Research on a Learning Model Presuming Representationalism, Functionalism, and Neural Darwinism

Jen-Hao Chang
Tian-Li Yu

TEIL Technical Report No. 2012001
June, 2012

Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
No.1, Sec. 4, Roosevelt Rd., Taipei, Taiwan

http://teilab.ee.ntu.edu.tw

Research on a Learning Model Presuming Representationalism, Functionalism, and Neural Darwinism

Jen-Hao Chang backitup@teilab.ee.ntu.edu.tw
Tian-Li Yu tianliyu@cc.ee.ntu.edu.tw

June, 2012

Abstract

Even the best artificial intelligence technology can not compare with humans on the inductive ability. Humans can learn general rules from observed instances based only on bare-bone prior knowledge. Lots of research on the human mind tries to explain the learning ability of humans. Of all these hypotheses about the human mind, this thesis designs a learning model based on the ideas of representationalism, functionalism, and neural Darwinism. This thesis researches on the learning ability of the proposed model when the model is given only bare-bone prior knowledge. The proposed model is composed of three parts, which are respectively founded on representationalism, functionalism, and neural Darwinism. The knowledge representation of the proposed model is designed based on representationalism: symbols are the basic components of the knowledge representation of the model. The combination of knowledge of the proposed model is designed based on functionalism: the knowledge in the model can be combined in the form of functions. The inductive ability of the proposed model is designed based on neural Darwinism: the model exploits evolutionary processes to learn general rules. This thesis analyzes the proposed learning model. The expressive power of the model is the same as the universal Turing machine. The inductive process of the model is sound but incomplete. Three experiments are used to test the learning ability of the proposed model. Results shows that the learning model is able to learn general rules in these experiments. The results shows that the learning model is able to induct general rules from input instances of these experiments. This thesis proves that a learning model presuming representationalism, functionalism, and neural Darwinism is able to do inductive learning. Even based on bare-bone prior knowledge, the learning model still can learn general rules from instances. The learning model can exploit additional analogical knowledge to increase its inductive learning ability.

Introduction

Of all the abilities of the human mind, learning ability is one of the most influential abilities in humans' life. Humans have the ability to conclude inductive rules from instances even based on little prior knowledge. The learning ability of the human mind has been researched for years but is still not fully explained. One of the research fields about the human mind is cognitive science.

Cognitive science can be roughly summed up as a scientific interdisciplinary study of the mind, including philosophy, psychology, linguistics, artificial intelligence, robotics, and neuroscience. Its theoretical perspective on the mind is information processing. According to Ulric Neisser (20), information processing refers to all processes whereby the sensory input is transformed, reduced, elaborated, stored, recovered, and used. In other words, as an information processor, the mind represents and manipulates information. Computer science thus finds its ways into cognitive science. A specific information-processing model of a mental ability can be simulated on a computer. Aspects of the simulation can also yield insights which cause researches to develop new models. Computer science thus accelerates the development of cognitive science. This thesis proposes a model of the learning process of the human mind and simulates it on the computer.

Based on three hypotheses the thesis proposes a learning model to simulate the learning process of human. The three hypotheses are representationalism, functionalism and neural Darwinism (9).

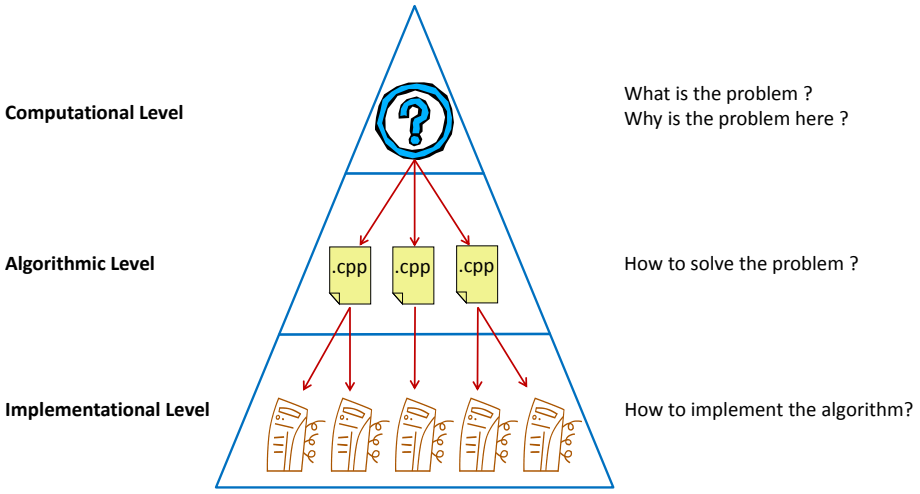


Figure 1: Mental or artificial information-processing events can be evaluated on three different levels.

Representationalism, also known as indirect realism, states that the mind uses symbols to refer the external world. These symbols are called representations. Functionalism states that mental states are functions of sensory inputs, other mental states, and behavioral outputs. Neural Darwinism applies the idea of evolutionary processes to neural learning. In short, neural Darwinism asserts that many cognitive processes are not hardwired or pre-specified at birth. Instead, a variety of webs of neural circuits are pruned and selected by experience. Moreover, it states that all higher level activities in brains are the results of evolutionary processes. The learning model presumes only these three hypotheses of cognitive science and can be used to investigate these hypotheses.

According to Marr (18), mental or artificial information-processing events can be evaluated on at least three different levels, as shown in Figure 1. The highest level is the computational level. The computational level defines what the information problem is and why does the problem arise. Stepping down one level is the algorithmic level, which is a description of steps to carry out when solving the problem. At the bottom is the implementation level. The implementation level describes the actual physical characteristics of the information processing system. There is only one computational description of one particular problem. There can be multiple algorithms which solve the problem and different ways in which an algorithm can be physically implemented. This thesis focuses on the algorithmic level of the learning process of human. Particularly, this thesis focuses on a model of the learning process of human. Why does human learns and how is the learning process physically implemented are not covered in the thesis.

For decades, evolutionary computation (8; **?**) has been used to solve various problems such as search and optimization. Evolutionary computation implements the idea of evolution. First, it encodes candidate solutions into chromosomes, which are strings of variables. Then, under a predefined measurement of qualities of candidate solutions, the betters from a collection of candidate solutions are selected to recombine. Resulted candidate solutions after the recombination process are expected to have higher qualities. The selection-recombination process can be repeated until an acceptable solution is created. In the thesis, the idea of neural Darwinism is practiced by applying evolutionary computation techniques. The processes of creating and selecting functions of representations are implemented with evolutionary computation algorithms.

This thesis utilizes representations to express knowledge. The proposed model manipulates information by recombining knowledge as functions. In order to reflect how the learning model performs, the model learns with bare-bone prior knowledge. By conducting experiments on this model, insights into these hypotheses can be gained. Because representations are composed of symbols in the model, both intermediate processes and results are more understandable compared to models utilizing methods like Bayesian networks (14; 15) or neural networks (4; 1). In particular, this thesis is dedicated to exploring the learning ability of the proposed model.

Thesis Objectives

The objectives of this thesis are

1. Based on representationalism, functionalism, and neural Darwinism, design a learning model at the algorithmic level.

2. Analyze the learning ability of the proposed model.

Particularly, this proposed learning model uses representations to express knowledge. Knowledge is treated like functions and can combine with each other. Evolutionary computation algorithms are implemented to simulate the process described in neural Darwinism. The learning model is developed based on only hypotheses about the human mind. It therefore could be used as a tool to investigate these hypotheses and the human mind.

Roadmap

This thesis is composed of four chapters.

Chapter 1 presents the background knowledge for this thesis. First, Representationalism, functionalism, and neural Darwinism are elaborated. Then, evolutionary computation is described along with an example to show how evolutionary computation is adopted when solving optimization problems. At the end of this chapter, inductive theorem proving is introduced. The concepts presented in the chapter should provide enough background for readers to follow this thesis.

Chapter 2 proposes the learning model, which is designed based on representationalism, functionalism, and nerual Darwinism. The connections between these hypotheses and the design of the model are first described. An overview of the model is then illustrated. The implementation of the model is described in detail, including knowledge representation, combination of knowledge, and the inductive process. It explains how evolutionary computations cooperate with these hypotheses to form the proposed learning model. At the end of this chapter, the proposed model is compared with some existing cognitive models and inductive theorem proving systems.

Chapter 3 first analyzes the expressive power, soundness, completeness, and decidability of the proposed model. Three experiments are then conducted to test the learning ability of the model. Each of these experiments tests different aspects of the learning model.

Chapter 4 first gives a summary of this thesis. It then concludes this thesis. At the end, several possible directions to extend the work are presented.

1 Background

This chapter gives an introduction to representationalism, functionalism, neural Darwinism, some background of evolutionary computation, and inductive theorem proving. These background knowledge will be used later in the thesis.

1.1 Representationalism

Representationalism is the philosophical position that the world which humans see in conscious experience is not the real world itself, but an internal representation of the world (9). As illustrated in Figure 2, the human mind refers to objects in the external world as representations. Those objects are referents of these representations. The mind explains and manipulates the external world via these representations.

Various hypotheses about the representations are proposed, however, they all have the following six characteristics (13).

1. A *representation bearer* such as a human or a computer must realize a representation.

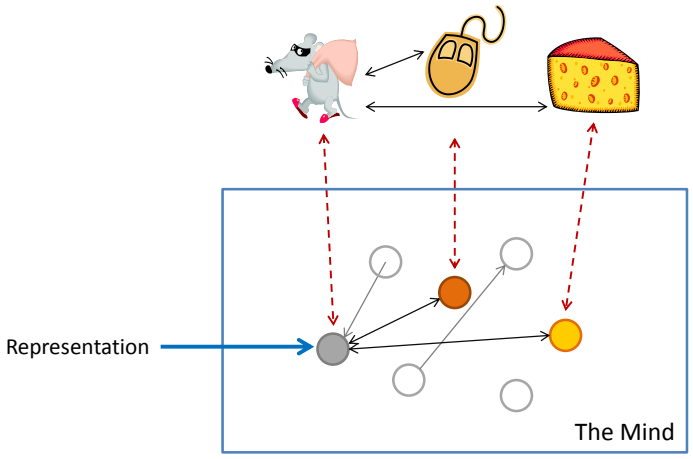


Figure 2: An illustration of representationalism.

2. A representation must have content–meaning that it stands for one or more objects.
3. A representation and its referent must be related.
4. A representation must be interpretable by representation bearers.
5. A representation bearer can map different aspects of a representation onto its referent, and vise versa. It is called *isomorphism*.
6. Causal relations between referents must be preserved by their representations.

Any implementation of representations should satisfy these characteristics.

Researches in cognition suggest that there are numerous forms of representation. Paul Thagard proposes the following four forms (23).

Concept is the most basic form of representation. A concept is an idea that represents things which have been grouped together. For example, the concept "chair" does not refer to a specific chair but refers to all possible chairs no matter what their colors, sizes, shapes. Concepts can also refer to abstract ideas, like "hot" or "love". Concepts are related in a hierarchical fashion, where a concept at one level stands for all concepts directly below it.

Proposition is a statement or assertion typically posed in the form of a simple sentence. An essential feature of a proposition is that it can be proved true or false. Proposition can be expressed by:

[Relationship between elements] ([Subject element],[Object element])

where the relationship and elements are concepts.

Rule is a conditional statement of the form: "If x, then y," where x and y are propositions.

Analogy represents the similarity between two situations. Thinking analogically involves applying one's knowledge about a situation to another situation. Suppose one had never taken an airplane before, but has taken ships. The experiences of ship taking can be represented by analogy and used to represent the airplane taking. Analogies provide the ability to generalize learning and can also be classified as a form of reasoning.

This thesis utilizes all these forms of representation to express information. Section 2.1.1 will shows how this thesis utilizes these forms of representations.

1.2 Functionalism

Functionalism is a hypothesis about the mind in philosophy. The core idea of functionalism is that the mind is composed of various *functional kinds*, which are distinguished by their function to sensory inputs, behavioral outputs, and other functional kinds (9). In other words, functionalists consider the mind to be a system of functions. Each kind of these functions is responsible for a certain purpose.

One implication of functionalism is that the same functional kind can be realized in a quite different way in two separate physical systems (10). It can be illustrated with computing devices. The same program can be written in different programming languages and can be executed on different computer architectures, such as reduced instruction set computing (RISC) or complex instruction set computing (CISC). According to this implication, different physical substrates can give rise to mind as long as they are composed of correct combinations of functional kinds. This view provides the foundation for computer simulations of mental processes. Section 2.1.2 will shows how this thesis utilizes the ideas from functionalism.

1.3 Neural Darwinism

Neural Darwinism was first proposed by Gerald Edelman in 1978 in order to reconcile two sets of observations that seemed inconsistent with the then prevalent notions of brain function (6). First, individual nervous systems show enormous structural and functional variability. Although there is an obvious commonality of neural structure within a species, the degree of individual variation far exceeds that which could be tolerated for reliable performance in any machine constructed according to current engineering principles (5). Second, the world of stimuli encountered by a newborn animal can not be described adequately as preexisting, unambiguous information ready to be manipulated according to a set of rules similar to those followed by a computer executing a program (5).

Gerald Edelman proposes the theory of neuronal group selection (6) to explain these two observations. It argues that the ability of organisms to categorize an unlabeled world and behave in an adaptive fashion arises not from instruction or information transfer, but from processes of selection upon variation. Like the theories of natural selection and of clonal selection in immunity, the theory of neuronal group selection is a population theory.

According to the theory, three interactive processes exist for selection upon variation. The first process occurs largely in embryonic and postnatal development, during which adjacent neurons tend to be strongly interconnected in collectives of variable size and structure and form neuronal groups. The second process consists of alterations in synaptic strengths during an animal's activity, selecting the correlated responses of these neuronal groups that yield adaptive behavior. The third process is the correlated activities between neuronal groups. These correlated activities can resolve conflicts arising between the synaptic activities of different neuronal groups and can enforce basic rules such as spatiotemporal continuity on resulted neuronal groups.

Although the theory of neuronal group selection only accounts for the selection and variation of neuronal groups, another work (7) proposes a mechanism for the multiplication of neuronal groups. Therefore neuronal groups go through the processes of variation, selection, and multiplication. These processes together form the foundation of neural Darwinism.

Further, Edelman states that all higher level activities in brains, including memory and learning, are the results of evolutionary processes. In his opinions, memory is the enhanced ability to categorize or generalize associatively, not the storage of features or attributes of objects as a list. The minimal unit capable of carrying out the *recategorizations* is a global mapping buttressed by learning, both of which are related to the satisfaction of systems linked to evolutionary survival.

1.4 Evolutionary Computation

Evolutionary computation (8; **?**) adopts the principles of natural selection to solve search and optimization problems. Evolutionary computation is good at solving problems which can be decomposed into parts which have little dependencies between other parts. The problem therefore can be encoded, and the result is composed of variables which represent different part of the problem. The coding result is often called chromosome, and these variables are called genes. Chromosomes

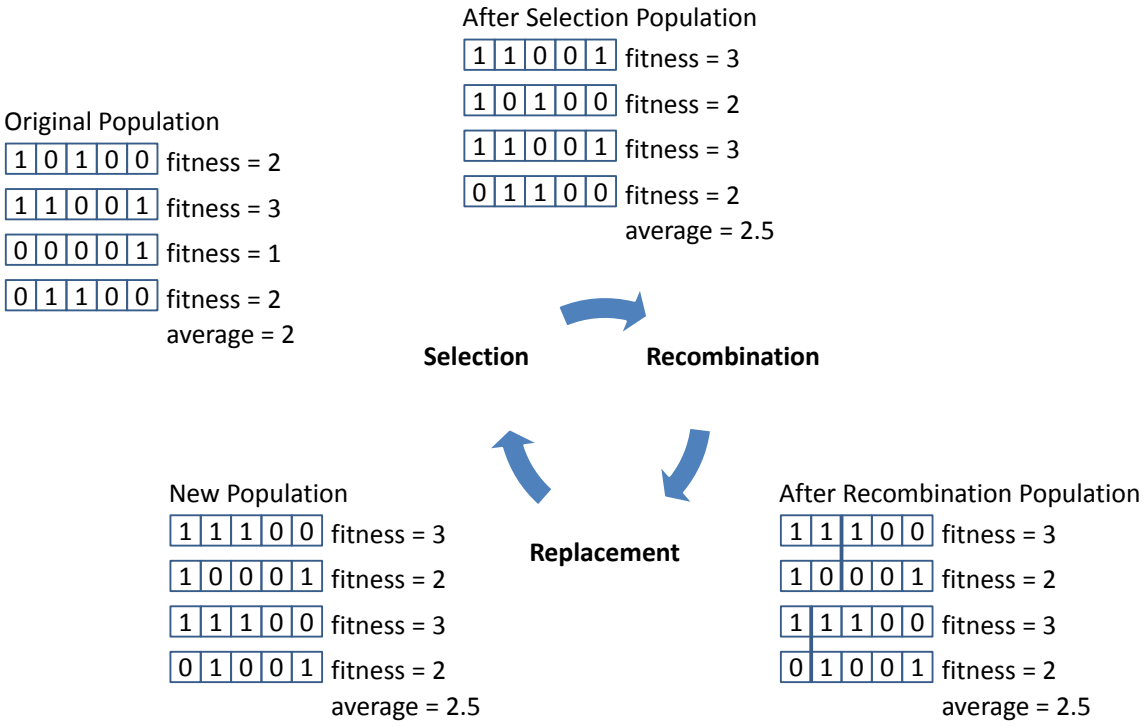


Figure 3: An example of using evolutionary computation to solve an optimization problem.

are candidate solutions to the problem. Evolutionary computation implements its algorithms on chromosomes.

