference strategies, and we show that structure-mapping produces favorable results on all three ontology curation tasks.

Introduction

When users create and revise task models and ontologies, they have a propensity for making mistakes, often through directly introducing inconsistencies or failing to perform proper maintenance between components. Users may continue using these models or ontologies with the risk that it no longer matches its intended use, unknowningly introducing risk to task execution. We call this phenomenon *model drift*. Automatically discovering whether a change represents an incompatibility with a model's usage would help reduce the risk associated with making model changes. This paper describes progress on Marshal, a system that assists users as they extend task models and ontologies in order to prevent model drift.

Marshal observes users as they author procedural models and provides feedback on how they may improve. One of Marshal's hallmark contributions is its ability to adapt its representation of the user's task model to match how the user constructs procedures or plans. Instead of constantly reminding users of a violated constraint in their procedure, Marshal can learn contexts in which the constraint is no longer enforced. Marshal can also learn new features of the task model by allowing users to enter semi-structured text that describes steps. Marshal faces the major challenge of readily distinguishing new terms from synonyms of known terms.

In the work described in this paper, we seek out contextual clues that help Marshal disambiguate. To do this, we apply previously explored techniques for structure-mapping. Structure-mapping matches a base description to find potential analogs in a target description, and constructs mappings between the two. From there, Marshal can infer information from the base to the target (e.g., category labels), thereby generating assumptions about the target via the base. In the context of Marshal, this provides the ability to help curate an ontology. For instance, we can query a user whether a newly introduced object (e.g., a `Crew` member `sally`) would fit within a particular category or should be part of a new (or super) category, dependent on that object's usage. This will help detect mis-categorized entities, propose new categorizations, and propose new categories altogether based on instance similarity.

Though structure-mapping techniques have power in their ability to represent and detect analogies between structures of models, we explore other possibilities as well. A simpler, and less computationally intensive approach involves computing a vector of all predicates that describe each object (e.g., `sally`), and then normalizing these vectors and compare them using their dot product. We compared two vector-based encodings and show that structure-mapping produces favorable results on three ontology curation experiments: (1) inducing new categories; (2) detecting categorization anomalies; and (3) classifying unlabeled entities into a gold-standard ontology.

We begin by describing the setting and research challenges involved in Marshal. We then describe background relevant to Marshal, and we describe our experiments to support the above claims. We close with a discussion of the implications and future work, including developing additional strategies for Marshal.

Marshal

Marshal interfaces with the PRIDE knowledge entry interface (Schreckenghost, Milam, and Billman, 2014) for subject matter experts— who have no knowledge representation expertise— to specify procedures in Procedure Representation Language (PRL) (Kortenkamp, Bonasso, and Schreckenghost, 2007). PRIDE and PRL have been used to specify

diverse procedures, from using an EpiPen during a severe allergic reaction to conducting repair and maintenance on the International Space Station (ISS). PRL is converted to PDDL (Boddy and Bonasso, 2010, Mcdermott et al., 1998), which comprises the data for our three experiments.

As mentioned above, Marshal is designed to fix model drift over time as well as make improvements/corrections as non-technical users specify or modify a procedure. The main module of Marshal operates by identifying executed procedures and discerning potentially drifting constraints from them.

Non-technical users use a graphical interface to specify their procedure, so they do not type PRL or PDDL directly. This means Marshal is not concerned with syntactic errors such as typos, arity, or argument order; rather, Marshal must address downstream semantic issues made by the user:

1. **Ontology extension**: if the user specifies categories (e.g., `wrench` and `spanner`) with similar extensions and affordances in their procedures, Marshal should propose a common super-category, if none yet exists.
2. **Anomaly detection**: if the user mis-categorizes an entity (e.g., as category `inhibit-command` instead of `turn-off-command`), then Marshal should detect the anomaly and signal the user.
3. **Classification**: after Marshal detects an anomaly— or if the user is unsure of the appropriate category— Marshal should provide a prioritized list of categories for an entity, using the remainder of the entity's description as evidence.

Our three experiments address these three challenges with domain-general similarity-based reasoning techniques and ontology comparison strategies described next.

Background & Related Work

Our work on Marshal uses PDDL for knowledge representation and leverages previous work in similarity-based reasoning and ontology comparison. We review these topics before describing our approach.

PDDL

We represent our tasks using the Planning Domain Description Language (Mcdermott et al., 1998), a commonly used standard for describing task planning domains and problems.[1] PDDL uses relational descriptions of objects to represent facts and actions. We restrict ourselves to STRIPS-based PDDL models, though we believe most of the techniques we discuss in this work would be uneffected by more complex variants of the language.

