The Effect of Intermediate Concepts in Hierarchical Relational Learning

by

Hout (Peter) Nga

B.S. Computer Science, Massachusetts Institute of Technology, 2010

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

© Massachusetts Institute of Technology 2011. All rights reserved.
The Effect of Intermediate Concepts in Hierarchical Relational Learning

by

Hout (Peter) Nga

Submitted to the Department of Electrical Engineering and Computer Science on August 22, 2011, in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

Abstract

The DARPA Bootstrapped Learning project uses relational learners to ladder concepts together to teach a final concept, essentially narrowing the search space at each step. However, there are many ways to structure background knowledge to teach a concept and it is uncertain how different ways of structuring knowledge affects the accuracy and performance of learning. In this paper, we examine the effect of having intermediate concepts when learning high level concepts. We used Quinlan’s First Order Inductive Learner to learn target selection for a real-time strategy game and did cross-validation tests with varying levels of intermediate concept support. The results show that including intermediate concepts does not always improve performance and accuracy.

Thesis Supervisors:
Leslie Kaelbling, Professor of Computer Science and Engineering
Howard Reubenstein, BAE Systems Principal Investigator

This material is based upon work supported by the United States Air Force Research Laboratory (AFRL) under Contract No. FA8650-07-C-7722. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. DISTRIBUTION STATEMENT A: Approved for Public Release, Distribution Unlimited
Acknowledgments

First off, I want to thank BAE Systems and the VI-A Internship Program of MIT. I would not have had this opportunity without these organizations.

This paper would not be possible had it not been for the continual support from Howard Reubenstein. Not only did he work towards finding me the project, he gave me a great deal of advice and found me a wonderful group of people to aid me in my project. Thanks also go to Terran Lane, Candace Sidner, and Brian Roberts for their support.

I would like to thank my MIT supervisor, Leslie Pack Kaelbling, for helping me until the very end of the thesis. Her patience and guidance has helped me pull through and finish.

Additionally, I must thank Tom Lee for his inspirational words when I was at a point of stagnation. He rekindled my fire to work, which allowed me to push through difficult times. An affirmation is also in order for Steven Huynh. He made sure that I maintained my pace when I felt overwhelmed by life.

Lastly, I want to thank my family, especially my brother. They have always been there to help me and give me words of encouragement when I am in need.
## Contents

ABSTRACT ................................................................. 2
Acknowledgments ...................................................... 3
List of Tables ............................................................ 5
List of Figures ............................................................. 6
1 Introduction ............................................................ 8
   1.1 Research Goals ................................................ 8
   1.2 Organization .................................................. 10
2 Bootstrapped Learning .............................................. 11
   2.1 Overview ....................................................... 11
   2.2 Terminology .................................................. 11
   2.3 System ........................................................ 12
3 First Order Inductive Learner ..................................... 16
   3.1 Overview ....................................................... 16
   3.2 Terminology .................................................. 16
   3.3 System ........................................................ 17
4 Learning Domain ...................................................... 20
   4.1 Real-Time Strategy Games ................................... 20
   4.2 Wargus ........................................................ 20
   4.3 Concept Structure ............................................ 22
Research Methodology .................................................. 25
  5.1 Data Collection ...................................................... 25
  5.2 Testing .............................................................. 25
Results and Discussion ................................................. 27
  6.1 Accuracy .......................................................... 27
  6.2 Performance ....................................................... 28
  6.3 Analysis .......................................................... 29
  6.4 Implications on Bootstrapped Learning System ............... 31
Conclusion ............................................................... 32
Future Directions ......................................................... 33
  8.1 Structure in Hierarchical Learning ....................... 33
  8.2 Partially Correct Knowledge ............................... 33
  8.3 Self-Organizing Hierarchical Relational Learner ....... 33
References ............................................................... 34
Appendix A: FOIL-6 Relation Definitions ....................... 36
Appendix B: Horn Clauses Produced by FOIL-6 .................. 38
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Original Concept Structure Accuracy</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>Intermediate Concepts as Background Knowledge Accuracy</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>Intermediate Concepts Removed Accuracy</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Original Concept Structure Performance Speed</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Intermediate Concepts as Background Knowledge Performance Speed</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>Intermediate Concepts Removed Performance Speed</td>
<td>29</td>
</tr>
</tbody>
</table>
List of Figures

1. Diagram of the parts of the Bootstrapped Learning system and how they connect with one another. .................................................. 13

2. Screenshot of a Wargus scenario .................................................. 21

3. Concept Structure ..................................................................... 24

4. Code for getting time in FOIL-6 .................................................. 26

5. Accuracy Comparison Across the Test Cases ............................... 28

6. Performance Speed Comparison Across the Test Cases ............... 30
1 Introduction

There is a lot of promise in the fields of relational learning and the use of intermediate concepts in hierarchical learning. Relational learning has shown application potential in several different areas [4] and the inclusion of intermediate concepts has also demonstrated gains in accuracy and performance [2],[11]. However, the two techniques have not been previously combined. While it has been shown that adding more information via background knowledge in relational learning can decrease the hypothesis space of the search and in turn increase the accuracy of learned rules [7], it is unclear how intermediate concepts will behave in the relational learning framework.