For a given a problem, the quality of a candidate solution, also called fitness, can be defined. Under a definition of fitness, evolutionary computation utilizes three processes to solve a problem. The three processes are selection, recombination, and replacement. They are shown in Figure 3. Evolutionary computation repeatedly goes through these processes until a candidate solution is accepted.

Suppose higher quality indicates higher fitness. As Darwin's theory of natural selection suggests that species which better fit the environment are more likely to reproduce, selection is the process which replicates higher-fitness candidate solutions. The basic idea is that candidate solutions with higher fitness have higher chances to be replicated. Two methods are commonly used as selection. One is roulette-wheel selection, and the other is tournament selection (19). Roulette-wheel selection replicates candidate solutions proportionally to their fitness. Tournament selection on the other hand replicates the bests of randomly chosen groups of candidate solutions.

Recombination generates new candidate solutions based on those selected candidate solutions. Two main methods are used to recombine candidate solutions. One is crossover, and the other is mutation. Crossover is the process that two or more candidate solutions exchange their genes. Mutation is the process to modify genes of one candidate solution. Because these resulted candidate solutions are generated from those betters in the original candidate solutions, these resulted candidate solutions are expected by evolutionary computation to have higher fitness than original candidate solutions. These resulted candidate solutions, or the offspring, will proceed to the replacement process to replace original candidate solutions, the parent population.

Replacement is the procedure for replacing the parent population with the offspring. Certain candidate solutions of the offspring are chosen and replaces certain candidate solutions in the parent population. The simplest way to do replacement is replacing the entire parent population with the entire offspring. This kind of replacement is called full replacement. Sometimes the diversity of the population is taken into consideration. One way to keep the diversity of the population is that only replacing parent candidate solutions with higher-fitness offspring candidate solutions which are similar to these parents.

Here, a simple problem, shown in Figure 3, is used as an example to illustrate how evolutionary

computation solves optimization problems. Suppose that the problem can be decomposed into five independent parts, and each part has two possible values. Encoded candidate solutions are therefore composed of an array of binary variables. The fitness is defined as the total number of *one* in a candidate solution. The goal of the optimization problem is to find a candidate solution with the highest fitness. Note that the definition of the fitness is unknown to evolutionary computation algorithms, and so is the highest fitness value. When using evolutionary computation, selection, recombination, and replacement need to be designed. For simplicity, roulette-wheel selection and full replacement are chosen. When doing recombination, pairs of candidate solutions are picked and exchange their genes preceding a random location. At first, four candidate solutions are generated randomly. After running each process once, the average fitness of the population rises. The three processes can be repeated until a candidate solution is accepted.

1.5 Inductive Theorem Proving

Inductive theorem proving (3) is a field in automated reasoning and theorem proving (12). Inductive theorem proving (inductively) deduces more general instances from many instances of formulas. The very general *principle of well-founded induction*, for proving some property P depending on a parameter x, may be stated in the form of an inference rule as follows:

$$\frac{\forall x\,[\,(\,\forall y\; y < x \Rightarrow P(y)\;) \Rightarrow P(x)\,]}{\forall x\; P(x)}$$

, where $<$ is a well-founded order (on the domain of x).

An inference system based on some set G of assumptions defines the derivability of a formula F, denoted by $G \vdash F$. If F is derivable, it is a deductive theorem. An interpretation satisfying a set G of assumptions is called a *model* of G. A formula F is *semantic entailed from* or *consistent with* a set G of assumptions, denoted by $G \models F$, if F holds in all models of G. Suppose F is an inductive theorem, G is the set of assumptions, and $\forall f,\ G \vdash f \Rightarrow G \models f$. To prove that F is consistent with G, one can use

$$G \models F \Longleftrightarrow \forall \sigma\; G \vdash \sigma F$$

, where the substitution σ ranges over all ground substitutions over the given signature.

1.6 Summary

This chapter first introduces representationalism, functionalism, and neural Darwinism. Representationalism states that there are four types of representations in the human mind. Functionalism states that the mind is s system of functions. Neural Darwinism states that there are evolutionary processes happening in the brain. The chapter then gives some introductions to evolutionary computation algorithms and inductive theorem proving. Based on these hypotheses of cognitive science and evolutionary algorithms, the thesis proposes an inductive theorem proving model. This proposed model is then analyzed and tested for its learning ability.

2 Design the Learning Model

This chapter proposes the learning model which presumes representationalism, functionalism, and neural Darwinism. The model simulates the learning process of the human mind depicted by these three hypotheses. It tries to generalize from input instances with only bare-bone prior knowledge about these input instances. Therefore the model can also be used to investigate these hypotheses.

The chapter is organized as follows. It first introduces how the learning model is inspired by these three hypotheses, and how the model is designed based on these hypotheses. An overview of the learning model is then provided along with the position and the block diagram of the proposed model. Prior knowledge needed by the proposed model is described. Then, the chapter describes the detail of the implementation of the model. At the end, the proposed model is compared to existing cognitive models and inductive theorem proving systems.

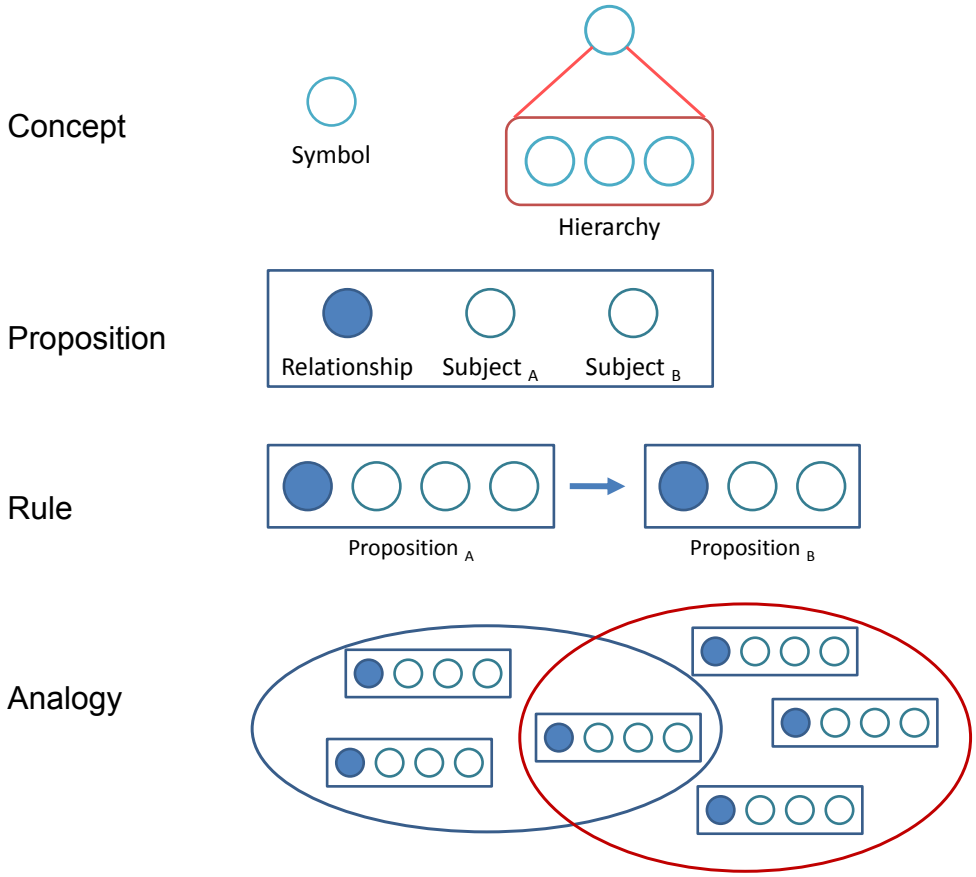


Figure 4: This figure illustrates the knowledge representation implemented in the model.

2.1 Based on Cognitive Hypotheses

A learning model is composed of three components. The first is the knowledge representation used in the model. The second is the operations on these knowledge representations. The third is the learning algorithm. Each of these components of the proposed model is designed based on one cognitive hypothesis. Representationalism, functionalism, and neural Darwinism are hypotheses about the human mind. Representationalism is the foundation of the knowledge representation. Functionalism is the foundation of the operations on knowledge representations. Neural Darwinism is the foundation of the learning algorithm of the model. This section will describes how these components are designed based on these hypotheses.

2.1.1 Based on Representationalism

According to representationalism, objects in the external world are represented by representations in the human mind. The mind manipulates and explains the external world via these representations. Therefore, representations are implemented in the proposed model to represent knowledge. Section 1.1 introduces different forms of representations, including concept, proposition, rule, and analogy. These forms of representations provides different ways to represent knowledge. Each of them will be described below, and Figure 4 illustrates the summary of knowledge representation implemented in the model.

A concept represents things which have been grouped together. Concepts are isomorphisc, meaning that a concept can always represent the same group of things regardless of aspects to see these things. Based on the characteristics of concepts, in practice the proposed model implements concepts with symbols. In this model, the smallest unit to expressing the world is symbol. Each symbol has its domain and value defined. For example, the symbol representing Tina, who is a girl, has domain `Girl` and value `Tina`. A symbol always represents the same group of things, and

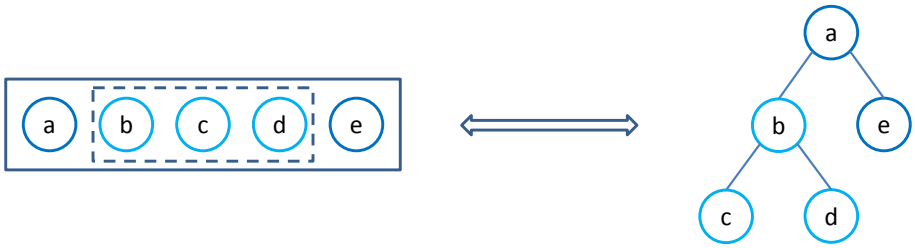


Figure 5: The tree-wise structure of a proposition representation.

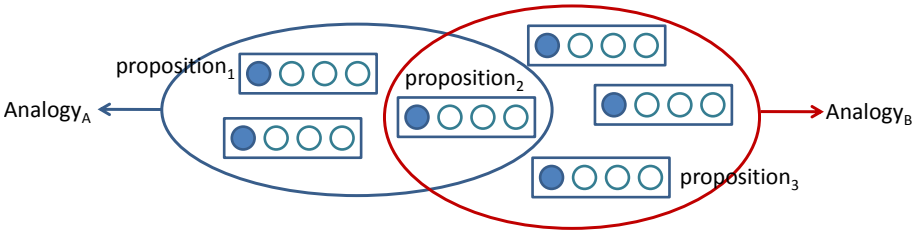


Figure 6: An example of analogy representations.

the equality of symbols is defined. Two symbols are equal if they have the same domain and the same value. Given an object, all symbols which can represent the object can be found. Concepts are related in a hierarchical fashion, where a concept at one level stands for all concepts directly below it. Therefore, some symbols can represent a group of symbols and form a unary relation of symbols. In order to distinguish this kind of symbols from other, in the following of the thesis, this kind of symbols will be called `relation`. When mentioning symbols, all symbols are referred, including `relations`.

The second form of representation, the proposition representation, is composed of concepts. Propositions can be expressed by:

[Relationship between elements] ([Subject element],[Object element])

where the relationship and elements are concepts. In the proposed model, a proposition is composed of a sequence of symbols. The first symbol in the sequence is the symbol representing the relationship among the following symbols. A proposition can be proved true or false, and so is a sequence of symbols. Because the first symbols in a proposition representation represents the relationship among the following symbols, the proposition representation is actually a syntactically pre-ordered sequence of symbols. This syntactic ordering forms a tree-wise structure in the sequence, which is illustrated in Figure 5. Each parent in the tree is a symbol representing the relationship among its children.

Rule representation is another form of representation. A rule representation is of the form: "If x, then y," where x is the conditional proposition and y is the consequent proposition. Causal relations among concepts can be expressed by rules. In the proposed model, a rule representation is composed of two sequences of symbols, in which the first sequence stands for the conditional proposition and the other stands for the consequent proposition.

The last form of representation, the analogy, enables the mind to apply knowledge about a situation to another situation. Analogies implemented in the thesis are predefined sets of proposition representations. Each set of propositions is considered as a type of analogy. Propositions belonging to the same set are analogically equal. Figure 6 illustrates an example of analogy representations. In this example, proposition_1 and proposition_2 belong to Analogy_A; proposition_2 and proposition_3 belong to Analogy_B; proposition_1 and proposition_3 are not analogically equal. Note that one proposition can belong to no analogy, and the proposed model does not know what belongs to an analogy. The model can only test whether two propositions belong to the same analogy.

Analogy allows the model to apply knowledge about one proposition to another which is analogically equal to the proposition. If two propositions are analogically equal and one of them has known consequent proposition, the model can assume that the other proposition has the same

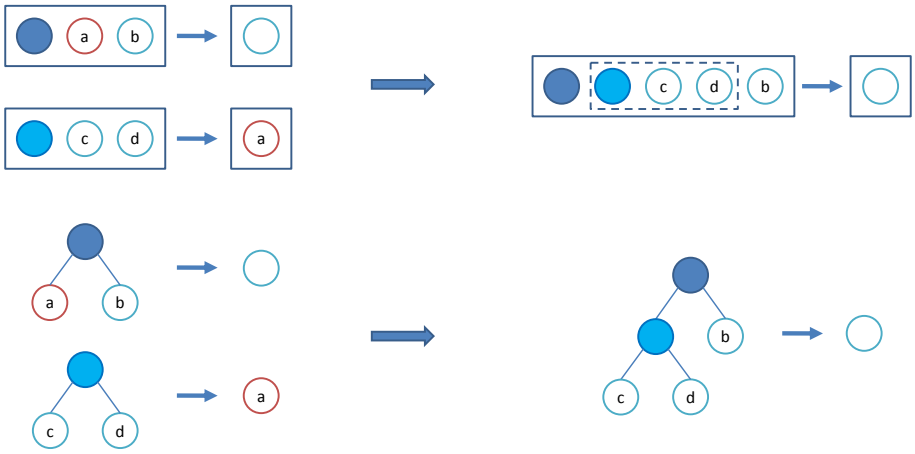


Figure 7: The combination of two rule representations.

consequent proposition. Analogical equality of two propositions does not guarantee that these two propositions will have the same consequent proposition. Sometimes two analogically equal propositions have different consequent propositions. In this situation, this analogy should not be applied to these propositions. The model learns the type of analogy which can be applied to one proposition by testing whether any analogically equal proposition in the knowledge base conflicts with the tested proposition. If any conflict happens, this analogy is considered inconsistent to all propositions analogically equal to the tested proposition.

2.1.2 Based on Functionalism

According to Functionalism, the mind is composed of various functional kinds, which are distinguished by their function to sensory inputs, behavioral outputs, and other functional kinds. The mind therefore can be thought as a system of functions.

