Learning Propositional Logic From Scratch

Abdul Rahim Nizamani[1] and Claes Strannegård[2,3]

[1]Department of Applied Information Technology, University of Gothenburg.
abdulrahim.nizamani@gu.se
[2]Department of Applied Information Technology, Chalmers University of Technology;
[3]Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg.
claes.strannegard@gu.se

Abstract

We present a computational model for developing intelligent agents that are able to reason in multiple symbolic domains. The agents have deductive and inductive reasoning abilities. The deductive mechanism is based on a simple cognitive model with bounded cognitive resources. The main learning mechanism is a formalization of Occam's razor. Agents constructed in our model can learn generalized knowledge from concrete examples. For example, an agent can learn elementary arithmetic and propositional logic, and then compute correct answers to previously unseen questions such as "what is $27*9$?" and "is $\mathtt{P} \vee (\mathtt{P} \to \mathtt{Q})$ a tautology?". We illustrate the learning process in the case of propositional logic, where an agent first learns the syntax and then some basic logical laws. The performance of this agent beats average human scores in terms of accuracy when tested on tautologies used in an experiment reported in [16].

Keywords: multi-domain agents, logical reasoning, propositional logic.

1 Introduction

Artificial Intelligence (AI) is concerned with mathematical modeling of intelligence. Cognitive science, on the other hand, focuses on constructing formal models of human cognition and reasoning. One of the methods used in cognitive science is cognitive modeling that develops computational models of mental processes and structures. It is a method for understanding human intelligence and can be a valuable tool for AI as well:

> "Cognitive modelers attempting to explain human intelligence share a puzzle with artificial intelligence researchers aiming to create computers that exhibit human-level intelligence: how can a system composed of relatively unintelligent parts (such as neurons or transistors) behave intelligently?" [3]

As argued in [3], cognitive modeling and artificial intelligence can help each other to solve their shared puzzle. This has been experimented in some recent AI designs that incorporated simple cognitive models to achieve human level performance in their relevant domains (e.g., [13, 16]).

Cognitive models can be constructed as standalone models, or as components within cognitive architectures that are general frameworks to simulate and study specific mental processes. Many such cognitive architectures exist, such as ACT-R [1], Soar [7], and CHREST [5].

ACT-R is a general cognitive architecture containing multiple modules that represent different mental structures (such as memory, vision and speech). The basic assumption in ACT-R is that memory is divided in declarative and procedural knowledge.

Soar is a general cognitive architecture that is intended for developing intelligent systems. Though it is similar to ACT-R, it does not require the cognitive models to be plausible with respect to human cognition.

Both ACT-R and Soar are based on production systems and use explicit production rules that are roughly of the form "if *condition* then *action*".

The working memory (WM) is one of the core components in cognitive models of reasoning. The WM model was first developed by Baddeley and Hitch [2]. WM is a mental workspace and a precious cognitive resource with severe restrictions on the number of information chunks it can hold at any point in time. Earlier findings suggest its capacity to be between 5 and 9 chunks [10]. WM capacity is highly correlated with general intelligence (Spearman's *g* factor) [6].

Other cognitive resources used in cognitive modeling include declarative memory, procedural memory, visual short-term memory, and attention.

Artificial general intelligence (AGI) focuses on the "general" intelligence as opposed to traditional AI that aims at more specific applications of intelligence [18]. Cognitive modeling is a commonly used method in AGI systems. Research in AGI designs has utilized many cognitive architectures, e.g., ACT-R [11], CHREST [8] and Sigma [12]. New cognitive architectures have also been proposed [17].

Occam's razor is a simple scientific principle that expresses a preference for the simplest solution to a problem. Mathematical formulations of Occam's razor include Solomonoff complexity, Kolmogorov complexity, and also Levin complexity [9].

The work presented in this paper builds upon our earlier models of deductive reasoning [13, 16, 4] and inductive reasoning [15]. Our objective is to develop a unified design for inductive and deductive reasoning in symbolic domains. We use the same formal design as in [14] (which uses the example of arithmetic), and apply it to the domain of propositional logic.

In this paper, we present a formal model for intelligent agents that can be trained to compete with humans in propositional reasoning. An agent is presented that learns the syntax and standard axioms of propositional logic from scratch. It can then prove tautologies and perform above average human score.

Section 2 presents the computational model, section 3 shows the training and evaluation of an agent, section 4 discusses the results, and section 5 concludes the paper.

2 Computational model

Our computational model is inspired by cognitive modeling and employs a simple cognitive model with bounded cognitive resources. An explicit representation of working memory enables the agent to perform deductive computations, and its long-term memory stores symbolic concepts and axioms. Learning is achieved by a simple formulation of Occam's razor together with a small set of strategies.

Definition 1 (Tag) *A* tag *is a string of unicode characters with no punctuation marks such as period, comma or colon.*

Definition 2 (Variable) *A* variable *is a string of the form* $\sigma:\tau$*, where* τ *is a tag and* $\sigma \in \{x, y, z\}$.