The addition of intermediate concepts can have a negative effect on the learning process. Intermediate concepts can propagate error along the learning hierarchy if it incorrectly learns what it needed to know from the background concepts. Intermediate concepts can also slow down performance if it is irrelevant to the higher level concepts [3].

However, the hypothesis is that the extra knowledge inherent in the intermediate concepts will reduce the error that propagates from learning a sequence of concepts. Since the intermediate concepts have additional information about the structure of the learning hierarchy and they in turn should narrow the search space, the learner is expected to perform better with the aid of the intermediate concepts than when learning without them. There are some choices of intermediate concepts that are advantageous and others that are harmful, but it is unclear how best to choose these concepts.

1.1 Research Goals

The primary focus of this work was to determine the accuracy and performance effects of the inclusion of intermediate concepts in relational learning. Achieving this goal would also give insight to the effectiveness of the Bootstrapped Learning system's structured curriculum design. Several tasks were required to accomplish this goal.
1.1.1 Understand the Bootstrapped Learning System Design

The Bootstrapped Learning system was designed using many individual learning modules developed by several teams of developers. While very modular, each part was supposed to be taken as a black-box rather than have its implementation examined. As a result, the availability and scope of documentation was inconsistent. Documentation was done on a high level for the most part. Therefore, I had to review the code base to have a deeper appreciation and understanding of the system design.

1.1.2 Find the Learning Domain and Comparison Learner

One of the goals of this research was to validate the Bootstrapped Learning system design. In accordance with this goal, it was necessary to find a learner that was similar to that of the Bootstrapped Learning system and evaluate it in a learning domain. The learning domain had to be one that had a simulator that was publicly available or quick enough to implement.

1.1.3 Develop and Refine the Scope of the Domain

The scope of the domain dictates how difficult the learning problem will be. The scope of the domain should have depth and breadth large enough to showcase the effects of the intermediate concepts without having the runtime exceed twenty four hours. The concepts in this scope of the domain must also build upon one another to have the hierarchical structure desired.

1.1.4 Implement Method to Extract Data and Test

The learner needed to be supplied data before any learning could be done. This data was extracted from the learning domain and formatted so the learner could understand it. After obtaining the data, the learner was run against the data points to collect accuracy and performance metrics.
1.1.5 Analyze Implications of Results on Bootstrapped Learning System

The results of these tests were then used to analyze the possible advantages or disadvantages intermediate concepts have on the Bootstrapped Learning system. This analysis is accompanied by suggestions of areas to focus development.

1.2 Organization

Throughout this paper, I propose and analyze the effect of intermediate concepts in relational learning and its implication on the Bootstrapped Learning system. Section 2 introduces the Bootstrapped Learning system as it currently stands. Section 3 discusses the comparison learner, the First Order Inductive Learner, chosen to demonstrate the effects of intermediate concepts. Section 4 examines the testing domain and the concept structure used in the tests. Section 5 describes the methods used for testing. Section 6 details the results and the analysis of the results.
2 Bootstrapped Learning

This section gives some information on the Bootstrapped Learning project necessary to understand the paradigm that is being analyzed. The section gives a brief overview, followed by some terminology and a description of the system.

2.1 Overview

One of the many goals of the DARPA Bootstrapped Learning project is to be able to learn from a small data set but still remain accurate and general. Its approach is to provide a set of instructions from a teacher to guide the learner towards the target concept. These instructions can be broken down into modules (rungs) that can be built on one another. Essentially, the instructions allow the learner to obtain knowledge in small chunks and build additional knowledge off of previously learned knowledge. The learning algorithms used in the system are not new, but the composition of these algorithms is novel [6].

2.2 Terminology

Before delving into the details, it would be useful to clarify some of the terms used to describe the bootstrapped learning system.

- Agent - a module in the system that manipulates data
- Student - the agent that performs all of the learning tasks
- Teacher - the agent that organizes the data to be learned and passes the data to the student
- World - the agent that maintains the state of the simulator as the student performs tasks
- Simulator - the program that provides a simulation of a domain
• Framework - the part of the system that first receives messages and determines which agent to pass the messages to next

• Percept - an observation in the world

• Concept - the knowledge to be taught

• Learning Lesson - the instructions given to teach a concept

• Testing Lesson - the instructions given to test whether a concept has been properly learned

• Curriculum - a sequence of lessons to teach a higher level concept

• Background Knowledge - declarative prior knowledge

2.3 System

The Bootstrapped Learning framework provides a context in which:

• A teacher may teach and evaluate a student

• Both teacher and student can interact with a simulated world

• The simulated world may provide percepts to both the teacher and student whenever changes occur in the simulation

All interactions between the world, teacher, and student are preserved on a timeline that records student and teacher utterances and imperatives along with percept postings from the world.

