Exploiting Graph Structure to Abstract & Compress Relational Data

Scott E. Friedman FRIEDMAN@SIFT.NET
SIFT, 319 1st Ave North, Suite 400, Minneapolis, MN 55401 USA

Abstract

AI systems consume volumes of relational knowledge to extract patterns and model the world. We face the challenges of identifying more sophisticated patterns and concepts, scaling to growing volumes of knowledge, and compressing and indexing relational knowledge for future access. This paper describes the domain-general *path-specific isomorphic abstraction* (ψ-abstraction) approach to encoding relational knowledge as relational graphs, and exploiting paths and cycles in these graphs to guide pattern abstraction and summarization. ψ-abstraction automatically summarizes relational graphs by (1) identifying cyclic and acyclic paths; (2) generalizing the graph along those paths; (3) explicitly annotating generalizable sequences; (4) encoding qualitative relations over generalized quantity sequences; and (5) optionally pruning generalized knowledge to compress the graph. We demonstrate ψ-abstraction on seventeen examples spanning diverse domains: 2D spatial; 3D spatial; temporal (calendar) concepts; abstract feedback cycles; narratives; cell signaling; and anatomy. We evaluate ψ-abstraction for data compression, pattern identification, structural similarity of ψ-abstracted examples within and across categories, and one-shot learning of sequence-based concepts.

1. Introduction

Intelligent systems— ranging from higher-order cognitive models, scientific collaboration systems, and domain-specific data miners— consume volumes of relational knowledge to learn informative models of the world and extract patterns of interest. For example, symbolic AI systems consume relational knowledge to induce logic programs, perform comparative analysis (e.g., McLure et al., 2015), induce qualitative and quantitative models (e.g., Friedman et al., 2009), and perform case-based reasoning. These techniques compute mappings, hypotheses, and abstractions over *specific* symbols and statements within the relational knowledge, but most of these approaches are less effective at automatically learning and expressing categories that involve *arbitrary many* symbols in aggregations, sequences, and cycles. This is a representational (or *re*-representational) challenge for cognitive systems.

This paper presents *path-specific isomorphic abstraction* (ψ-abstraction) designed to address the above scalability and representational challenges. ψ-abstraction traverses relational knowledge as a graph— without domain- or task-specific knowledge— and selectively identifies and generalizes isomorphic subgraphs using an enhanced SAGE (Sequential Analogical Generalization of Exemplars) algorithm (McLure et al., 2015). ψ-abstraction compactly represents arbitrarily-long relational paths and cycles (e.g., of temporal, causal, flow, or spatial relations) within a matrix of isomorphic entities and values. It then augments the initial relational knowledge to explicitly describe this matrix as an ordered sequence of entities, and it describes bounded qualitative relationships over generalized qualitative or numerical values. Finally, ψ-abstraction optionally removes the generalized graph structure, leaving the summarized sequence description and qualitative descriptions, as a form of compression. Compressing graph paths has the added benefit of reducing storage requirements and search complexity for massive relational graphs.

Rerepresenting the graph with ψ-abstraction jointly improves similarity-based reasoning and compacts the representation. The claims of this paper are as follows:

1. ψ-abstraction reduces the relational knowledge required to express graphs with repetitive or sequential structure.
2. ψ-abstraction increases structural similarity of category exemplars that involve sequences and cycles.
3. ψ-abstraction summarizes relational regularities of examples for efficient concept learning.

We support these claims with experimental results of ψ-abstraction on seventeen examples spanning diverse domains: 2D spatial representations; 3D spatial representations; temporal calendar concepts; abstract feedback systems; narratives; human anatomy; and cell signaling. We measure information compression, sequence identification, and the increased structural similarity of within-category examples (e.g., two stacks of blocks, with different height) and across-category examples (e.g., a stack and a ladder) after processing them with ψ-abstraction. Finally, we present results of learning the sequence-based concepts of *stack*, *row*, and *staircase* from single examples within a cognitive system. Since ψ-abstraction increases within-category and across-category similarity, it is a promising knowledge representation strategy for learning-by-generalization as well as learning with near-misses.

We begin by describing background work and then we describe the ψ-abstraction approach and our experiments to support the above claims. We close with a discussion of related work, implications, and future work, including developing additional strategies and constraints for ψ-abstraction.

2. Background

ψ-abstraction 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 cases (e.g., Falkenhainer et al., 1989) and within cases (e.g., Ferguson, 1994). These techniques are based on the psychological Structure-Mapping Theory of analogy and similarity (Gentner, 1983). Structure-Mapping Theory holds that people match entities and relationships between relational representations (a *base* and a *target* representation) when they reason by analogy, 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.
2. Parallel connectivity: If a base relation maps to a target relation, so must each argument, in order.
3. Tiered identicality: A relation can only map to another relation with the same predicate.

In graph theory terms, the first (one-to-one) constraint is shared with definition of an isomorphism. The second (parallel connectivity) constraint states that if we treat a relational graph as a directed graph where relations are nodes that have index-labeled edges to their arguments, then if two relation nodes are isomorphic, their child nodes must also be isomorphic. Applied recursively, the first two constraints state that isomorphic (i.e., mapped) nodes must have isomorphic reachable subgraphs.

Structure-mapping algorithms (e.g., Falkenhainer et al., 1989) compute one or more *mappings* between the base and target. Each mapping includes a set of *correspondences* (i.e., isomorphisms) between base and target, a positive real *similarity score* of the correspondences, and *candidate inferences* that can be projected across graphs using the correspondences as support. We can also compute a *normalized similarity score* s in a closed interval $[0, 1]$, using the similarity score of the mapping s_m and the self-similarity scores (computed by summing all graph components) of the base s_b and target s_t:

$$s = \frac{2s_m}{s_b + s_t} \tag{1}$$

Like other research in structure-mapping (e.g. McLure et al., 2015), use this normalized similarity score to measure similarity in our experiments (Section 4).

Computational structure-mapping techniques support analogical generalization (e.g., Friedman et al., 2009, Kuehne et al., 2000) and inter-category comparative analysis (e.g., McLure et al., 2015, Winston,

1970) with encouraging results. However, the constraints of structure-mapping, which make it tractable and practical, also present challenges in certain cases. For example, consider a spatial description of a ladder with six rungs and another of a ladder with twelve rungs. Structure-mapping alone may easily map the side-rails of each ladder together, but due to the one-to-one constraint, at most half the rungs on the 12-rung ladder will have corresponding rungs. Intuitively, people probably mentally *rerepresent* the knowledge to account for repetition prior to performing similarity-based reasoning.

Rerepresentation techniques allow us to preserve structure-mapping constraints with more practical results. Yan et al. (2003) changes the granularity/connectivity of knowledge representation while maintaining semantics. Conversely, Falkenhainer (1988) uses subsumption to relax the category/relation semantics, while maintaining the granularity/connectivity of the graph structure. However, the implementations of these techniques do not directly solve the ladder example above.