For example, `x:Formula` and `y:Digit` are variables.

Definition 3 (Term) *A* term *is a finite binary tree whose nodes contain tags or variables, with the restriction that variables can only appear in the leaf nodes.*

Hereafter, we will write terms in conventional linearized form using infix notation, such as $2+3$ and $\mathtt{P} \wedge \mathtt{Q}$. For the purpose of implementing this model as a computer program, we use a standard parser and linearizer to read and write terms in text format.

Definition 4 (Axiom) *An* axiom *has the form* $(\tau, a \Longmapsto b)$ *or* $(\tau, a \longmapsto b)$*, where* τ *is a tag, and a and b are terms.*

Example 1 (Axioms) *Following are some axioms.*

$$(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q} \longmapsto \mathtt{P}) \tag{1}$$

$$(\mathtt{Taut}, \bot \vee \mathtt{x:Formula} \Longmapsto \mathtt{x:Formula}) \tag{2}$$

$$(\mathtt{Arith}, 3 * 2 \Longmapsto 6) \tag{3}$$

$$(\mathtt{Arith}, \mathtt{x:Number} + 0 \Longmapsto \mathtt{x:Number}) \tag{4}$$

Definition 5 (Theory) *A* theory *is a finite set of axioms.*

Definition 6 (Chunk) *A* chunk *is an item of the form* $(\tau, \mathtt{t})$ *where* τ *is a tag and* $\mathtt{t}$ *is a term.*

Definition 7 (Assignment) *An* assignment *is a partial function from variables to terms.*

An instance of an assignment is the following.

$$\sigma = \{(\mathtt{x} : \mathtt{Formula}, \top), (\mathtt{y} : \mathtt{Formula}, \mathtt{P})\}$$

By extention, assignments are defined from terms to terms. E.g.,

$$\sigma(\mathtt{x} : \mathtt{Formula} \to \mathtt{y} : \mathtt{Formula}) = \top \to \mathtt{Q}$$

For a variable-free term t, $\sigma(t) = t$.

Definition 8 (Shallow rewrite) *A shallow rewrite replaces a term t' with a term t'' if there is an axiom $(\tau, t_1 \longmapsto t_2)$ in the theory and there is an assignment σ such that $\sigma(t_1) = t'$ and $\sigma(t_2) = t''$.*

$$\frac{t'}{t''}\,(\tau, t_1 \longmapsto t_2)$$

Definition 9 (Deep rewrite) *A deep rewrite replaces a single occurrence of a subterm t' in a term t (which can be t itself) with another term t'', if there is an axiom $(\tau, t_1 \Longmapsto t_2)$ in the theory and there exists an assignment σ such that $\sigma(t_1) = t'$ and $\sigma(t_2) = t''$.*

$$\frac{t(t')}{t(t'')}\,(\tau, t_1 \Longmapsto t_2)$$

By extension, we shall call an axiom *deep* if it has the form $(\tau, t' \Longmapsto t'')$ and *shallow* if it is like $(\tau, t' \longmapsto t'')$.

Example 2 *Below is an example of a shallow rewrite.*

$$\frac{\mathtt{P} \vee \mathtt{Q}}{\mathtt{P}}\,(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q} \longmapsto \mathtt{P})$$

Example 3 *An instance of a deep rewrite is the following.*

$$\frac{(\bot \vee \mathtt{P}) \to \mathtt{Q}}{\mathtt{P} \to \mathtt{Q}}\,(2)$$

Note that a shallow rewrite operates only on the term itself, while a deep rewrite can also apply on a subterm within the term. Thus, a deep rewrite encompasses the shallow rewrite, but not vice versa.

Semantically, deep axioms are intended to preserve equivalence and shallow axioms to preserve sufficiency. Thus, a shallow axiom such as $(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q} \longmapsto \mathtt{P})$ means that, in order to prove the logical truth of $\mathtt{P} \vee \mathtt{Q}$, it is sufficient to prove that $\mathtt{P}$ is true. Although deep axioms preserve equivalence, the deep rewrite is unidirectional. For example, the axiom $(\mathtt{Taut}, \neg\neg\mathtt{P} \Longmapsto \mathtt{P})$ rewrites $\neg\neg\mathtt{P}$ to $\mathtt{P}$, but not vice versa.

Definition 10 (Computation) *A* computation *is a sequence of distinct terms $(t_0, \ldots, t_n)$ such that t_{i+1} is computed from t_i by applying deep or shallow rewrite.*

Note that computations are acyclic. Hereafter, computations will be written vertically from top to bottom, with every transition marked with the relevant axiom used by the rewrite rules.