The system is composed of three agents: the student, the teacher, and the world. These agents communicate with one another through this framework using a messaging language, which in the current implementation is a custom language called Interlingua [5]. The teacher takes in a curriculum file and sends out appropriate lessons. The framework passes these lessons to the student, which then tries to perform the set of instructions. It does this by sending messages to the world, which then performs the actions in the simulator, maintaining the state of the simulator for the bootstrapped system. This can be better seen in Figure 1.
Figure 1: Diagram of the parts of the Bootstrapped Learning system and how they connect with one another.

### 2.3.1 Inputs and Outputs

One of the key distinguishing characteristics of this system compared to other learning systems is its ability to learn from instruction. Inputs to the system are taken in the form of the curriculum passed to the teacher. For the purposes of this research, an electronic teacher was used but an interactive human teacher module could also be used. The curriculum contains the set of instructions that dictate the order of learning and testing, the number of times to repeat each lesson, and grading criteria.

When the student is done with the learning lessons, the testing lessons evaluate the student's ability to do tasks that should have been learned in the learning lesson such as concept classification or procedure execution. The output of the student is a response to the teacher indicating what the student believes to be the correct answer to the test or the appropriate actions to perform a procedure. If the student is confused and cannot proceed to respond, it defaults to a response of "I don't know".

### 2.3.2 Representation of Knowledge

The system uses a custom messaging language called Interlingua to describe its knowledge. Interlingua provides several primitives that facilitate interactions with the world. It allows for logic and conditional statements, definition of new procedures, and communication with the other agents.
in the system. Communication with the other agents can be in the form of an Utterance or an Imperative, which are statements and commands, respectively.

Lesson segments are the basic building block of knowledge in the system. They are built from a series of Interlingua messages. They take as parameters the model of the world, a series of messages, an optional initial state, and an optional message generator. One or more lesson segments form a lesson and several lessons form a curriculum.

Natural instruction methods (NIMs), specify how a lesson should be taught. As their name implies, they represent methods of instruction that are easy to understand or natural to a human instructor. There are three key types of NIMs: byExample, byTelling, and byFeedback. Each method of instruction is further broken down into the type of knowledge it is trying to convey, which can either be a condition or a procedure.

Knowledge is also represented as a Concept class. Lessons can refer to concepts as targets to learn or also as prerequisites to the lesson being taught. If a concept is labeled as a prerequisite, the system must have that knowledge in order to proceed with the lesson.

2.3.3 Algorithm

The student has three main components: preprocessor, learner (consisting of multiple learning modules), and translator.

The preprocessor decides which messages from the teacher to pay attention to, removes unresolved symbols and gestures, and provides various utilities for the learning component. When the student is being taught a concept, the student waits until the teacher is done presenting information before doing anything. When the teacher is done, it extracts any messages labeled as relevant and then prunes unresolved symbols and gestures.

The learning component presents the problem in first-order logic, calls a suite of learners to solve the problem, and re-poses the problem if the learned hypothesis is deemed incorrect or too general. The component gets the messages from the preprocessor and translates the messages into first-order
logic. It then applies a least-general-generalizations procedure on the examples given to it to come up with a clause that represents its hypothesis of what the target procedure is. If the hypothesis is too general, each example is made more specific by adding in all first-level properties of each argument. If the hypothesis is still too general, all literals that are related to the original arguments are added in to the hypothesis space.

The translator takes the result of the learning component and translates it back into Interlingua. Before the translator does this, however, it needs to check if all of the bindings can be made. If any cannot and cannot be resolved, the translator returns an error.
3 First Order Inductive Learner

This section gives some information on the comparison learner chosen, the First Order Inductive Learner (FOIL), and a brief explanation of why the learner was chosen. The section then proceeds to outline some of the terminology and give a description of how the FOIL system works.

3.1 Overview

For this test setup, we needed to choose a comparison learner that bears enough resemblance to the Bootstrapped Learning system. As a reminder to the reader, the Bootstrapped Learning system learns through relationships among concepts. The particular relational learner chosen is the latest version of Quinlan’s First Order Inductive Learner (FOIL), FOIL-6 [1]. This was chosen because of it was readily available, it has great significance in relational learning, and it was well-tested and documented in the academic literature.

FOIL belongs in a class of learning methods developed in the 1990s known as Inductive Learning Procedures [4]. The general idea behind this family of algorithms is to learn rules one at a time and remove the data it covers, repeating until a condition is met. The key differences between the individual ILP algorithms are how they learn each rule and how the algorithms evaluate the performance of these rules. FOIL is a top-down system that uses a separate and conquer strategy. It applies a hill-climbing search guided by an information-based evaluation function. One of the primary benefits of these kinds of learning methods is that they are able to learn with few examples.

3.2 Terminology

For completeness, some predicate logic terms are presented to help describe FOIL.

- Literal - a basic formula or its negation
- Ground literal - a literal that does not contain any variables
• Clause - a collection of literals combined by logical ORs

• Horn Clause - a clause containing at most one positive literal

• Antecedents - statement or condition in a conditional statement

• Consequent - rule that logically follows if the antecedent is true

3.3 System

FOIL is an implementation of the empirical ILP algorithm written by Ross Quinlan. Given examples for relations, it will generate a specification for each relation and produce a Horn clause. It attempts to cover all of the positive examples without covering any of the negative ones.