The MAGI algorithm, also based on structure-mapping, detects self-similarity in relational representations by computing and then decomposing global analogical mappings over the entire case/graph (Ferguson, 1994). MAGI may indeed identify the rungs as similar components, but it does not perform the generalization, compression, and qualitative summaries of ψ-abstraction. Further, ψ-abstraction does not need to construct a global mapping, and its time complexity is decoupled from the size of the entire relational description, as we describe below.

3. Approach

Here we describe the algorithms and representations of ψ-abstraction. While ψ-abstraction is not a cognitive model *per se*, it builds on the same key principle of previous structure-mapping research: reasoning is guided by the [graph-based] structure of relational knowledge. ψ-abstraction employs this principle at a different granularity, by traversing the knowledge graph itself, along paths and cycles, to perform similarity-based generalization and graph summarization.

We walk through the entire process of ψ-abstraction using the example input of a six-rung ladder. Since a ladder is a sequence of rungs, we show how ψ-abstraction can identify the sequence without any domain knowledge, generalize along the sequence to abstract relations over the current, next, and previous sequence terms, compress the graph using the abstraction, and summarize qualitative knowledge (e.g., monotonic changes over coordinate quantities) over the sequence.

3.1 Relational graphs

ψ-abstraction takes structured relational knowledge as input. Consider the predicate calculus statement:

```
(valueOf (YPosFn rung-2a) 1).
```

This statement includes a relation `valueOf` associating two arguments: a functional term `(YPosFn rung-2a)` referring to the y-position of the entity symbol `rung-2a`; and the number 1. This and other statements can be unambiguously represented in a directed graph. Figure 1 illustrates this graph, and Figure 2 shows a graph of this relation— and many others— that jointly describe a six-rung ladder. We refer to this as a *relational graph*.

We draw relational graphs with squares denoting atomic (i.e., symbol or numerical) nodes, ovals denoting non-atomic (i.e., relational or functional) nodes, and top-level assertions marked with * (e.g., the `valueOf` formula is asserted at the top-level but the `YPosFn` formula is not). Edges are labeled by argument index, starting with zero. The relational graph is acyclic, since no formula can contain itself as an argument.

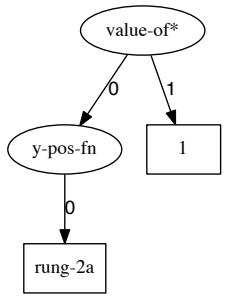

Figure 1. Simple relational graph.

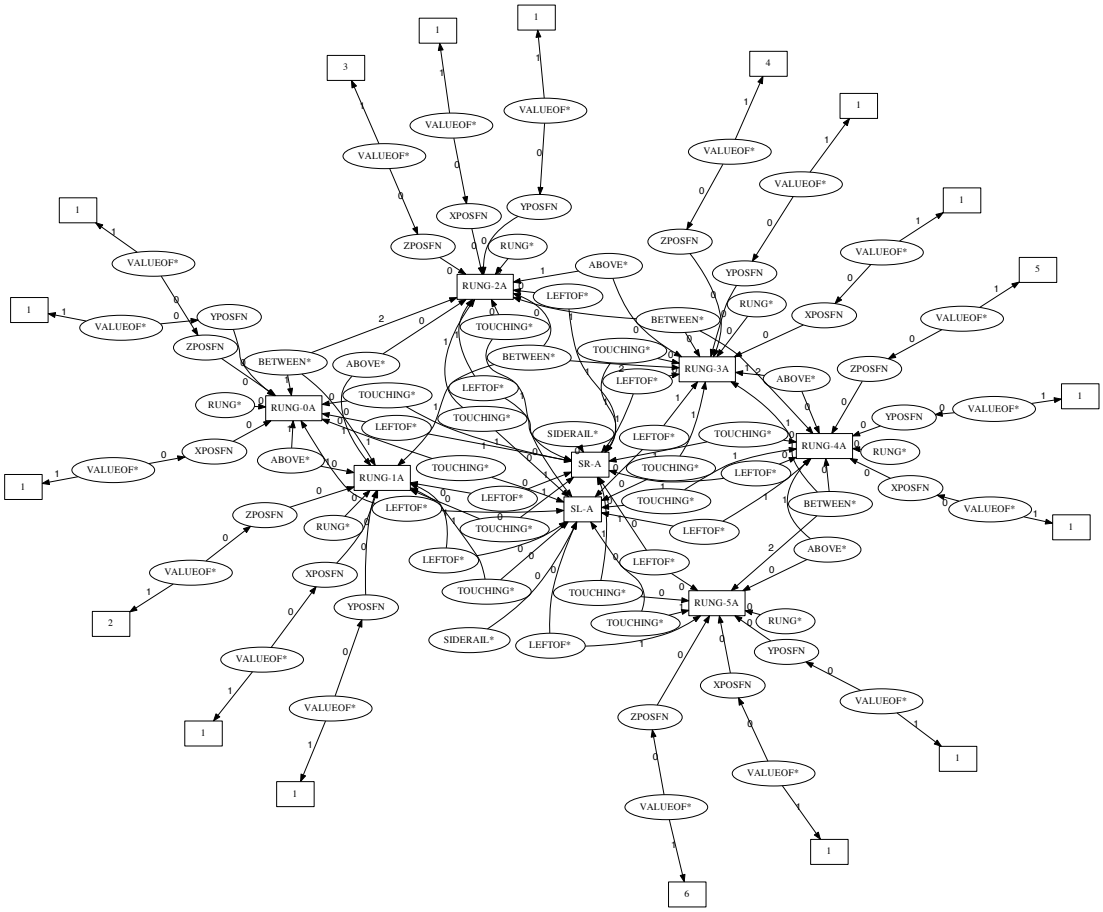

Figure 2. Relational knowledge describing a six-rung ladder displayed as a graph.

From any given entity, e.g., `rung-2a`, we can identify all statements mentioning it by regressing (i.e., traversing backward) along edges to identify all top-level assertions. From any top-level assertion, we can identify all atomic and non-atomic formulas it includes by traversing forward to compute the reachable subgraph. When we perform these operations in succession (i.e., finding all statements mentioning an entity and then finding all formulas in those statements), we call the result the *relevant subgraph* of an entity. [1] Importantly, computing the relevant subgraph of an entity does not involve iterating over all statements, so the time complexity is based solely on the number and structure of statements containing the entity.