Example 4 *Below is an example of a computation in the domain of propositional logic.*

$$\frac{\dfrac{\neg\top \to \mathtt{Q}}{\bot \to \mathtt{Q}}\,(\mathtt{Taut}, \neg\top \Longmapsto \bot)}{\top}\,(\mathtt{Taut}, \bot \to \mathtt{x} : \mathtt{Formula} \Longmapsto \top)$$

Example 5 *Following is a computation in the domain of Arithmetic.*

$$\frac{\dfrac{(3+0)*2}{3*2}\,(4)}{6}\,(3)$$

Definition 11 (Term size) *The size of a term t, denoted $s(t)$, is the number of nodes in t.*

The definition of *size* is extended to axioms as, for an axiom $a = (\tau, t_1 \Longmapsto t_2)$, its size is defined as $s(a) = s(t_1) + s(t_2)$. The size of a theory T is $s(T) = \sum\{s(a) : a \in T\}$.

Definition 12 (Agent) *An* agent *is a tuple (T, C, W, L, D), where*

T *is a theory, representing beliefs in the declarative memory;*
C *is a set of chunks, that represents concepts in the declarative memory;*
W *is a natural number that represents working memory capacity;*
L *is a natural number that limits the maximum length of computations (representing attention span);*
D *is a natural number that boundes the maximum size of theories.*

Definition 13 (Bounded computation) *Given an agent $A = (T, C, W, L, D)$, a* bounded computation *is a sequence of distinct terms $(t_0, \ldots, t_n)$ such that:*

1. *$n \leq L$;*
2. *for each term t_i, $s(t_i) \leq W$;*
3. *each transition from t_i to t_{i+1} uses an axiom from T;*

4. *for* $1 \leq i < j \leq n$, t_i *does not appear as a subterm in* t_j.

Condition 4 above is a weak restriction on increasing sizes of terms in a computation.

Definition 14 (Induction Item) *An* Induction Item *is a tuple* (τ, t_1, t_2, u)*, where* τ *is a tag,* t_1 *and* t_2 *are terms, and u is the utility (an integer). In addition,* t_1 *and* t_2 *do not contain any variables.*

Definition 15 (Induction Problem) *An* Induction Problem *(IP) is a finite set of induction items.*

Example 6 *Below is an example of an IP.*

$$(\mathtt{Taut}, \mathtt{P} \vee \top, \top, 1) \tag{5}$$
$$(\mathtt{Taut}, \mathtt{Q} \vee \top, \top, 1) \tag{6}$$
$$(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q}, \mathtt{Q}, -1) \tag{7}$$

Definition 16 (Item computation) *Agent A computes an induction item* (τ, t_1, t_2, u) *if there is a bounded computation from* t_1 *to* t_2 *satisfying the following condition.*

For any term t_i *in the computation and any axiom* $(\tau, w_1 \Longmapsto w_2)$ *or* $(\tau, w_1 \longmapsto w_2)$*, if* w_1 *contains a variable* $(\mathtt{x}:\mathtt{Tag})$ *that matches with a subterm* t' *in* t_i*, then* t' *must be of type* Tag*, i.e., there must be a bounded computation (for type-checking) from* t' *to* Tag *using only axioms with tag* Lang*.*

Example 7 *Example 4 contains a bounded computation of induction item* $(\mathtt{Taut}, \neg\top \rightarrow \mathtt{Q}, \top, 1)$*, in addition to the following type-checking computation.*

$$\dfrac{\dfrac{\mathtt{Q}}{\mathtt{Boolean}}\,(\mathtt{Lang}, \mathtt{Q} \Longmapsto \mathtt{Boolean})}{\mathtt{Formula}}\,(\mathtt{Lang}, \mathtt{Boolean} \Longmapsto \mathtt{Formula})$$

Item computations are goal-driven bounded computations. We use the standard A^* search algorithm to find bounded computations of minimum length.

Definition 17 (Performance) *The* performance *p of agent A on induction problem IP is computed as* $p = \sum\{u : (\tau, t_1, t_2, u) \in IP \text{ and } A \text{ computes } (\tau, t_1, t_2, u)\}$*. We use the convention* $\sum\emptyset = 0$ *to ensure that p is always defined.*

Induction items with $u < 0$, e.g., (7), are negative examples. Such an induction item, if solved, will reduce the performance of the agent. Thus, to achieve maximum performance, an agent must solve all items with $u > 0$ and must not solve any item with $u < 0$.

As we shall see now, if an agent cannot compute an induction item with $u > 0$, it tries to extend its theory in a way that enables it to compute that item.

Definition 18 (Strategy) *A* strategy *is a method to form axioms for constructing new theories.*

Our model includes the following strategies.

Strategy 1 (Abstraction) *Axiom* $(\tau, t' \Longmapsto w')$ *is formed from the induction item* (τ, t, w, u) *by* abstraction*, where* t' *is obtained by replacing one or more subterms of t with variables, and* w' *is obtained similarly from w.*

Abstraction is a straightforward way of generalizing observations. Below is an example.

Example 8 *Here are some of the axioms constructed by* abstraction *from item* $(\mathtt{Taut}, \mathtt{P} \wedge \mathtt{P}, \mathtt{P}, 1)$.