All representations implemented in the proposed model are functions. Given a symbol, whether or not an object is represented by the symbol is either true or false. Any proposition can be proved true or false. A rule representation expresses a causal relation of propositions, and therefore a rule representation is itself a function. Note that different rules with the same conditional proposition and different consequent propositions are considered as different functions. Analogy representations also satisfies the definition of function, because the set-belonging property is a function.

Functions can be connected with each other by cascading. Outputs of a function can be the inputs of another function. This characteristic of function is the foundation of combining representations in the proposed model. In the proposed model, propositions can connect with each other. Parts of a proposition can be replaced by another proposition if the resulted proposition is syntactically correct. As a result, rule representations can combine with each other. Figure 7 shows an example of combining two rules. As shown in this figure, the combinations of rules can also be considered as replacing subtrees.

2.1.3 Based on Neural Darwinism

Neural Darwinism states that the adaptive behaviors of the mind sources from the variation, replication, and selection of neuronal groups. Each neuronal group is considered as a function in the thesis. It means that the evolutionary processes upon neuronal groups are implemented with the evolutionary processes upon functions. In particular, the evolutionary processes are operating on proposition and rule representations.

The implementation is supported by functionalism and representationalism. As the neural Darwinism considers neuronal groups as the unit of evolutionary processes, functionalism states that the mind is composed of functional kinds. Neuronal groups therefore can be considered as functional kinds. Under the scope of learning, these functional kinds represent knowledge. Representationalism states that knowledge is expressed by representations in the mind. The proposed

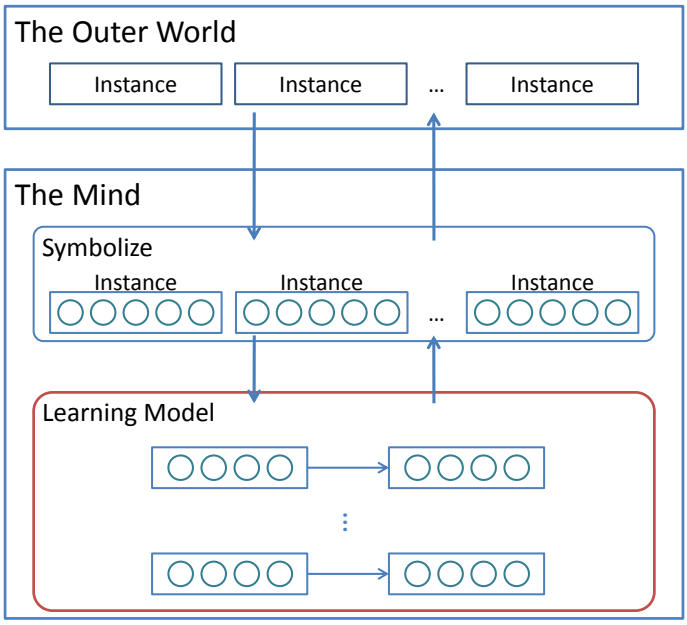


Figure 8: The position of the model.

model uses symbols to express representations. Therefore the evolutionary processes are performing on symbols.

Performing evolutionary processes on symbols has one immediate advantage. Operations on symbols are understandable by human when comparing to operations on neurons. If on neurons, evolutionary processes need to adjust structures and synapses of neurons, and these adjustments can be difficult to trace and understand by human. On the other hand, this proposed learning model involves only operations on symbols. These operations includes replacing, reordering, duplicating symbols, and they are all understandable by human.

The evolutionary processes are implemented with evolutionary algorithms. Neural Darwinism states that there are selection, replication, and variation of neuronal groups. In the proposed model, each of these processes is simulated by corresponding evolutionary algorithms operating on proposition and rule representations. In particular, selection of neuronal groups corresponds to selection of representations; replication of neuronal groups corresponds to duplication of representations; variation of neuronal groups corresponds to recombination of representations. Neural Darwinism states that while forming neuronal groups some basic rules such as spatiotemporal continuity are enforced on these neuronal groups. Likewise when recombining symbols the syntactic correctness in each propositions is enforced.

2.2 Overview of the Model

This section gives an overview of the learning model. First, this section shows the position of the model in the architecture of learning. Then, this section describes the learning procedure. Last, the block diagram of the learning model is described.

2.2.1 Position of the Model

The proposed learning model focuses on the knowledge-learning procedure following the symbolization of the inputs. Figure 8 illustrates the position of the learning model. Because representationalism suggests that in the mind there exist symbols representing the external world, symbolization can be thought as an interface between the mind and the external world. The proposed model learns upon symbolized input instances, and the symbols available to model are determined in the symbolization process.

Ideally, the learning model and the symbolization process should cooperate. First, the symbolization process provides symbolized instances to the learning model. Then, the learning model

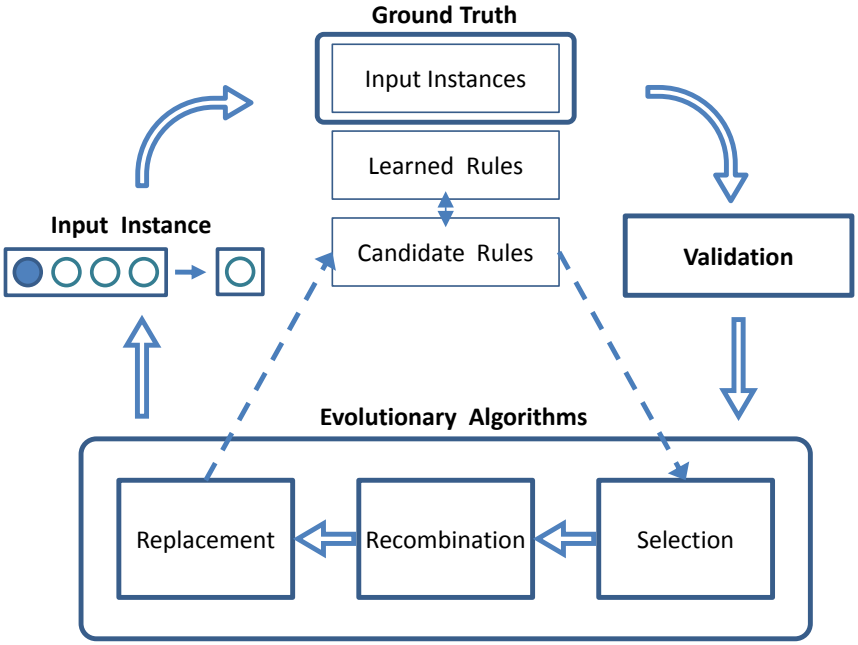


Figure 9: The learning procedure and the block diagram of the model.

learns from these instances and provides knowledge back to the symbolization process. Last, the knowledge can guide and adjust the symbolization process. These procedures can proceed cyclically. However, it complicates the learning problem. As the first attempt, this thesis focuses on the second procedure in the cycle.

2.2.2 The Learning Procedure and Block Diagram

The purpose of the proposed learning model is to do inductive theorem proving. Particularly, the model learns general rule representations from input instances. The model takes symbolized instances as inputs. These instances are rule representations, which express the casual relations between propositions. A rule representation is of the form:

$$[\text{Proposition}] \rightarrow [\text{Proposition}]$$

, where propositions can simply be concepts which are symbols not representing relationships, or propositions describing relationships between symbols. The learning model tries to learn general rule representations which satisfy all input instances.

The learning procedure runs cyclically. In each run an input instance is given to the model. An input instance is a rule representation which dose not contain any `relation` ,and its consequent proposition does not contain any symbol describing relationships. In other words, an input instances is a ground rule representation whose consequent proposition is composed of only concept representations. This input instance is stored in a knowledge base, which also contains the learned rules and candidate rules which might or might not be consistent with input instances. The evolutionary algorithm uses the candidate rules as its population of candidate solutions. After a generation (including selection, recombination, and replacement), the resulted candidate rules are stored back into the knowledge base. Those candidate rules which have been satisfied by all input instances in the knowledge base are considered consistent to the knowledge base and stored with other learned rules. Figure 9 shows the learning procedure and the block diagram of the learning model.

2.3 Provided Prior Knowledge

This learning model is designed to test the learning ability of a model presuming the three cognitive hypotheses. In order to investigate the ability of the model, the prior knowledge should be carefully

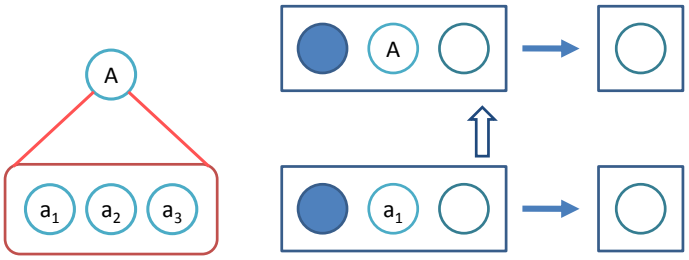


Figure 10: An example of the inductive process used in the model.

provided because prior knowledge has significant influence on the ability of a learning model. It needs to be sure that the knowledge learned by the model is actually learned because of the learning algorithm itself, not because of the prior knowledge. Therefore in the thesis, only bare-bone prior knowledge is provided to the learning model.

The followings are the prior knowledge provided in the thesis:

1. The domains of symbols.
2. The equality of symbols.
3. The hierarchy of symbols.
4. The analogical information of the problem (not necessary).
5. The memory usage of each representation.

The first four listed prior knowledge are based on representationalism. The last listed prior knowledge is needed in order to compute the fitness function used in the proposed model. Knowledge other than those listed above is not provided to the thesis. Therefore, the model only has bare-bone knowledge about the problem.

2.4 Implementation Detail

This section describes the implementation of the learning model in detail. In short, the learning model is an inductive theorem proving system, and the input instances are the ground knowledge. The inductive process is discussed in Section 2.4.1. Recall the procedure of the learning process shown in Section 2.2.2. Every iteration an instance is provided. Because this instance may change the ground knowledge, all learned rules and candidate rules need to be validated again. The validating process is time-consuming, and the architecture of the knowledge base is designed to speed up the process. The knowledge base architecture is discussed in Section 2.4.2. After the re-validating process, the learning model tries to generalize knowledge from knowledge base. The evolutionary algorithm is in charge of the inductive process. The fitness function used in the model is discussed in Section 2.4.3. The implementations of selection, recombination, and replacement are described in Section 2.4.4. Finally, the difficulties encountered in the implementation are discussed in Section 2.4.5. Note that this thesis focuses on only deterministic knowledge, and all input instances are correct.

2.4.1 Induction and Validation

The learning model generalizes rules from input instances. The induction sources from the hierarchy of symbols. Recall that in a symbol hierarchy, a (`relation`) symbol represents all symbols directly below it. Given the hierarchy and a rule representation, the rule can be generalized by replacing symbols in the rule with higher level symbols in the hierarchy (, called upgrading the rule). The resulted rule contains the same amount of knowledge as the conjunction of rules each containing a symbol at the original level. Therefore, the resulted rule is more general than the original rule. Figure 10 shows an example of the inductive process.

Some `relations` in a rule must represent the same ground symbol concurrently. For example, suppose [R] is a `relation` representing symbol [A] and [B]. Seeing two rules, [C][A] $\rightarrow$ [A] and [C][B] $\rightarrow$ [B] can result in an inductive rule [C][R] $\rightarrow$ [R] with the two [R] both representing the same symbol concurrently. Without specifying the identicalness of these two [R]s the inductive rule will

instead represent four ground rules, [C][A] → [A], [C][A] → [B], [C][B] → [A], and [C][B] → [B]. The model uses additional information attached to a rule representation to specify identical `relations` in the rule. Particularly, the model uses *identical links* to specify two identical `relations`. An identical link points from `relation` R_1 to `relation` R_2 confines R_1 representing the same symbol as R_2. In order to simplify the parse of rules, identical links point from the second `relation` to the first `relation`. Back to the example above, the inductive rule is expressed by [C][R] → [R] (2 = 1). The number in the parentheses represents the index (starting from 0) of the symbol in the rule.

Most of the time, replacing symbols results in rules which are inconsistent with the knowledge base. Therefore, the resulted rules need to be validated. By using the inductive proof technique described in Section 1.5, the entailment of the resulted rules from the knowledge base can be proved by showing that all substitutions of the resulted rule ranging over all ground symbols of the `relation` can be inferred from the knowledge base. In other words, the process to validate a rule checks whether every rule resulting from every substitution of ground symbols for every `relations` in the rule can be inferred from the knowledge base.

To validate an inductive rule the model checks whether the rule can be inferred by the knowledge base. However, the inference itself is challenging. In particular, the inference in the proposed model will be proved to be undecidable in Section 3.1.3. Therefore, to simplify the inductive process, all input instances are always remembered by the model. Keeping all input instances makes each inference process a series of table-lookups and substitutions. After constructing the syntax-trees of condition part and consequence part of the rule, starting from leafs of each tree, each subtree, which is a one-level proposition, is searched in the knowledge base and substituted for the searched consequence proposition. If some proposition can not be found in the knowledge base, it means that the proposition is unknown yet and the proposition is marked unknown and is stored with the to-validate rule. When finished and all propositions are known, the resulted condition and consequence propositions are compared to each other. If they are the same, the rule can be inferred and is consistent with the knowledge base. If not the same, the rule is known to be inconsistent. In this way, the inference process becomes decidable (, which will be proved in Section 3.1.3).

2.4.2 Knowledge Base Architecture

The knowledge base stores input instances, learned rules, and candidate rules. These instances and rules are stored separately. The knowledge base organizes these instances and rules to avoid duplicated storage of information. The knowledge base is designed to speed up the processes to retrieving and storing knowledge.

The knowledge base is composed of hash-maps. Symbols, propositions, and rules are stored in hash-maps and are identified by their own keys. It speeds up the retrieval and storage of knowledge. Any information about a symbol, proposition, or a rule is grouped and stored in the hash-map. This design avoids duplicated storage of information, and allows easy access and updates to all information. For example, a symbol stores the keys of all propositions containing the symbol. A proposition stores the keys of all rules containing the proposition. Remember that when a new instance comes, the candidate rules need to be re-validated. Having access to the propositions and further the rules containing a symbol narrows down the search space of the rules related to the instance. Using hash-maps has another advantage which is that everything can be expressed by keys. It reduces that amount of memory used by the learning model. The hash-map has amortized $O(1)$ time complexity to access and to insert data. Therefore, using hash-maps only has limited additional costs.

2.4.3 Fitness Function

Fitness functions are important for evolutionary algorithms. The first step to use evolutionary algorithms is to define the fitness function. In normal optimization problems the fitness functions are given in advance, and evolutionary algorithms or other optimization techniques are used to search for the optimal solution(s).

Unlike normal optimization problems, the problem focused on by the thesis is an inductive theorem proving problem. The search space of an inductive theorem proving problem is usually unbounded, and the goal of an inductive theorem proving problem is to find useful theorems, where

the usefulness is defined by the fitness function. There exist no ultimate solutions to stop the searching. Therefore, what is important for inductive theorem proving problems is the sequence of found inducted theorems. A good inductive learner would be expected to learn useful information as quick as possible. As a result, the fitness function of a inductive theorem proving problem can be thought as a guide to the search for useful theorems.

This proposed model adopts the minimum description length principle (21) to design the fitness function which guides the search for inductive theorems. The minimum description length principle is a formalization of Occam's Razor. Occam's Razor is a principle urging that one to select hypothesis which makes the fewest assumptions and thereby offers the simplest explanation. Given that any set of data can be represented by a string of symbols from a finite alphabet, the minimum description length principle states that the best hypothesis for a given set of data is the one that leads to the best compression of the data, where the compressions are choosing strings of symbols from a finite alphabet to represent the set of data. The minimum description length principle can be used in information theory and learning theory, and it is also perfectly suited for the purpose of the model. The instances are expressed by symbols, and the model tries to learn inductive rule representations which explains ground instances. Using the minimum description length principle as the fitness function provides the model the ability to learn simpler rules.

Using the minimum description length principle requires a method to describe (encode) datasets and hypotheses, and a measure of the descriptions. In the model, the datasets are the ground instances which are described by symbols. This model uses the following method to measure the descriptions:

1. The description length of a symbol, s, is

$$M_S(s) = \begin{cases} log_2(T_G) & , s \text{ is a ground symbol} \\ log_2(T_R) & , s \text{ is a } \texttt{relation} \text{ symbol} \end{cases}$$

   , where T_G is the number of different ground symbols in the knowledge base, and T_R is the number of different `relation` symbols in the knowledge base.