3.2 Identifying the abstraction path

Given a relational graph as input, such as the six-rung ladder description in Figure 2, ψ-abstraction first identifies productive paths or cycles to traverse for abstraction purposes. It computes the *binary projection*

1. The result is equivalent to the `CaseFn` of the entity in dynamic case construction (Mostek et al., 2000).

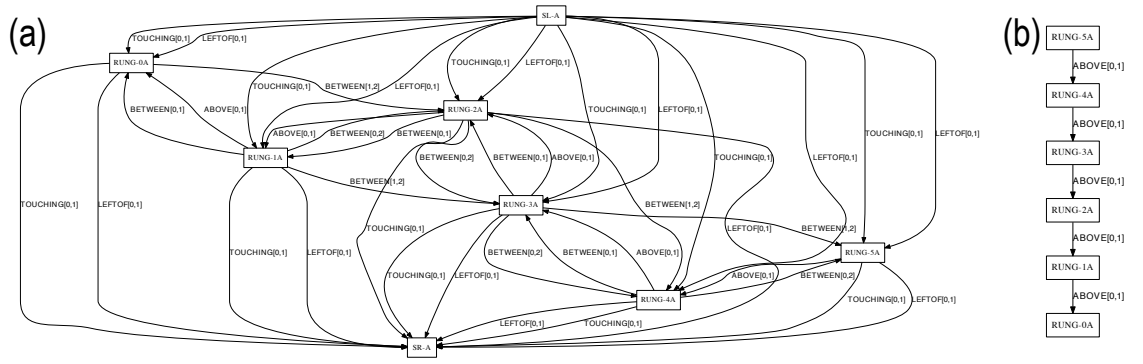

Figure 3. Automatically computing an abstraction path within the relational graph of a six-rung ladder: (a) binary projection of the relational knowledge; (b) binary projection of just the *above* relation.

of the relational graph, as shown in Figure 3a. This graph is comprised of all entities or functional formulas that are associated with other entities or functional formulas (not including numerical terms) in the relational graph. Edges between the entities are predicate-argument combinations linking the entities. For example, `rung-2a` is linked with `rung-1a` with `above[0,1]` since `(above rung-2a rung-1a)` appears in the relational graph. Similarly, `rung-2a` is linked with `rung-4a` with `between[1,2]` since `(between rung-3a rung-2a rung-4a)` appears in the relational graph.

ψ-abstraction then computes the longest traversal or simple cycle in the graph using any single edge label. In this case, it is the `above[0,1]` label. Figure 3(b) shows the `above[0,1]` subgraph of Figure 3(a). This will be the path ψ-abstraction traverses to abstract the relational graph. This is a domain-general heuristic for identifying the abstraction path; however, domain-specific or culture-specific strategies (e.g., following causal links, following temporal relations, or following specific spatial relations) may be used as well, eliminating this path-selection step. Note that this is the only time ψ-abstraction accesses or traverses the global graph; it never accesses the entire graph to perform structure-mapping or subsequent analysis.

3.3 Generalizing the abstraction path

After ψ-abstraction selects the abstraction path, it iterates over each entry (i.e., vertex) in the path (which may be a cycle). For each entry in the path, ψ-abstraction generalizes the path with the following operations:

1. Identify the next formula f.
2. Compute the relevant subgraph G_f for f.
3. If no generalization exists yet, set the generalization to G_f; otherwise:
   (a) Compare G_f with the existing generalization via structure-mapping to compute the best mapping m, where f is constrained to map to the previous formula.
   (b) Merge G_f into the generalization along m with the SAGE merge operation (McLure et al., 2015).

The result of generalizing the relational graph in Figure 2 along the path shown in Figure 3(b) is the generalization shown in Figure 4. ψ-abstraction generalizations include the generalized relations (shown in green), where brightness indicates frequency. ψ-abstraction generalizations also retain the sequence of values of each generalized entity. For instance, note that the formerly-numerical `valueOf` arguments are generalized entity (i.e., `GE-<nbr>`) nodes. Other generalized entities, e.g., `currterm-path-<nbr>`,

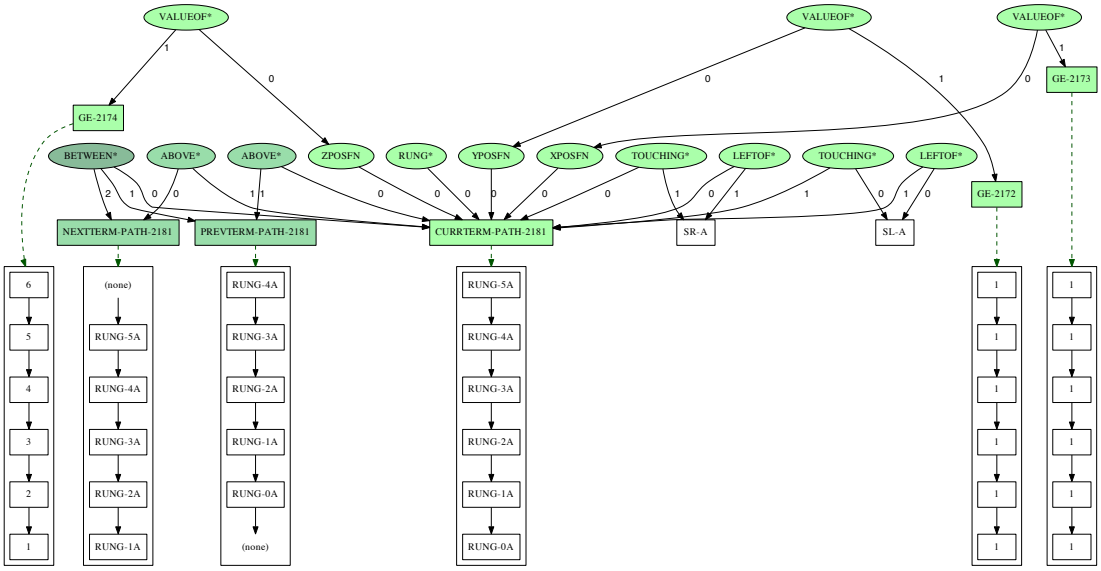

Figure 4. Generalization along the `above` abstraction path of a six-rung ladder.

have been renamed by ψ-abstraction, which we describe later. Beneath generalized entities are the columns of numerical or symbolic values that assumed that node-space in the graph.

Each row across columns describes a single iteration in ψ-abstraction's generalization, e.g., each rung in the sequence: `rung-5a` has `ZPosFn` 6 and is above `rung-4a`; `rung-4a` has `ZPosFn` 4 and is above `rung-3a`; and so on. We call this an isomorphic matrix, or more succinctly, an *isomatrix*. Each column in the isomatrix is a *slot* in the isomorphism, and each row describes the values for each slot for a single iteration. This effectively uses structure-mapping to induce a database table that describes a sequence of redundancies in the graph. This preserves numerical and symbolic data without duplicating the relational data, however it is lossy with respect to retaining which relations are incomplete over which iterations.[2]

Note that some entities, including the siderails `sr-a` and `sl-a` were not generalized against other entities, since they were mapped against themselves for the entire sequence of rungs.

3.4 Summarizing the abstraction path

ψ-abstraction summarizes the graph based on the generalization and associated isomatrix. It first compares each column in the isomatrix against the abstraction path. If a column can be produced with a single left- or right-shift (for acyclic paths) or a single left- or right-rotation (for cyclic paths), then ψ-abstraction will assert that they are the previous or next term in the sequence. ψ-abstraction renames the generalized entity corresponding to the abstraction path as `currterm` it renames the generalized entities corresponding to the previous term as `prevterm` and next term as `nextterm` accordingly, as shown in Figure 4. This rerepresents the generalization to include the following relation, among others:

```
(between currterm-path-2181 prevterm-path-2181 nextterm-path-2181)
```

This states that the current term (i.e., rung) is spatially between the previous and the next in the sequence.

