Semantic Network as Knowledge Representation for Human-Like Agent Solving Raven's Progressive Matrices

Jarrod F. Parkes

Georgia Institute of Technology

Abstract

This paper explores the design of a knowledge representation using semantic networks purposed for a human-like agent solving Raven's progressive matrices (RPMs). For clarity, the semantic network described in this paper is referred to as the Parkes semantic network (PSN). The goal of the agent—and in conjunction PSN—is to not only solve RPMs, but also to reason towards their solutions in a human-like manner. Since RPMs are visually-oriented human intelligence problems, PSN attempts to incorporate the same rich, visual relationships perceived by humans. The preliminary design of PSN draws heavily upon a semantic network constructed by Goel and Joyner (2014)—hereinafter referred to as the Goel and Joyner semantic network (GJSN)—but deviates in order to solve motivating RPMs not previously solvable by GJSN. For each motivating RPM, PSN addresses the shortcomings of GJSN while simultaneously trying to incorporate an element of human-like intuition into its representation. This paper reveals how the design of PSN requires a keen understanding of human visual processing, classification, and cognition.

Keywords: Raven's progressive matrices, Parkes semantic network, Goel and Joyner semantic network

Semantic Network as Knowledge Representation

for Human-Like Agent Solving Raven's Progressive Matrices

Semantic networks, and more generally knowledge representations, provide a basis for solving problems in cognitive systems. For the task of solving RPMs, a task laden with visual reasoning, cognitive systems undoubtedly require a knowledge representation—in this case a Semantic Network—that models meaningful visual semantics. PSN, a semantic network designed for the purposes of a human-like agent, strives to solve RPMs by modeling representations based on the intuitions of humans as they reason through RPMs. The initial structure of PSN is inspired by the GJSN developed by Goel and Joyner (2014), but makes adjustments to account for problems encountered by more complex RPMs (see Appendix A for more information about mentioned RPMs). This paper discusses those motivating problems and the design of PSN by examining its representation scheme and application towards solving RPMs.

1. Motivating Prompt

How would you use Semantic Networks for knowledge representation to design an agent that can answer Raven's progressive matrices? Please illustrate with examples and figures.

2. Description and Articulation of Target Concept

One way of understanding semantic networks is by considering their strong correlation to percepts. Stated more directly, semantic networks can be seen as mappings of knowledge—percepts and their relationships—that are derived from inputs arriving into a cognitive system. Relating this to the design of a human-like agent to solve RPMs, one must consider which knowledge is useful to humans when solving these types of problems. When humans attempt to answer RPMs, one will likely observe that the knowledge relevant is largely goal-driven. And,

because solving visual puzzles is the goal of RPMs, the design of PSN focuses on answering the fundamental question, "What knowledge leads me to correctly solving RPMs?"

2.1. Knowledge through Self-Experimentation

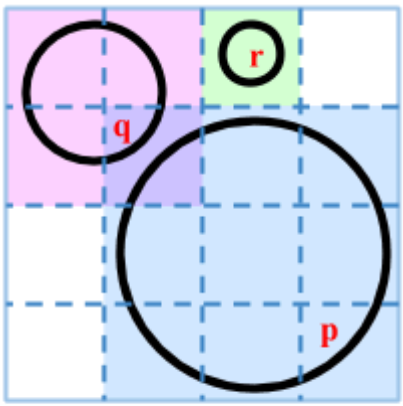
In order to categorize the knowledge necessary for humans to solve RPMs, I attempted a series of RPMs (see Appendix A for more information about mentioned RPMs). Simultaneously, I recorded the steps in my reasoning process as well as the visual information I was using in order to arrive at solutions. Initially, this was problematic because certain questions were so intuitive, it felt as if I skipped the reasoning process; however, when encountering more difficult questions, I was forced to pause and deliberate. From this experiment, I examined a few patterns that consistently allowed me to arrive at correct answers: these can be further reduced into patterns of classification and correspondence. From these patterns, I was able to identify the pertinent percepts and knowledge for solving RPMs.

2.2. Knowledge Related to Classification

A core problem to address when solving RPMs is that of the classification of visual objects—stated in plain English, "What am I looking at?" For humans, this process is quick and intuitive, but for an artificial agent it is difficult and can be riddled with ambiguity. So, to simplify the requirements for designing PSN, I made the following assumption that the agent in question can already understand the concept of shapes and their rotations. But, given that the agent can recognize a shape and shape, how might it describe its size or shading? For this, I surveyed the entire set of RPMs at my disposal and developed a classification system using grids.