2. The description length of an identical link, i, is

$$M_E(e) = 2log_2(\ell) - log_2(T_R)$$

   , where ℓ is the length of the rule containing i.

3. The description length of a rule, r, is

$$M_R(r) = \sum_{s \in r}(M_S(s)) + \sum_{e \in r}(M_E(e))$$

   , where s is the symbols in the rule, e is the identical links.

The model uses binary encoding. It uses $log_2(T)$ bits to represent each of T different symbols. Representing an identical link in a rule with length ℓ takes two descriptions for the starting and ending symbols, and the original description length of one of the two symbols can be saved. The description length of a rule takes account of all symbols and identical links in the rule. These calculations of description lengths actually models the memory required to remember these symbols and rules.

The fitness function of a rule is defined as the description length of the rule and the ground instances of the rule. Suppose that r is a rule, G is the set of ground instances of r which are consistent with r, and C is the set of ground instances of r which are inconsistent with r. The fitness of r, $F(r)$, is defined as:

$$F(r) = M_R(r) - \sum_{i \in G}(M_R(i)) + P\sum_{i \in C}(M_R(i))$$

, where P is a constant named punishment factor. The fitness function calculates the amount of memory needed to remember the rule r. The evolutionary algorithms minimize the fitness function.

Under the definition of the fitness function, a useful rule consumes less memory, or even can save memory (by having negative fitness value).

The definition of the fitness has some characteristics. First, when two rules with the same description length compare to each other, the one has more consistent ground instances is preferred. Second, when two rules have the same number of consistent and inconsistent ground instances, the one with less description length is preferred. Last, when two rules with the same description length have the same number of consistent ground instances, the one with less inconsistent ground instances is preferred. There is no fitness gap between consistent and inconsistent rules. It is important for using evolutionary algorithms to search consistent rules. During the search process, inconsistent rules might be frequently generated. These inconsistent rules might just need a little adjustments to become consistent. If there exist wide gaps between consistent and inconsistent rules, these inconsistent rules are likely to be discarded, and it makes searching for consistent rules like looking for a needle in a haystack. The fitness function therefore guides the search process.

2.4.4 Evolutionary Algorithms

The model uses evolutionary algorithms to simulate the variation-selection process described by the neural Darwinism. Rule representations in the model are considered as neuronal groups of the neuronal group selection theorem. Evolutionary algorithms are composed of three main processes: selection, recombination, and replacement.

The selection process replicates the rules with lower fitness. These rules are considered more useful and are replicated. The fitness function influences the process through the components in the candidate population. After selection, rules with lower fitness values will share expectedly bigger proportion of the candidate population. The selection process is implemented with tournament selection.

The recombination process recombines the after-selection candidate population and generates new rule representations. This process can be considered as the exploration of the space of all possible rules. Two processes are included in the recombination. The first process is crossover. Recall that the propositions are pre-ordered sequences of symbols and can be expressed by syntax trees. In order to maintain the syntactic correctness of resulted rules, two rules crossovers by exchanging subtrees. The domains of the symbols are required to be the same in order to maintain the syntactic structures. In the current implementation of the model, one subtree of each rule is chosen to exchange. The second process is mutation. During mutation each symbol has some probability to be replaced with a symbol randomly chosen from the same symbol hierarchy.

The replacement process decides whether the newly generated rules are kept in the candidate population or discarded. The size of the candidate population of the proposed model is limited. If the population is full, to remember a new rule, one rule in the population must be discarded. In order to learn a variety of rules the population the diversity of the population should be maintained. Therefore, these generated rules (offspring) are compared with the most similar rule in a randomly selected group of rules in the population (parent). If the offspring has lower fitness it will be kept in the population and the parent is discarded.

The similarity of representations is measured by distance between representations. Because the model does not have the prior knowledge about the distance between any two rule representations, the distance is measured by equality of symbols and the hierarchy of symbols.

1. The distance between symbol s_A and s_B:

$$D_S(s_A, s_B) = \begin{cases} 0 & , s_A == s_B \\ 0.5 & , s_A \text{ and } s_B \text{ are in the same hierarchy} \\ 1 & , \text{otherwise} \end{cases}$$

2. The distance between proposition p_A and p_B:

$$D_P(p_A, p_B) = \sum_{n \in N} \left({b_{max}}^{-d(n)} D_S\left(s_{A_n}, s_{B_n}\right) \right)$$

, where N is the set of nodes in the tree, $d(n)$ is the depth of the node, b_{max} is the max branching factor of syntax tree in the knowledge base.

3. The distance between rule r_A and r_B:

$$D_R(r_A, r_B) = D_P\left(c\left(r_A\right), c\left(r_B\right)\right)$$

, where $c(r)$ is the conditional proposition of the rule r.

The distance is defined based on the prior knowledge mentioned in Section 2.3. Because the prior knowledge provided to the model is limited, the ability of the distance is confined. For example, the distances between any different symbols of two different symbol hierarchies are the same, and the distances between any different symbols of one symbol hierarchy are the same. The definition of $D_P(p_A, p_B)$ emphasizes the nodes with less depth and allows quick terminations of the calculations. The working hypothesis is that if the symbols representing the relationship of two propositions are different, the two representations describes different things and can be considered totally different no matter what the rest of the symbols are. The distance is proportional to ${b_{max}}^{-d(\ell)}$ so that the distance caused by all child nodes does not exceed the distance caused by two different parent nodes. Two propositions with equal parent symbols and completely different child nodes are more similar that two propositions with different root symbols but identical child nodes. Therefore, the calculation starts from the roots of the two syntax trees of p_A and p_B. If the roots of each subtree are equal, the calculation continues. Otherwise, the calculation terminates.

2.4.5 Encountered Problems

The following problems are encountered when implementing the model. First is related to crossover. Crossover exchanges contents of two rules. When exchanging a subtree with a symbol, crossover increases the length of one rule and therefore usually results in complex rules. If the crossover sites are uniformly chosen in rules, it has high probability that leaf nodes are chosen. Considering balanced binary trees in which the difference between the number of leaves and parents is equal to one. It means that a leaf node would be chosen with probability = 0.5. In order to exchange higher-level subtrees with higher probabilities, in the model the probabilities for all levels to be chosen ares the same. For example, suppose there are n levels in a proposition. The probability for choosing a node in level i is $\frac{1}{n}$. However, in this way the resulted propositions will have longer length expectedly. To prevent longer rules occupying the population, the crossover probability should not be too high.

Another problem is also related to crossover. Because there are various rules in the after-selection population, if two rules are chosen uniformly in the after-selection population, two rules with syntactically different structure will have the great probability to be chosen. Crossover two syntactically different rules might end up ruining both rules. Therefore when choosing crossover candidates the model chooses two similar rules in a randomly selected group of rules.

The next problem is related to mutation. Mutation replaces symbols with other symbols. If the mutation is frequently preformed, the candidate rules in the population are like randomly generated. The opportunity for replicating rules with lower fitness decreases when the mutation probability is high. Therefore, the mutation probability should be low.

Another problem is related to maintaining the diversity of the population. The population can lose its diversity. It happens when the population is filled up with low-fitness (useful) rules. It might prevent newly generated rules entering the population because new generated rules usually have higher fitness values. Therefore the population sometimes needs to be reinitialized. However, less-fitness rules can provides expectedly low-fitness propositions to the recombination process. Therefore, how often to reinitialize the population is a trade-off between the diversity of the population and the supply of the useful components in the population. In the model, a threshold is set to trigger the reinitialization process. Once the number of the consistent rules exceeds the threshold, the population is reinitialized by input instances and learned rules.

2.5 Compared to Existing Methods

The proposed learning model is an inductive theorem proving system based on cognitive hypotheses about the human mind. This section compares the proposed learning model to existing cognitive models and inductive theorem proving methods.

Soar (17) is a cognitive model focusing on simulating the architecture of memory of the mind. The memory architecture of Soar is composed of two main parts which are the long-term memories and the working memories. Soar's learning methods tightly cooperates with its memory architecture. It has three kinds of learning methods. The first is chunking, which is the main learning method used by Soar. Chucking is a deductive learning method. After a goal state is reached, Soar deduces new production rules from all the knowledge retrieved from the long term memories. When facing the same problem, these resulted rules help Soar skipping some deductive processes and increase its performance. The second method is reinforcement learning (22). Reinforcement learning uses feedback from the environment, or what is often called a *reward*, to adjust the memory content of Soar. The reward can come from the "body" in which Soar is embedded, or it can be generated when a goal is achieved. Soar use reinforcement learning to learn rules that test the appropriate features of the states and operators and to learn the appropriate expected rewards for each rule. The third method is semantic learning. Semantic learning learns from the co-occurrence of structures in working memory. The resulted knowledge, or a semantic memory, represents the relationships between a few elements in the working memory. There are three differences between the proposed learning model and Soar. First, the proposed learning model focuses on the learning procedure of the mind, and Soar focuses on the memory architecture of the mind. Second, the proposed model uses inductive learning methods, and Soar uses mostly deductive learning methods. Third, the proposed model does generalization by upgrading a symbols and giving analogical information, and Soar uses features characterizing states and operations.

ACT-R (2) is a cognitive framework which allows researchers to create models (or programs) that incorporating the ACT-R's view of cognition or to add their own assumptions about the particular task. Researches on ACT-R mainly focuses on the implementation level; they research on how a particular portion of the mind is physically implemented. Comparing with ACT-R, the proposed model focuses on the algorithmic level. The model focuses on the learning algorithm based on representations, functions, and evolutionary processes and does not simulate the physical implementations of them in the mind.

The proposed model is different from most of the existing inductive theorem proving systems (16; **?**; **?**; **?**) on two aspects. First, these existing systems are based on logical framework (first-order or higher-order logic). By using logical expressions of knowledge, these systems are embedded with the prior knowledge about logics and the dependencies between different knowledge. The proposed model uses symbols to represent knowledge, by which the prior knowledge about logics is not needed by the model and all knowledge are considered independent if not provided with the analogical information. Second, the goal of the proposed model is different from the goal of these existing systems. The proposed model wants to see whether only based on the three cognitive hypotheses can still learn inductive theorems, and the goal of these existing models is to create a practical inductive theorem proving system. Therefore, instead of using mathematical proving methods like the existing systems, the proposed model uses evolutionary algorithms which is founded on neural Darwinism as its inductive algorithm.

2.6 Summary

This chapter describes the proposed learning model. It first describes how the three hypotheses about the human mind form a system. Representationalism forms the knowledge representation; functionalism forms the combination of knowledge; neural Darwinism forms the inductive process. Knowledge in the model is expressed by symbols. These symbols are syntactically structured and form syntax trees. Cascading syntax trees combines knowledge. Inductive process in the model is modeled by evolutionary algorithms. This chapter then describes the model in detail. Knowledge base is hashed to increase the performance. The fitness function utilizes the idea of minimum description length and measures the amount of memory used when remembering a rule representation. The implementation of each stage of evolutionary processes is then described, including the inductive process, the validation process, the design of the knowledge base, the fitness function, the implementation of evolutionary algorithms, and some encountered problems. At the end of this chapter, the proposed learning model is compared to two existing cognitive models and existing inductive theorem proving systems. The proposed model differs from Soar on its learning algorithm. The proposed model differs from ACT-R on its implementation level. The

proposed model differs from most existing inductive theorem proving systems on its knowledge representation and its goal. The description of the proposed model is now finished. The next chapter will analyze and perform experiments on the proposed model to test the learning ability of the model.

3 Analyses and Experiments

This chapter analyzes the proposed learning model, and performs three experiments to test its learning ability. The expressive ability, soundness, completeness, and decidability of the model are analyzed. In these experiments, the problem definition and the encoding will first be described. Results and some observations will then be provided.

Section 3.1 analyzes the model. Section 3.2 uses the proposed model to learn general rules from a one-dimensional minesweeper game. Section 3.3 provides the analogical information about the one-dimensional minesweeper game and shows how analogy affects the ability of the model. Section 3.4 applies the proposed model on the mathematical ring problem which allows rules with unseen syntactic structure to be generated.

3.1 Analyses

As described in Section 2, the proposed model is an inductive theorem proving system. This section analyzes the expressive ability, soundness, completeness, and decidability of the model. The expressive ability confines the kinds of problems the model can be adopted. Soundness tells whether those learned theorems are consistent in the knowledge base. Completeness tells whether all theorems which can be expressed by the model can be learned. Decidability tells whether some properties of the model can be decided by any algorithms. This section will analyze the above properties in order.

3.1.1 Expressive Ability

The proposed model can be formally defined as $M = (S, I, R)$, where S is the set of concept representations or the symbols, I is the set of proposition representations of input ground instances, and R is the set of rule representations of the form $\alpha \to \beta$ where α and β are proposition representations or strings of symbols in S and α is not the empty string. Given a model M, the one step derivation relation $\Rightarrow$ is defined by $x\alpha y \Rightarrow x\beta y$ for all strings $x, y \in S^*$ if and only if there is a rule representation $\alpha \to \beta \in M$. The language of the model is the set of proposition representations or strings which can be derived from ground propositions I with a sequence of one step derivation relations $\Rightarrow$.

The language of the proposed model characterizes the recursively enumerable languages. It means that for every model there exists a Turing machine which can simulate the model and vice-versa. Because unrestricted grammars characterizes the recursively enumerable languages, this can be proved by showing that the model can be simulated by unrestricted grammars and vise-versa. An unrestricted grammar is a formal grammar $G = (\sigma, P, S')$, where σ is a set of symbols, P is a set of production rules of the form $\alpha \to \beta$ where α and β are strings of symbols in σ and α is not the empty string, and $S' \in \sigma$ is a specially designated start symbol.

A proposed model $M = (S, I, R)$ can be simulated by an unrestricted grammar $G = (\sigma, P, S')$ if $S = \sigma$, and for every ground proposition $i \in I$ there is a corresponding production rule $p = S' \to i \in P$, and for every rule representation $r = \alpha \to \beta \in R$ there is a corresponding production rule $p = \alpha \to \beta \in P$, and P contains nothing else. The one step derivation from a ground proposition i can be simulated by first replacing S' with the production rule $S' \to i$ and then applying the corresponding production rule in P. Every derivation after then can be simulated by applying corresponding production rule in P. Therefore for any proposed model there exists an unrestricted grammar simulating it. The reverse construction is the same. Any unrestricted grammar $G = (\sigma, P, S')$ can be simulated by a proposed model $M = (S, I, R)$ by setting $S = \sigma \cup S'$, $I = \{S'\}$, and $R = P$. Every derivation after then can be simulated by applying corresponding rule representation in R. Therefore for any unrestricted grammar there exists a proposed model which can simulate it.

3.1.2 Soundness and Completeness

The definition of soundness is that every learned theorem is consistent with the knowledge base. The definition of completeness is that every consistent theorem which can be expressed by the model can be learned by the model. Most inductive theorem proving models are sound but not complete (11).

The proposed model is sound, because every learned theorem is validated and checked whether it is consistent with every input instances in the knowledge base. The proposed model is not complete. The incompleteness is caused by the characteristics of evolutionary algorithms. The evolutionary algorithms are guided by the fitness function. Every generation the fitness function determines which rules to replicate and as a consequence decides the direction to explore the search space. During the selection-replacement processes, some symbols or propositions might always be considered worse by the fitness function and never appear in the population. As a result, rules composed of these *extincted* symbols or propositions might never be learned.

3.1.3 Decidability

Two questions are usually asked about an inductive theorem proving problem. The first is that given a knowledge base and a ground theorem whether or not the theorem is consistent with the knowledge base. The second is that given a knowledge base and an inductive theorem whether or not the theorem is inductively consistent with the knowledge base. For most inductive theorem proving systems, the first question is undecidable but semi-decidable, and the second question is neither decidable nor semi-decidable (11). Under the assumptions used in the thesis–the consequent propositions of input instances are concepts–both of the questions are decidable.

