CS7637: Overall Project Description (Fall 2016)

This document gives high-level learning goals and guidance for the sequence of class projects in CS7637: Knowledge-Based AI: Cognitive Systems for Fall 2016. The first half covers the overall goals of the project. The second half covers some of the more specific details about the project deliverables and requirements.

Quick-Link Table of Contents:

In a (Large) Nutshell

Your project for this Fall is to create an AI agent that can pass a human intelligence test. You'll download a code package that contains the boilerplate necessary to run an agent you design against a set of problems inspired by the Raven's Progressive Matrices test of intelligence. Within it, you'll implement the Agent.java or Agent.py file to take in a problem and return an answer.

There are four sets of problems for your agent to answer: B, C, D, and E. Each set contains four types of problems: Basic, Test, Challenge, and Raven's. You'll be able to see the Basic and Challenge problems while designing your agent, and your grade will be based on your agent's answers to the Basic and Test problems. Each project will add a problem set or two: on Project 1, your agent will answer set B; on Project 2, your agent will answer sets B and C; and on Project 3, your agent will answer sets B, C, D, and E. Thus, Projects 1 and 2 build toward Project 3, the ultimate deliverable. Your grade will be based on two components: how

well your agent performs on the problems, and a project reflection you turn in along with your agent. If your agent improves on an earlier project's problems, that will also be used to bump up your score on those earlier projects.

Different problems will also give your agent different amounts of information. Certain problems in problem sets B and C (specifically, the Basic and Test problems) will provide "verbal" representations. Verbal representations are structured representations that verbally describe what's inside a figure in a problem. For example, in the first problem below, a verbal representation would describe figure A as "a large, unfilled circle inside a very large, unfilled circle", and figure B as "a small, unfilled square inside a very large, unfilled circle", using a more structure representation. Your agent would take those descriptions and produce an answer from the eight choices. In all the other problem sets, however, your agent will *only* be given the images themselves -- what we call a "visual" representation -- in .png format. It will have to take in the image of the figures themselves and do its reasoning based on those.

Every problem set provides visual representations, so you can try approaching these problems using visual representations (instead of or in addition to using verbal representations) as early as you want. Project 3's problem sets (D and E) *only* provide visual representations, so you'll have to try a visual approach eventually. However, verbal approaches tend to be somewhat easier because a human has already interpreted the figure, so you may find it best to rely mostly on the verbal representations for the first two projects. Note that *all* the optional problems (the Challenge and Raven's problems) only provide visual representations, so if you want to try those problems during Projects 1 and 2, you'll want to try a visual approach then. Your agent will run against every problem on Project 3, though, so you'll never miss out on the chance to give those a try.

Don't worry if the above doesn't make sense quite yet -- the projects are a bit complex when you're getting started. The goal of this section is just to provide you with a high-level view so that the rest of this document makes a bit more sense.

Background and Goals

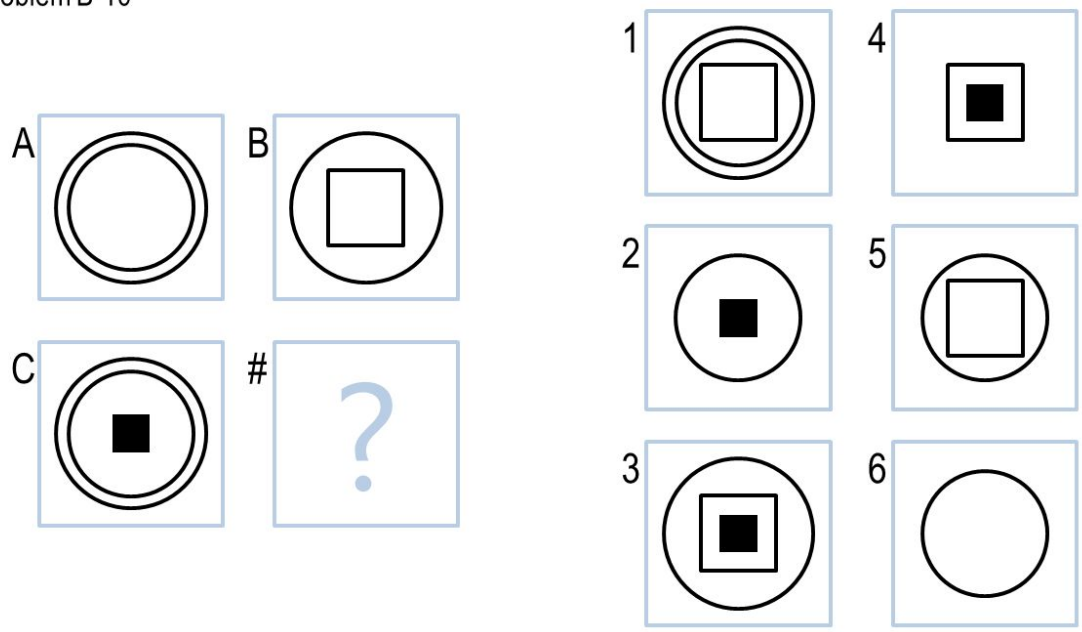
This section covers the learning goals and background information necessary to understand the projects.

