Andrew (Joo Hun) Nam
First-Year Project Proposal
Advisor: Jay McClelland
Second Reader: Tobias Gerstenberg

# A Study in Extensible Algorithmic Thinking

## Introduction

A fundamental aspect of intelligence is the ability to apply a procedure across multiple inputs that the reasoner hasn't seen before, transferring what was learned in one domain to another. An extreme case of this is when the procedure may be applied near invariantly to the input, an approach often taken in mathematics and algorithms where functions transform inputs by following a prescribed process. A necessary feature to this reasoning is attention, isolating the relevant features of the total input/task/problem/feature space, searching for applicable transformations that would yield useful results with the relevant subset, and applying the procedure with sure accuracy. This cognitive feature of extensibility allows for reasoning agents to abstract the procedure away from the specific exemplars by which it was trained and extrapolate to parts of the domain it has never encountered before with no error, the core of deductive reasoning. Despite its clear significance in intelligence, modern models of cognition struggle to capture extensibility. Symbolic processing models only trivially demonstrate extensibility by having procedures and corresponding variables built into the model. Meanwhile, connectionist models are often severely limited in their capacity to perform extensible reasoning.

In this study, I will explore the concept of extensible reasoning through the game of Sudoku, a logic puzzle with the following rules:

1. Each row, column, and 3x3 box in the 9x9 grid called 'the Sudoku board' must contain exactly one of each digit from 1 to 9.
2. Each puzzle (a valid Sudoku board with missing entries) must yield exactly one solution.

Application of the first rule yields a total of 6.671e21 valid Sudoku grids (Mathematics of Sudoku I - Felgenhauer, 2006), yet it also allows for interesting isomorphisms by exploiting equivalencies (e.g. renumbering 1 -> 2 -> … -> 9 -> 1) and symmetries (rotate the grid 90 degrees), resulting in only 5.473e9 canonically distinct grids (Mathematics of Sudoku II - Felgenhauer, 2006). This highlights the incredible extensible nature of Sudoku where identifying structural regularities can reduce the number of possible grids by 12 orders of magnitude.

Moreover, while the rules are simple enough to be captured in two short sentences, Sudoku produces a complex set of techniques and procedures for players to use during solving. The most trivial is when a house (defined as a row, column, or 3x3 box) contains all but one of the nine digits, in which the blank cell must contain the last digit. Even in the simplest technique, Full House, extensibility is necessary to make this tractable. Consider an agent that learns through memorization of key-value pairs. To learn the simplest instances of the simplest Sudoku technique, this learner must observe a training instance for each of the 9 rows, 9 columns, 9 boxes, and 9 digits with each of the possible spatial permutations, a combinatorial complexity of (9+9+9)*9*9! = 88,179,840.

In this study, I plan to understand the nature of extensibility of human reasoning, specifically the mental facility that identifies interchangeable elements of a set that can be treated isomorphically in algorithmic procedures through the medium of Sudoku puzzles and build towards a connectionist model that is able to solve Sudoku puzzles by exploiting extensible reasoning. Though none would question whether humans can reason with extensibility, no empirical research exists measuring the degree of extensibility. To establish empirics and build intuition, I will conduct a series of experiments by teaching people unfamiliar with Sudoku a few techniques and test their capacity to carry them out across varying conditions. Moreover, while Sudoku-solver models already exist (Recurrent Relational Networks - Palm, 2017) and still more generic models can be fitted to the task, there are none yet that do so extensibly without having the property built into the program. Through this research, I hope to contribute to both our understanding of human cognition as well as artificial intelligence.