For the proposed model, determining whether a rule is consistent with the knowledge base is decidable. Because the proposed model remembers all ground knowledge, and the syntactic information is available to the model, any syntactically correct ground proposition representations can be simplified from leaves in the syntax trees and finally result in one proposition without any relationship symbols. The two simplified propositions of a ground rule can be compared. If they are the same the rule is considered consistent with the knowledge base. There are finite symbols in a rule representation, and every simplification decreases the number of symbols. Therefore, finite number of simplifications are needed, so the process is decidable. Particularly, if not every ground input instances are not remembered by the model, the problem becomes undecidable. It loses the properties that for every proposition with more than one symbols there must be a simplification which can decrease the number of the proposition. Because the model is actually an universal Turing machine, determining a rule is consistent is the same as determining whether the Turing machine accepts the rule, which is undecidable.

The second question is also decidable for the proposed model. Because any inductive theorem in the model can be downgrade to finite number of ground rule representations and validating a ground rule is decidable (shown by the first question), checking whether an inductive theorem is consistent is therefore decidable.

3.2 Experiment 1: Minesweeper

In this experiment, the proposed model is applied to a one-dimensional minesweeper game. The proposed model is expected to learn general rules about how the bombs are deployed. In the following subsections, the game will first be described, and then the encoding of the game used in the experiment is shown. The learning results will be illustrated at the end of the section.

3.2.1 Problem Description

The game, one-dimensional minesweeper, is simplified from the original two-dimensional minesweeper game. There are a one-dimensional N-cell array, $c_0, c_1, ..., c_{N-1}$. At each cell, a bomb is deployed by random. B_i shows the number of bombs at the cell c_i, where $i = 0, ..., N-1$, and $B_i = 0, 1$. All $B_i, i = 0, ..., N-1$ are hidden from the player, in this case, the learning model. Each cell, c_i has an indicator, I_i, which is revealed to the player. An indicator shows the total number of bombs

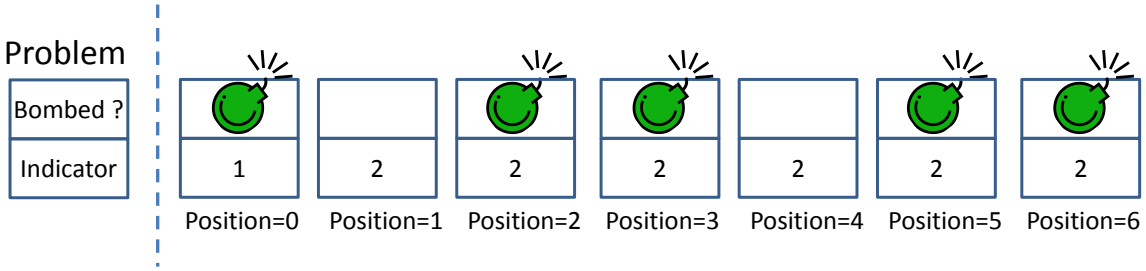


Figure 11: An example of the one-dimensional minesweeper game.

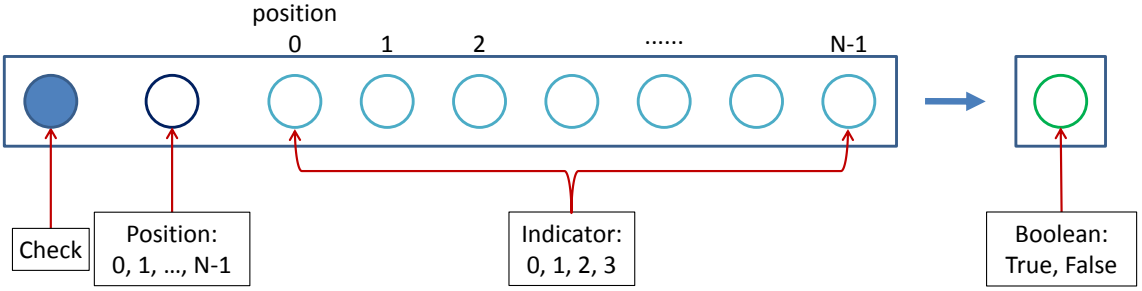


Figure 12: An example of the encoding of the one-dimensional minesweeper game.

around the cell.

$$I_i = \begin{cases} B_0 + B_1 & , i = 0 \\ B_{i-1} + B_i + B_{i+1} & , i = 1, ..., N-2 \\ B_{N-2} + B_{N-1} & , i = N-1 \end{cases} \quad (1)$$

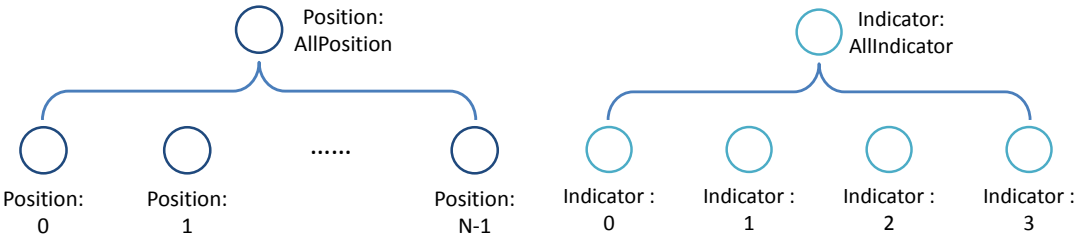
Given indicators of all cells, the player is asked to guess whether or not a bomb is deployed at each cell. Note that the game can have deterministic answer if $N \, mod \, 3 \neq 2$. Figure 11 shows an example of the game with $N = 7$.

3.2.2 Problem Encoding

The proposed model takes strings of symbols as inputs. Therefore the problem should be encoded by symbols before provided to the model. In the experiment, a straight-forward encoding is adopted.

Given the indicators of all cells, in order to represent whether or not a bomb is deployed at cell i, the instance is encoded in the form shown in Figure 12. Remind that an input instance is a rule representation, and the first symbol in a proposition representation indicates the relationship among the following symbols. In the conditional proposition of the instance, the first symbol, `Check`, represents that the proposition checks the following symbols. The second symbol in the proposition indicates the position to check the bomb. This symbol has the domain `Position`, and its value starts from 0 and ends at $N-1$. All of the following symbols in the proposition represent the indicators. For convenience, these symbols are arranged in ascending order of the positions of the cells. Each of them has the domain `Indicator`, and its value can be one in zero to three. In the consequent proposition of the instance, only one symbol represents whether there is a bomb at the position. This symbol has domain `Boolean`, and its value can be `True` or `False`. Note that this encoding does not attach any information other than the position to check, the indicators, and whether there is a bomb at the position. All symbols in the instances are considered independent by the model.

Two hierarchies of symbols are provided to the learning model. One has the domain `Position`, and the other has domain `Indicator`. Both of the hierarchies are shown in Figure 13. Only one `relation` exists in each hierarchy. The symbol, `AllPosition`, represents all possible positions, and the symbol, `AllIndicator`, represents all possible indicators.



(a) The hierarchy of symbols representing positions.

(b) The hierarchy of symbols representing indicators.

Figure 13: The hierarchy of symbols provided in the experiment.

3.2.3 Experiment Results

In the following experiments, the population size is fixed, the crossover probability = 0.25, the mutation probability = 0.5, no reinitialization is performed, and the group size used in the replacement is one tenth of the population size. In each round an instance is randomly generated and provided to the model, and the evolutionary algorithms runs one generation. The results are averages of ten different runs.

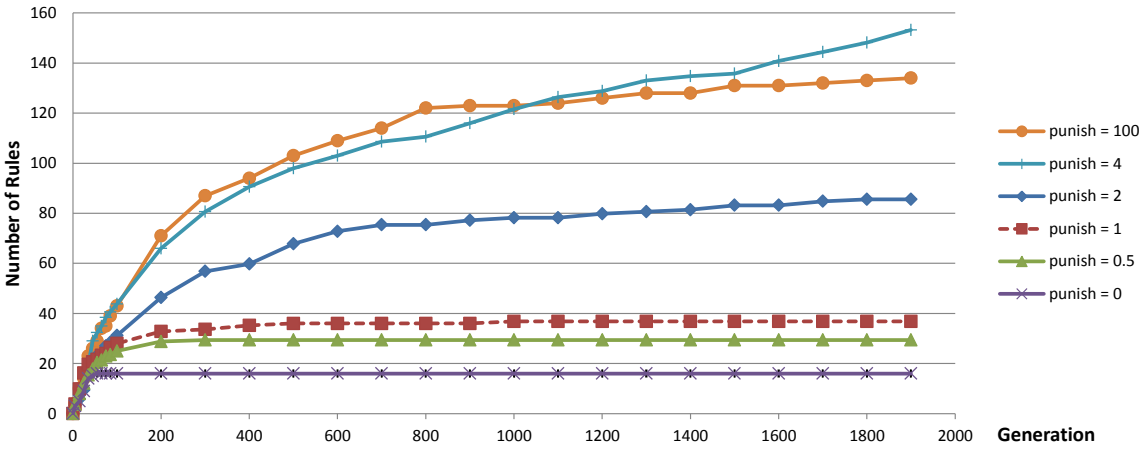
The first experiment focuses on the punishment factor in the fitness function. In this experiment, $N = 7$, population size = 25, and the punishment factor = $0, 0.5, 1, 2, 4, 100$. The results are shown in Figure 14. Figure 14a shows the number of learned consistent rules. Figure 14b shows the number of consistent rules in current population.

The results shows that when the punishment factor is less than or equal to one, the number of consistent rules in the population decreases as the generation increases. As a result, no new rules are learned after 200^{th} generation. The reason of the phenomenon is that when no or little punishment is practiced, conflicted rules usually have the lower(better) fitness values. Upgrading a rule does not decrease the number of consistent ground instances and usually will increase the number. However, most of the time upgrading a rule increases a lot of conflicted ground instances and makes the rule inconsistent. If the number of conflicted ground instances does not been considered in the fitness function, the population will prefer highly upgraded rules which usually are inconsistent.

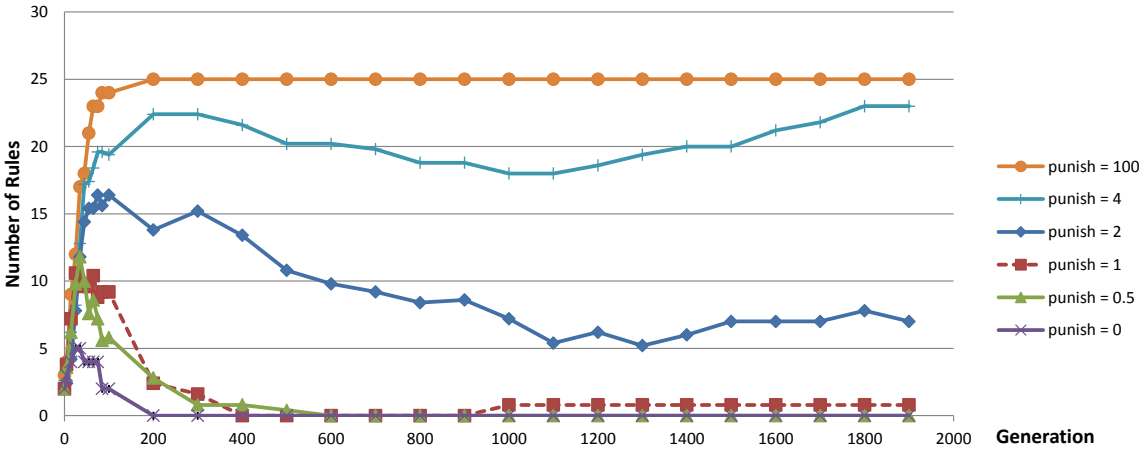
Also shown in Figure 14 is that when the punishment factor is too large, the number of learned rules will at last less than that of a moderate punishment factor. High punishment makes the population prefer the consistent rules. Therefore at first the number of learned rules will be higher. However as the times goes higher punishment will prevent the inconsistent rules appearing in population and further prevent new rules being learned. As a consequence, without loss of generality, in the following experiments the punishment factor is set to four.

The next experiment shows the effect of the fitness function. Figure 15 shows the number of rules seen by the model (Total Seen), rules seen by population (Population Seen), learned consistent rules, and consistent rules in current population with population size = $25, 100$. Note that the vertical axis is in log scale. This figure shows that a lot of rules in the offspring are filtered out by the fitness function and do not enter the population. This filtering effect makes the learning model can learn with a relatively small amount of population size. Rules which save more memory have lower fitness values and are kept in the population with higher probability. Rules which save less memory therefore are considered unimportant and are forgotten. This figure also shows that when the population size increases, the filtering effect does not prevent the learning model utilizing additional population size.

The next experiment shows the generalization ability of the model. Figure 16 shows the coverage of ground instances during the learning. An instance is covered if it belongs to at least one set of ground instances of a consistent rule in the population. The figure shows: the number of all covered ground instances (All Covered Instance) which might contain instances with input propositions which will never appear in the input ground instances, the number of input ground instances (Input Instance), the number of covered consistent ground instances (Covered Consistent Instance), and the number of covered input ground instances existing in the knowledge base (Covered Input



(a) The number of learned consistent rules.



(b) The number of consistent rules in current population.

Figure 14: This experiment focuses on the punishment factor in the fitness function.

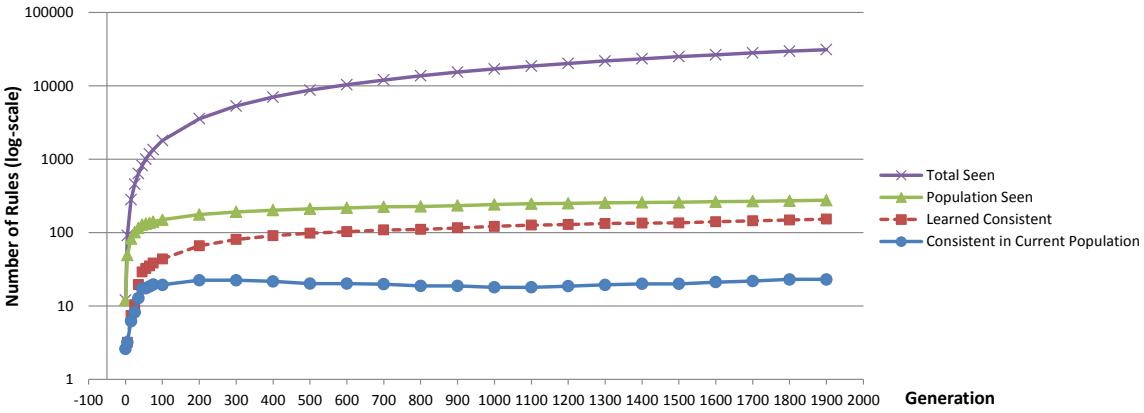
Instance). These results show that on average each consistent rule covers more than one ground instances. The result also show that these consistent rules can cover input instances which have not yet been seen by the model. This figure also shows that the increase of the population size will raise the generalization ability of the model. The above results shows that the proposed learning model has the ability to generalize from input instances in this problem.

Figure 17 shows the proportion of consistent rules in the all seen rules (, which are rules seen by both population and offspring) and all rules seen by the population with population size = 25 when all input instances are given at once at the beginning of the experiment. The dashed line shows the probability that consistent rules are found when randomly guessing. This figure shows that the probability for a consistent rule to be found in the offspring is gradually increasing. It means that the search direction of the model becomes more efficient as the generation goes.

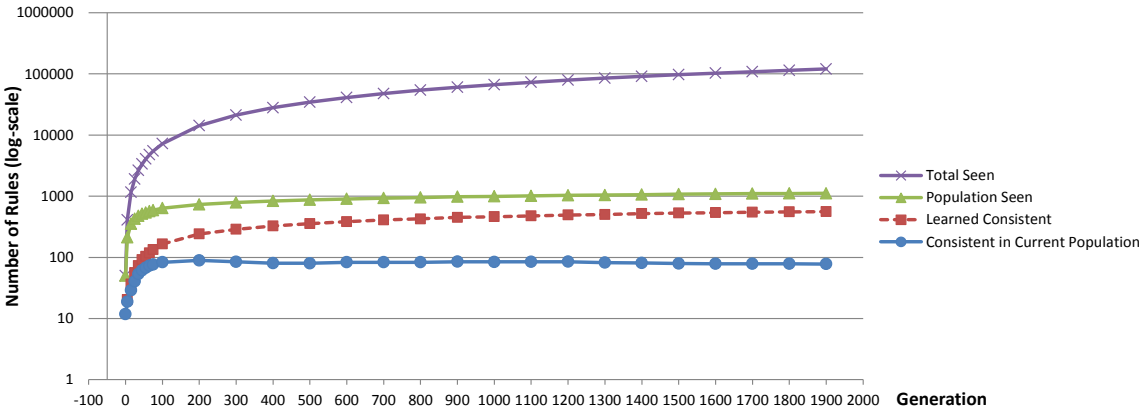
Figure 18 shows some rules learned by the proposed model with population size = 25. Because there are about 150 rules learned, only some of them can be shown here. Because the ground instances are provided one at a time, some of the rules might be considered consistent at one generation and become inconsistent later.