2.2.1 Size grid method. To classify the size of shapes in RPMs, an agent must be able to reason about a shape's size relative to another shape or relative to the image frame. Indeed, one

could state the Earth is large, but when compared to the size of the Milky Way, it is very small. So, as a basis for comparison, a shape's size is related to the area of RPM image frames. In order to make an accurate classification, however, a level of classification granularity must be decided upon. By surveying the set of RPMs, it was determined the small, medium, and large would be sufficient. Unfortunately, this still does not solve the problem of classifying shape size. In order to do this, an objective specification must be given for classifying shapes as small, medium, or large—introduce the size grid method.



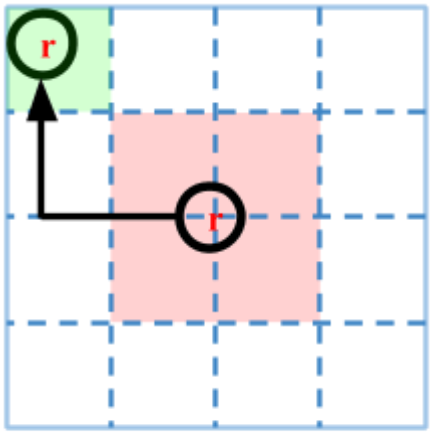
Rules
small = 1 grid square
medium = 2-4 grid squares
large = 4+ grid squares

p = 9 grid squares (large)
q = 4 grid squares (medium)
r = 1 grid square (small)

Figure 2.2.1a - Primitive size grid method

As seen in Figure 2.2.1a, the classification of size can be determined by applying a simple rule where size is based on the number of grid squares a shape inhabits. Generally, this approach works, but it does not account for corner cases when shapes may reside on the borders of grid squares as seen in Figure 2.2.1b. To solve this problem, shapes must be translated, or normalized, to the origin so that a proper comparison can take place. While this solves the problem, it is not the most elegant solution. One could argue that computing the number of grid squares a shape inhabits is as simple as dividing the shape's area by the area of a single grid square. In theory, that approach does work; however, it is not guaranteed the agent will have the data on-hand to compute such areas. Regardless, each approach allows the agent to classify size

and establishes the first piece of knowledge included in PSN. This knowledge was also included in the GJSN.



Rules
small = 1 grid square
medium = 2-4 grid squares
large = 4+ grid squares

Before Correction
r = 4 grid squares (medium)
After Correction
r = 1 grid square (small)

Figure 2.2.1b - Improved size grid method

2.2.2 Fill grid method. Similar to the size grid method, the fill grid method uses a grid system local to each shape for classifying its fill type. Through the survey of RPMs, I determined the highest granularity required was by quadrant—see the square in Figure 2.2.2a. Thus, by using quadrants, each shape's fill can defined as belonging to one of the following classes: unfilled, fill-1, fill-12, fill-123, filled, and so on. Figure 2.2.2b gives an example of these types of classifications. In comparison, GJSN does not include this extra bit of knowledge and, as a result, can only represent fills as all-or-nothing.