**Experimental Design**

Participants (taken from Amazon Mechanical Turk) will be assigned tasks to learn 3 techniques in Sudoku with 4 levels of extensibility across the tasks. First, participants will be taught the basic rules of Sudoku and taught the trivially simple Full House technique described in the introduction above where they are shown 8 digits in a house and the last digit being filled in at the blank cell. To demonstrate the extensible elements of Sudoku, participants will also be displayed the application of the technique across each of the 3 house types (row, column, box) and in varying locations with different digits in place.

| *Set* | Digit | Cell | House Index | House Type |
|---|---|---|---|---|
| *A* | 1 | 2 | 3 | 4 |
| *B* | 2 | 3 | 4 | 1 |
| *C* | 3 | 4 | 1 | 2 |
| *D* | 4 | 1 | 2 | 3 |

After the introduction phase, participants enter the 3 learning phases where in each phase, they are given a tutorial of a technique and 4 puzzles across varying invariances that utilize the technique. Following each tutorial, the participant will be quizzed on the last technique learned through 4 puzzles, each building upon a different invariant: digit, cell location, house index, and house type. Each puzzle will add an additional invariant so that the first puzzle only changes the digit from the tutorial, the second puzzle changes both the digit and the cell location, and so on. To counterbalance the ordering effect, the order in which the digit, cell, and house index conditions are presented will be divided across 6 permutations. During this quiz phase, participants will be allowed to reference the original tutorial if desired. Participants will be assigned one of 4 instances of each quiz. If the participant correctly solves the puzzle, the next puzzle is given until all 4 are completed, upon which the program shifts to the tutorial phase of the next technique until all 3 techniques are completed. If the participant is unable to solve the puzzle (too many incorrect attempts or time-out), the explanation grid is displayed similar to those in the tutorials to reinforce understanding.

Once the three learning phases are completed (a total of 12 puzzles), the participant proceeds to the test phase where new puzzles are displayed, again one for each technique and variant (again, cumulative), but the puzzle order is fully shuffled so that the techniques and variants are in random sequence. During the test phase, participants may not return to the tutorial and are given no explanatory feedback when the participant fails to solve.

Throughout the program, each of the participants' interaction with the application will be logged such as buttons clicked and entries written/erased along with their timestamps.

**Methods of Analysis**

This experiment tests for how easily participants can solve variants of the technique that they have learned where each variant introduces a different dimension of fungible items. Without any level of extensibility, learning by example should not generalize to a technique and participants would not be able to solve any other puzzles other than the specific ones seen so far. With ideal extensibility where every interchangeable element is immediately recognized and the technique uniformly applicable across each instance of the puzzle, participants would demonstrate no discernable differences in accuracy and solving times. Most likely, human extensible reasoning falls somewhere in the middle with countless other factors in effect such as the difficulty of the instance of the puzzle or number of distractors. Therefore, the models constructed to capture extensible reasoning will attempt to control for the various confounding factors specific to each instance of the puzzle while isolating out the variable of interest, degrees of fungibility.

If extensible reasoning does fall somewhere between nonexistence and the ideal, then it is possible that different dimensions of variation also vary in the difficulty of extending. Shuffling only the digits involved from another puzzle, for instance, is likely more easily recognized as a variant of the other and can be described as more easily extensible by a human reasoner. On the other hand, changing the house-type from Box to Row is likely more difficult and the translation not immediately obvious. Two hypotheses can be made based on these assumptions. First, at least some proportion of participants will be able to solve both types of puzzles. Second, most participants will find the change of house-type more difficult than just change of digits, resulting in lower accuracy and longer solving times.

A distant goal using this data is to go beyond describing the phenomenon with simple regression models, but to model this form of reasoning through a computational model. It is hoped that such a model will mimic the strengths and weaknesses, biases and errors of human agents. Above all, however, the goal of the model will be to solve the same types of puzzles the human participants are presented with the same training inputs the human participants are presented, just a single example of the technique in use.

**Challenges and Obstacles**

Prior to data collection, it is impossible to estimate how well participants will perform on these tasks, especially when the participant pool is Amazon MTurk Workers. Depending on the participants' educational background, experience with logic puzzles, and specifically experience in Sudoku, the data may vary wildly across participants. Moreover, it is possible that the data will reveal a ceiling and/or floor effect should the puzzles prove too be too difficult or too easy. Participants may also exhibit a learning effect, picking up concepts and extensible regularities at a pace faster than the program can measure.