2. For instance, if `leftOf` was not asserted for some rung with the right siderail `sr-a`, the isomatrix would not retain *which* rung this was.

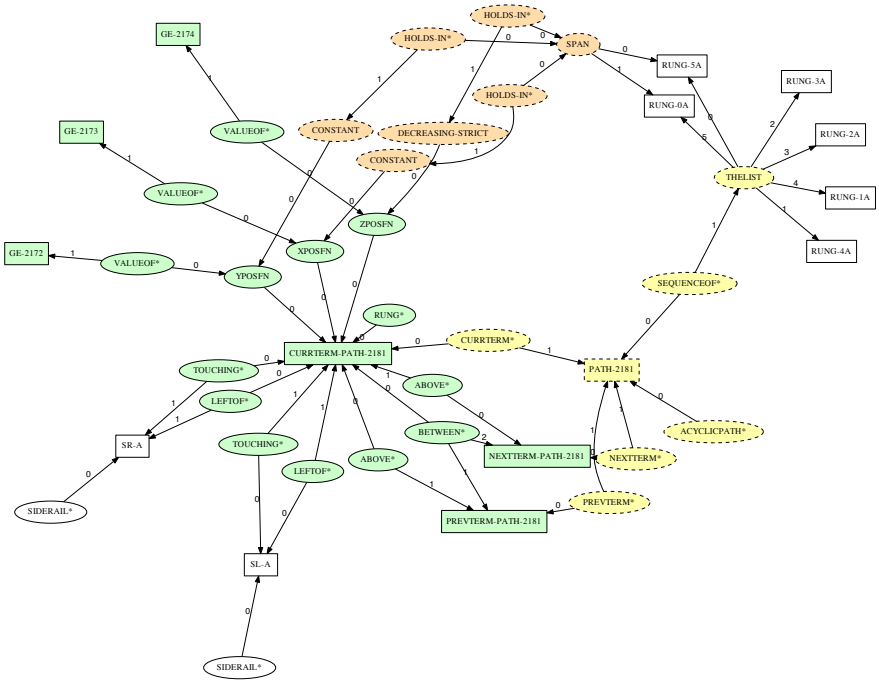

Figure 5. The ψ-abstracted graph of a six-rung ladder, with generalized (green) structure, sequence annotation (yellow), and qualitative descriptions of generalized quantity behaviors (orange).

If there are other columns of generalized entities that are (1) *not* quantity values and (2) *not* next or previous terms, ψ-abstraction returns to its generalization stage and appends this path to its abstraction path, so that each generalization iteration computes the relevant subgraph of the next item in every path sequence. It iterates until no more paths of entities can be generalized.

ψ-abstraction then removes completely-generalized entities, such that no more statements describe the entity in the relational graph that are not included in the generalization, to produce the ψ-abstracted relational graph in Figure 5. The ψ-abstracted graph contains generalized (green) structure as well as unabstracted (in white) structure from the original graph. It describes the generalized path(s) with the yellow, dashed summary knowledge in Figure 5. This includes annotating which entity is the current (and optionally, next and previous) term, whether the path is cyclic or acyclic, and listing the entities in the sequence.

ψ-abstraction performs ordered qualitative analysis of the various quantity behaviors over the isomatrix, and describes this with the orange, dashed summary knowledge in Figure 5. This asserts that for the (sub)sequence `(span rung-5a rung-0a)`, the current term's y-position and x-position are constant, but the z-position is strictly decreasing over the sequence.

This is a more satisfying description of a ladder for exemplar-based learning and reasoning, and ψ-abstraction has constructed it in a domain-general manner, by traversing and generalizing a path in the original relational graph. The isomatrix contains information for reasoning about each rung in isolation, and for partial reconstruction of the original representation if necessary.

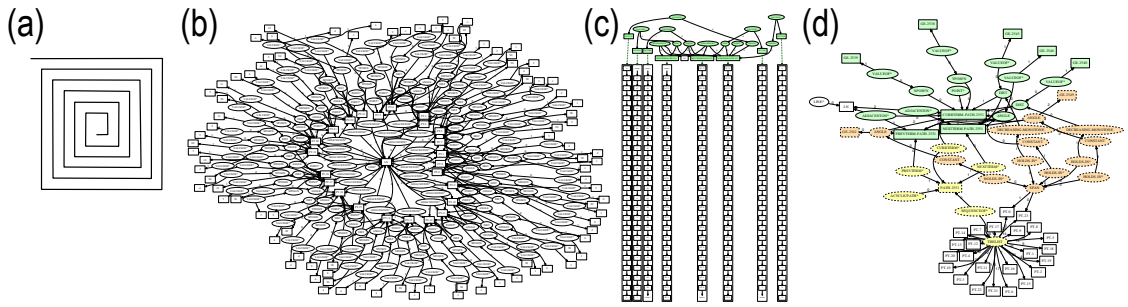

Figure 6. Results of abstracting a spatial description of a square spiral: (a) sketch; (b) initial relational graph; (c) generalization matrix; and (d) ψ-abstracted graph.

4. Evaluation

We next describe three ψ-abstraction experiments, spanning seventeen examples over multiple domains. We relate each experiment to the overall claims of this paper, including ψ-abstraction's ability to abstract relational knowledge and its effect on structural similarity.

4.1 Experiment 1: Demonstrating ψ-abstraction across domains

In our first experiment, we ψ-abstract relational graphs in multiple domains to demonstrate ψ-abstraction's ability to summarize different graph structures and multi-sequence expansions. This includes (1) a spatial description of a square spiral, (2) a description of the human circulatory system with qualitative O_2 concentrations from (Friedman & Forbus, 2011), (3) activation of RAF, MEK, and MAPK for cell signaling, and (4) a summarized plot of the children's book "The Spiffiest Giant in Town."

For each input graph, we ran ψ-abstraction as described in Section 3. For the circulatory system example and the children's book, ψ-abstraction repeats its generalization stage at least three times, having found more abstractable sequences. Figure 6 illustrates (a) the spiral, (b) its relational graph, (c) its generalization and isomatrix, and (d) the resulting ψ-abstracted graph. Figure 7 illustrates the resulting ψ-abstracted graphs of the remaining three examples. Of the remaining three, only the circulatory system example in Figure 7(a) has quantity data, so ψ-abstraction asserts that O_2 decreases from the left atrium (`lt-a2`) to the right ventricle (`rt-v2`), and increases in the complement of the cycle. This is due to O_2 consumption in the body and O_2 infusion in the lungs, respectively.

All three of the ψ-abstracted graphs in Figure 7 contain more than one sequence. This is because ψ-abstraction identified multiple distinct sequences in the isomatrix: in Figure 7(a), ψ-abstraction identified distinct sequences of `FluidFlow` instances, `ContainedFluid` instances, path instances, and container instances; in Figure 7(b), ψ-abstraction identified sequences of activat*ing* entities, activat*ed* entities, `Activation` instances, and base compounds; and in Figure 7(c), ψ-abstraction identifies sequences of clothing (given by the giant), animals (recipients of the clothing), missing entities (mitigated by the clothing), and the associated `Encounter` events. All of these sequences are joined by generalized structure describing how the different sequences combine semantically.