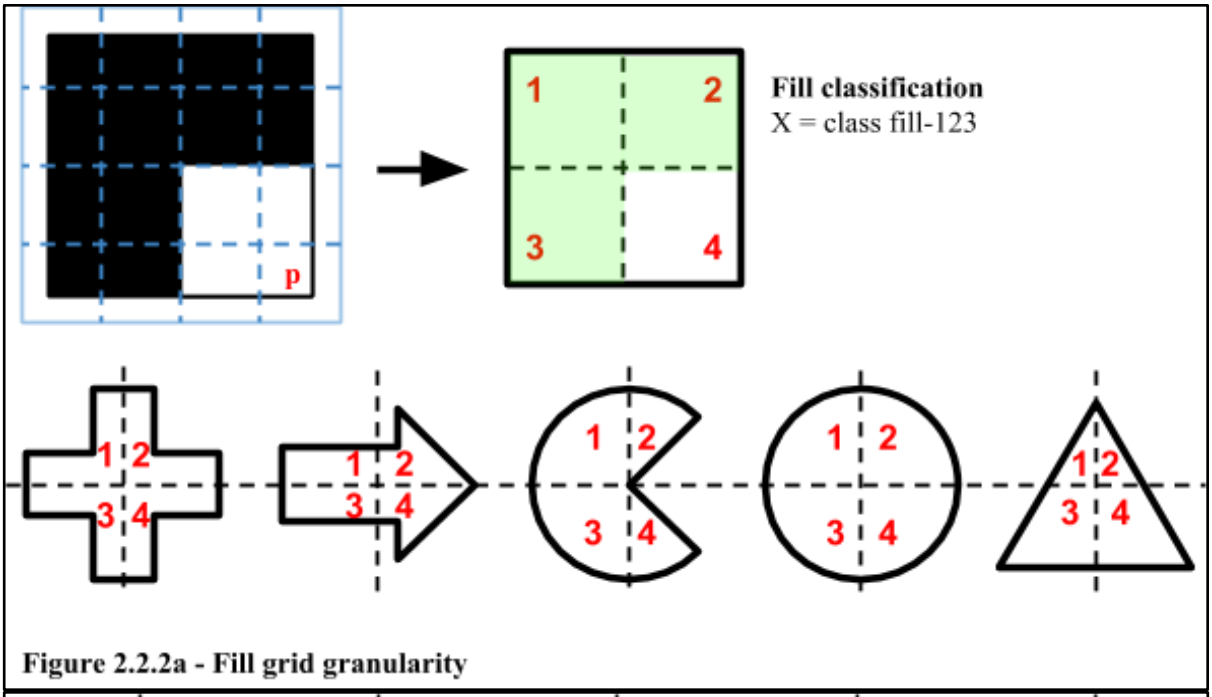


Figure 2.2.2a - Fill grid granularity

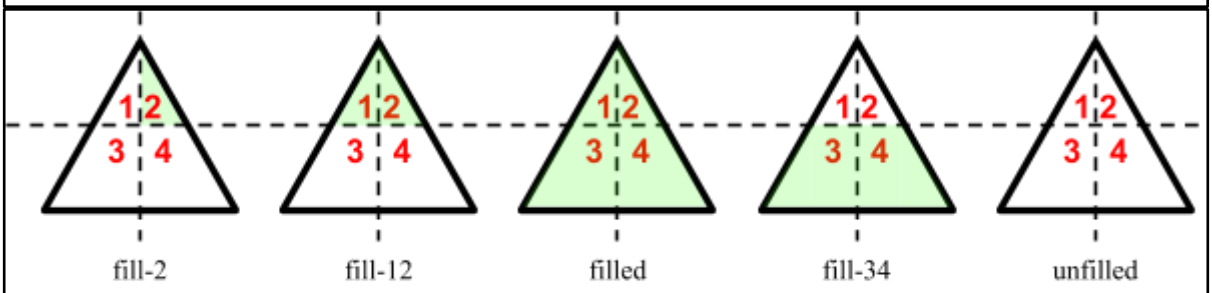


Figure 2.2.2b - Fill classification examples

2.3. Knowledge Related to Correspondence

In RPMs, after classifying objects as shapes with size, fill, and rotation, a second challenge presents itself. How do we determine which shapes correspond from one image to another? In other words, how do we solve the correspondence problem? For me, it starts with another intuitive, human process. First, I classify two images as visually equivalent—either positional, reflective, or rotational equivalence. Next, I compare shapes, sizes, and any additional properties to resolve ambiguities and establish the actual correspondence between shapes. For

me, as well as PSN, generating a correspondence between shapes is the basis for later synthesizing the relationships between shapes.

2.3.1 Visual Equivalence. In the context of PSN, visual equivalence is a concept that relates the center points of shapes between images. By using the center points of shapes, a human or artificial agent can make sense of the translations occurring in RPMs. Besides looking at the center points, additional reasoning is often required to establish a full correspondence between shapes. For the design of PSN given in this paper, it is assumed that this additional reasoning takes place by utilizing the shape, fill, rotation, and other positional knowledge gained through classification. An example the different visual equivalences are given in Figure 2.3.1.