In our work, we focus only on PDDL facts and state information (including the states and goal description). A PDDL fact is a relation between objects, described by a predicate. As an example, we can describe that a object `ge-0` *possesses* an object `brt_1` by writing `(possesses ge-0 brt_1)`, where `possesses` is a predicate and `ge-0` and `brt_1` are objects or entities. In PDDL, we specify a problem for a particular domain (consisting of actions) by providing an initial state by listing a set of facts about the world, and a goal set. We must find an ordered set of actions that make the entire goal set simultaneously true.

[1]Numerous repositories for PDDL planning domains exist. The (perhaps arguably) best are hosted by the websites of the International Planning Competitions (`http://www.icaps-conference.org/index.php/Main/Competitions`).

Graph Similarity & Generalization

Marshal builds on previous research on computational models of *structure-mapping* that have produced techniques for describing and quantifying *isomorphic* (i.e., similarly-structured) portions of relational knowledge across descriptions (Falkenhainer, Forbus, and Gentner, 1989). These techniques match entities and relationships between relational representations (a *base* and a *target* representation) with three constraints:

1. One-to-one: An entity or relation in the base can map to at most one in the target, and the reverse for target-to-base. This constraint is part of the definition of a graph isomorphism.
2. Parallel connectivity: If a base relation maps to a target relation, so must each of their arguments, in order.
3. Tiered identicality: A relation can only map to another relation with the same predicate.

Matching a base and target description according to these constraints produces one or more *mappings* between the base b and target t. Each mapping $\langle M_{b,t}, s_{b,t} \rangle$ describes a set of matched correspondences $M_{b,t}$ between entities and relations in b and t, and a similarity score $s_{b,t}$ that increases monotonically with the elements in $M_{b,t}$. We convert $s_{b,t}$ into a normalized score $\|s_{b,t}\| = [0, 1]$ by scaling it by the average self-similarity score (i.e., the similarity score of a description against itself) of the base s_b and target s_t:

$$\|s_{b,t}\| = \frac{2s_{b,t}}{s_b + s_t} \quad (1)$$

We use normalized similarity scores in all three experiments to infer new categories, detect anomalies, and categorize unlabeled instances.

We compute a *generalization* of the mapping by creating new entities to represent heterogenous corresponding entities in $M_{b,t}$, and asserting new relations over those entities. Consider the following relations in the base:

```
(crew sally)
(o2use-rate sally slow)
(has-iss-location sally intra-vehicle)
(possesses sally brt_1)
(possesses sally pgt_2)
```

and in the target:

```
(crew joe)
(o2use-rate joe fast)
(has-iss-location joe intra-vehicle)
(has-operator ssrms joe).
```

The resulting generalization contains the following statements, including generalized entities (`ge` terms) and constituent probabilities:

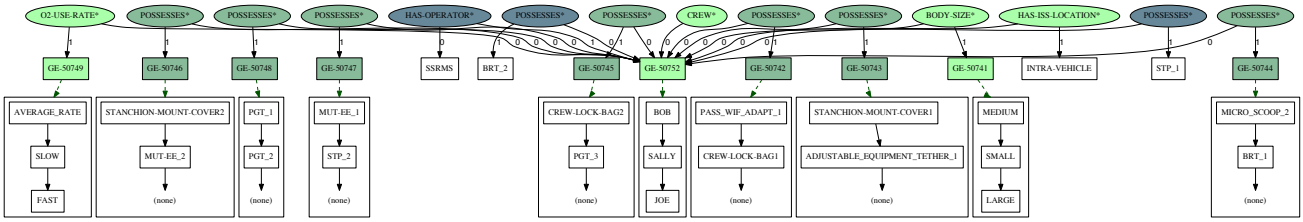

Figure 1: Generalization of three relational graphs describing three `crew` instances: `bob`, `sally`, and `joe`. Relations are colored according to frequency: light green statements occur over all three instances, and darker green statements occur over two or one (darkest).

```
(crew ge-0) = 1.0
(o2use-rate ge-0 ge-1) = 1.0
(has-iss-location ge-0 intra-vehicle) = 1.0
(possesses ge-0 brt_1) = 0.5
(possesses ge-0 pgt_2) = 0.5
(has-operator ssrms ge-0) = 0.5
```

The probabilities on each statement are soft constraints on similarity scores for future comparison against this generalization. Generalization can be performed iteratively, to generalize entire categories (e.g., `crew`) with arbitrarily many instances. This process is called Sequential Analogical Generalization of Exemplars (SAGE) (McLure, Friedman, and Forbus, 2015). We use a SAGE algorithm to generalize categories in our experiments. The results of sequential generalization over three `crew` instances are shown in Figure 1. The generalization is shown as colored links, and the entities comprising each generalized entity are shown beneath. Note that some generalized entities have `(none)` rows where no isomorphic entity was found, e.g., since `joe` does not possess any entities in this scenario.