3.3 Experiment 2: Minesweeper plus Analogy

In Section 3.2, each of the instances and each of the symbols are considered independent. One rule can only be applied to ground propositions which are the same as the conditional proposition of the rule. This requirement makes the learning process inefficient when there might be some



(a) Population size = 25.



(b) Population size = 100.

Figure 15: This figure shows the number of rules seen by the model (Total Seen), the number of rules seen by the population (Population Seen), the number of learned consistent rules (Learned Consistent), and the number of consistent rules in current population (Consistent in Current Population) of each generation in the minesweeper problem.

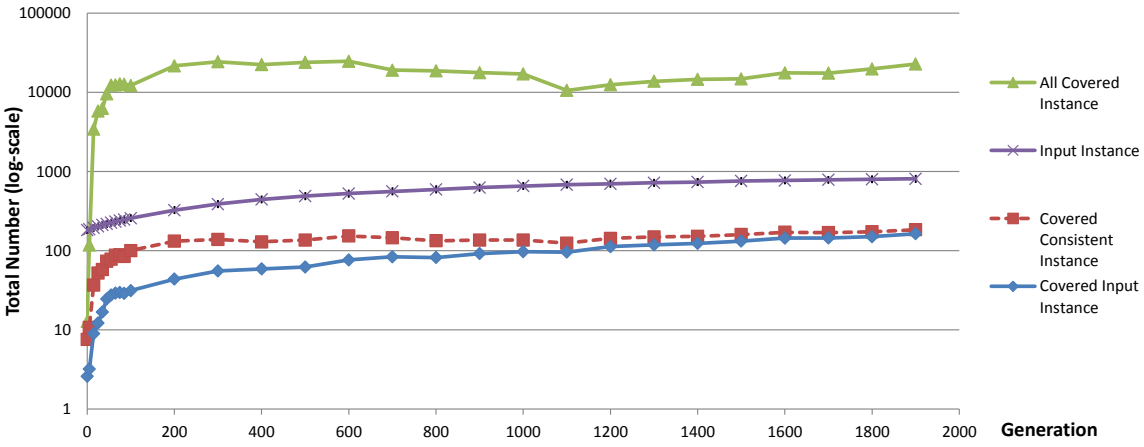
dependencies and similarities in the instances. Providing analogy representations gives the model the ability to connect seemed-independent instances. In this section, the experiment is still performed on the minesweeper game used in Section 3.2. The problem and the instance encoding are the same. Additionally, the analogical information is provided to the learning model.

3.3.1 The Analogy Information

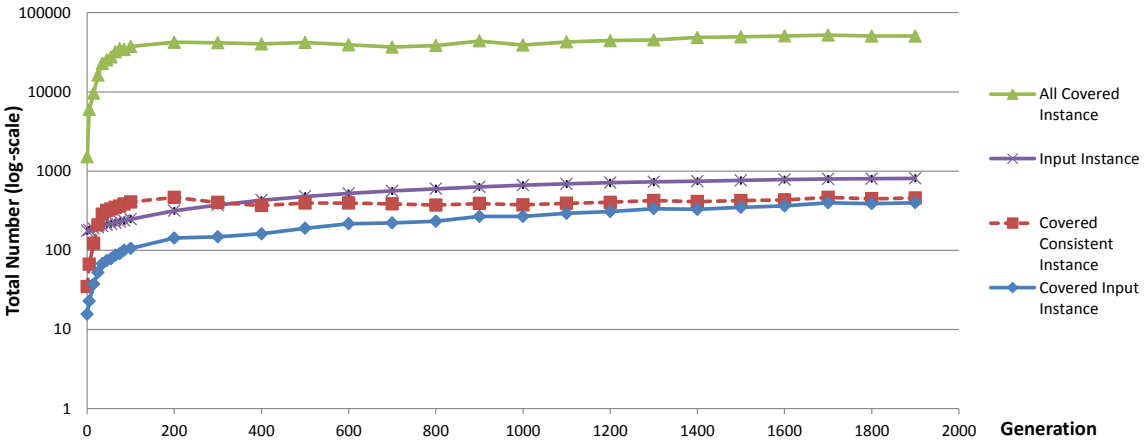
In the section, twenty types of analogies are defined based on the symbol `Check`. It means that when learning propositions leading by `Check`, there are total twenty one options (twenty analogies and no analogy) for the model to link instances. The model does not know how these analogies works and what these analogies are composed of. It can only test whether two instances satisfy a type of analogy.

Suppose a and b are propositions. Let $x(a)$ be the position of the cell to check in proposition a, $I(x)$ be the indicator of the cell at position x, and $a \approx b$ means that a and b are analogically equal. Ten types of the analogies are defined as follows:

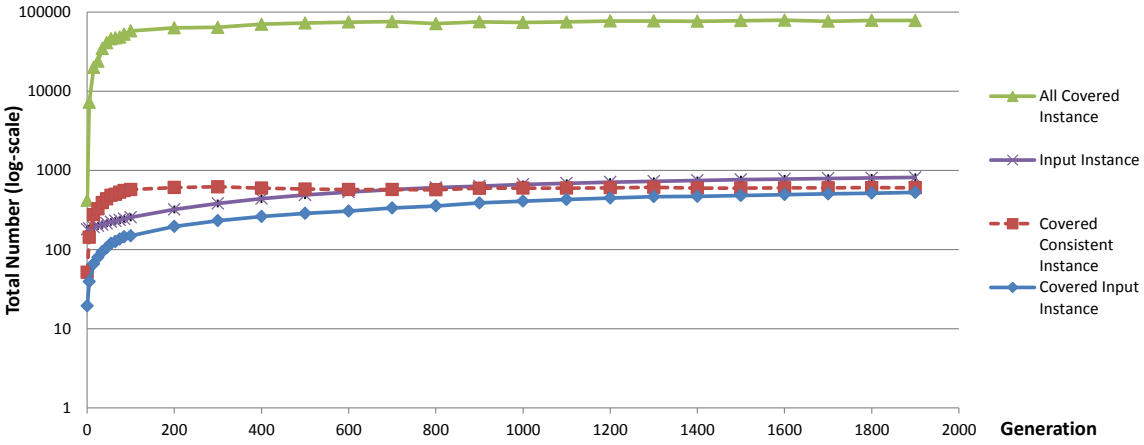
1. Current Cell:
   $a \approx b \Leftrightarrow I(x_a) = I(x_b)$, $x_a = x(a)$, $x_b = x(b)$
2. Left 1:
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i})$, $x_{a_i} = x(a) + i$, $x_{b_i} = x(b) + i$, $i = -1, 0$



(a) Population size = 25.



(b) Population size = 100.



(c) Population size = 200.

Figure 16: The figure shows the coverage of ground instances in the minesweeper problem. The number of all covered ground instances (All Covered Instance) which might contain instances with input propositions which will never appear in the ground instances, the number of input ground instances (Input Instance), the number of covered consistent ground instances (Covered Consistent Instance), and the number of covered input instances existing in the knowledge base (Covered Input Instance).

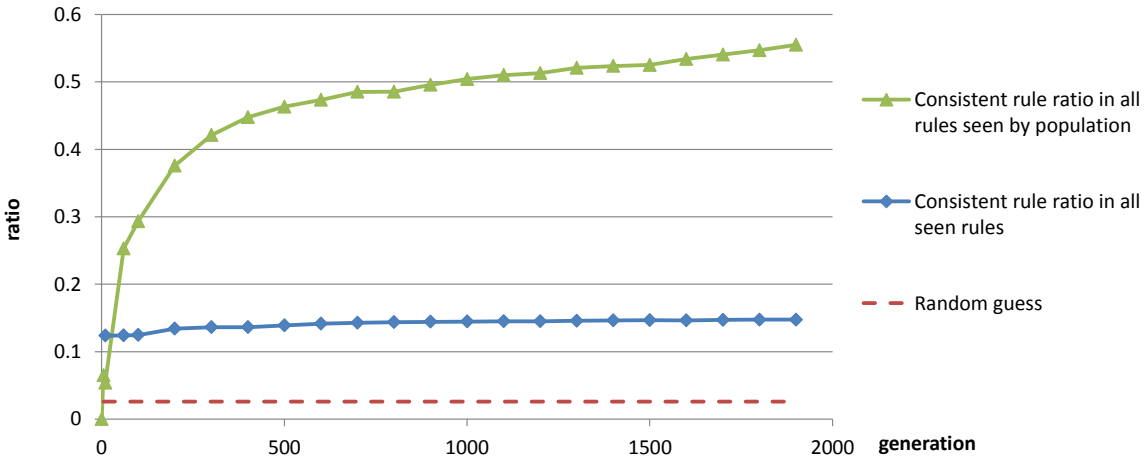


Figure 17: The proportion of consistent rules in the minesweeper experiment.

3. Right 1:
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = 0, 1$
4. Left 2:
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = -2, -1, 0$
5. Left 1 Right 1:
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = -1, 0, 1$
6. Right 2:
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = 0, 1, 2$
7. Left 3:
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = -3, -2, -1, 0$
8. Left 2 Right 1 :
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = -2, -1, 0, 1$
9. Left 1 Right 2 :
   $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = -1, 0, 1, 2$
10. Right 3 :
    $a \approx b \Leftrightarrow I(x_{a_i}) = I(x_{b_i}),\ x_{a_i} = x(a) + i,\ x_{b_i} = x(b) + i,\ i = 0, 1, 2, 3$

Another ten types of analogy are defined as those above but without checking whether $I(x(a)) = I(x(b))$. If any index is out of bound the two propositions are considered not analogically equal. Figure 19 shows some examples of the analogies. Note that some types of analogies can cause two analogically equal conditional propositions conflict with each other. Figure 20 shows an example in which the conditional propositions of the two instances are analogically equal (Left 1 Right 1) but their consequent propositions are not the same. The model has to learn the type of analogies to used when dealing with different conditional propositions. When two seemed analogically equal conditional propositions have different consequent propositions, the type of analogy is considered illegal to both of the conditional propositions.

3.3.2 Experiment Results

In the following experiments, $N = 7$, crossover probability $= 0.25$, mutation probability $= 0.5$, no reinitialization is performed, and the group size of replacement is one tenth of the population size. Each round an instance is randomly generated and provided to the model, and the evolutionary algorithms runs one generation. Each symbol `Check` is labeled with one type of analogy, including not using analogy. Two symbol `Check` labeled with different type of analogies are considered different symbols.

Three experiments are preformed to test the influence of analogy. The first experiment only uses the analogy "Current Cell". The second one uses all ten analogies described above. The third one uses these ten analogies without checking $I(x(a)) = I(x(b))$. The coverage of ground instances of these experiments are shown in Figure 21a, Figure 21b, and Figure 21c respectively.

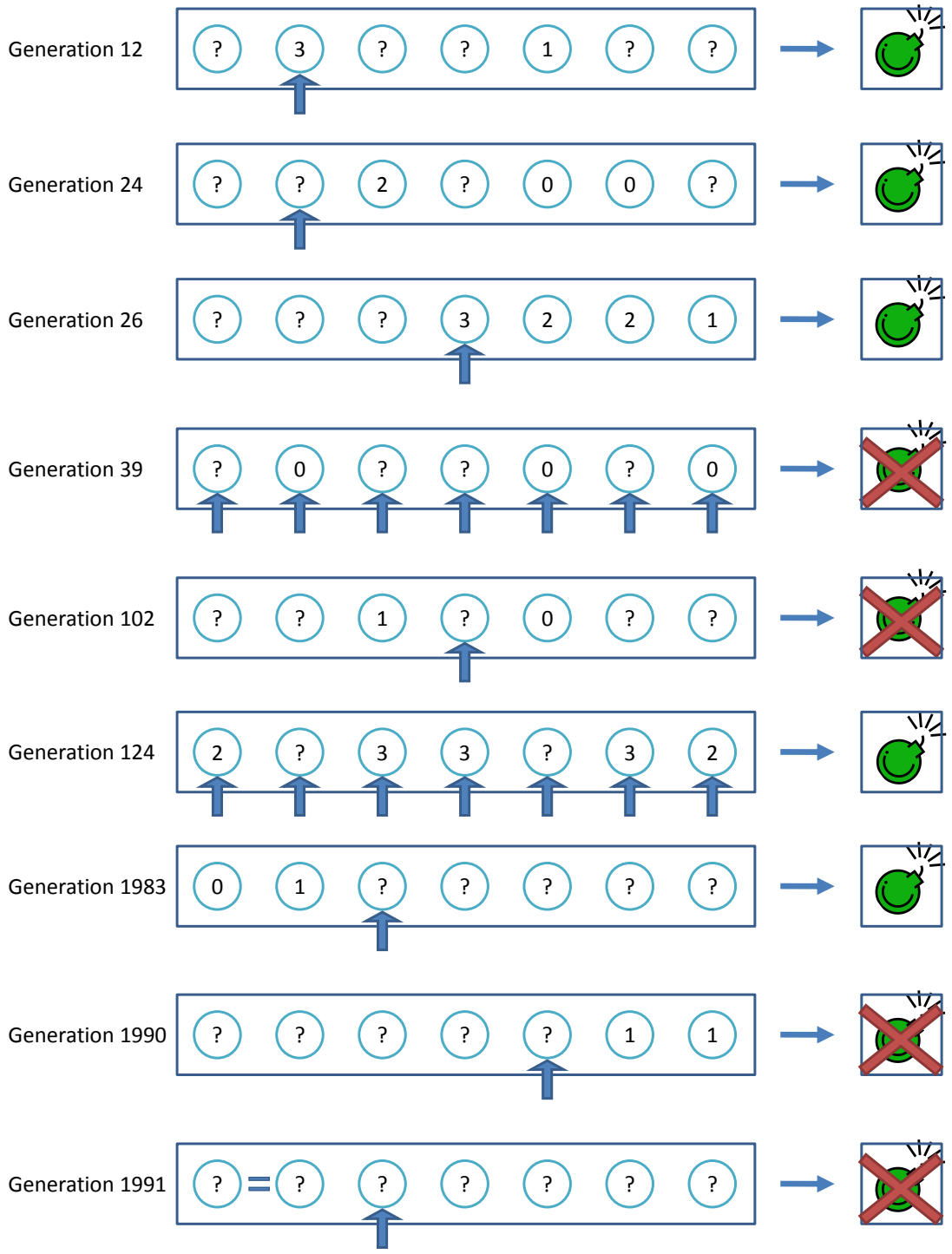


Figure 18: The learned rules of the minesweeper experiment with population size = 25. The arrows mean the position to determine bombed. The question mark means the indicator value of the position can be any value. The equal sign means the indicator values are the same.