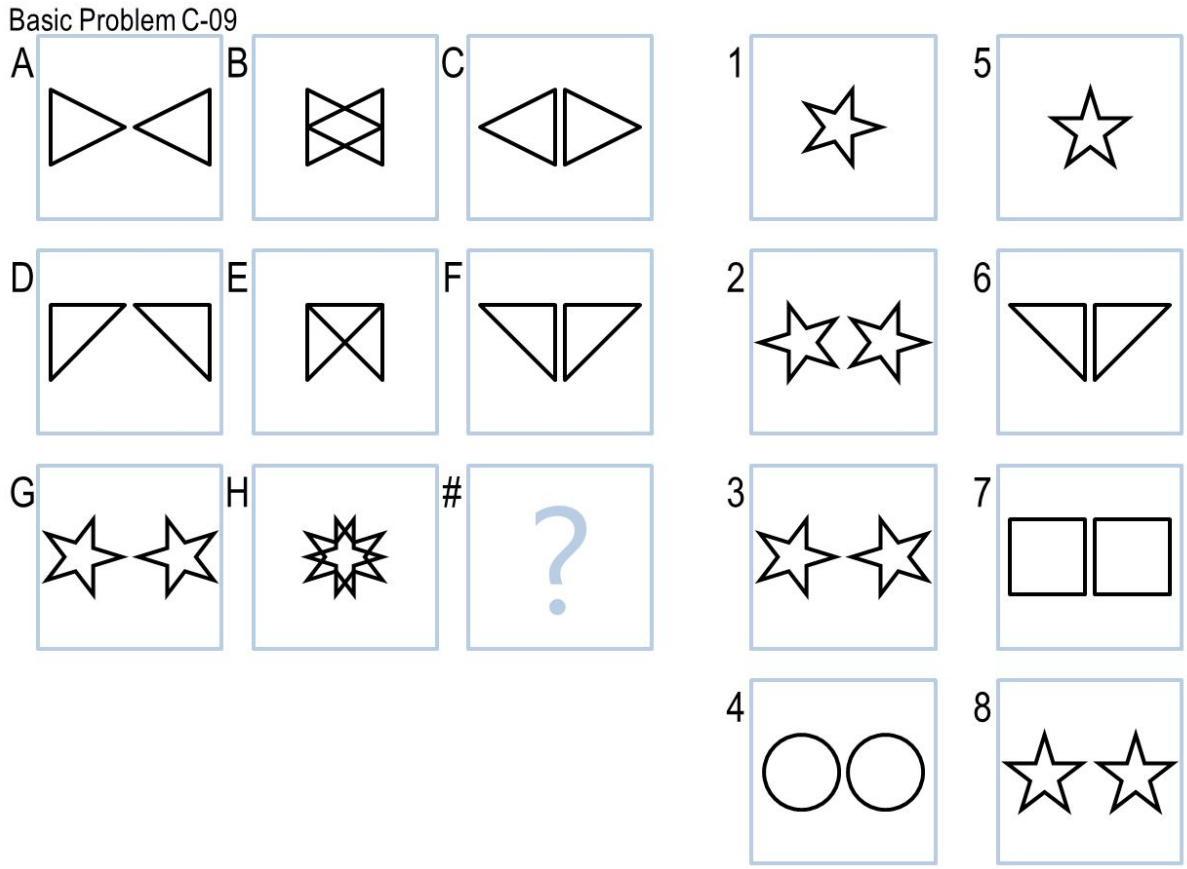
Learning Goals

The goal of Knowledge-Based Artificial Intelligence is to create human-like, human-level intelligence. If this is the goal of the field, then what better way to evaluate intelligence of an agent than by having it take the same intelligence tests that humans take?