Ontology Similarity

Since Marshal is designed to automatically curate ontologies over time, we use established metrics to measure Marshal's accuracy in automatically clustering leaf categories and inducing supercategories.

Previous research in unsupervised learning uses the Rand index (RI) and adjusted Rand index (ARI) to measure clustering accuracy (e.g., Liang and Forbus, 2014), and research in ontology-learning uses the Onto-Rand index (ORI) (Brank, Mladenic, and Grobelnik, 2006) to measure clustering accuracy with respect to a supercategory lattice. We report clustering results with all three measures. Since the ORI uses a proximity metric between categories, we use the following ancestor-based proximity metric $\delta(c, c')$ between two leaf categories of an ontology, where $Supers(c)$ is the set containing c and all super-categories:

$$\delta(c, c') = \frac{|Supers(c) \cap Supers(c')|}{|Supers(c) \cup Supers(c')|} \quad (2)$$

Intuitively $\delta(c, c')$ is 1.0 when $c = c'$, 0 when c and c' share no ancestors, and (0,1) in all other cases.[2] We use this ontology proximity metric within ORI for Experiment 1 and as a classification accuracy metric for Experiment 3.

Evaluation

We conducted three experiments to validate Marshal's approaches to ontology curation, including ontology induction (Experiment 1), ontological anomaly detection (Experiment 2), and automated classification of new entities into an existing ontology (Experiment 3). All three experiments use an ontology that describes procedures for the ISS. The ontology contains 1774 instances of 420 concrete categories, and 230 additional abstract supercategories that have no instances.

We explore three similarity metrics in our experments for the sake of comparison. We provide examples of each using the description of `sally` from the previous section.

- **Predicate vector**: We compute a vector of all predicates that describe the entity (e.g., `sally`), irrespective of argument index, such as $\langle$`crew`, `o2use-rate`, `has-iss-location`, `possesses`$\rangle$. We normalize these vectors and compare them with dot product.
- **Weighted predicate-argument vector**: We compute a vector of argument positions that describe the entity, such as $\langle$`crew`$_0 = 1$, `o2use-rate`$_0 = 1$, `has-iss-location`$_0 = 1$, `possesses`$_0 = 2\rangle$. We normalize these vectors and compare with dot product.
- **Structure-mapping**: We use structure-mapping graph similarity as described above to compute the normalized similarity score $\|s\|$ of two relational descriptions.

These metrics are listed in order of efficiency, with graph similarity being two orders of magnitude slower than the others; however, all three computations are efficient enough to analyze individual categories and classify individual entities online. All three similarity metrics compute a similarity score [0,1], so these are interchangeable and can be combined with a portfolio or weighted voting approach, as we demonstrate in Experiment 3.

Exp 1: Ontology Induction

In lieu of having an incomplete user-generated ontology for Marshal to extend, we provide Marshal with a set of labeled

[2] We can also measure category proximity with tree distance, but it does not support multiple inheritance as well, among other complications (Brank, Mladenic, and Grobelnik, 2006).