One of the most difficult features to effectively operationalize and capture through data is the rate at which participants pick up on the extensibility of the concepts compared to simply learning Sudoku and its techniques. Certainly, both will contribute to the accuracy and duration of the participants.

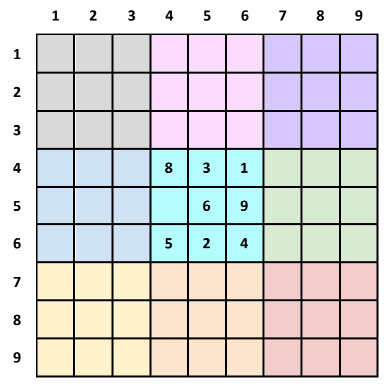
Furthermore, the puzzles are difficult to design such that they isolate the features of interest while also balancing the difficulty and variety. A puzzle with 4 hints (e.g. a hidden single task 6 cells are invalidated by 2 cells through same-Box constraints and 2 cells invalidated by 2 cells through same-Column constraints) may be equally valid as one with 8 (e.g. a hidden single task where 8 cells are invalidated by 8 cells through same-Column constraints). Other factors may contribute to time such as the spatial distance between hint cells, the number of items to track (e.g. naked-single requires finding the single missing digit by scanning the 8 existing ones), the number of distractor cells (all example puzzles in this proposal have 0), and program-related issues such as UI, latency, color-contrast, etc.

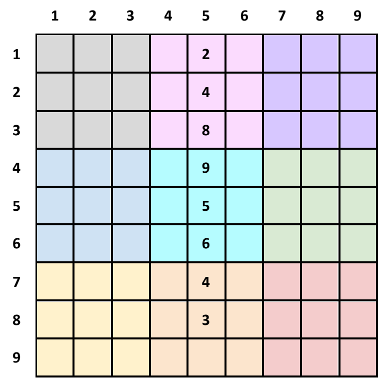
Some of the issues may be addressed by raising (or lowering) the difficulty of the puzzles, shuffling the order of invariants introduced, and even substituting the techniques with more difficult versions such as *hidden pairs* instead of *hidden singles*.

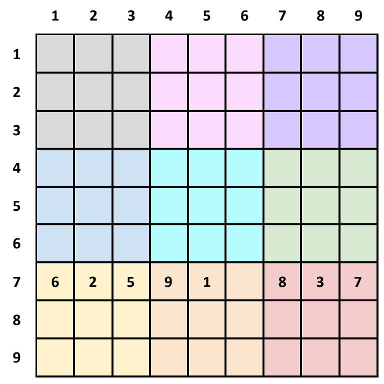
## Examples of Materials

Full House Tutorial

- Given 8 cells in a house (row, column, or box) with known correct digits, the remaining cell in the house must be the missing digit
- The left grid contains a blank cell in R5C4 (Row 5, Column 4) and every digit except a 7 in Box Cyan, so R5C4 must be a 7.
- The center grid contains a blank cell in R9C4 and every digit except a 1 in Column 5, so R9C4 must be a 1.
- The right grid contains a blank cell in R7C6 and every digit except a 4 in Row 7, so R7C6 must be a 4.