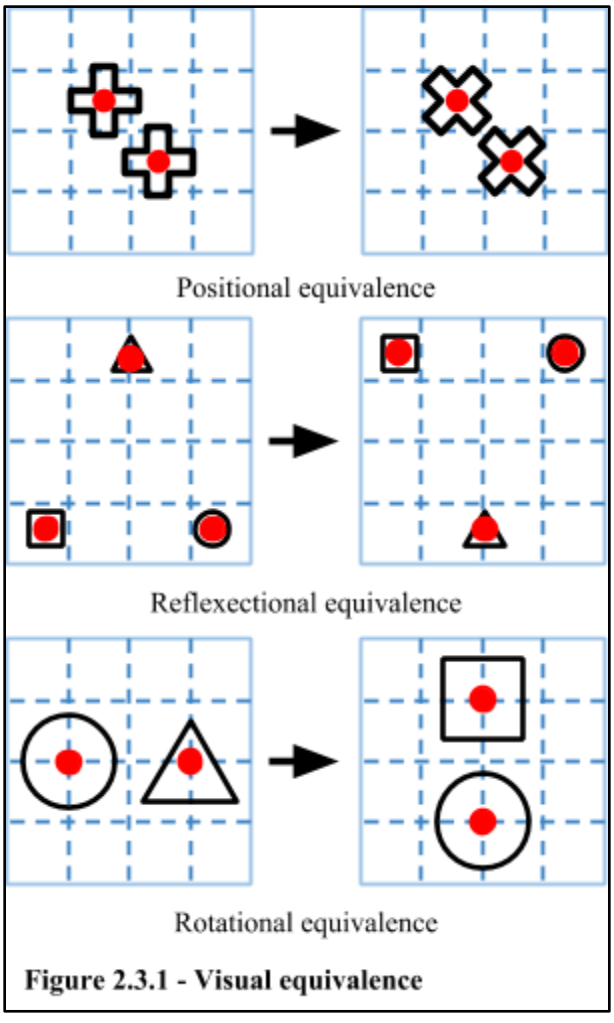


Figure 2.3.1 - Visual equivalence

2.4. Synthesizing Translations

Understanding classification and correspondence are the first steps in solving RPMs, and they provide a basis for the knowledge represented in PSN. Further, if one can assume that these preliminary steps complete without error, then generating the translations between shapes in RPMs becomes a matter of inductive reasoning. For example, imagine that correspondence is established between an unfilled square and fully-filled square—called Square-p. Using this knowledge, one can induce that Square-p underwent the filled (fill-1234) translation to become fully-filled. In some cases, however, ambiguity may exist between two corresponding shapes. If this occurs, then weights can be assigned to the possible translations based on some heuristic measure. Using these weights, the possible translations can be analyzed until one translation is revealed to be the most optimal candidate. Finally, once the translations have been synthesized, they can be used to test or predict possible solutions to RPMs—see the PSN solving algorithm.

3. Designing a Suitable Semantic Network

Using my analysis of the knowledge pertinent to solving RPMs as humans, a knowledge vocabulary and structure revealed itself for PSN. Using GJSN as a basis, PSN is composed of graphical nodes connected by directed links that establish structure and relationship semantics. Additionally, PSN expands on GJSN with two main enhancements—the matrix grid and state containers—and other small changes like more descriptive translations and content like the fill-12 translation described in the fill grid method.

3.1. Matrix Grid

In order to delineate between images in RPMs, PSN adds the subtle notion of the matrix grid. The matrix grid is an abstract representation of the locations of PSN lexicons and their

connected inner networks in relation to each other. Depending on the dimensions of the RPMs, the matrix grid grows or shrinks to mirror the dimensions of the problem matrix. In Figure 3.1, an example of the matrix grid is given a 2x2 RPM.