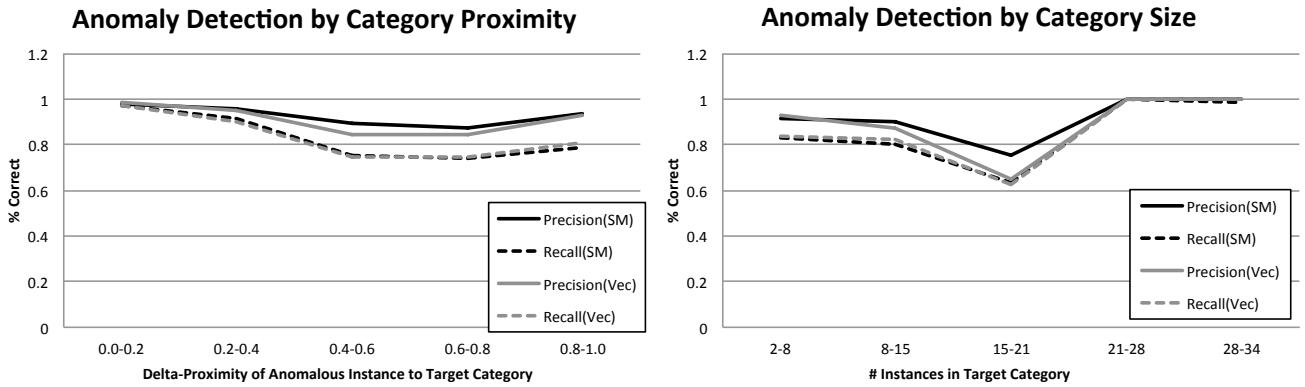

Figure 2: Results for Marshal's anomaly detection, where a single anomaly is seeded in a target category of varying size and distance from the anomaly's true category. At left: Effect of δ-proximity between the target and true category. At right: Effect of the size of the target category.

example descriptions— like the snippets of `crew` instances `sally` and `joe` shown above— and have Marshal recreate all supercategories of a gold-standard ontology from these instances and leaf categories. We had Marshal recreate the ontology using each of the three representations and similarity metrics described above.

For each of the three representation/similarity settings, Marshal first generalizes all of the instances into their leaf categories. For the two vector settings, Marshal sums and then normalizes the vectors of the instances to calculate the category vectors. For the structure-mapping setting, Marshal uses a SAGE algorithm to generalize the instances into categories, as shown for the `crew` category in Figure 1.

After generalizing each leaf category, Marshal uses a greedy algorithm similar to (Liang and Forbus, 2014) to induce new supercategories and incorporate categories into supercategories. Marshal computes pairwise similarity using the three similarity metrics described above, and iteratively selects the highest-scoring pair:

- If the highest-scoring pair are two categories without supercategories, it constructs a new supercategory and compares it against all existing categories.
- If the highest-scoring pair are a leaf category and an induced category, it adds the leaf to the induced category.
- If the pair contains exactly one category without a supercategory, it takes the complement's supercategory.
- Otherwise, it continues with the next highest-similarity pair until it has no more pairs to process.

Results are shown in Table 1. Rand index (RI) measures clustering accuracy regardless of the ontology, adjusted Rand index (ARI) corrects RI for chance, and the OntoRand index (ORI) measures differences in ontological proximity using the δ-proximinity distance. ORI is by far the most meaningful measurement in this experiment, since ORI evaluates the entire ontology lattice and RI/ARI only measure sibling/co-clustering accuracy.

Table 1: Comparison of similarity metrics for ontology induction, on Rand (RI), Adjusted Rand (ARI), and OntoRand (ORI) indices.

	RI	**ARI**	**ORI**
Pred Vec	0.913	0.311	**0.743**
Pred-Arg Vec	0.899	0.278	**0.760**
Structure-Mapping	0.940	0.251	**0.785**

Exp 2: Anomaly Detection

In our second experiment, we used the gold standard ontology from Experiment 1 for an anomaly detection task. Here we are interested in *local* anomaly detection (i.e., identifying an anomaly within a category by looking at *within*-category instances), since the next experiment involves across-category comparison.

In this experiment, we iterate through every possible ordered pair of categories $\langle c_1, c_2 \rangle$, where $c_1 \neq c2$, and use each instance in c_1 as a separate anomaly in the *target* category c_2. We also test Marshal with category c_2 *without anomalies* to see whether Marshal will identify false positives. In every case, Marshal attempts to find at most one anomalous instance seeded into c_2 using either structure-mapping or predicate argument vectors, as described above. [3]

For each anomaly detection round, instead of comparing every instance to every other instance— which would incur n^2 similarity computations— we calculate each instance's *omission score* as the similarity of each instance i to the generalization of all remaining instances. This incurs $2n$ similarity computations: n computations to build the vector-or-structural generalization G, and n computations to score each i against $G \setminus \{i\}$. For a vector representation, Marshal

[3]This experimental setup is similar to the visual *oddity task*, where subjects choose exactly one image that does not belong in a set of images, and human performance has been modeled with structure-mapping (Lovett and Forbus, 2011).