There are numerous tests of human intelligence, but one of the most reliable and commonly-used is Raven's Progressive Matrices. Raven's Progressive Matrices, or RPM, are visual analogy problems where the test-taker is given a matrix of figures and asked to select the figure that completes the matrix. Examples of 2x2 and 3x3 RPM-style problems are shown below.

Basic Problem B-10





In these projects, you will design agents that will address RPM-inspired problems such as the ones above. The goal of this project is to authentically experience the overall goals of knowledge-based AI: to design an agent with human-like, human-level intelligence; to test that agent against a set of authentic problems; and to use that agent's performance to reflect on what we believe about human cognition. As such, you might not use every topic covered in KBAI on the projects; the topics covered give a bottom-up view of the topics and principles KBAI, while the project gives a top-down view of the goals and concepts of KBAI.

About the Test

The full Raven's Progressive Matrices test consists of 60 visual analogy problems divided into five sets: A, B, C, D, and E. Set A is comprised of 12 simple pattern-matching problems which we won't cover in these projects. Set B is comprised of 12 2x2 matrix problems, such as the first image shown above. Sets C, D, and E are each comprised of 12 3x3 matrix problems, such as the second image shown above. Problems are named with their set followed by their number, such as problem B-05 or C-11. The sets are of roughly ascending difficulty.

For copyright reasons, we cannot provide the real Raven's Progressive Matrices test to everyone. Instead, we'll be giving you sets of problems -- which we call "Basic" problems -- inspired by the real RPM to use to develop your agent. Your agent will be evaluated based on

how well it performs on these “Basic” problems, as well as a parallel set of “Test” problems that you will not see while designing your agent. These Test problems are directly analogous to the Basic problems; running against the two sets provides a check for generality and overfitting. Your agents will also run against the real RPM as well as a set of Challenge problems, but neither of these will be factored into your grade.

Overall, by Project 3, your agent will answer 192 problems. More on the specific problems that your agent will complete are in the sections that follow.

Verbal vs. Visual

Historically in the community, there have been two broad categories of approaches to RPM: verbal and visual. Verbal approaches attempt to solve RPM based on verbal representations of the problems. In these representations, a human initially describes the contents of the figures of a problem using a formal vocabulary, and an AI agent then reasons over those representations. Visual approaches, on the other hand, attempt to solve RPM based strictly on the images themselves: they take as input the raw image data and perform their analysis from there. Examples of verbal approaches include Carpenter, Just & Shell 1990 and Lovett, Forbus & Usher 2009. Examples of visual approaches include Kunda, McGreggor & Goel 2013 and McGreggor & Goel 2014.

Much research has been done examining the differences between these approaches in humans (e.g. Brouwers, Vijver & Hemert 2009). Within artificial intelligence, visual approaches are generally more robust in that an initial phase of human reasoning is not necessary. Thus, the ultimate goal for these projects will be to solve RPM-inspired problems visually. However, visual problem-solving tends to be significantly more difficult. Thus, you will start by having the option to use both verbal and visual approaches (using verbal representations we have produced for you), and by the last project you will use only visual methods.

Grading and Learning Goals

Your grade on the project is based on three criteria: your agent’s performance on the Basic problems, your agent’s performance on the Test problems, and the project reflection you submit along with your agent.

What this means is that a significant portion of your grade is determined by how well your agent performs on these problems. However, the learning goal of this project is not to understand how an agent might approach RPM. The goals are to explore how to design an agent inspired by human-like intelligence, how to test that agent against authentic problems, and how to use that agent’s performance to reflect on human cognition itself. In the process of exploring those goals, however, it is our belief that you will create an agent whose

performance improves as you move through that cycle; that agent's performance, then, will serve as a decent barometer for the exploration of those goals. The goal is not to design an agent that does well on RPM-inspired problems; however, we believe you can't design an agent that does well on RPM-inspired problems without *also* accomplishing those learning goals.