The ψ-abstracted graphs yielded compression ratios shown in Figure 9. The compression rate increases with the length of the sequence, all else being equal. In graphs with no cycles or repetition, ψ-abstraction will not compress or abstract any structure.

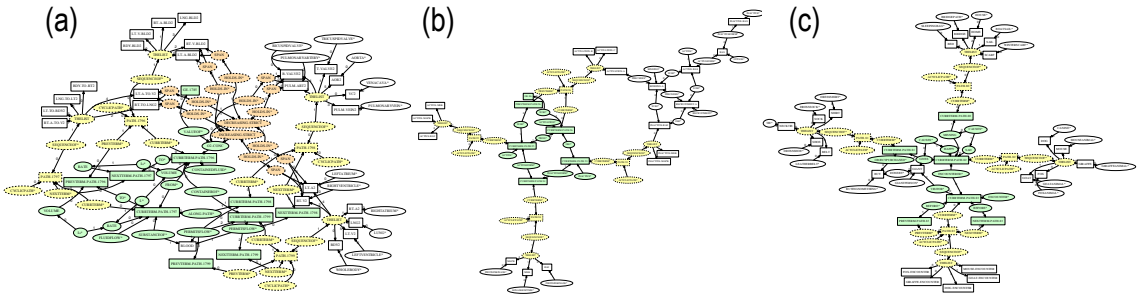

Figure 7. Three ψ-abstracted graphs: (a) double-loop model of the circulatory system with oxygen concentration; (b) activation of RAF, MEK, and MAPK; and (c) the plot of the children's book "The Spiffiest Giant in Town."

4.2 Experiment 2: Evaluating ψ-abstraction's effect on similarity

Next we compare pairs of relational graphs with structure-mapping to compute an *original mapping*, and then we compare the respective ψ-abstracted graphs with structure-mapping to compute an *abstracted mapping*. We then compare the numerical graph similarity scores of the original mapping and the abstracted mapping to determine the effect of ψ-abstraction on graph similarity. Our hypothesis is that ψ-abstraction will significantly increase similarity (i.e., the abstracted mapping will have a higher similarity score than the original mapping). As in previous work on similarity-based reasoning with structure-mapping (e.g. McLure et al., 2015), we compute numerical similarity using the structure-mapping *normalized similarity score*, defined in Section 2, Equation 1. A normalized similarity score of 1.0 indicates perfectly isomorphic graphs, and a score of 0.0 indicates no isomorphisms across graphs.

The pairs of examples used for this experiment include: (1) expert vs. novice circulatory system descriptions (without O_2); (2) ladders of different sizes; (3) seasons vs. months of the year with their average Minneapolis temperatures; (4) stacks of blocks of different sizes and dimensions; and (5) a four-node feedback cycle of viral videos vs. a three-node feedback cycle of mosquito bites.

Figure 8 illustrates the results in the above order, including the original graphs (left), and the resulting ψ-abstracted graphs (right). The ψ-abstracted circulatory system models are mapped as cycles of blood flows through vessels into different containers. The ψ-abstracted feedback cycles are mapped as a cycle of qualitative proportionalities. The ψ-abstracted seasons and months descriptions are mapped as a cycle of states, with the temperature increasing monotonically for half of the cycle, and monotonically decreasing for the complement.

The original and abstracted mappings for the seasons and months are shown in Figure 10. The original mapping between seasons and months (Figure 10a) is too small to read, but it is included for coarse size comparison against the abstracted mapping (Figure 10b). In both mappings, black nodes indicate correspondences (i.e., isomorphism), red nodes indicate non-isomorphic structure in the months description, and blue nodes indicate non-isomorphic structure in the seasons description. In the abstracted mapping, everything corresponds as a cycle of temporal relations, except for the fact that the current term of the sequence is a `CalendarSeason` or `CalendarMonth`. The monotonic descriptions of temperature change are also isomorphic, with the boundary states corresponding (at the right of Figure 10b): `mnsummer` to `jul`, and `mnwinter` to `jan`.

Numerical similarity results are summarized in Figure 11. ψ-abstraction significantly increased the similarity scores in each domain, indicating that the isomorphic portions of the graphs increased significantly.