3.3.1 Inputs and Outputs

There are two key inputs to FOIL-6: the specification of types and the extensional definitions of relations. Test cases can also be provided but they are optional.

The types specified can be declared to be ordered, unordered, or continuous. The size of the types dictate the space that needs to be searched through when figuring out the Horn clauses.

The relations in FOIL-6 must be defined extensionally. It takes in a set of tuples that positively describe the relation and optionally, a set of tuples that negatively describe the relation. If there is no set of negative tuples, all tuples of the corresponding types not declared in the positive set of tuples will be assumed to be negative. Additionally, relations can be declared to be either a target relation or background relation.

The output of the program is a Horn clause for every target relation. If test cases are provided, the program also indicates which test cases are incorrectly classified by the Horn clause.
3.3.2 Representation of Knowledge

Knowledge in FOIL-6 is represented in predicate logic. The inputs are represented as ground literals and the output is represented as a Horn clause that can predict when an argument will satisfy a designated relation. While learning, FOIL-6 represents its hypothesis as a clause and updates the clause at each iteration.

3.3.3 Algorithm

The basic FOIL algorithm is as follows [8, 9]:

- Initialize clause to the target relation and create a training set that contains all tuples of the target relation. The antecedent is the target relation.
- While the training set contains negative tuples, find the literal that can be added to the clause that removes the most negative tuples but not any positive tuples and add it to the consequent of the clause.
- Update the training set to reflect the tuples that still satisfy the clause and repeat
- Remove any unnecessary literals and output

When only positive examples are covered by the rule, the process of adding in more specialization literals of the rule ends. All positive examples covered by the rule are then removed from the training set. If there are still positive examples left in the training set, a new search for another rule begins with these positive examples.

The FOIL algorithm is inherently a greedy search. Its choice in literals is dependent on how many example tuples it covers. So when choosing literals, it needs to find one that will maximize its gains. To determine the gain of each literal, we must first describe how much information a literal provides. Given the number of positive examples a literal covers, the information it provides is described by
\[ I(T) = - \log_2 \frac{\text{positive}(T)}{\text{size}(T)} \]

Letting \( T \) be the training set, \( T' \) be the training set after the new literal is added, and \( s \) be the number of tuples that are in the intersection of the two training sets; the information gained from adding in a literal is given by

\[ \text{literalgain} = s \times (I(T) - I(T')) \]

Each step of the iterative process examines the gains of adding literals and disregards exploring literals that do not have a high gain.

The FOIL algorithm is strict in its classification. The rules that it generates will only be those that satisfy the positive examples but none of the negative examples. In FOIL-6, it is possible to relax these constraints.
4 Learning Domain

This section describes the rationale behind choosing the learning domain, outlines the chosen learning domain itself, and details the concept structure built for the domain.

4.1 Real-Time Strategy Games

The learning domain being used for this project is a real-time strategy game called Wargus. The real-time strategy domain has many interesting concepts relevant to the artificial intelligence community. These include but are not limited to: resource management, decision making under uncertainty, spatial and temporal reasoning, group coordination, and opponent modeling. While there has been a large amount of work towards each of these topics, they continue to be focal points of research because of the scope of the problem they are trying to solve. In addition, game environments provide easily accessible AI testing environments because they generate large amounts of data.

4.2 Wargus

The particular real-time strategy game used for this project is Wargus. Wargus is an open source Warcraft II mod built on top of the Stratagus engine [10]. There is also an open source client created by Alan Fern of Oregon State University that is able to extract raw data from game play. Using this client, data will be obtained to pass into FOIL-6 for learning. The attributes being extracted are: unique unit id, current location, maximum hit points of unit, current hit points of unit, and current target of unit.

The unit ids are unique integer identifiers for each unit that never repeat in a single episode, or round, of the game. The current location is a pair of the unit’s x and y coordinates on the grid where the origin is defined to be the top left corner of the grid. The hit points of the unit is a number that refers to how much damage the unit can withstand before dying. The target is an enemy unit’s unit id that the current unit has decided to attack.
Figure 2: Screenshot of a Wargus scenario
Episodes of the game start out with an initial state that is paired with the map information. Maps can be of various sizes, with the unit of measurement of maps being a tile. So for example, a 36x36 map represents 36 tiles by 36 tiles. The game supports battles of many units from several different owners but only two owners and up to 10 units for each owner will be used for the purposes of this research.

The Wargus client creates AI players through the use of YAML configuration files to allow easy creation of different archetypes of players. However, this configuration doesn’t fit well with the way we want to manipulate data since it does not allow direct control of the units during gameplay. Therefore when implementing it for the Bootstrapped Learning system, the Wargus client is integrated into the world agent plug-in rather than using the YAML configuration files to create the player to represent the Bootstrapped Learning student. For the purposes of FOIL, data is extracted and then preprocessed. The preprocessing will format the data into something FOIL-6 can understand, and separate the positive and negative examples.