$(\mathtt{Taut}, \mathtt{x}:\mathtt{Formula} \Longmapsto \mathtt{P})$
$(\mathtt{Taut}, \mathtt{x}:\mathtt{Formula} \wedge \mathtt{x}:\mathtt{Formula} \Longmapsto \mathtt{P})$
$(\mathtt{Taut}, \mathtt{x}:\mathtt{Formula} \wedge \mathtt{x}:\mathtt{Formula} \Longmapsto \mathtt{x}:\mathtt{Formula})$

Strategy 2 (Crossover) *Axioms* $(\tau, t' \Longmapsto w')$ *and* $(\tau, t' \longmapsto w')$ *are formed from chunks* (τ, t) *and* (τ, w) *by* crossover*, where* t' *is obtained by replacing a subterm of t with a subterm of w (*w' *is obtained similarly).*

Crossover is a powerful but computationally expensive method of constructing new theories. All the chunks in the agent's memory and also a small number of variables are used to generate axioms by crossover. Simple heuristics are applied to reduce the number of generated theories.

Example 9 *Following are some of the axioms formed by* crossover *from chunks* $(\mathtt{Taut}, \mathtt{P} \wedge \mathtt{Q})$ *and* $(\mathtt{Taut}, \top \vee \bot)$.

$(\mathtt{Taut}, \mathtt{P} \Longmapsto \top)$
$(\mathtt{Taut}, \top \Longmapsto \mathtt{P} \wedge \mathtt{Q})$
$(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q} \Longmapsto \bot)$
$(\mathtt{Taut}, \mathtt{Q} \vee (\mathtt{P} \wedge \mathtt{Q}) \Longmapsto \bot)$

Strategy 3 (Memorization) *The axiom* $(\tau, t_1 \Longmapsto t_2)$ *is formed by* memorization *from the induction item* (τ, t_1, t_2, u) *if* $u > 0$*.*

Memorization is the strategy to remember the examples literally. It will be used if no other strategy from the previous strategies works.

Example 10 *Axiom* $(\texttt{Formula}, \neg\bot \Longmapsto \top)$ *is formed by* memorization *from item* $(\texttt{Formula}, \neg\bot, \top, 1)$.

The set of strategies can easily be extended if required. We also have a strategy to find recursive definitions of functions; for example, for solving number sequence problems such as $8, 11, 14, ?$. It is not described here however, being irrelevant for propositional logic.

Definition 19 (Update function) *The update function takes as input agent* $A = (T, C, W, L, D)$ *and an IP, and outputs agent* $A' = (T \cup T', C \cup C', W, L, D)$.

C' *is obtained from IP by taking all the subterms of the terms in the items of IP.*

$T' = \emptyset$ *if all items in IP with positive utility have bounded computations; otherwise* T' *is obtained by the following procedure.*

First, construct a set Θ *of new theories whose axioms are generated by the strategies mentioned earlier (strategies 1 to 3). Each theory* T^* *in* Θ *is subject to a set of conditions, such as the following:*

- *number of axioms in* $T^* \leq 3$*;*
- *no axiom of* T^* *appears in* T*;*
- $s(T^*) \leq D$ *if* T^* *is formed by Crossover.*

Theory T' *is selected from* Θ *by applying the following selection criteria in the listed order.*

1. *Agent* A' *performs optimally on the IP.*
2. $s(T')$ *is as small as possible.*
3. T' *contains the maximum number of deep axioms.*
4. T' *has the maximum number of variable tokens.*
5. T' *has the minimum number of variable types.*
6. T' *is lexicographically minimal.*

Condition 2 above is a formal application of Occam's razor and selects an explanation (i.e., theory) of the smallest size that can account for the observations (IP).

Strategies 1 to 3 are used one by one, such that if strategy n leads to a suitable T', the Update function halts, otherwise it proceeds to strategy $n+1$.

3 Results

3.1 Implementation

We implemented our model in the functional programming language Haskell. The code consists of approximatly 2,500 lines. The program, called `OccamStar`, takes an agent and an IP as input, and outputs a new agent with updated memory (that replaces the input agent). It also produces bounded computations of the items in IP.

3.2 Learning process

To illustrate the learning process, we show here a set of examples that allows the agent to acquire the elementary axioms of propositional logic.

Let the agent A be initiated with $T = \emptyset$, $C = \emptyset$, $W = 8$, $L = 10$, and $D = 4$.