Is it possible to accomplish the learning goals without designing a successful agent? Yes, which brings us to the next section...

Take Chances, Make Mistakes

As you're working on these projects, you may encounter a dilemma that students have reported encountering in the past. On the one hand, you might have an idea for a way of designing your agent that you believe would do pretty well, but may not be all that interesting. On the other hand, you might have an idea for a way of designing your agent that may not work at all, but would be more innovative and novel. You might hesitate to try out this latter method because there is some risk involved: it might not work, and because your grade is based on the number of problems your agent gets right, your performance might suffer because you tried to do something more interesting.

If you find yourself in this dilemma, take the risk. Try out the approach you find more innovative, novel, or interesting. If the approach ends up being successful, great! If the approach does not end up being successful, however, write about it thoroughly in your project reflection. Describe the "safe" idea that you did not use, as well as the novel idea that you did use. Explore why you believed the novel idea had potential, and try and explain why you believe it was not successful: were you unable to implement it successfully, or was there an inherent problem in the method? How will this experience inform your next project? How would you improve on this method in the future?

If you take a novel approach to the project and that novel approach does not pan out, we will compensate for that in your grade. In line with the learning goals above, we *want* you to take risks and explore innovative or interesting approaches to these problems. If you take a risk and it doesn't pay off, your grade won't suffer; just make sure to explain the risk you took in detail in your project reflection, as well as why it didn't pan out. In other words, demonstrate that you accomplished the learning goals of the project even though your agent did not perform that well.

Project Difficulty and Authenticity

As you go through these projects, you might realize as students have in the past that they are not easy. The projects in the course are very difficult. However, they are not difficult simply

for the sake of being difficult: they are difficult because they address a real problem that the artificial intelligence community is facing right now. There exists ongoing research by professors, research scientists, and PhD students *right now* with the *exact* same goal as your project here. Former KBAI students are pursuing this research in ongoing Master's theses. If the problem was easy, it wouldn't be the subject of ongoing research -- it would have been solved long ago.

This authenticity is exactly why we have chosen this as the project for the class. The learning goal, ultimately, is not to understand how an agent solves Raven's Progressive Matrices. The learning goals are instead the goals of the research community itself: to design an intelligent agent inspired by human cognition, to test that agent against authentic problems, and to use that agent's performance to reflect on human intelligence. The research community considers intelligence tests to be a useful place in which to explore these issues, and so, we too use these tests to explore these issues.

At the end of this document is a sampling of recent research on artificial intelligence and Raven's Progressive Matrices to demonstrate the authenticity and openness of this problem.

Details and Deliverables

This section covers the more specific details of the projects: what you will deliver, what problems your agents will solve, and what representations will be given to you.

Project Progression

In this Fall offering of CS7637, you will complete three projects:

- **Project 1:** Problem set B.
- **Project 2:** Problem sets B and C.
- **Project 3:** Problem sets B, C, D, and E.

Each problem set consists of 48 problems: 12 Basic, 12 Test, 12 Raven's, and 12 Challenge. Only Basic and Test problems will be used in determining your grade. The Raven's problems are run for authenticity and analysis, but are not used in calculating your grade. At the conclusion of each project, you'll receive a file describing your agent's problem-by-problem performance.

On each project, you will have access to the Basic and Challenge problems while designing and testing your agent; you will not have access to the Test or Raven's problems while designing and testing your agent. Challenge and Raven's problems are not part of your grade, though note that the Challenge problems will often be used to expose your agent to extra

properties and shapes seen on the real Raven's problems that are not covered in the Basic and Test problems.

As mentioned previously, the problems themselves ascend in difficulty from set to set. Additionally, only visual representations will be given for problem sets D and E, so for project 3 you'll be required to do some visual reasoning.