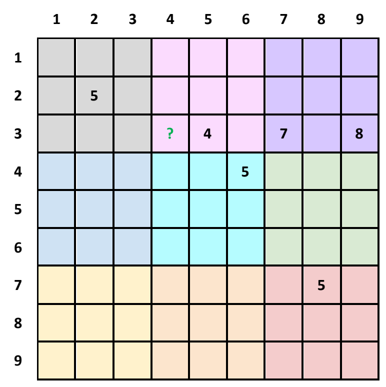


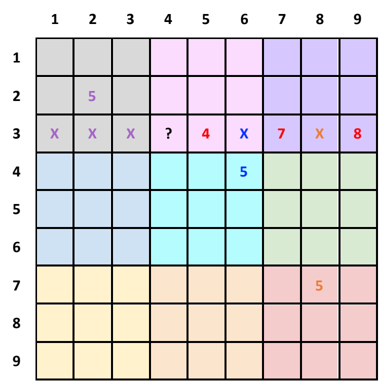


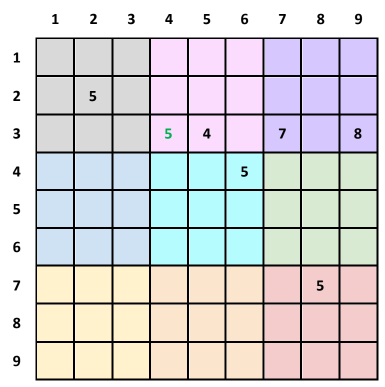
The first technique is the *hidden single* technique, a single-digit/cell variant of the *hidden set* class of techniques. Participants are shown the following tutorial sequence.

Hidden Single Tutorial

- If only one cell in a house can be a digit, that cell must be the digit
- In the left grid
  - We use the *hidden single* technique to resolve the **?** in (R3C4)
- In the center grid
  - The cells in **Row 3** containing **4** (R3C5), **7** (R3C7), and **8** (R3C6) cannot be **5** because they already have digits
  - The **X**s in (R3C1, R3C2, R3C3) in **Box Gray** cannot be **5** because **5** (R2C2) already exists in **Box Gray**
  - The **X** in (R3C7) cannot be **5** because **5** (R4C7) already exists in **Column 7**
  - The **X** in (R3C8) cannot be **5** because **5** (R7C8) already exists in **Column 8**
- In the right grid
  - Therefore, (R3C4) is the only cell in **Row 3** that can be a **5**
  - Use the *hidden single* technique to insert the correct digit in (R3C4).





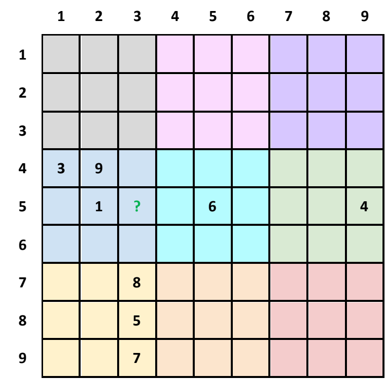


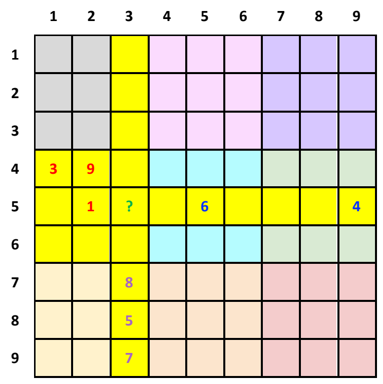
The examples of materials for the different invariances are provided at the end of this document.

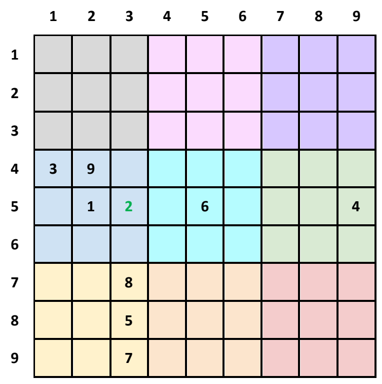
The second technique is the *naked single* technique, a single-digit/cell variant of the *naked set* class of techniques. Participants are shown the following tutorial sequence.