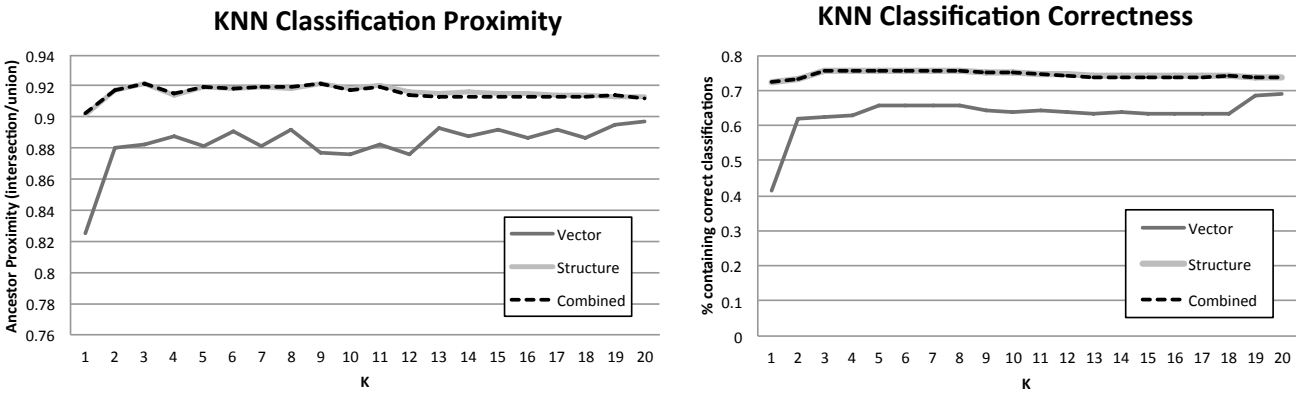

Figure 3: Results for Marshal's leave-one-out k-Nearest Neighbor classification of all instances into the gold-standard ontology. At left: Proximity of Marshal's highest-priority category to the true category, using δ-proximity. At right: Percent of trials where one of Marshal's three suggested categories was the true category (and not just a supercategory thereof).

computes $G \backslash \{i\}$ as a vector difference. For a structural generalization (e.g., Figure 1) Marshal trivially computes $G \backslash \{i\}$ by removing the row for i (and all subsequent relations) from the matrix of entities and relations that comprises G.

The instance with the minimum omission score is the least similar to the remaining instances, but Marshal must also prevent false positives. Consequently, Marshal only flags an instance as an anomaly if its omission score is 1.5 standard deviations less than the average omission score.

We compare the precision and recall of Marshal's anomaly detection results using vector (Vec) similarity and structure-mapping (SM) similarity in Figure 2. We report results over two independent variable. We report the effect of δ-proximity of c_2 and c_1 on anomaly detection in Figure 2 (left). We also report the effect of the number of instances in the target category c_2 on anomaly detection in Figure 2 (right). Both similarity measures perform better in larger categories, probably because higher cardinality allows Marshal to form more coherent generalizations. Both similarity measures have significantly better recall for distant categories. This is sensible, since the standard deviation will be lower for ontologically proximal (and presumably, more similar) categories.

Overall, structure-mapping (Precision=0.93, Recall=0.83) marginally outperformed vector similarity (Precision=0.90, Recall=0.82), but structure-mapping significantly outperformed vector similarity in precision and recall for mid-size and mid-proximity settings, as shown in Figure 2. Both settings produced unacceptable false positive rates in zero-anomaly categories: structure-mapping incorrectly inferred an anomaly in 75% of zero-anomaly categories; and vector similarity incorrectly inferred an anomlay in 87% of zero-anomaly categories. These negative results are overshadowed by the number of correct anomalies inferred in the rest of the experiemnt. We revisit this result in our discussion.

Exp 3: Classifying into an Existing Ontology

In our third experiment, Marshal classifies every instance in the problem into the existing ontology. Marshal is designed to classify entities in a collaborative setting with a subject matter expert, so Marshal should provide a prioritized list of categories— including supercategories— to point the user in the right direction. This means that we are not concerned with fully-automatic, one-shot classification of the perfect category.

We compare Marshal's classification results in three settings: (1) predicate-argument vector; (2) structure-mapping; and (3) a combined approach, averaging the two former similarity scores. For each of these settings, we evaluate classification accuracy and proximity with a simple k-Nearest Neighbor (KNN) algorithm for $k = [1, 20]$.

For each classification, Marshal calculates the top k instances with the given similarity metric, and then performs upward spreading activation from each of the k retrieved instances' categories into the supercategory lattice. The activation value is the $[0, 1]$ similarity score, and the decay is set to 0.25, determined empirically.

For the base case of $k = 1$, Marshal classifies an unlabeled instance by retrieving the most similar instance and asserting its category. Since decay is 0.25, Marshal is guaranteed to choose only leaf categories for $k \leq 4$. For $k > 4$, Marshal can choose both supercategories and leaf categories.