Example 11 *Let IP consist of the following items.*

$$(\texttt{Lang}, \top, \texttt{Boolean}, 1) \tag{8}$$
$$(\texttt{Lang}, \bot, \texttt{Boolean}, 1) \tag{9}$$

`OccamStar` *takes agent A and the above IP, and updates A by adding the following axioms to the theory.*

$$(\texttt{Lang}, \top \Longmapsto \texttt{Boolean}) \tag{10}$$
$$(\texttt{Lang}, \bot \Longmapsto \texttt{Boolean}) \tag{11}$$

Item 8 is computed with this theory as follows.

$$\frac{\top}{\texttt{Boolean}}\ (10)$$

With a similar IP, agent A learns the following axioms.

$$(\texttt{Lang}, \texttt{P} \Longmapsto \texttt{Boolean}) \tag{12}$$
$$(\texttt{Lang}, \texttt{Q} \Longmapsto \texttt{Boolean}) \tag{13}$$
$$(\texttt{Lang}, \texttt{R} \Longmapsto \texttt{Boolean}) \tag{14}$$

Example 12 *Now, continuing the learning process, let IP be given as follows.*

$$(\texttt{Lang}, \top, \texttt{Formula}, 1) \tag{15}$$
$$(\texttt{Lang}, \texttt{P}, \texttt{Formula}, 1) \tag{16}$$

Then, agent A learns the following axiom.

$$(\texttt{Lang}, \texttt{Boolean} \Longmapsto \texttt{Formula}) \tag{17}$$

Intuitively, agent A knows that a boolean symbol is a formula. The bounded computation for item (15) is as follows.

$$\frac{\dfrac{\top}{\texttt{Boolean}}\ (10)}{\texttt{Formula}}\ (17)$$

Example 13 *Let IP contain the following items.*

$$(\mathtt{Lang}, \top \wedge \bot, \mathtt{Formula}, 1) \quad (18)$$
$$(\mathtt{Lang}, \mathtt{P} \wedge \mathtt{Q}, \mathtt{Formula}, 1) \quad (19)$$

Then, the following axiom is added to the theory.

$$(\mathtt{Lang}, \mathtt{Formula} \wedge \mathtt{Formula} \Longmapsto \mathtt{Formula}) \quad (20)$$

When the IP contains the tag Lang, the agent prefers the most general tag. From (17), it knows that the tag Formula is more general than Boolean. Thus, it selects (20) instead of the following axiom that is equally valid.

$$(\mathtt{Lang}, \mathtt{Boolean} \wedge \mathtt{Boolean} \Longmapsto \mathtt{Formula}) \quad (21)$$

With similar IPs, agent *A* adds the following axioms.

$$(\mathtt{Lang}, \neg\mathtt{Formula} \Longmapsto \mathtt{Formula}) \quad (22)$$
$$(\mathtt{Lang}, \mathtt{Formula} \vee \mathtt{Formula} \Longmapsto \mathtt{Formula}) \quad (23)$$
$$(\mathtt{Lang}, \mathtt{Formula} \rightarrow \mathtt{Formula} \Longmapsto \mathtt{Formula}) \quad (24)$$
$$(\mathtt{Lang}, \mathtt{Formula} \leftrightarrow \mathtt{Formula} \Longmapsto \mathtt{Formula}) \quad (25)$$

Example 14 *Next, let IP consist of the following items.*

$$(\mathtt{Taut}, \top \vee \mathtt{P}, \top, 1) \quad (26)$$
$$(\mathtt{Taut}, \top \vee (\neg\mathtt{Q}), \top, 1) \quad (27)$$
$$(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q}, \mathtt{P}, -1) \quad (28)$$

Then, agent A acquires the following axiom.

$$(\mathtt{Taut}, \top \vee \mathtt{x} : \mathtt{Formula} \Longmapsto \top) \quad (29)$$

Agent *A* is now able to solve new problems such as the following.

$$\frac{\top \vee (\neg\mathtt{R})}{\top} \ (29)$$

The above computation requires an additional type-checking computation to verify that $\neg\mathtt{R}$ is a Formula.

$$\cfrac{\cfrac{\cfrac{\neg\mathtt{R}}{\neg\mathtt{Boolean}}\ (14)}{\neg\mathtt{Formula}}\ (17)}{\mathtt{Formula}}\ (22)$$

The inclusion of the negative item (28) in example 14 invalidates the following erroneous axiom, which would have been learned by the agent otherwise.

$$(\mathtt{Taut}, \mathtt{x} : \mathtt{Formula} \vee \mathtt{y} : \mathtt{Formula} \Longmapsto \mathtt{x} : \mathtt{Formula}) \quad (30)$$

Example 15 *Let IP have the following items.*

$$(\mathtt{Taut}, \bot \vee \mathtt{P}, \mathtt{P}, 1) \quad (31)$$
$$(\mathtt{Taut}, \bot \vee \neg\mathtt{Q}, \neg\mathtt{Q}, 1) \quad (32)$$
$$(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q}, \mathtt{Q}, -1) \quad (33)$$

Then, agent A learns the following.

$$(\mathtt{Taut}, \bot \vee \mathtt{x} : \mathtt{Formula} \Longmapsto \mathtt{x} : \mathtt{Formula}) \quad (34)$$

The agent can also learn shallow axioms when a negative item is included in the IP to invalidate a similar deep axiom. The following example illustrates this.