Naked Single Tutorial

- If a cell has only one possible candidate digit, it must be that digit.
- In the left grid
  - We use the *naked single* technique to resolve the ? in (R3C4)
- In the center grid
  - ? (R3C4) shares the same **Box** as 3 (R4C1) , 9 (R4C2), and 1 (R5C2) so it cannot be **1**, **3**, or **9**
  - ? (R3C4) shares the same **Column** as 8 (R7C3) , 5 (R8C3), and 7 (R9C3) so it cannot be **5**, **7**, or **8**
  - ? (R3C4) shares the same **Row** as 4 (R5C5) and 6 (R5C9) so it cannot be **1**, **3**, or **9**
- In the right grid
  - Therefore, the only remaining digit possible for (R5C3) is **2**.
  - Use the *naked single* technique to insert the correct digit in (R5C3).



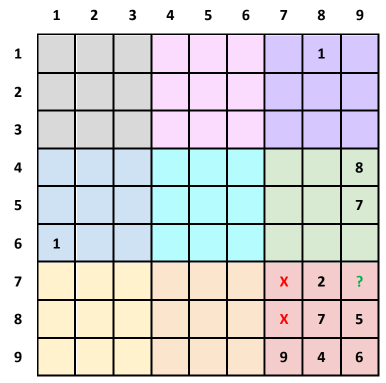


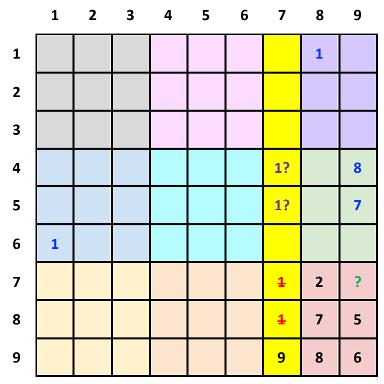


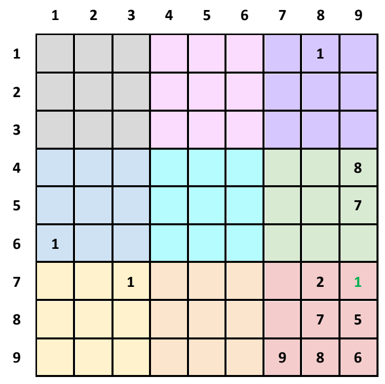
The third technique is the *locked candidates* technique. Participants are shown the following tutorial sequence.

Locked Candidates Tutorial

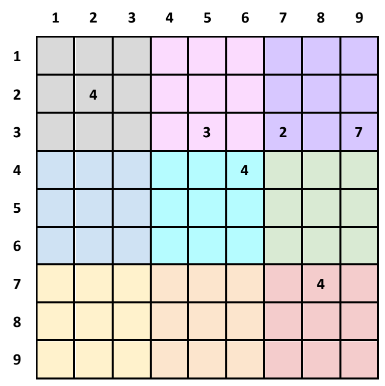
- If the intersecting cells of a Box and Row/Column are the only candidates for a digit for either of the intersecting houses, no other cells in the other house can be that digit.
- In the left grid
  - We use the *locked candidates* technique to resolve the **?** in (R7C9)
- In the center grid
  - The **1?**s (R4C7, R5C7) are the only cells in Box Green that could be **1** since **8** (R4C9) and **7** (R5C9) already have digits and **1**s (R1C8, R6C1) already exist in **Row 6** and **Column 8.**
  - Since **1** must be in either (R4C7) or (R5C7) due to constraints in Box Green, no other cell in **Column 7** could be **1**. Therefore, ~~**1**~~s (R7C7, R8C8) cannot be **1**.
  - The only remaining cell that can be a **1** in Box Red is **?** (R7C9) so it must be a **1**.
- In the right grid
  - Therefore, The only remaining cell that can be a **1** in Box Red is **?** (R7C9)
  - Use the *locked candidates* technique to insert the correct digit in (R7C9).