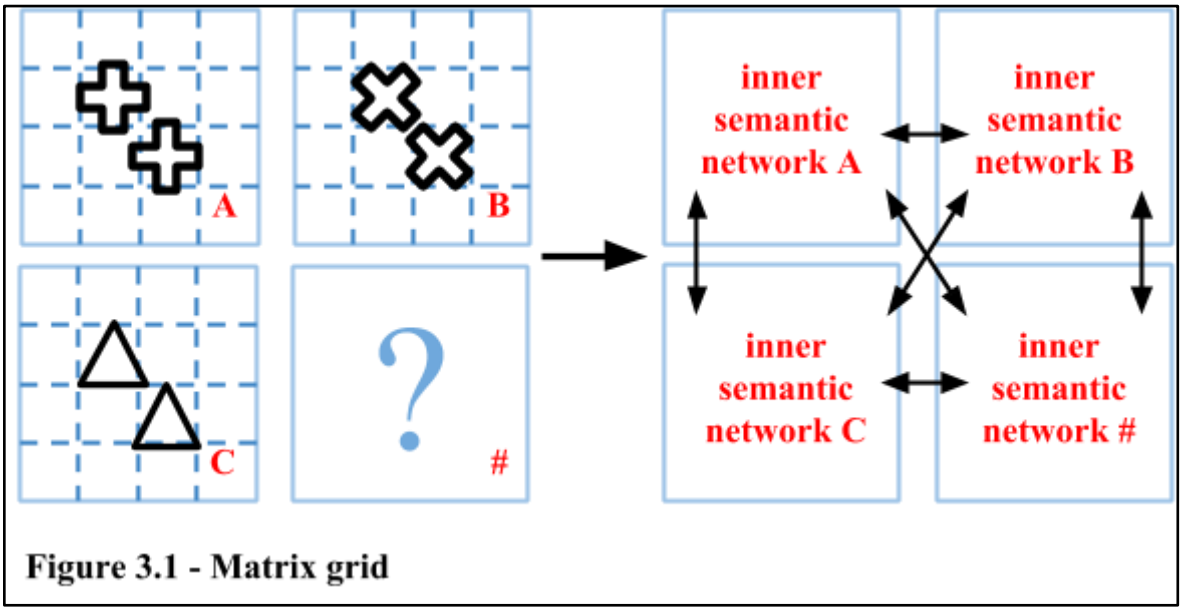
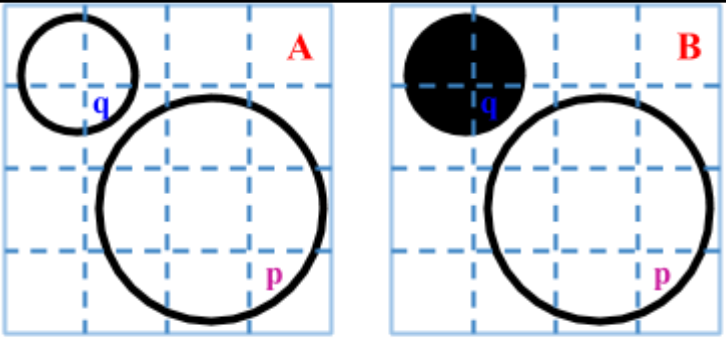
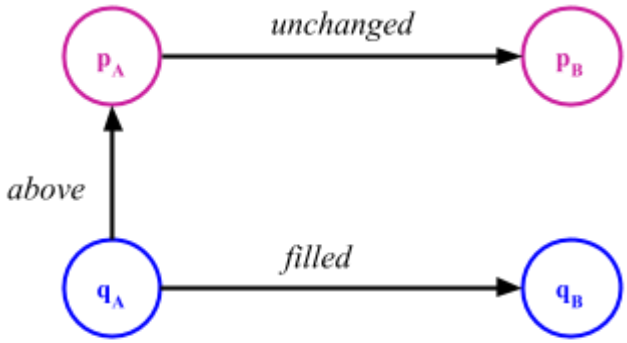


Figure 3.1 - Matrix grid

3.2. State Containers

To effectively describe the images in RPMs, PSN includes the notion of object state. This is a useful mechanism because it gives PSN the ability to create clearer translations between objects. For simplicity, state containers are visualized as separate frame-like entities like in Figure 3.2, but they could also exist as graphical nodes.





State containers for p_A and p_B

p_A {
shape: circle,
size: large,
quads: [5,6,7,9,10,11,13,14,15],
fill: unfilled,
rotation: 0,
center: 10,
...
}

p_B {
shape: circle,
size: large,
quads: [5,6,7,9,10,11,13,14,15],
fill: unfilled,
rotation: 0,
center: 10,
...
}

Figure 3.2 - State containers

3.3. Representation Scheme

See Appendix B for a listing of PSN's representation scheme.

4. Application of Semantic Networks

4.1. PSN Solving Algorithm

To solve a single RPM problem, the PSN is used alongside its associated agent—together they follow an algorithm that can be split into three parts: (1) classification and correspondence, (2) translation analysis, and (3) synthesizing or testing given solutions.

4.1.1 Classification and correspondence. To begin, the agent is responsible for generating PSN nodes given some image. These nodes are then inserted into the matrix grid where their positions correlate to the image's position in the RPM problem. To actually generate these nodes and populate them with content, the agent might utilize techniques such as the size

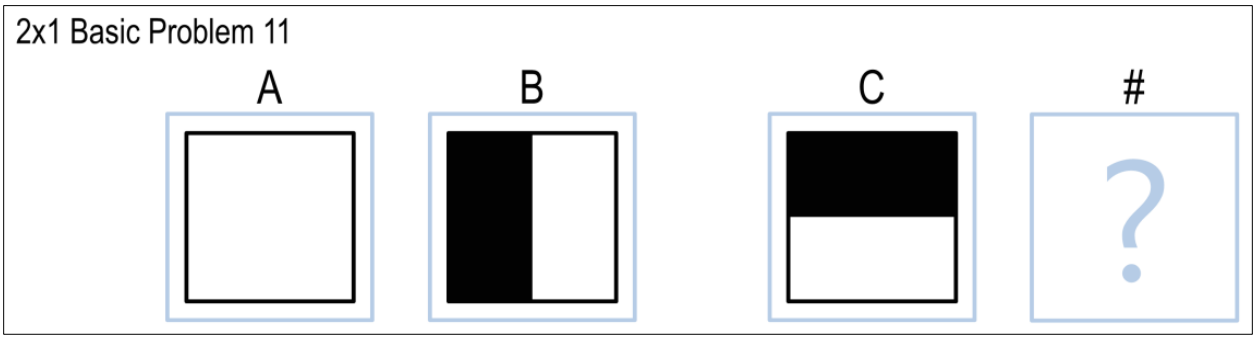
grid and fill grid methods mentioned earlier. Once complete, the nodes and their content form the agent's initial knowledge representation about the RPM problem.