Example 16 *Let IP contain the following items.*

$$(\mathtt{Taut}, \mathtt{P} \vee \mathtt{Q}, \mathtt{P}, 1) \quad (35)$$
$$(\mathtt{Taut}, \mathtt{P} \wedge (\mathtt{P} \vee \mathtt{Q}), \mathtt{P} \wedge \mathtt{P}, -1) \quad (36)$$

Then, agent A learns the following.

$$(\mathtt{Taut}, \mathtt{x} : \mathtt{Formula} \vee \mathtt{y} : \mathtt{Formula} \longmapsto \mathtt{x} : \mathtt{Formula}) \quad (37)$$

3.3 Evaluation

An experiment reported in [16] presented 40 tautologies and 40 non-tautologies, mixed randomly, to participants who were asked to decide which ones were tautologies. The subjects were university students of computer science who had received tuition in propositional logic. The mean accuracy of participants on tautologies was 32.1, with the maximum score of 35 (out of 40). All tautologies except one (trial 10) were solved by at least one of the participants.

The computer program OccamStar was used to train an agent LA with the basic axioms of propositional logic, a subset of the axioms used by [16] in their computational model. Parameters were set as follows: $T = \emptyset$, $C = \emptyset$, $W = 8$, $L = 10$ and $D = 4$. The training proceeded in the same fashion as illustrated with examples previously, starting with an empty theory and resulting in the theory that is given in Appendix A.

Agent LA was able to solve 34 tautologies out of the 40. It computed the solutions within a few minutes on a standard personal computer. Sample computations of tautologies produced by LA are presented below.

Computation of trial 3:

$$\cfrac{\cfrac{(\mathtt{P} \vee \mathtt{P}) \leftrightarrow \neg\neg\mathtt{P}}{\mathtt{P} \leftrightarrow \neg\neg\mathtt{P}}\ (A43)}{\cfrac{\mathtt{P} \leftrightarrow \mathtt{P}}{\top}\ (A18)}\ (A57)$$

Computation of trial 17:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}\vee(\mathsf{P}\vee\neg(\mathsf{Q}\wedge\mathsf{P}))}{\mathsf{P}\vee\neg(\mathsf{Q}\wedge\mathsf{P})}\,(\text{A83})}{\neg(\mathsf{Q}\wedge\mathsf{P})\vee\mathsf{P}}\,(\text{A45})}{(\mathsf{Q}\wedge\mathsf{P})\to\mathsf{P}}\,(\text{A60})}{\top}\,(\text{A22})$$

Computation of trial 33:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\mathsf{P}\leftrightarrow(\mathsf{Q}\leftrightarrow(\mathsf{P}\vee\mathsf{P})))\to\mathsf{Q}}{(\mathsf{P}\leftrightarrow(\mathsf{Q}\leftrightarrow\mathsf{P}))\to\mathsf{Q}}\,(\text{A43})}{(\mathsf{P}\leftrightarrow(\mathsf{P}\leftrightarrow\mathsf{Q}))\to\mathsf{Q}}\,(\text{A46})}{((\mathsf{P}\leftrightarrow\mathsf{P})\leftrightarrow\mathsf{Q})\to\mathsf{Q}}\,(\text{A52})}{(\top\leftrightarrow\mathsf{Q})\to\mathsf{Q}}\,(\text{A18})}{\mathsf{Q}\to\mathsf{Q}}\,(\text{A39})}{\top}\,(\text{A17})$$

Computation of trial 37:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\mathsf{P}\wedge\mathsf{Q})\to(\mathsf{Q}\vee\mathsf{R})}{\neg(\mathsf{P}\wedge\mathsf{Q})\vee(\mathsf{Q}\vee\mathsf{R})}\,(\text{A61})}{(\neg(\mathsf{P}\wedge\mathsf{Q})\vee\mathsf{Q})\vee\mathsf{R}}\,(\text{A50})}{(\neg(\mathsf{P}\wedge\mathsf{Q})\vee\mathsf{Q})}\,(\text{A82})}{(\mathsf{P}\wedge\mathsf{Q})\to\mathsf{Q}}\,(\text{A60})}{\top}\,(\text{A22})$$

4 Discussion

The model described here incorporates some elements of cognitive modeling to reduce the combinatorial explosion of the search space, and achieves human level performance in mutiple domains. We have demonstrated this in the case of propositional logic, where an agent trained in logic was able to compete with human scores in an experiment on tautologies. Nothing about our agent was geared specifically to logic, however. Indeed, the agent can equally well be trained in arithmetic as was done in [14]. When trained in arithmetic, the agent was able to solve previously unseen deductive problems (e.g., what is $65*7$?) and inductive problems (e.g., what number comes next in $8, 11, 14$?). A similar model developed earlier was able to solve number sequence problems not seen before from real IQ tests and it scored above average human performance [15].