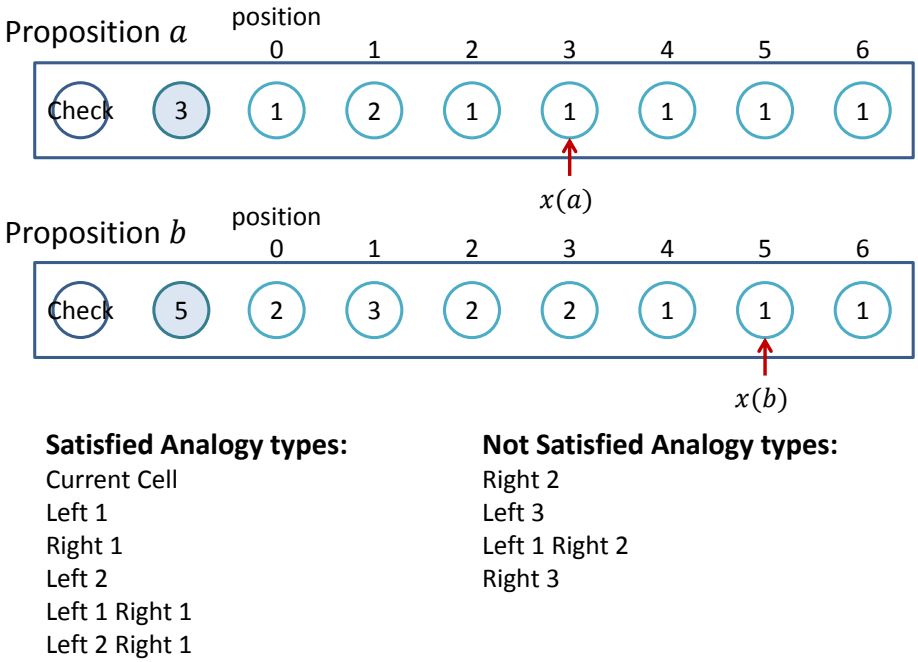


Figure 19: Some examples of the analogies.

Figure 21 shows that providing analogical information to the learning model increases its ability to generalize from instances. The lines labeled "Analogically Covered Instance" shows the number of input instances which are analogically covered only. All sub-figures in Figure 21 have positive number of analogically covered input instances. It means that the model benefits from all these three sets of analogies. Figure 21 also shows that the analogical information provided to the model can affect the performance of generalization. The number of total covered input instances when providing only the analogy "Current Cell" is much lower than that when providing ten types of analogy. In Figure 21b and 21c, the number of total covered input instances is much higher than the number of actually covered input instances and is close to the maximum number of input instances. Note that the fitness function used in the proposed model is not designed to maximize the number of input instances covered by the whole population. Some input instances might be covered by multiple rules. Nevertheless, the results still shows that the generalization ability of the proposed model is improved by providing analogical information.

3.4 Experiment 3: Mathematical Ring

In this section, the proposed model is applied to a mathematical ring problem. This problem is designed to test the learning ability of the model when combinations of propositions are allowed and rules with unseen syntactic structures can be generated. In Section 3.2 and 3.3, crossover does not have much effect because the syntactic structure confines the number of levels in propositions to one. The mathematical ring problem has the syntactic structure allowing combinations of propositions and can generate more complex syntactic structures.

A ring is defined on a set $\mathbb{Z}$ equipped with two binary operators '$+$': $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ and '$\cdot$': $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ (where $\times$ denotes the Cartesian product). To be qualified as a ring, the set and two operators, $< \mathbb{Z}, +, \cdot >$, must satisfy the following requirements known as the ring axioms:

Closure under '+' $\forall a, b \in \mathbb{Z}, a + b \in \mathbb{Z}$
Commutativity of '+' $\forall a, b \in \mathbb{Z}, a + b = b + a$
Associativity of '+' $\forall a, b, c \in \mathbb{Z}, (a + b) + c = a + (b + c)$
Existence of Identity of '+' $\forall a \in \mathbb{Z}, a + 0 = 0 + a = a$
Existence of Inverse Element of '+' $\forall a \in \mathbb{Z}, \exists b s.t. a + b = b + a = 0$
Closure under '·' $\forall a, b \in \mathbb{Z}, a \cdot b \in \mathbb{Z}$

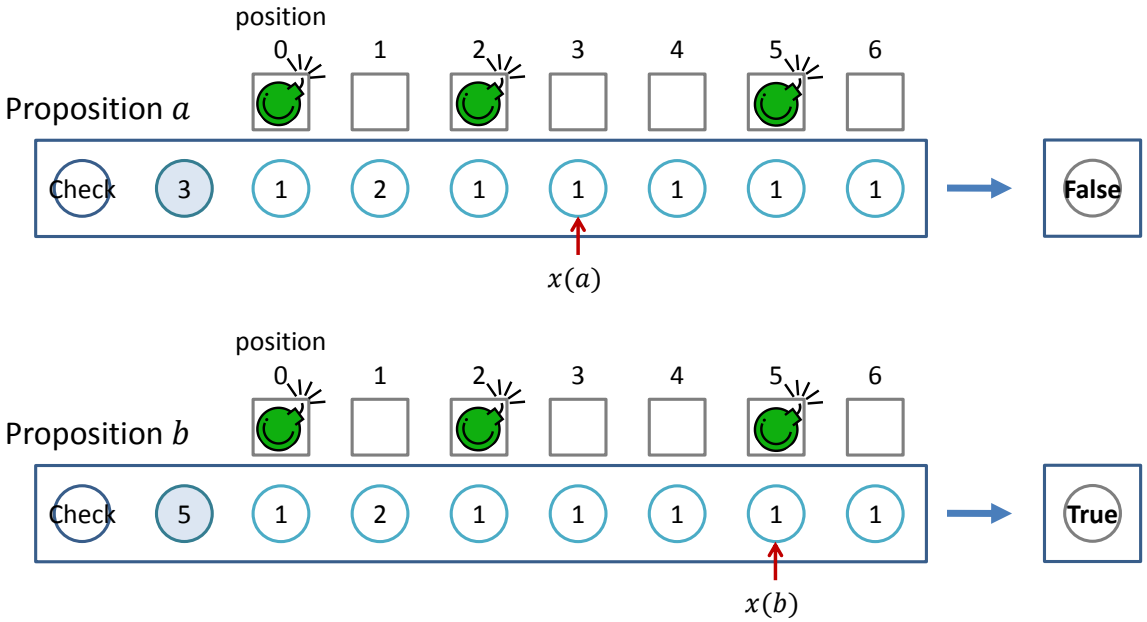


Figure 20: An example about the conflict of two analogically equal instances.

Associativity of '$\cdot$' $\forall a, b, c \in \mathbb{Z}, (a \cdot b) \cdot c = a \cdot (b \cdot c)$
Existence of Identity of '$\cdot$' $\forall a \in \mathbb{Z}, a \cdot 1 = 1 \cdot a = a$
Distributivity over '+' and '$\cdot$' $\forall a, b, c \in \mathbb{Z}, a \cdot (b + c) = a \cdot b + a \cdot c$

Given these nine axioms, any least residue system modulo N with normal definitions of addition and multiplication is a ring. In the experiment, the model is applied to learn the general rules in a least residue system modulo N.

3.4.1 Problem Description and Encoding

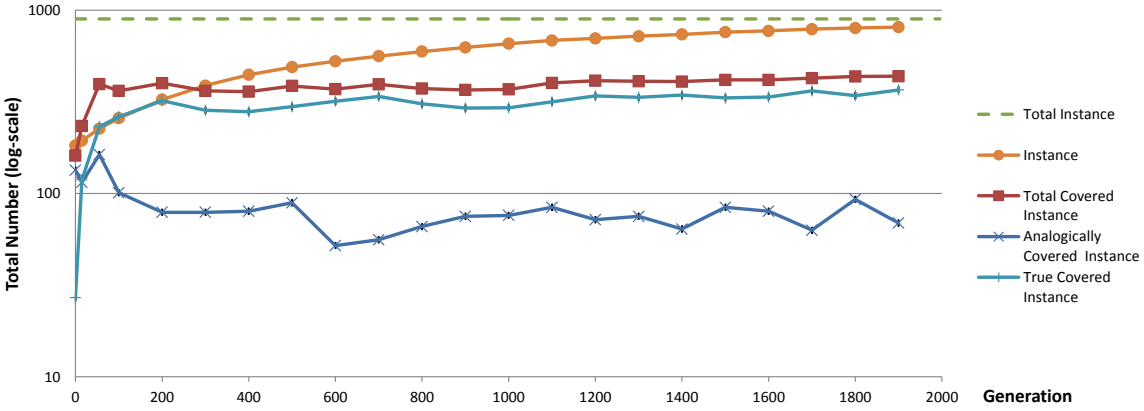
The problem is defined on a least residue system modulo N, which contains integer $0, 1, ..., N-1$. The operator '+' is defined as the integer addition operator with modulo N. The operator '$\cdot$' is defined as the integer multiplication operator with modulo N. Note that although the definition of the ring does not require the commutativity of '$\cdot$' the multiplication of integer does follow commutativity.

Every instance expresses a calculation of adding or multiplying two numbers. The conditional proposition contains the operator and the two numbers in pre-order, and the consequent proposition contains the result. There are two types of symbol; the first is `Operator`, and the second is `Number`. The values of `Operator` symbols can be `Add` or `Multiply`. The values of `Number` symbols can be one in the set $\{0, 1, ..., N-1\}$. Figure 22 shows an example of an instance expressing $3 + 4 = 7$.

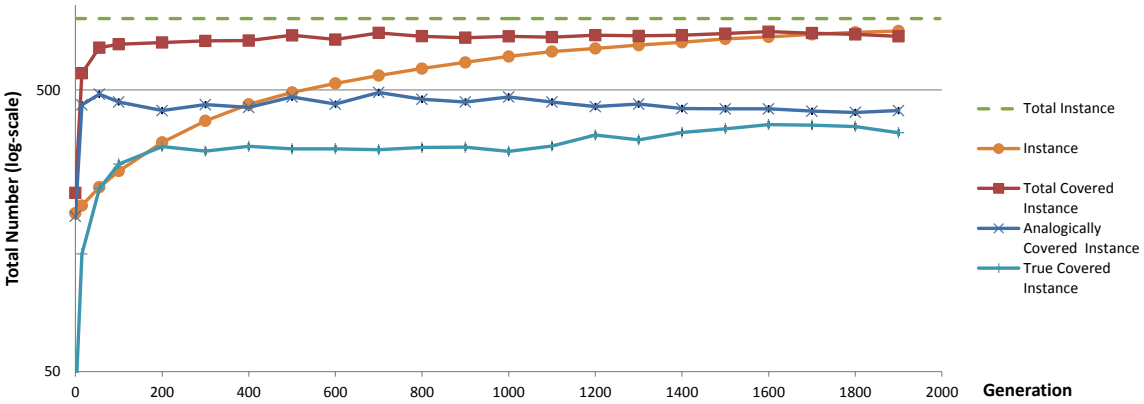
3.4.2 Experiment Results

In the following experiments, the population size is fixed, the crossover probability $= 0.25$, the mutation probability $= 0.5$, the punishment factor $= 4$, no reinitialization is performed, no analogical information is given, and the group size of replacement is the population size. In each round an instance is randomly generated and provided to the model, and the evolutionary algorithms runs one generation.

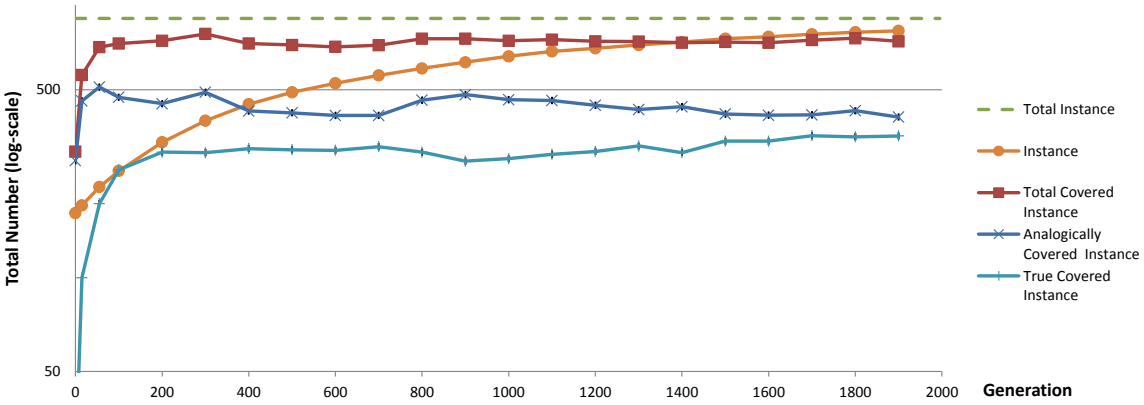
Crossover generates rules with syntactic structures which are not contained in the input instances. After recombination, there might be redundant propositions which contains no `relations` or are known to have consequent propositions which are independent to them. These redundant propositions make the rule contain more symbols or more ground instances. When calculating the fitness value of the rule, if directly using these ground instances as consistent ground instances or inconsistent ground instances, the rule will have lower fitness values than a simpler rule describing



(a) One analogy "Current Cell" is used.



(b) All ten analogies are used.



(c) All ten analogies are used without checking $I(x(a)) = I(x(b))$.

Figure 21: The coverage of input instances of experiments on minesweeper plus analogies with population size = 25. "Total Instance" is the maximum number of input instances. "Instance" is the number of input instances in the knowledge base. "Total Covered Instance" is the number of input instances covered actually and analogically by consistent rules in the population. "Analogically Covered Instance" is the number of input instances analogically covered by consistent rules in the population. "True Covered Instance" is the number of input instances actually (non-analogically) covered by consistent rules in the population.

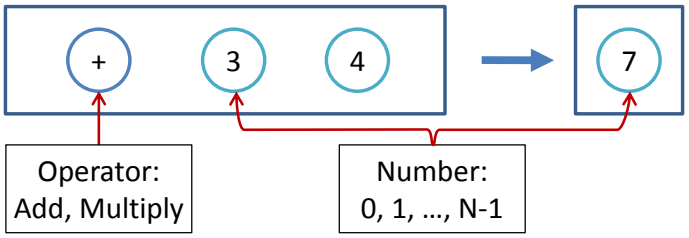


Figure 22: An example of the instance expressing $3 + 4 = 7$.

Population Size	$0 \cdot N = N \cdot 0 = 0$		Identity of $\cdot$		Identity of +		$2 \cdot N = N \cdot 2 = N + N$		Commutativity of +		Associativity of +		Associativity of $\cdot$		Distributivity
25	1.00	(65)	1.00	(182)	1.00	(265)	0.65	(281)	0.60	(283)	0.00		0.00		0.00
100	1.00	(10)	1.00	(52)	1.00	(83)	1.00	(75)	1.00	(155)	0.05	(1072)	0.10	(1223)	0.00
200	1.00	(4)	1.00	(14)	1.00	(34)	1.00	(33)	1.00	(88)	0.24	(1342)	0.24	(1228)	0.00
500	1.00	(0)	1.00	(6)	1.00	(4)	1.00	(22)	1.00	(28)	0.33	(700)	0.50	(921)	0.00

Table 1: The probabilities for the proposed model to learn these rules. The average generations needed to learn these rules are in parentheses.

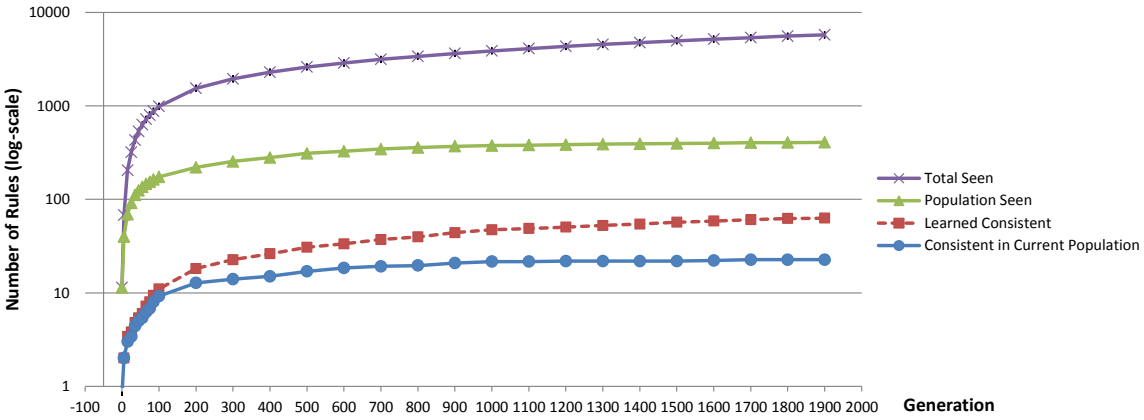
the same thing because rules with redundant propositions usually have more ground instances than concise rules. This phenomenon might even cause the model prefer inconsistent rules and result in the population filled with inconsistent and complex rules. As a result, redundant propositions need to be removed from rules. However, sometimes multiple propositions can combine with each other and become redundant. It makes identifying redundant propositions difficult. Therefore, the model currently only removes one-level redundant propositions and uses an alternative way to calculate (in)consistent ground instances of a rule. When calculating the fitness value of a rule, the model simplifies the ground instances until they become ground input instances. These resulted input instances are used as the (in)consistent ground instances of the rule. As a result, concise rules will be preferred to complex rules which cover the same input instances and have redundant propositions.