4.3 Concept Structure

In choosing a scope of the domain to teach, two factors have to be considered. The first factor is the complexity of the problem. The problem needs to have enough depth and breadth to showcase the kinds of effects that happen when differing amounts of intermediate concepts are used. However, the problem cannot be so complicated that the learner takes an inordinate amount of time to come up with a response.

The second is the goal of the research. While it is possible to create a scope of learning that can address all of the issues in the domain, the point of the research is not to create a robust artificial intelligence for the game, like those created using reinforcement learning or fuzzy logic, but rather to investigate effects of a learning technique.

4.3.1 Concept Definitions

The experiment is run using the following concepts:
Final Concept

- TargetCorrect - true if the target has less than 25% health and closer than 5 tiles

Intermediate Concepts

- TargetInDistance (Distance) - true if the target is closer than 5 tiles
- TargetHasLowHealth (Health) - true if target has less than 25% health
- PercentageLessThan25 (Percentage) - true if ratio of two items is less than 0.25
- Add - true if sum of first two elements equals the third
- Subtract - true if difference of first two elements equal the third

Background Knowledge

- LessThan25 - true if first item is less than 0.25
- Divide - true if the dividend of the first two elements equals the third
- Square - true if the second element is the square of the first
- SquareRoot - true if the second element is the square root of the first
- Increment (Incr) - true if the second element is one more than the first
- Decrement (Decr) true if the second element is one less than the first
- UnitID - true if the element is a unit id integer
- MapCoordinate - true if the element is a map coordinate value

The concepts are built on each other as shown in Figure 3.

There are two main branches here: TargetHasLowHealth and TargetInDistance. They can be learned independently of one another and they are each built on other intermediate concepts but TargetCorrect is built on both of them.
Figure 3: Concept Structure
5 Research Methodology

This section outlines the steps taken to gather the data and setup the experiments.

5.1 Data Collection

Data for the tests was obtained in two ways. Background concepts and other concepts that were not particularly about targeting were randomly generated from formulas corresponding to their name. For example, a data point for \( \text{add} \) would be generated by randomly selecting points from a generator that had the formula \( x, y, x+y \) and some nonsense function such as \( \text{random}(35), \text{random}(35), \text{random}(70) \). The range for the randomization is chosen because addition is done over map coordinates, which range from 0 to 35.

The reason why background concepts are not simply given a definition is a limitation in the FOIL-6 engine. FOIL-6 does not allow for intensional definition statements. Therefore, data must be collected or generated for the background knowledge to pass in as examples for the extensional relation definitions.[1]

Data for the targeting concepts were gathered by running the Wargus simulator and extracting data from battles. The battles were done on a 36x36 tile map with no obstacles between the units and a 9 units versus 9 units attack configuration. The simulator was queried for the state of the game and then filtered based on if a data point was a positive example or a negative example. Taking each data point from an entire episode would be too many to gather so again, data points were chosen randomly from the set of possible data points.

5.2 Testing

A 10-fold cross-validation test was used to get the accuracy and speed performance effects of these intermediate concepts. The subsamples were taken randomly at the beginning and the same subsamples were used throughout the entire experiment. Cross-validation testing is used because
float CPUTime()
{
    struct rusage usage;

    getrusage(RUSAGE_SELF, &usage);

    return (usage.ru_utime.tv_sec + usage.ru_stime.tv_sec) +
            (usage.ru_utime.tv_usec + usage.ru_stime.tv_usec) / 1E6;
}

Figure 4: Code for getting time in FOIL-6

it is useful when estimating performance of a learned model from available data using only one
algorithm.

Tests were performed on an AMD Turion 64 Mobile ML-40 2.21 GHz with 2.00 GB of RAM.
Accuracy is defined as the ratio of the actual results and the predicted results. FOIL-6 returns
times based on the getrusage function in the C time library. The exact method used is shown in
Figure 4.

The tests are performed against three different cases: with all intermediate concepts, with inter-
mediate concepts as background knowledge, and without intermediate concepts. The tests will
be evaluated against the concepts that rely on intermediate concepts: TargetCorrect, TargetH-
asLowHealth, and TargetInDistance. Refer to Appendix A for more detail about concept defini-
tions.
6 Results and Discussion

In this section, we report the experimental results. The final concept in the concept structure \textit{TargetCorrect} is omitted from these results because FOIL-6’s runtime when trying to learn the concept was approximately 130,000 seconds and running it multiple times for the cross-validations checks was not feasible with the resources available. However, this data point does make for a good upper-limit check to see how the learning scales at each step.

6.1 Accuracy

The following tables detail the cross-validation accuracy values for each subsample and the average of the accuracy values. These results are taken from running FOIL with the original concept structure, with the intermediate concepts as background knowledge, and with the intermediate concepts removed.