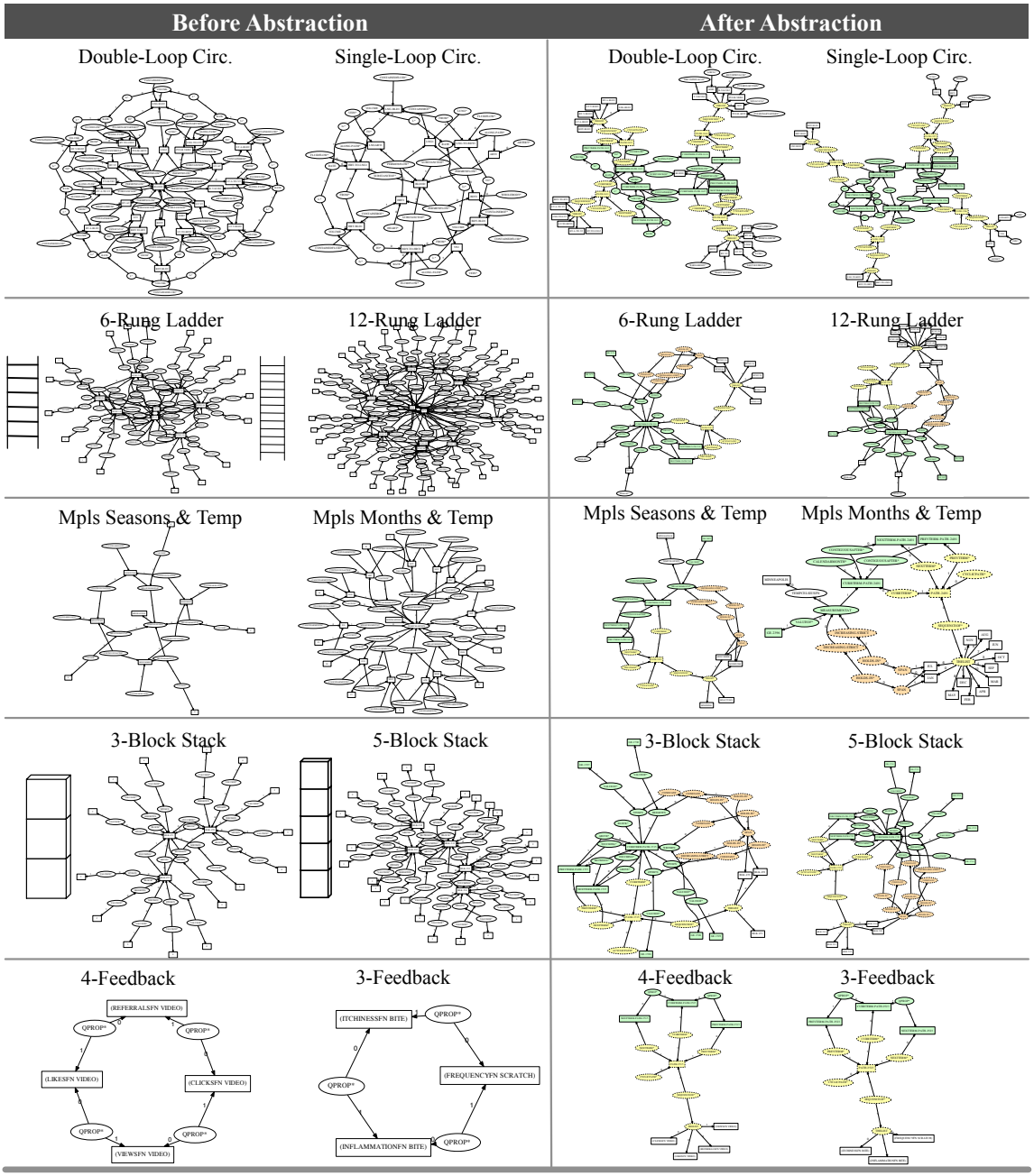

Figure 8. Table of ψ-abstraction results, displaying (at left) the two input representations and their resulting analogical mapping, and (at right) the same inputs after ψ-abstraction, and their resulting analogical mapping. Smaller mappings with more (white) overlapping structure and corresponding (boxed) entities indicate higher similarity.

Domain	Compression
Spiral	5.1
Circulatory	1.24
Cell Signaling	1.21
Narrative	1.18

Figure 9. Compression ratios across four example domains, including qualitative descriptions and sequence descriptions.

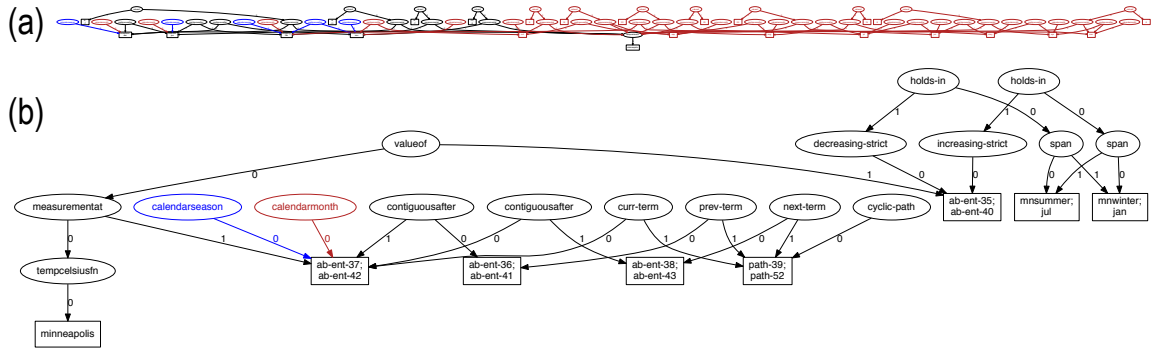

Figure 10. Mappings between descriptions of months and seasons and average Minneapolis temperature in each, (a) before ψ-abstraction and (b) after ψ-abstraction. Isomorphic subgraph in black, non-isomorphic subgraphs in blue (for seasons) and red (for months).

Domain	Abstracted Sim.	Increase in Sim.
2-Loop to 1-Loop Circulatory	0.87	29.9%
6-Rung to 12-Rung Ladders	1.0	37%
Seasons to Months	0.98	67%
3-Block to 5-Block Stacks	1.0	42.9%
4-Feedback to 3-Feedback	1.0	23.4%

Figure 11. Effect of ψ-abstraction on normalized similarity scores (Abstracted Sim.), including the increase from the original similarity score (Increase in Sim.) across five domains. Perfect isomorphism is 1.0 normalized similarity.

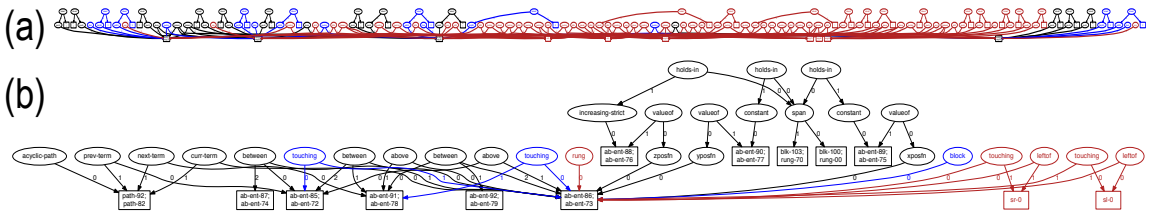

Figure 12. Mappings between a description of a 4-block stack and an 8-block ladder, (a) before ψ-abstraction and (b) after ψ-abstraction. Isomorphic subgraph in black, non-isomorphic subgraphs in blue (for stack) and red (for ladder).

4.3 Experiment 3: ψ-abstraction effect on near-misses

We have so far shown that ψ-abstraction increases *within-category* similarity, so that different examples of the same sequential category are described more uniformly and without redundancy. However, other analogical learning research (e.g. McLure et al., 2015, Winston, 1970) focuses on *cross-category* comparison to identify *near-misses*: similar positive-negative example pairs whose comparison might yield categorization criteria. For example, using structure-mapping to compare an arch with a similar near-miss structure that is *not* an arch (since its legs touch) might yield the candidate inference (defined in Section 2) that the category of *arch* requires that the legs *not* touch.

Our hypothesis is that ψ-abstraction will improve the utility of near-misses for cross-category comparative analysis. We present evidence for this hypothesis by comparing pairs of abstracted and non-abstracted descriptions with structure-mapping and listing the resulting candidate inferences.

For the comparison of seasons and months (and average temperatures) in Figure 10, the two candidate inferences in the ψ-abstracted mapping Figure 10b are intuitive and obvious: one cycle describes `CalendarSeason` elements (blue), and the other describes `CalendarMonth` elements (red).

We compared an original stack and ladder with a ψ-abstracted stack and ladder (i.e., an inter-category comparison), for a similarity increase of 140% (final ψ-abstracted similarity 0.75), shown in Figure 12. This similarity increase is due to structure-mapping utilizing the sequence objects, abstracted category structure, and qualitative descriptions in its mapping rather than putting specific `Block` and `Rung` instances in correspondence. The original mapping in Figure 12a has 34 candidate inferences, and many entities (e.g., rungs) have no correspondence, since there are more rungs than blocks. The candidate inferences describe the specific rungs and blocks themselves: the first block touches the second, and so forth. The abstracted mapping in Figure 12b has 10 candidate inferences, and they are much more category-general: blocks in a stack touch the next and the previous, and rungs do not; rungs all touch side-rails on the left and right, and blocks do not.