4.1.2 Translation analysis. The next step of the algorithm is to generate the translation knowledge that exists between nodes in the PSN—this happens both within an image and between images. Regardless of the type of translation, they are dealt with similarly. For translations within an image, they can be generated by applying inductive reasoning directly to the knowledge stored in nodes; however, for translations occurring between images, the images must first be compared to establish some visual equivalence. Once complete, inductive reasoning is again used to determine the translations occurring between images and their corresponding shapes. In either case, when inductive reasoning cannot resolve ambiguity, heuristic measures can be taken by assigning weighted values to the possible translations until an optimal solution is found. The final step in translation analysis would involve assigning some certainty factors to the generated translations. These certainty factors, when coupled with a solving strategy such as generate and test, can be used to compare multiple possible solutions.

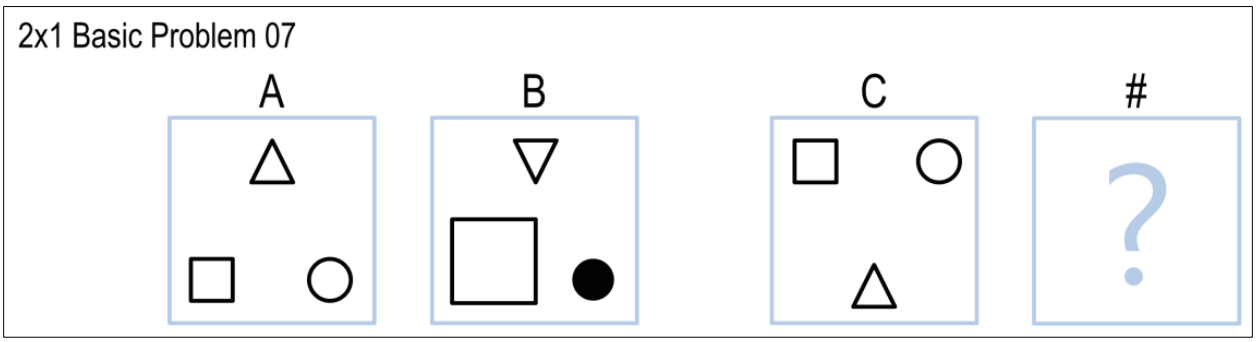
4.1.3 Synthesizing or testing given solutions. The question as to whether synthesize solutions or test given solution is largely based on the solving strategy chosen. Because the focus of this paper is on semantic networks as knowledge representation, and not the solving strategies, we will not discuss those concepts here. But, regardless of the strategy, the correct solutions will likely only be found if PSN is an acceptable knowledge representation of the problem.

4.2 Addressing GJSN Shortcomings

In this section, I will address the RPM examples which presented issues with the GJSN, and will address how they motivated additions to the PSN.



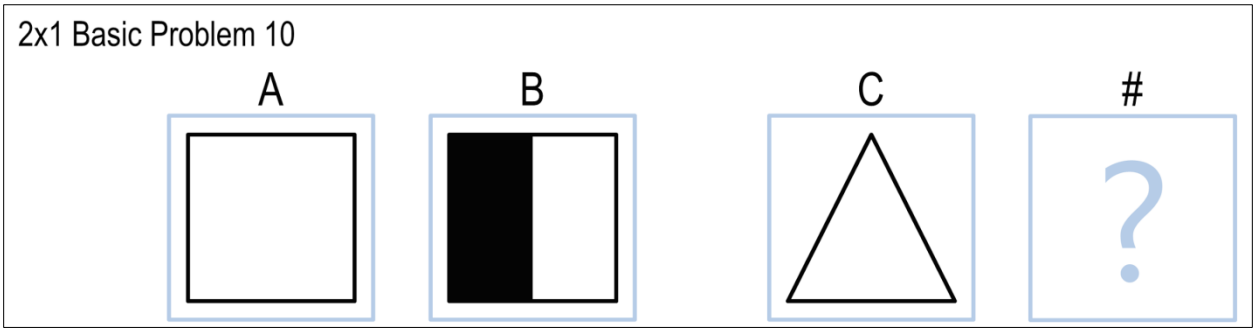
4.2.1 Basic Problem 11. This problem presents a challenge for GJSN because it is unable to specify more granular levels of fill in its knowledge representation. In GJSN, the best possible translation between A and B would be "fill". To address this problem, PSN captures more information in nodes using state containers so that the square in A is considered "unfilled" and the square in B is considered "fill-13". Thus, a "fill-13" translation could be synthesized between the squares in A to B. By applying that same translation to C you arrive at the correct answer of 4 (not shown).