The Framework

To make it easier to start the project and focus on the concepts involved (rather than the nuts and bolts of reading in problems and writing out answers), you'll be working from an agent framework in your choice of Python or Java. You can get the framework in one of two ways:

- Clone it from the master repository with 'git clone --recursive https://github.gatech.edu/Dilab/KBAI-package-java.git' (Java) or 'git clone --recursive https://github.gatech.edu/Dilab/KBAI-package-python.git' (Python). This makes it easier to 'git pull' any (rare) framework changes or fixes that must be made after the project is released.
- Download Project-Code-Java or Project-Code-Python as a zip file from this folder. This method allows you to obtain the code if you are having trouble accessing the Georgia Tech Github site.

You will place your code into the Solve method of the Agent class supplied. You can also create any additional methods, classes, and files needed to organize your code; Solve is simply the entry point into your agent.

The Problems

As mentioned previously, in project 3, your agent will run against 192 problems: 4 sets of 48 problems, with each set further broken down into 4 categories with 12 problems each. The table below gives a rundown of the 16 smaller sets, what will be provided for each, and when your agent will approach each.

Key:

- P1?, P2?, and P3?: Whether that set will be used on that project.
- Graded?: Whether your agent's performance on that set will be used in determining your grade for the project (Basic and Test are used for grading, Challenge and Raven's are just used for authenticity and curiosity).
- Provided?: Whether you'll be given a copy of those problems to use in designing your agent (you'll see Basic and Challenge problems, but Test and Raven's will remain hidden).
- Visual?: Whether your agent will have access to visual representations of the set (which it will for all problems).

- Verbal?: Whether your agent will have access to verbal representations of the set (you'll have verbal representations for sets B and C, but not for sets D and E).

Set	Type	P1?	P2?	P3?	Graded?	Provided?	Visual?	Verbal?
B	Basic	✔	✔	✔	✔	✔	✔	✔
	Test	✔	✔	✔	✔		✔	✔
	Challenge	✔	✔	✔		✔	✔	
	Raven's	✔	✔	✔			✔	
C	Basic		✔	✔	✔	✔	✔	✔
	Test		✔	✔	✔		✔	✔
	Challenge		✔	✔		✔	✔	
	Raven's		✔	✔			✔	
D	Basic			✔	✔	✔	✔	
	Test			✔	✔		✔	
	Challenge			✔		✔	✔	
	Raven's			✔			✔	
E	Basic			✔	✔	✔	✔	
	Test			✔	✔		✔	
	Challenge			✔		✔	✔	
	Raven's			✔			✔	

Thus, for the first two projects, you'll be addressing the easier two sets of problems using both their visual and verbal representations. For the final project, you'll address the final two sets of problems using their visual representations only. It might, therefore, be prudent to get an early start on the visual methods!

Within each set, the Basic, Test, and Raven's problems are constructed to be roughly analogous to one another. The Basic problem is constructed to mimic the relationships and transformations in the corresponding Raven's problem, and the Test problem is constructed

to mimic the Basic problem very, very closely. So, if you see that your agent gets Basic problem B-05 correct but Test and Raven's problems B-05 wrong, you know that might be a place where your agent is either overfitting or getting lucky. This also means you can anticipate your agent's performance on the Test problems relatively well: each Test problem uses a near-identical principle to the corresponding Basic problem. In the past, agents have averaged getting 85% as many Test problems right as Basic problems, so there's a pretty good correlation there *if* you're using a robust, general method.

Penalties for Incorrect Answers

Raven's Progressive Matrices are multiple-choice problems. So, an agent that just guesses randomly will get about 1/6th of the 2x2 and 1/8th of the 3x3 problems correct. We need a way to compensate for that random guessing. Additionally, in Knowledge-Based AI, we're interested in building agents that are metacognitive, that think about their own reasoning. Thus, we want to build agents that not only can answer these questions, but also can self-evaluate and decide whether it's actually in their best interest to answer questions.

So, in grading, a correct answer is worth 1 point. An incorrect answer is worth -0.20 points for 2x2 problems, and -0.14 points for 3x3 problems. These penalties mean that random guessing will average out to 0. A skipped problem will not incur any penalty, however. To skip a problem, an agent should just return a negative number.

Repeated Problem Sets

You'll notice that in Project 2, your agent again looks at problem set B. On Project 3, your agent again looks at problem sets B and C. This is to encourage the iterative nature of these projects: we want you to keep revising and improving your agent, not only for the new problems but also for the older ones. Our goal is to build an agent that can address all these problems, not to build three agents that can each address a subset of these problems.