This supports the hypothesis that ψ-abstraction increases cross-category similarity and can generate high-utility near-misses for comparative analysis, and it supports our claim that ψ-abstraction summarizes relational regularities for efficient concept-learning. Future work will utilize ψ-abstraction for comparative analysis in broader experiments, as discussed in Section 6.

4.4 Experiment 4: Learning sequential, composable concepts

We present preliminary results using ψ-abstraction to learn sequence-based composable models in a blocks-world domain with our linguistic agent architecture CLiC. We base CLiC's knowledge representation on compositional modeling (Falkenhainer & Forbus, 1991, Friedman & Forbus, 2011) and generative lexicon theory (Pustejovsky, 1991) so that its spatial and conceptual knowledge natively supports qualitative modeling, language understanding, and language generation.

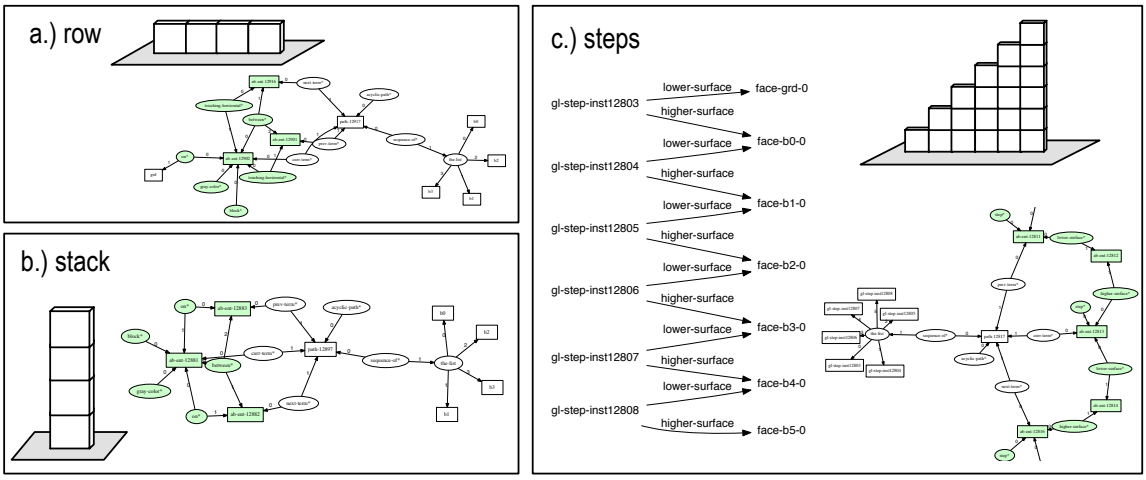

Figure 13. ψ-abstraction of (a) a row of blocks, (b) a stack of blocks, and (c) a stairway of blocks within a blocksworld simulation. The steps involve ψ-abstraction of a sequence of smaller, composable entities (i.e., automatically-instantiated *step* instances) to represent a larger sequence of composable entities (i.e., a stairway).

In our ψ-abstraction experiments with CLiC, we initialize a three-dimensional world with a predetermined configuration of `Block` entities (named with ascending indices `b0`, `b1`, etc.) resting on an entity `grd` of type `Ground`, with numerical data describing position and orientation. CLiC infers qualitative spatial relations over solids and faces, including coordinate-wise abutting/touching, alignment, adjacency, and more. CLiC then instantiates composable concepts (e.g., `step` instances) comprised of multiple constituents in the scene (e.g., `face` instances of blocks) arranged according to *constitutive qualia* (Pustejovsky, 1991) which is analogous to existence constraints in compositional modeling (Friedman & Forbus, 2011).

We then manually invoke CLiC's experiential learning— analogous to people verbally labeling entities or events in the world— which subsequently invokes ψ-abstraction. CLiC attempts ψ-abstraction over primitive entities (e.g., `Block` instances for the row and the stack) and composed entities (e.g., `Step` instances, for the stairs). For primitive entities, CLiC computes a graph of arguments that share relations,[3] and then it selects the highest-weighted path through those arguments. For composed entities, such as `step` instances, CLiC computes a bipartite graph over the entities and their constituents— e.g., a `step` has a `higher-surface` and a `lower-surface`, as shown in Figure 13c— and then it selects the highest-weighted path over the composed entities. CLiC uses these paths for ψ-abstraction, and then produces the abstracted concept descriptions shown in Figure 13. We have hidden the abstracted numerical data from the graphs in Figure 13 to illustrate the central qualitative structure of the concept.

CLiC then uses the ψ-abstracted graphs to generate conceptual sequential models that can be used for recognition and reasoning. The specification of a row is shown below in Equation 2, as a sequence of blocks that are `touching-horizontal` (abbreviated as `touchH`), and all are `on` the ground `grd`.

$$row = \left[\begin{array}{l} \textbf{sequence}(\{\texttt{block}, \texttt{grayColor}\} : \langle b_{i-1}, b_i, b_{i+1} \rangle) : \\ begin = \{\texttt{touchH}(?x, b_i), \texttt{between}(?x, ?y, b_i), \texttt{on}(b_i, \texttt{grd})\} \\ middle = \{\texttt{touchH}(b_i, b_{i-1}), \texttt{touchH}(b_{i+1}, b_i), \texttt{between}(b_i, b_{i+1}, b_{i-1}), \texttt{on}(b_i, \texttt{grd})\} \\ termination = \{\texttt{touchH}(b_i, b_{i-1}), \texttt{between}(b_{i-1}, b_i, ?x), \texttt{on}(b_i, \texttt{grd})\} \end{array} \right] \quad (2)$$

3. See (Raghavan & Mooney, 2013) for a similar argument-graphing approach for learning inference rules.

The row concept is not as general as we would expect; at present, each element must be on the ground. We might circumvent this by subsequently showing CLiC rows that are on other blocks or tables, and then have it generalize from there, using a standard SAGE algorithm. Further, rows must be comprised entirely of `gray-color` entities. While we could trivially remedy this by varying the color of the blocks, it illustrates that ψ-abstraction will only generalize as far as its observations permit.

The stack concept is shown in Equation 3:

$$stack = \left[\begin{array}{l} \textbf{sequence}(\{\texttt{block}, \texttt{grayColor}\} : \langle b_{i-1}, b_i, b_{i+1} \rangle) : \\ begin = \{\texttt{on}(?x, b_i), \texttt{between}(?x, ?y, b_i)\} \\ middle = \{\texttt{on}(b_i, b_{i-1}), \texttt{on}(b_{i+1}, b_i), \texttt{between}(b_i, b_{i+1}, b_{i-1})\} \\ termination = \{\texttt{on}(b_i, b_{i-1}), \texttt{between}(b_{i-1}, b_i, ?x)\} \end{array} \right] \tag{3}$$

Unlike the row concept, the stack concept does not include the `Ground` entity `grd`, since only *one* element of the sequence (i.e., the bottom block `b0`) is on the ground.