<table>
<thead>
<tr>
<th>Concept</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>AVERAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>0.500</td>
<td>0.500</td>
<td>0.750</td>
<td>0.500</td>
<td>0.250</td>
<td>1.000</td>
<td>1.000</td>
<td>0.330</td>
<td>1.000</td>
<td>0.750</td>
<td>0.659</td>
</tr>
<tr>
<td>Subtract</td>
<td>0.800</td>
<td>0.600</td>
<td>0.600</td>
<td>0.800</td>
<td>0.600</td>
<td>0.750</td>
<td>0.750</td>
<td>0.600</td>
<td>0.200</td>
<td>0.800</td>
<td>0.650</td>
</tr>
<tr>
<td>Distance</td>
<td>0.950</td>
<td>0.857</td>
<td>1.000</td>
<td>0.905</td>
<td>0.952</td>
<td>0.900</td>
<td>0.950</td>
<td>0.950</td>
<td>1.000</td>
<td>1.000</td>
<td>0.947</td>
</tr>
<tr>
<td>Percentage</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Health</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 1: Original Concept Structure Accuracy

The results shown in Table 1 contains the accuracy values for the intermediate concepts of \textit{Health} and \textit{Distance}. Whereas the accuracy for the \textit{Add} and \textit{Subtract} are noisy, the accuracy for \textit{Percentage} is perfect.

<table>
<thead>
<tr>
<th>Concept</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>AVERAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>0.950</td>
<td>0.857</td>
<td>1.000</td>
<td>0.905</td>
<td>0.952</td>
<td>0.900</td>
<td>0.950</td>
<td>0.950</td>
<td>1.000</td>
<td>1.000</td>
<td>0.947</td>
</tr>
<tr>
<td>Health</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 2: Intermediate Concepts as Background Knowledge Accuracy

Comparing the data in Table 1 and Table 2, it does not appear that setting the intermediate concepts as background knowledge has any effect on the accuracy of the higher level concepts.
In terms of accuracy, the addition of intermediate concepts seems to have a negative effect. The exclusion of the intermediate concepts improves the accuracy of the noisy concept, Distance, and it does not harm the accuracy of the perfect concept, Health. The difference between the test cases can be more clearly seen in Figure 5.

Table 3: Intermediate Concepts Removed Accuracy

<table>
<thead>
<tr>
<th>Concept</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>AVERAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.950</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.995</td>
</tr>
<tr>
<td>Health</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Figure 5: Accuracy Comparison Across the Test Cases

6.2 Performance

The performance of FOIL across the different test cases has similar trends as the accuracy data but there are some slight differences.

Performance shown here makes sense. The concept that has more dependencies, Distance, takes longer than Health. The other concepts that depend on only a small subset of the background knowledge have short run times and are very similar to each other in terms of performance.
Whereas the data for accuracy did not change between the original concept structure and the one with intermediate concepts as background knowledge, there is a slight performance boost when it is no longer necessary to figure out the intermediate concepts. This performance boost is magnified when there are more intermediate concepts that the final concept depends on, as seen in the *Distance* concept.

Like in accuracy, performance decreases with the inclusion of intermediate concepts but the difference here is much more drastic. As shown in Figure 6, there is a very large difference between the *Distance* concept with the original concept structure and without any intermediate concepts.

### 6.3 Analysis

The results of the tests are somewhat surprising. Instead of improving the performance, the intermediate concepts actually hinder the system. Both accuracy and performance suffer from the inclusion of the intermediate concepts.
One reason why this may be the case is that the intermediate concepts hold redundant information. Since the intermediate concepts derive their meaning from the background knowledge, it comes to reason that the higher level concepts can do the same without having to go through the extra overhead of figuring out the intermediate concepts. This agrees with the data showing that the original concept structure and the concept structure with the intermediate concepts as background knowledge have no difference in accuracy. The differences between the two are in performance and that is in favor of the intermediate concepts as background knowledge.

Another possible reason is that since FOIL-6 does not have any idea of the structure of the system, it is still searching through all of the concepts to find dependencies. So rather than narrowing the search space, the intermediate concepts are actually expanding the search space. By removing all of the intermediate concepts, the search space narrows and performance increases.

An explanation for the accuracy results of the two high level concepts can be attributed to the concepts they are building on top of. In the case of Health, the components were all perfect so the result had perfect accuracy regardless of the concept structure. In the case of Distance, the intermediate concepts were noisy and building on top of those noisy concepts led to a less accurate description of the concept. Additionally, FOIL-6 may not always find the optimal definitions for these concepts because the FOIL algorithm is a greedy hill-climbing algorithm that is susceptible
to local maxima. Please refer to Appendix B for the Horn clauses produced by FOIL.

6.4 Implications on Bootstrapped Learning System

Given these results, the implications on the Bootstrapped Learning system are still unclear.

The Bootstrapped Learning system has many parts required to set it up. It needs a percept mapping, context files, a world model, package declarations, agent interface definitions, and test classes among many others. While it may be excessive for small learning tasks, many interesting learning tasks are large in scale.

It is not possible to add intermediate concepts without a dependency structure in the Bootstrapped Learning system. This observation means that an expert is required to know how concepts relate to each other beforehand in order to teach the target concept in the system. This also means that setting up the same test structure as in this research for the Bootstrapped Learning system would be difficult, if not impossible, because the required dependency structure would prevent removing the intermediate concepts. The possible problem FOIL-6 had with the lack of structure is largely avoided in this system because of this pre-defined structure.