4.2.2 Basic Problem 07. If the establishing correspondence fails between the images A and C, then this problem also presents a challenge for GJSN. Because the correspondence methods associated with GJSN are not provided directly, it cannot be said with assurance that this problem would cause GJSN to generate an inaccurate knowledge representation; however, PSN tries to accommodate this type of problem by specifying additional content in using state containers. For example, PSN would include that the triangle in A has a center point that is

contained within, or on the border of, two grid squares—likely grid squares 1 and 2 (assuming at 16-square grid with indexing beginning at zero). Using this representation, an agent would be able to determine that a triangle in C with the same size as the triangle in A is now centered over grid squares 13 and 14. If this determination is extended to the other shapes from A to C, it can be determined the A has reflection equivalence to C. Armed with this information, the agent could then begin to synthesis new translations from C to D.

4.3 Applying PSN: 2x1 RPM



For demonstration purposes, 2x1 Basic Problem 10 is used to show how PSN could function as a knowledge representation for an agent solving RPMs. For simplicity, the PSN for this problem is split into two separate figures. Figure 4.3a contains the nodes and their translations and Figure 4.3b shows the state containers for each node.

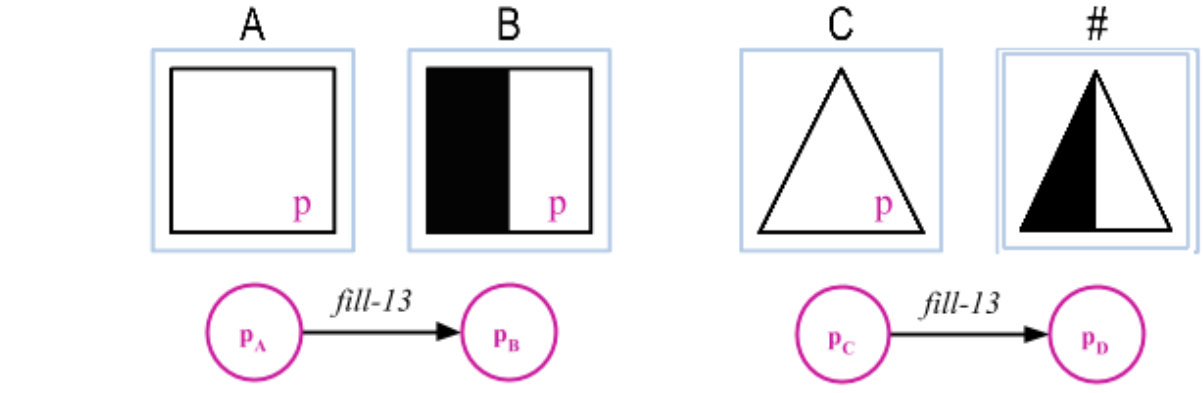


Figure 4.3a - Nodes and translations (possible solution shown) for 2x1 Basic Problem 10

State containers for p_A, p_B, p_C, and p_D

p_A {
shape: square,
size: large,
quads: all,
fill: unfilled,
rotation: 0,
center: [5,6,9,10],
...
}

p_B {
shape: square,
size: large,
quads: all,
fill: fill-13,
rotation: 0,
center: [5,6,9,10],
...
}

p_C {
shape: triangle,
size: large,
quads: [1,2,5,6,7,8,9,10,11,12,13,14,15],
fill: unfilled,
rotation: 0,
center: [5,6,9,10],
...
}

p_D {
shape: triangle,
size: large,
quads: [1,2,5,6,7,8,9,10,11,12,13,14,15],
fill: fill-13,
rotation: 0,
center: [5,6,9,10],
...
}