However, students in the past have noted that this also is something like double jeopardy: if you do poorly on Project 1, you're likely penalized for that negative performance again on Project 2. We don't want that, of course; we want everyone to start on an even footing on each project.

So, while your agent will be run against problem set B on all three projects and problem set C on both projects 2 and 3, *only* the new problems will count toward each project's grades. Project 2's grade will be based only on problem set C. Project 3's grade will be based only on problem sets D and E.

However, we still want to preserve the incentive to improve your agent's performance on earlier problem sets. So, if your agent's performance on problem set B improves in Project 2 relative to Project 1, we'll average in that improvement: we'll take your agent's best performance on problem set B and average it with your agent's performance on problem set B in Project 1. So, improving your agent's performance on problem set B in Project 2 will retroactively improve your grade on Project 1. If your agent answered 12/24 problem set B problems right in Project 1 and 16/24 problem set B problems right in Project 2, then your grade for Project 1 would be based having gotten 14/24 problems right on problem set B.

The drawback here is that this creates no incentive to maintain your agent's performance on earlier problem sets throughout later ones. Your agent could just skip all of problem set B in Project 2 without hurting your grade. We ask that you do not do this. The goal of the project is to build an agent that can answer all 192 problems, and skipping the earlier sets goes against this goal.

Project Grades and Evaluation

Your project grades will be based on three pieces: your agent's performance on the Basic problems, your agent's performance on the Test problems, and your project reflection. Performance on the Challenge and Raven's problems won't be included in the grade, although if your agent somehow performs well on the Challenge and/or Raven's problems without performing well on the Basic and Test problems, we'll take that into consideration.

These three pieces of your project grade are not simply averaged and summed; the grading scheme may be more complicated. In the past, for example, we have applied explicit constraints to reduce the incidence of overfitting by requiring minimum levels of performance for both Basic and Test problems. Make sure to pay attention to the grading criteria of individual projects to make sure you're properly prioritizing your time.

Finally, it's important to note that the threshold for "success" on these projects should not be thought of as 90%. Even answering 50% of the problems correctly is an incredible achievement. We generally normalize project grades, so make sure to read the stats posts at the end of each project to get a feel for how you're doing relative to the rest of the class.

Relevant Resources

Goel, A. (2015). Geometry, Drawings, Visual Thinking, and Imagery: Towards a Visual Turing Test of Machine Intelligence. In *Proceedings of the 29th Association for the Advancement of Artificial Intelligence Conference Workshop on Beyond the Turing Test.* Austin, Texas.

McGreggor, K., & Goel, A. (2014). Confident Reasoning on Raven's Progressive Matrices Tests. In *Proceedings of the 28th Association for the Advancement of Artificial Intelligence Conference.* Québec City, Québec.

Kunda, M. (2013). Visual problem solving in autism, psychometrics, and AI: the case of the Raven's Progressive Matrices intelligence test. Doctoral dissertation.

Emruli, B., Gayler, R. W., & Sandin, F. (2013). Analogical mapping and inference with binary spatter codes and sparse distributed memory. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE.

Little, D., Lewandowsky, S., & Griffiths, T. (2012). A Bayesian model of rule induction in Raven's progressive matrices. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society*. Sapporo, Japan.

Kunda, M., McGreggor, K., & Goel, A. K. (2012). Reasoning on the Raven's advanced progressive matrices test with iconic visual representations. In *34th Annual Conference of the Cognitive Science Society*. Sapporo, Japan.

Lovett, A., & Forbus, K. (2012). Modeling multiple strategies for solving geometric analogy problems. In *34th Annual Conference of the Cognitive Science Society*. Sapporo, Japan.

Schwering, A., Gust, H., Kühnberger, K. U., & Krumnack, U. (2009). Solving geometric proportional analogies with the analogy model HDTP. In *31st Annual Conference of the Cognitive Science Society*. Amsterdam, Netherlands.

Joyner, D., Bedwell, D., Graham, C., Lemmon, W., Martinez, O., & Goel, A. (2015). Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests. In *Proceedings of the Sixth International Conference on Computational Creativity*. Provo, Utah.

...and many more!