Part of the power of the Bootstrapped Learning system is its ability to represent hierarchies of knowledge to its learning module via the curriculum that is passed in. In cases where the structure of the knowledge is unknown or incorrect, other types of learners will likely be superior.

Regardless of how well the structure is made, the inclusion of redundant, noisy, or irrelevant data will make the learner suffer. This means that poor curriculum authoring can affect both the accuracy and performance negatively.
7 Conclusion

The effect of intermediate concepts in a hierarchical relational learning structure is not uniformly positive. While the inclusion of intermediate concepts can save computation time and improve accuracy, it can also do quite the opposite. In tests that vary the amount of support provided by intermediate concepts in a hierarchical learning structure, it has been demonstrated that accuracy and performance can suffer severely. The poor accuracy and performance may be due to redundant information, lack of structure, or other reasons, but it cannot be said that the inclusion of these kinds of concepts will improve the system. Particular care must be taken when authoring these concept structures to avoid any redundant information and to convey that structure to the learning system.

However, there are still many theoretical advantages to using intermediate concepts if given a well-authored concept structure that is internalized by the learning system. The information provided by the structure can narrow the search space of the learning system, reducing the number of examples it has to process and therefore reducing the error resulting from exploring the extra examples. More work is needed to fully understand the advantages and disadvantages of structure in hierarchical learning.
8 Future Directions

8.1 Structure in Hierarchical Learning

Although it has been shown in this paper that intermediate concepts that are built on background knowledge do not inherently reduce the search space of the system, the next step is to determine how large a role structural knowledge is to the learning process. This can explore not only how useful it is to know that a concept is dependent on another concept, but how useful it is to know that a concept is entirely unrelated.

8.2 Partially Correct Knowledge

The tests that have been made can be further refined with sophisticated examples that explore partially correct knowledge. The examples in this paper that come closest to that are the concepts with noisy data. However, noisy data and partially correct knowledge are very different ideas. The partially correct knowledge should be measured against a metric that rates the amount of information conveyed to the dependent concept by the partially correct concept. Then it should be tested to see the effects of its inclusion.

8.3 Self-Organizing Hierarchical Relational Learner

Another interesting task would be to have a learner set up its concept structure based on the dependencies that it learns. Assuming that information about the structure can be constructed from the first pass through the data, subsequent passes can retain this information to guide the learning process.
References


Appendix A: FOIL-6 Relation Definitions

The type specifications introduce the world of values that will be searched.

Types: N, J, K, H, G

N ranges from 0 to 60. It has a range of values that spans all possible numbers extracted from the simulator. 0, 1, and 60 are theory constants that can appear in the definition.

Note: 0 to 35 is the range of possible map coordinates J ranges from $0^2$ to $35^2$. It covers the squares of the numbers within that range.

K ranges from 0 to 70. It spans the set of adding any two numbers from 0 to 35.

H ranges from 0.00 to 60.00. It represents the possible values when dividing numbers from 0 to 60.

G ranges from -35 to 35. It spans the set of subtracting any two numbers from 0 to 35.

Relations:

Note: a * represents the relation is background knowledge, a # represents the particular argument must be bound, and a - represents that argument can be bound or unbound

*unitid(N) # has values from 0 to 17 representing the range of possible unitid values in the tests.

*mapCoordinate(N) # has values from 0 to 35 representing the range of possible map coordinate values in the tests.

*incr(K, K) #- has increment pairs from 0 to 70.

add(N, N, K) takes in arguments and checks if the first two arguments sum up to the third.

*decr(G, G) #- has decrement pairs from -35 to 35.

subtract(N,N,G) takes in arguments and checks if the difference of the first two arguments equals the third.
*square(N, J) #* has pairs where the second argument is the square of the first argument. It is fully defined for first argument values of 0 to 35.

*sqrt(J, N) #* has pairs where the second argument is the square root of the first argument, rounded down. It is fully defined for all sums of squares in the system.

distance(N,N,N,N,N) takes in five arguments. The first two represent the first pair of coordinates on the map. The second two represent the second pair of coordinates on the map. The last argument is the numerical distance between these two points, rounded down. It is true when the distance between the two points is less than 5

*doubleLT25(H,H) #* has comparisons of all values defined in the H type less than 0.25. It is fully defined.

*divide(N,N,H) has three arguments. The third argument represents the division of the first two, rounded to two significant figures. This is fully defined for all pairs that are not divided by 0 in the range of 0 to 60.

percentage(N,N) has pairs of values that when divided, will be less than 0.25

health(N,N,N,N,N,N,N,N,N) has two groups of four arguments. The first four represent the current unit id, maximum health, current health, and target id. The second four represent the enemy id, maximum health, current health, and target id. It is true when the ratio of the enemy’s current health to maximum health is below 0.25

choice(N,N,N,N,N,N,N,N) has eight arguments. The first three are the current unit id and map coordinates. The second three are the enemy unit id and map coordinates. The last two are the enemy unit’s current health and maximum health. It is true when the ratio of the enemy’s current health to maximum health is below 0.25 and the distance between the two units is less than 5
Appendix B: Horn Clauses Produced by FOIL-6

Percentage

percentage(A,B) :- divide(A,B,C), doubleLT25(C,D).