The stairway concept is shown in Equation 4, with `loSurf` and `hiSurf` to abbreviate the full relations in Figure 13:

$$steps = \left[\begin{array}{l} \textbf{sequence}(\{\texttt{step}\} : \langle s_{i-1}, s_i, s_{i+1} \rangle) : \\ begin = \{\texttt{loSurf}(s_i, ?x)\} \\ middle = \{\texttt{loSurf}(s_i, ?x), \texttt{hiSurf}(s_{i-1}, ?x), \texttt{hiSurf}(s_i, ?y), \texttt{loSurf}(s_{i+1}, ?y)\} \\ termination = \{\texttt{loSurf}(s_i, ?x), \texttt{hiSurf}(s_{i-1}, ?x)\} \end{array} \right] \tag{4}$$

CLiC's **sequence** semantics allows the row/stack/stairs to start in what we might consider the *middle* of a sequence. For example, the row can start at any block b_i on the ground that is related `touching-horizontal` to another entity. Thus, a row (and similarly a stack and a stairway) can be a subsequence of another such instance, and does not necessarily have to span the entire satisfied sequence. This allows for flexibility of language and reference, and CLiC automatically prioritizes different sequence-based instances based on superset relationships to favor the "whole" concept over strict subsets. This will ultimately allow lazy instantiation of sub-sequences to prevent explosions of concept instances.

After CLiC has learned these concepts, we invoke CLiC to *generate* imaginary examples of these concepts using the above induced knowledge structures (i.e., CLiC generates the KR, but does not build it out of real blocks). We specify a constant sequence length (e.g., of steps or blocks) *a priori* and then CLiC generates a relational graph describing blocks or steps of the given sequence length, generating symbols as necessary for imaginary blocks and steps. Thus, CLiC can correctly use the concepts it learned via ψ-abstraction in a generative fashion.

5. Related Work

Graph rewriting and graph mining have been research topics in computer science for decades, and these techniques have been used to optimize cybersecurity, compilers, and provenance-tracking.

In the graph compression research, previous work focuses on contracting graphs by pairwise chunking (e.g., Matsuda et al., 2000), which iteratively compresses the graph at single points. By comparison, ψ-abstraction iteratively *generalizes subgraphs* of larger relational graphs along a single path, leaving the vertices behind in case they comprise the subgraph of the next iteration. It later summarizes the generalized subgraphs into a sequential structure.

In graph-mining research, machine learning techniques use kernel methods (e.g., Deshpande et al., 2005, Horváth et al., 2004) to select the most discriminating structures within graph-mining and machine learning classifiers. Kernel methods and ψ-abstraction both perform subgraph discovery, but ψ-abstraction exploits *paths* of common subgraphs to directly rewrite (and optionally, prune) the graph along these paths.

6. Conclusion & Future Work

This paper presented the ψ-abstraction approach to generalizing, compressing, and summarizing relational graphs. We have supported our claim that ψ-abstraction compresses sequences of isomorphic subgraphs into summary descriptions on multiple examples across domains. We also supported our claim that ψ-abstraction increases the structural similarity of same-category exemplars when the category permits arbitrary repetition. We showed that ψ-abstraction makes subtle cross-category differences apparent (e.g., in the ladder-to-stack comparison in Figure 12) for learning by near-misses and inducing logical hypotheses for categorization, which supports our third claim. Finally, we demonstrated ψ-abstraction's use for one-shot learning of concepts, to further support our third claim.

The ψ-abstracted concepts learned by CLiC in Section 4.4 suggests that ψ-abstraction can describe sequential procedures to support automated planning and execution. For example, CLiC's ψ-abstracted **sequence** statements describe how to *envision* or *build* a generic stack or row or stairway, of any size— provided a planner or executive can achieve the lower-level categories and spatial relations— after observing a single example. Further, in Section 4.1 we demonstrated ψ-abstraction of explicit causal and temporal chains, and this might support planning and intervention by cognitive systems as an area of future work.

We demonstrated ψ-abstraction as a proof-of-concept on examples without noisy numerical data or distracting entities and relations, when data in the real world is not so charitable. ψ-abstraction's qualitative summarization is sensitive to some noise by asserting monotonic quantity changes in addition to strict increasing/decreasing changes, but other approaches for computing qualitative behaviors from numerical data (e.g., Žabkar et al., 2013) might be more robust.

Acknowledgements

This work was supported by Contract W911NF-15-C-0238 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Thanks to J. Benton, Mark Burstein, David McDonald, and Dan Thomsen for feedback during the development and evaluation of this approach, thanks to James Pustejovsky for Generative Lexicon discussions, and thanks to the reviewers for the 28th International Workshop on Qualitative Reasoning for useful feedback on an ealier version of this manuscript.

References

Deshpande, M., Kuramochi, M., Wale, N., & Karypis, G. (2005). Frequent substructure-based approaches for classifying chemical compounds. *Knowledge and Data Engineering, IEEE Transactions on*, *17*, 1036–1050.

Falkenhainer, B. (1988). *Learning from physical analogies: a study in analogy and the explanation process*. Technical report, DTIC Document.

Falkenhainer, B., & Forbus, K. D. (1991). Compositional modeling: finding the right model for the job. *Artificial intelligence*, *51*, 95–143.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, *41*, 1 – 63. URL `http://www.sciencedirect.com/science/article/pii/0004370289900775`.

Ferguson, R. W. (1994). Magi: Analogy-based encoding using regularity and symmetry. *Proceedings of the 16th annual conference of the cognitive science society* (pp. 283–288). Citeseer.

Friedman, S., Taylor, J., & Forbus, K. (2009). Learning naïve physics models by analogical generalization. *Proceedings of the 2nd International Analogy Conference*.

Friedman, S. E., & Forbus, K. D. (2011). Repairing incorrect knowledge with model formulation and metareasoning. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*.

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

Horváth, T., Gärtner, T., & Wrobel, S. (2004). Cyclic pattern kernels for predictive graph mining. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 158–167). ACM.

Kuehne, S., Forbus, K., Gentner, D., & Quinn, B. (2000). Seql: Category learning as progressive abstraction using structure mapping. *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society* (pp. 770–775).

Matsuda, T., Horiuchi, T., Motoda, H., & Washio, T. (2000). Extension of graph-based induction for general graph structured data. In *Knowledge Discovery and Data Mining. Current Issues and New Applications*, (pp. 420–431). Springer.

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

Mostek, T., Forbus, K. D., & Meverden, C. (2000). Dynamic case creation and expansion for analogical reasoning. *AAAI/IAAI* (pp. 323–329).

Pustejovsky, J. (1991). The generative lexicon. *Computational linguistics*, *17*, 409–441.

Raghavan, S., & Mooney, R. J. (2013). Online inference-rule learning from natural-language extractions. *AAAI Workshop: Statistical Relational Artificial Intelligence*.

Winston, P. H. (1970). *Learning structural descriptions from examples.*. Technical report, DTIC Document.

Yan, J., Forbus, K. D., & Gentner, D. (2003). A theory of rerepresentation in analogical matching. *Proceedings of the 25th Annual Conference of the Cognitive Science Society*.

Žabkar, J., Možina, M., Bratko, I., & Demšar, J. (2013). Learning qualitative models from numerical data. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence* (pp. 3195–3199). AAAI Press.