Figure 23 shows the number of rules seen by the model, the number of rules seen by the population, the number of learned consistent rules, and the number of consistent rules in current population of population size $= 25, 100$. The filtering effect of the fitness function still occurs in the problem. It gives the proposed model the ability to use relatively small amount of rules to learn.

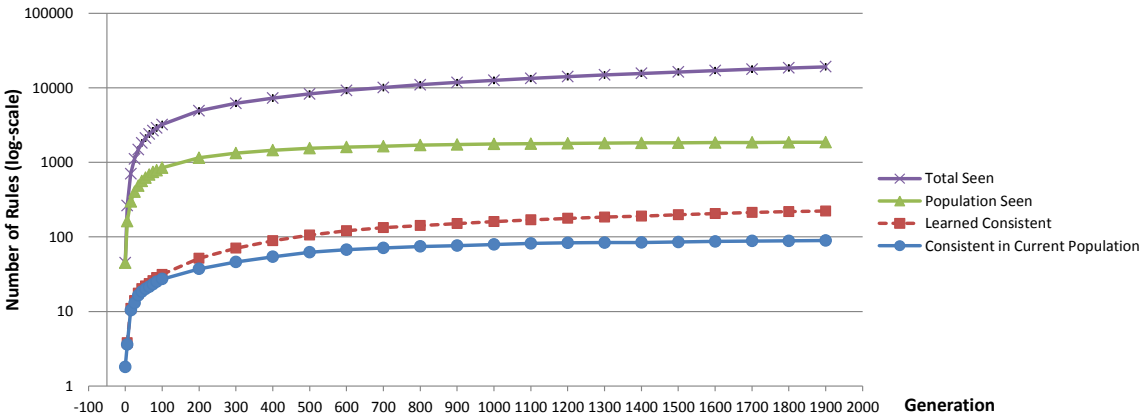
Figure 24 illustrates the coverage of instances with population size $= 25, 100, 200$. This figure shows that the coverage of input instances increases as the population size increases. The difference between the "All Covered Instance" and "Covered 1-Level Instance" shows the number of covered instances which have two-level or higher-level conditional propositions. It shows that the proposed model can combine and generate rules with syntactic structures which do not exist in the input instances.

Figure 25 shows the proportion of consistent rules in the all seen rules (, which are rules seen by both population and offspring) and all rules seen by the population with population size $= 25$ when all input instances are given at once at the beginning of the experiment. Note that because higher level propositions can be generated, the search space is much larger than the input instance space. This figure shows that the probability for a consistent rule to be found in the offspring is gradually increasing. It means that the search direction of the model becomes more efficient as the generation goes. The result shows that even when the search space becomes larger when higher level propositions generated, the ability of the model to find consistent rules does not decrease.

Table 1 shows the probabilities and the average numbers of generations needed for some rules to be learned with population size $= 25, 100, 200$. Those rules shown in the table are arranged from easy (left) to difficult (right) to learn by the proposed model. Associativities and distributivity are difficult for the model to learn. The reason is that the fitness function computes the input



(a) Population size = 25.



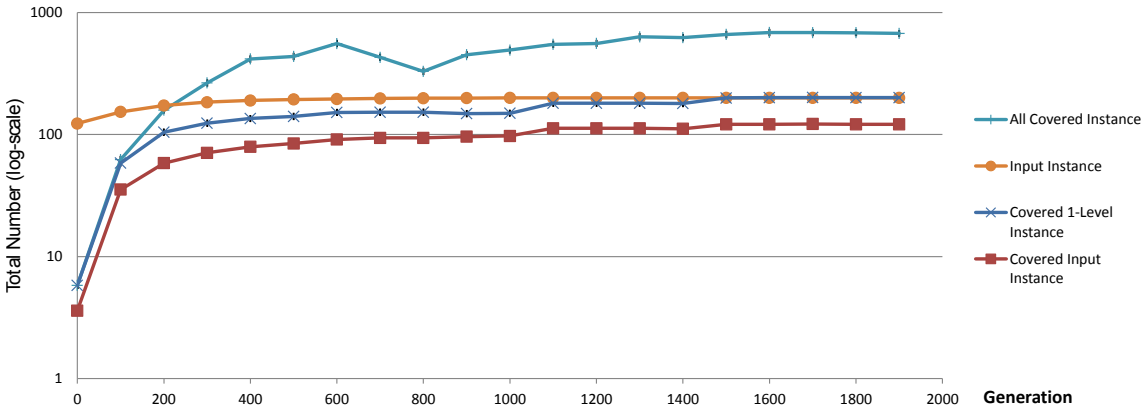
(b) Population size = 100.

Figure 23: The number of rules seen by the model (Total Seen), rules seen by the population (Population Seen), learned consistent rules (Learned Consistent), and consistent rules in current population (Consistent in Current Population) of each generation in the mathematical ring problem.

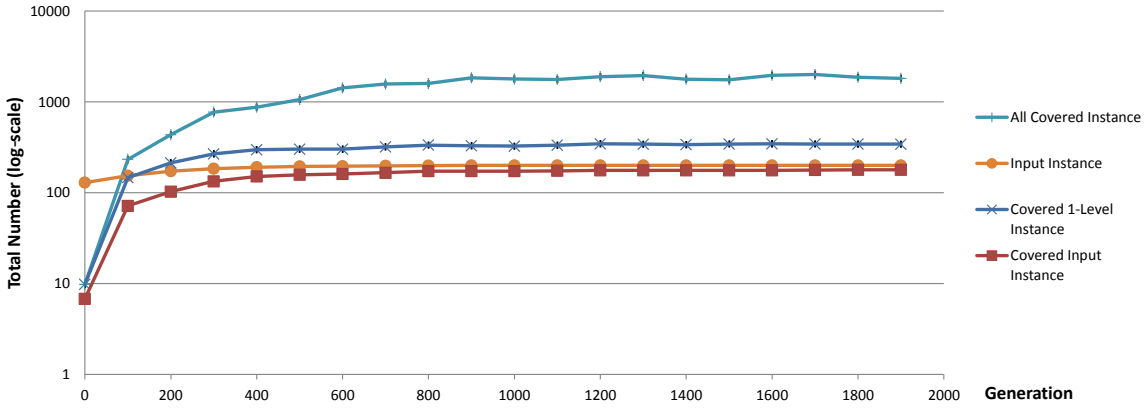
instances covered by a rule. Associativities and distributivity do not cover more input instances than for example commutativity of +, and they require more memories to remember. As a result, associativities and distributivity are not preferred by the proposed model to commutativity of +. Nevertheless, this figure still shows that the proposed model can generalize most of the ring axioms from input instances.

3.5 Summary

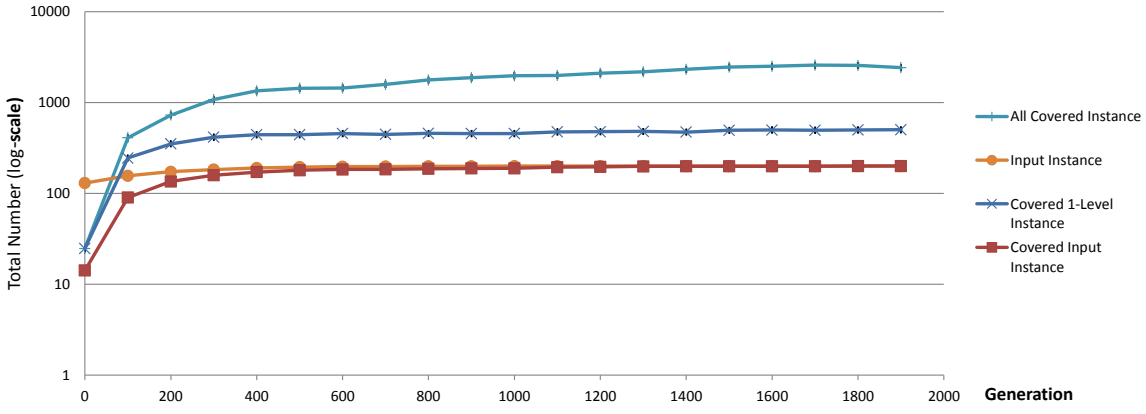
This chapter first analyzes the proposed model. The model has the same expressive power as the universal Turing machine. The inductive process of the model is sound but incomplete. Determining whether a theorem is (inductively) consistent is decidable. This chapter then describes three experiments performed on the proposed model. In the first experiment, only one syntactic structure is allowed. The problem, one-dimensional minesweeper, is used to test the effect of punishment factor, the fitness function, and the generalization ability of the model. The punishment factor should be at least one but not too large, because it makes the model prefer consistent rules but not overly prefer so that inconsistent rules can still occur in the population and keep the diversity of the population. The fitness function has filtering effect which helps the model learns with small amount of rules. The model has the ability to generalize input instances to unseen instances. In the second experiment, the analogical information is provided to the model. This experiment shows that the model can utilize the analogical information to enhance its generalization ability.



(a) Population size = 25.



(b) Population size = 100.



(c) Population size = 200.

Figure 24: The coverage of instances by consistent rules in the population in the mathematical ring problem. "All Covered Instance" is the number of covered instances. "Input Instance" is the number of input instances in the knowledge base. "Covered 1-Level Instance" is the number of covered instances whose conditional propositions are one-level. "Covered Input Instance" is the number of covered input instances.

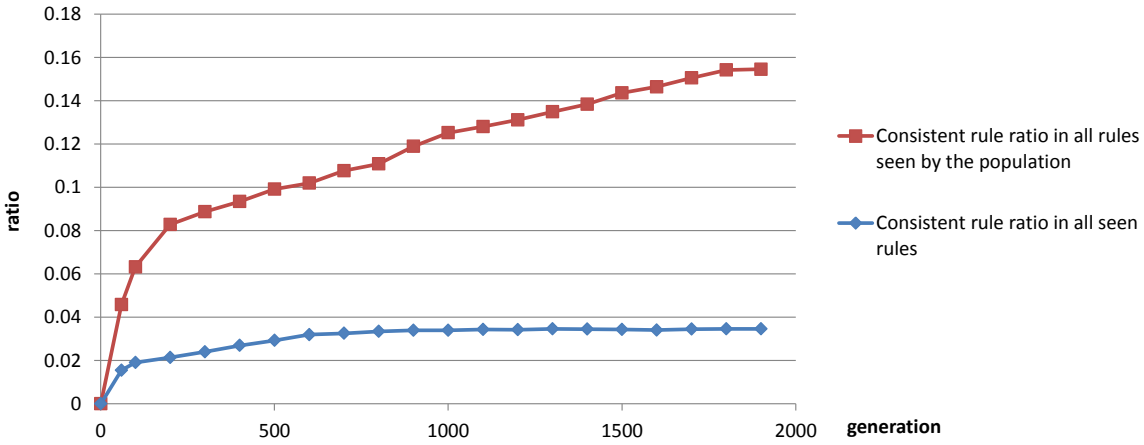


Figure 25: The proportion of consistent rules in the mathematical ring experiment.

In the third experiment, multiple syntactic structures are allowed in the problem. This experiment shows that the model has the ability to combine and generate knowledge with unseen syntactic structures. Results of the experiment shows that the proposed model can generalize most of the ring axioms from input instances. It also shows that when the fitness function uses the coverage of input ground instances to determine the saved memories, some rules might not be learned by the model.

4 Conclusion

This thesis researches on a learning model presuming representationalism, functionalism, and neural Darwinism. This learning model is designed based only on these three cognitive hypotheses and provided with only bare-bone prior knowledge about the learning problem. This learning model is analyzed and tested by three experiments. This learning model has the same expressive power as an universal Turing machine. This learning model shows that it is able to generalize from input instances in these three experiments. The second experiment shows that the model can utilize three different sets of analogical information to enhance its generalization ability on the minesweeper problem. The last experiment shows that the model can learn rules with unseen syntactic structures in the mathematical ring problem.

This thesis has three conclusions. First, a learning model based only on the three cognitive hypotheses–representationalism, functionalism, and neural Darwinism–can do inductive learning. Second, even provided with bare-bone prior knowledge, this learning model can still learn general rules from input instances. Third, this learning model can easily utilize analogical information to enhance its learning ability.

Lots in the proposed learning model can be improved. First, the inference method can be more sophisticated. Currently, the model memorizes all input instances, and when validating a rule, the rule is simplified by replacing propositions with consequent propositions of corresponding input instances. This method can not infer rules which require concurrently applying two or more learned rules. Better inference methods can especially improve the calculation of fitness values in problems which allow rules with unseen syntactic structures, like the mathematical ring problem. With better inference methods, the part related to the consistent inconsistent instances in the fitness function can be computed more accurately. Second, the model can learn the representation information itself. Currently, the representation information including the hierarchies of symbols and analogical information is provided as prior knowledge. The learning of representation can be included into the learning process of the model. The evolutionary processes can try different representational candidates and select those help saving the memory usage. Third, multiple levels of the proposed model can be researched. The learning results of one model can be used as the input instances of another model. It forms a hierarchical structure of the models. In this way, the higher level models are expected to learn concepts which might be more abstract than those

learned by lower level models.

References

[1] J. A. Anderson. *An Introduction to Neural Networks*. The MIT Press, Mar. 1995.

[2] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychol Rev*, 111(4):1036–1060, Oct. 2004.

[3] S. Baker, A. Ireland, and A. Smaill. On the use of the constructive omega-rule within automated deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, LPAR '92, pages 214–225, London, UK, UK, 1992. Springer-Verlag.

[4] Y. Bengio. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2:1–127, Jan. 2009.

[5] G. Edelman. Neural Darwinism: Selection and reentrant signaling in higher brain function. *Neuron*, 10(2):115–125, Feb. 1993.

[6] V. B. Edelman, Gerald M.; Mountcastle. *The mindful brain: Cortical organization and the group-selective theory of higher brain*. Oxford, England: MIT Press, 1978.

[7] C. Fernando, K. K. Karishma, and E. Szathmáry. Copying and Evolution of Neuronal Topology. *PLoS ONE*, 3(11), Nov. 2008.

[8] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (IEEE Press Series on Computational Intelligence)*. Wiley-IEEE Press, 2006.

[9] J. D. D. Friedenberg and D. G. Silverman. *Cognitive Science: An Introduction to the Study of Mind*. Sage Publications, Inc, 2005.

[10] J. L. Garfield. *Cognitive Science: An Introduction*. Cambridge: MIT Press, 1995.

[11] B. Gramlich. Strategic issues, problems and challenges in inductive theorem proving. *Electron. Notes Theor. Comput. Sci.*, 125(2):5–43, Mar. 2005.

[12] B. Gramlich, H. Kirchner, and F. Pfenning. Editorial: Strategies in automated deduction. *Annals of Mathematics and Artificial Intelligence*, 29(1-4):0–0, Jan. 2001.

[13] C. Hartshorne. Collected Papers of Charles Sanders Peirce. Cambridge, MA: Harvard University Press, 1931-1958.

[14] J. Hawkins and D. George. Hierarchical Temporal Memory: Concepts, Theory, and Terminology. Technical Report version 3/27/2007, Numenta Inc., 2006.

[15] D. Holmes and L. Jain. Introduction to Bayesian Networks. pages 1–5. 2008.

[16] D. Kapur and H. Zhang. An overview of rewrite rule laboratory (rrl). *J. of Computer and Mathematics with Applications*, 29:91–114, 1995.

[17] J. E. Laird. *A Gentle Introduction to Soar*. The MIT Press, 2012.

[18] D. Marr. *Vision: A computational investigation into the human representation and processing of visual information*. W.H. Freeman, July 1982.

[19] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evol. Comput.*, 4(2):113–131, June 1996.

[20] U. Neisser. *Cognitive Psychology*. Prentice Hall, 1st edition.

[21] J. Rissanen. *Minimum-Description-Length Principle*. Wiley, 2006.

[22] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. A Bradford Book, 1998.

[23] P. Thagard. *Mind: Introduction to cognitive science*. A Bradford Book, 2000.