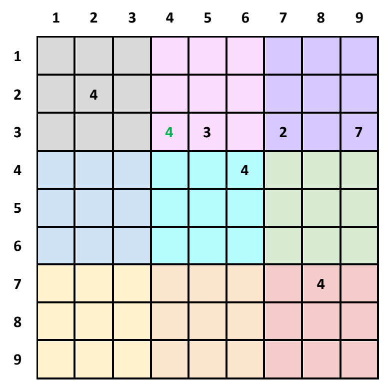




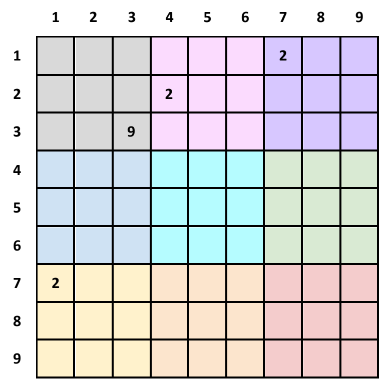


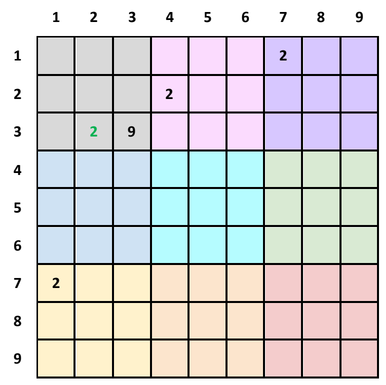
Example of an instance of digit variant for hidden single.



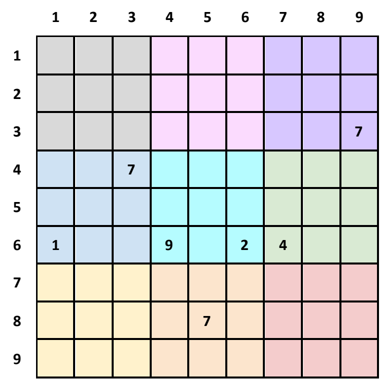


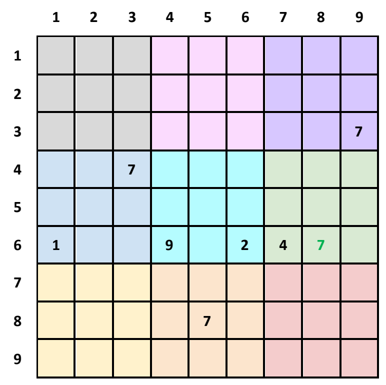
Example of an instance of cell (and digit) variant for hidden single



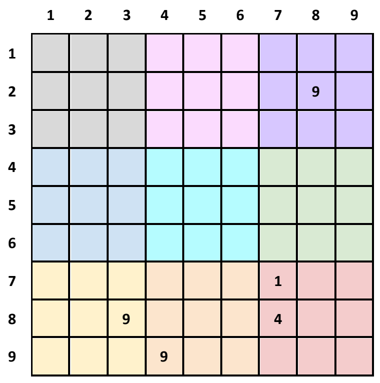


Example of an instance of house index (and cell and digit) variant for hidden single

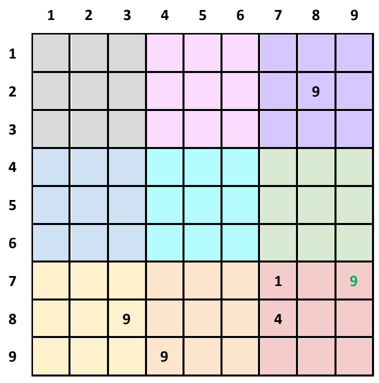




Example of an instance of house type (and cell, digit, and house index) variant for hidden single



|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  | 9 |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  | 1 |  |  |
| 8 |  |  | 9 |  |  |  | 4 |  |  |
| 9 |  |  |  | 9 |  |  |  |  |  |



|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  | 9 |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  | 1 |  | 9 |
| 8 |  |  | 9 |  |  |  | 4 |  |  |
| 9 |  |  |  | 9 |  |  |  |  |  |