Results of Experiment 3 are shown in Figure 3. Structure-mapping outperforms the vector-based representation across all $k = [1, 20]$ in classification δ-proximity (Figure 3, left). δ-proximity estimates average error distance in the ontology, e.g., if Marshal chooses a supercategory or an incorrect sibling category. Structure-mapping also outperforms the vector representation in absolute category accuracy (Figure 3, right), which measures whether any of Marshal's three highest-priority categories are the true category of the instance.

A combined structure-mapping and vector-similarity ap-

proach performed almost identically to the structure-mapping alone.

Conclusion & Future Work

This paper presented work in progress on Marshal's approach to automatically improving and repairing ontologies. We described empirical results for using similarity-based reasoning to (1) inductively extend an ontology, (2) detect anomalies, and (3) classify unlabeled instances within an existing ontology. Our results indicate that performing structure-mapping over relational graphs yields more accurate ontologies, anomaly detectors, and classifiers than dot product vector similarity.

The ontology used for our experiments contains 103 categories whose instances have identical predicate structure to other categories' instances. For example, the leaf categories `orbit_installed_gap_spanner_[18-21-in]` and `orbit_installed_gap_spanner_[30-45-in]` are disjoint, but their instances comprise the same argument indices of the same predicates, so they are not readily differentiable by any of our similarity metrics. Thus, we do not believe that similarity-based reasoning is a complete solution to addressing model drift— to the contrary, Marshal contains formal planning methodologies as well— but our results suggest that similarity is a practical component for collaborative ontology curation.

Similarity-based inference also presented an unacceptable false positive rates for anomaly detection in Experiment 2: structure-mapping erroneously inferred anomalies in 75% normal categories. This false positive rate might be improved by combining the local anomaly detection (Experiment 2) strategies with the KNN (Experiment 3) strategies, e.g., to only flag an anomaly if it differs sufficiently from its local category *and also* is sufficiently similar to another category.

This paper explored different similarity metrics in isolation as well as in a portfolio setting (e.g., Experiment 3), but previous work (Forbus, Gentner, and Law, 1995, McLure, Friedman, and Forbus, 2015, e.g.) indicates that these can also be combined in serial, e.g., by performing vector dot product as a coarse-but-efficient initial filter and then using structure-mapping as a finer comparator. We are investigating this in the near future, since it holds practical benefits for tractability when scaling to massive ontologies.

This preliminary investigation with Marshal uses only the PDDL `:init` (i.e., problem initialization) and `:objects` (i.e., instance declarations) clauses; however, PDDL also contains action knowledge, goal knowledge, and derived predicates that Marshal could also analyze to extend the ontology and classify instances. We believe that action knowledge— with structured preconditions and consequences— will be especially valuable for learning and classification. We plan to extend Marshal's similarity-based reasoning to use these data and integrate the work presented here with Marshal's other reasoning components.

References

Boddy, M., and Bonasso, R. 2010. Planning for human execution of procedures using anml. In *ICAPS 2010 Scheduling and Planning Applications Workshop*. ICAPS.

Brank, J.; Mladenic, D.; and Grobelnik, M. 2006. Gold standard based ontology evaluation using instance assignment. In *Workshop on Evaluation of Ontologies for the Web, EON*. Edinburgh, UK.

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

Forbus, K. D.; Gentner, D.; and Law, K. 1995. Mac/fac: A model of similarity-based retrieval. *Cognitive science* 19(2):141–205.

Kortenkamp, D.; Bonasso, R. P.; and Schreckenghost, D. 2007. Developing and executing goal-based, adjustably autonomous procedures. In *Proceedings AIAA InfoSysAerospace Conference*.

Liang, C., and Forbus, K. 2014. Constructing hierarchical concepts via analogical generalization. In *Proceedings of the Cognitive Science Society*.

Lovett, A., and Forbus, K. 2011. Cultural commonalities and differences in spatial problem-solving: A computational analysis. *Cognition* 121(2):281–287.

Mcdermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. Pddl - the planning domain definition language. Technical Report TR-98-003, Yale Center for Computational Vision and Control,.

McLure, M.; Friedman, S.; and Forbus, K. 2015. Extending analogical generalization with near-misses. *Proceedings of AAAI 2015*.

Schreckenghost, D.; Milam, T.; and Billman, D. 2014. Human preformance with procedure automation to manage spacecraft systems. In *Proceedings of IEEE Aerospace*.