We have defined percentage to be pairs of values that are less than 0.25 when the first term is divided by the second term. The result says that it is true when C, which is the result of dividing A by B, is less than D. D is always 0.25 in the case of doubleLT25 so this is exactly what we want.

Health


The Horn clause has 100% cross-validation accuracy but it is not actually the definition we want. Here, it uses the B term instead of the F term. In all the examples, however, they are the same. This is because they both represent the maximum health of the respective units. The unit types in the examples are all the same and thus, have the same maximum health.

Health With Intermediate Concepts Removed


The result without the intermediate concepts is unchanged. This is because the Horn clause produced with the intermediate concepts did not rely on percentage, the intermediate concept.

Add

add(0,B,B).

The additive identity rule.

add(A,B,A).
This rule is also the additive identity rule but FOIL chooses to use the variable B instead of the theory constant 0. There are no negative examples in the set of examples that indicate otherwise and FOIL attempts to make the most general rules that apply to all examples.

\[
\text{add}(1, B, C) : - \text{incr}(B, C).
\]

The rule indicates that \(1 + B = C\), which is exactly what the increment concept means. It correctly identifies that addition builds on top of increment.

\[
\text{add}(A, B, C) : - \text{incr}(B, D), \text{incr}(D, C).
\]

A convoluted way to say that \(A + 2 = C\). The constant 2 is not labelled as a theory constant so increments are chained to produce the same effect.

\[
\text{add}(A, B, C) : - \text{incr}(B, A).
\]
\[
\text{add}(A, B, C) : - \text{incr}(A, D), \text{incr}(D, B).
\]

The Horn clause is overfitting the examples. It thinks that the third argument in the addition concept does not matter as long as the first argument is one more than the second argument.

The definition for addition FOIL-6 has covers 83% of the example set. Further attempts to choose proper clauses required more literals than allotted by FOIL-6 for the number of examples it would cover and are not added to the definition set.

**Subtract**

\[
\text{subtract}(A, B, A).
\]

The subtractive identity rule using the variable B instead of the theory constant 0. There are no negative examples in the set of examples that indicate otherwise and FOIL attempts to make the most general rules that apply to all examples.
subtract(A,B,C) :- decr(A,C).

$A - 1 = C$ but uses the variable B instead of the theory constant 1. However, the variable B is disregarded in the Horn clause so it is just a rephrasing of what the decrement concept means. It correctly identifies that subtraction builds on top of decrement.

subtract(A,B,C) :- decr(A,B).
subtract(0,B,C) :- decr(B,D), decr(D,E), mapCoordinate(E).

$subtract(0,B,C)$ should produce a clause that recursively subtracts to get to C but the value of C is disregarded instead. While these clauses may fit some of the examples, it is not generalizable.

subtract(A,A,C) :- decr(A,D), decr(D,C).

Covers the case of A=2. This is a very specific rule.

subtract(A,A,C) :- decr(A,D), decr(D,E), decr(E,F), mapCoordinate(F).
subtract(A,B,C) :- decr(A,D), decr(C,E), decr(D,F), decr(F,G), mapCoordinate(G),
    decr(E,H), mapCoordinate(H).
subtract(A,B,C) :- decr(B,D), decr(D,A).
subtract(1,B,C) :- subtract(B,B,D).

These rules are all a result of overfitting. Some of them find intricate ways to relate to subtract but do not make any sense in the context of what we know about subtraction. The last Horn clause $subtract(1,B,C) :- subtract(B,B,D)$ recursively calls earlier cases; namely, the $subtract(A,A,C)$ case.

The definition for subtraction FOIL-6 has covers 83% of the example set. Further attempts to choose proper clauses required more literals than allotted by FOIL-6 for the number of examples it would cover and are not added to the definition set.

**Distance**
The definition for distance produced by FOIL-6 is convoluted and is not close to the definition of distance we know. The Horn clause of the formula for distance should be:

distance(A,B,C,D,E) :- subtract(C,A,F), subtract(D,B,G), subtract(F,H),
square(G,I), add(H,I,J), sqrt(J,E).
Despite having a very complicated definition that does not match the actual definition, it had a classification accuracy close to 95%.

**Distance with Intermediate Concepts Removed**

distance(A,B,C,D,E) :- square(E,F), unitid(F), sqrt(E,G).
distance(A,B,C,D,E) :- decr(C,F), square(E,G), unitid(G), incr(H,F).

Without the intermediate concepts, a much simpler definition is found for distance that is even more accurate than the definition with intermediate concepts. An explanation for this is that with the extra concepts, FOIL-6 finds local maxima that are inferior to the maxima found with fewer concepts. FOIL-6 uses a greedy hill-climbing algorithm so this is not absurd. However, the definition found for Distance without intermediate concepts is still far from the actual definition.