Figure 4.3b - State containers for 2x1 Basic Problem 10

4.4 Concerns and Improvements

Before PSN can be fully realized, more RPMs should be provided. At this point, the design of PSN is based entirely on small subset of 2x1 RPM problems. If more 2x1 problems, as well as 2x2 and 3x3 RPMs, are incorporated into the design of PSN, it is very possible that new knowledge would be required to solve them. For example, if a new 2x1 problem was introduced in which the fill of a shape is more granular than quadrants, then the knowledge representation present in PSN would need to change. The same would apply if shape fills could be striped or patterned. This is obviously a limiting factor of the PSN design; however, I believe the approach for what knowledge should be represented by PSN is valid. Thus, the more difficult question may be how to generate complex knowledge representations given increasingly complex RPMs.

References

Goel, A., & Joyner, D (2014). *Knowledge-Based AI: Cognitive Systems, Semantic Networks*. Retrieved from http://udacity.com.

Appendix A

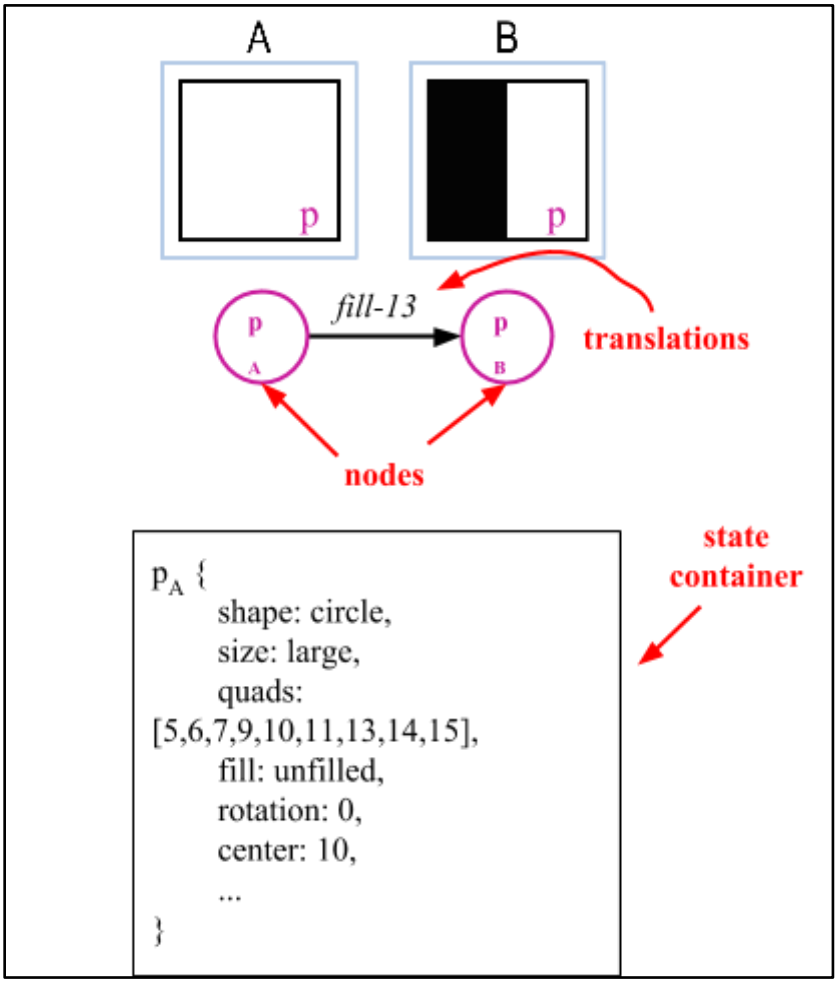
RPMs Mentioned in Paper

The RPMs used in this paper were distributed as part of the Spring 2015 offering of CS 7637: Knowledge-Based Artificial Intelligence at the Georgia Institute of Technology. If you would like access to the RPMs, first send correspondence to jparkes8@gatech.edu or to Jarrod F. Parkes at 111 N Rengstorff Avenue, Mountain View, CA 94043 so your request is acknowledged. Then, pending approval, the RPMs can be distributed for your access.

Appendix B

PSN Representation Scheme

The following image and table describe the PSN representation scheme:



Possible Translations	above/below, left/right, contains, unchanged, deleted, appeared, fill-#, unfill-#, translate-left/right, translate-up/down, shrink, grow, rotate-#, reflect-X, reflect-Y, change-shape, intersects, ...
Possible State Information	shape, size, quads (present-in), fill, rotation, center (present-in), overlaps, above/below, left/right, contains, intersects, ...