Cognitive modeling for the purposes of AI does not need to be psychologically verified. AI aims to achieve human level performance (or beyond) on intelligence problems, therefore the methods are not required to be cognitively grounded. Thus, AI does not need psychologically plausible cognitive models, rather only particular elements from the field of cognitive modeling that can help reduce the computational complexity of the solution. This can be compared with the biologically inspired computing methods such as neural networks, ant colony optimization algorithms or genetic algorithms. Such algorithms do not necessarily imitate any biological systems, rather are designed with inspiration from certain properties of natural systems.

The bounds over cognitive resources, such as working memory capacity (W), can be manipulated to vary the agent performance and learning complexity. For instance, with $W = 10$, agent `LA` correctly solved 36 tautologies out of 40. Learning additional axioms helps it score even higher. It is still an open question whether the agent can acquire a complete theory of propositional logic, and this requires further work.

5 Conclusion

The results presented here suggest that resource bounded cognitive models can be helpful in AI for constructing intelligent agents who perform at or above human level. For the purposes of AI, such cognitive models need not be psychologically plausible, as long as they help solve the problems. We believe that a more refined cognitive model would strengthen our model further however, for example in acquiring larger theories and reducing the search complexity.

We have presented an agent architecture that can acquire knowledge in multiple symbolic domains. Thus, it represents a restricted form of artificial general intelligence. Potential application areas of such agents include automated theorem proving, inductive program synthesis, intelligent pedagogical systems, and computational linguistic systems.

Acknowledgement

This research was supported by The Swedish Research Council (grant 421-2012-1000).

A Agent theory

Following is the theory learned by agent `LA`. For readability, `x:Formula` is shortened to `x`, `y:Formula` is short-

ened to y, and z:Formula is written z.

Language syntax:

A1. $(\mathsf{Lang}, \top \Longmapsto \mathsf{Boolean})$
A2. $(\mathsf{Lang}, \bot \Longmapsto \mathsf{Boolean})$
A3. $(\mathsf{Lang}, \mathsf{P} \Longmapsto \mathsf{Boolean})$
A4. $(\mathsf{Lang}, \mathsf{Q} \Longmapsto \mathsf{Boolean})$
A5. $(\mathsf{Lang}, \mathsf{R} \Longmapsto \mathsf{Boolean})$
A6. $(\mathsf{Lang}, \mathsf{Boolean} \Longmapsto \mathsf{Formula})$
A7. $(\mathsf{Lang}, \neg\mathsf{Formula} \Longmapsto \mathsf{Formula})$
A8. $(\mathsf{Lang}, \mathsf{Formula} \vee \mathsf{Formula} \Longmapsto \mathsf{Formula})$
A9. $(\mathsf{Lang}, \mathsf{Formula} \wedge \mathsf{Formula} \Longmapsto \mathsf{Formula})$
A10. $(\mathsf{Lang}, \mathsf{Formula} \to \mathsf{Formula} \Longmapsto \mathsf{Formula})$
A11. $(\mathsf{Lang}, \mathsf{Formula} \leftrightarrow \mathsf{Formula} \Longmapsto \mathsf{Formula})$

Tautologies:

A12. $(\mathsf{Taut}, \neg\bot \Longmapsto \top)$
A13. $(\mathsf{Taut}, \mathsf{x} \vee \top \Longmapsto \top)$
A14. $(\mathsf{Taut}, \top \vee \mathsf{x} \Longmapsto \top)$
A15. $(\mathsf{Taut}, \mathsf{x} \to \top \Longmapsto \top)$
A16. $(\mathsf{Taut}, \bot \to \mathsf{x} \Longmapsto \top)$
A17. $(\mathsf{Taut}, \mathsf{x} \to \mathsf{x} \Longmapsto \top)$
A18. $(\mathsf{Taut}, \mathsf{x} \leftrightarrow \mathsf{x} \Longmapsto \top)$
A19. $(\mathsf{Taut}, \mathsf{x} \vee \neg\mathsf{x} \Longmapsto \top)$
A20. $(\mathsf{Taut}, \neg\mathsf{x} \vee \mathsf{x} \Longmapsto \top)$
A21. $(\mathsf{Taut}, (\mathsf{x} \wedge \mathsf{y}) \to \mathsf{x} \Longmapsto \top)$
A22. $(\mathsf{Taut}, (\mathsf{x} \wedge \mathsf{y}) \to \mathsf{y} \Longmapsto \top)$
A23. $(\mathsf{Taut}, \mathsf{x} \to (\mathsf{x} \vee \mathsf{y}) \Longmapsto \top)$
A24. $(\mathsf{Taut}, \mathsf{y} \to (\mathsf{x} \vee \mathsf{y}) \Longmapsto \top)$

Contradictions:

A25. $(\mathsf{Taut}, \neg\top \Longmapsto \bot)$
A26. $(\mathsf{Taut}, \mathsf{x} \wedge \bot \Longmapsto \bot)$
A27. $(\mathsf{Taut}, \bot \wedge \mathsf{x} \Longmapsto \bot)$
A28. $(\mathsf{Taut}, \mathsf{x} \wedge \neg\mathsf{x} \Longmapsto \bot)$
A29. $(\mathsf{Taut}, \neg\mathsf{x} \wedge \mathsf{x} \Longmapsto \bot)$
A30. $(\mathsf{Taut}, \mathsf{x} \leftrightarrow \neg\mathsf{x} \Longmapsto \bot)$
A31. $(\mathsf{Taut}, \neg\mathsf{x} \leftrightarrow \mathsf{x} \Longmapsto \bot)$

Identity:

A32. $(\mathsf{Taut}, \mathsf{x} \vee \bot \Longmapsto \mathsf{x})$
A33. $(\mathsf{Taut}, \bot \vee \mathsf{x} \Longmapsto \mathsf{x})$
A34. $(\mathsf{Taut}, \mathsf{x} \wedge \top \Longmapsto \mathsf{x})$
A35. $(\mathsf{Taut}, \top \wedge \mathsf{x} \Longmapsto \mathsf{x})$
A36. $(\mathsf{Taut}, \top \to \mathsf{x} \Longmapsto \mathsf{x})$
A37. $(\mathsf{Taut}, \mathsf{x} \to \bot \Longmapsto \neg\mathsf{x})$
A38. $(\mathsf{Taut}, \mathsf{x} \leftrightarrow \top \Longmapsto \mathsf{x})$
A39. $(\mathsf{Taut}, \top \leftrightarrow \mathsf{x} \Longmapsto \mathsf{x})$
A40. $(\mathsf{Taut}, \mathsf{x} \leftrightarrow \bot \Longmapsto \neg\mathsf{x})$
A41. $(\mathsf{Taut}, \bot \leftrightarrow \mathsf{x} \Longmapsto \neg\mathsf{x})$

Idempotence:

A42. $(\mathsf{Taut}, \mathsf{x} \wedge \mathsf{x} \Longmapsto \mathsf{x})$
A43. $(\mathsf{Taut}, \mathsf{x} \vee \mathsf{x} \Longmapsto \mathsf{x})$

Commutativity:

A44. $(\mathsf{Taut}, \mathsf{x} \wedge \mathsf{y} \Longmapsto \mathsf{y} \wedge \mathsf{x})$
A45. $(\mathsf{Taut}, \mathsf{x} \vee \mathsf{y} \Longmapsto \mathsf{y} \vee \mathsf{x})$
A46. $(\mathsf{Taut}, \mathsf{x} \leftrightarrow \mathsf{y} \Longmapsto \mathsf{y} \leftrightarrow \mathsf{x})$

Associativity:

A47. $(\mathsf{Taut}, (\mathsf{x} \wedge \mathsf{y}) \wedge \mathsf{z} \Longmapsto \mathsf{x} \wedge (\mathsf{y} \wedge \mathsf{z}))$
A48. $(\mathsf{Taut}, \mathsf{x} \wedge (\mathsf{y} \wedge \mathsf{z}) \Longmapsto (\mathsf{x} \wedge \mathsf{y}) \wedge \mathsf{z})$
A49. $(\mathsf{Taut}, (\mathsf{x} \vee \mathsf{y}) \vee \mathsf{z} \Longmapsto \mathsf{x} \vee (\mathsf{y} \vee \mathsf{z}))$
A50. $(\mathsf{Taut}, \mathsf{x} \vee (\mathsf{y} \vee \mathsf{z}) \Longmapsto (\mathsf{x} \vee \mathsf{y}) \vee \mathsf{z})$
A51. $(\mathsf{Taut}, (\mathsf{x} \leftrightarrow \mathsf{y}) \leftrightarrow \mathsf{z} \Longmapsto \mathsf{x} \leftrightarrow (\mathsf{y} \leftrightarrow \mathsf{z}))$
A52. $(\mathsf{Taut}, \mathsf{x} \leftrightarrow (\mathsf{y} \leftrightarrow \mathsf{z}) \Longmapsto (\mathsf{x} \leftrightarrow \mathsf{y}) \leftrightarrow \mathsf{z})$

Distributivity:

A53. $(\mathsf{Taut}, \mathsf{x} \wedge (\mathsf{y} \vee \mathsf{z}) \Longmapsto (\mathsf{x} \wedge \mathsf{y}) \vee (\mathsf{x} \wedge \mathsf{z}))$
A54. $(\mathsf{Taut}, \mathsf{x} \vee (\mathsf{y} \wedge \mathsf{z}) \Longmapsto (\mathsf{x} \vee \mathsf{y}) \wedge (\mathsf{x} \vee \mathsf{z}))$
A55. $(\mathsf{Taut}, (\mathsf{x} \wedge \mathsf{y}) \vee (\mathsf{x} \wedge \mathsf{z}) \Longmapsto \mathsf{x} \wedge (\mathsf{y} \vee \mathsf{z}))$
A56. $(\mathsf{Taut}, (\mathsf{x} \vee \mathsf{y}) \wedge (\mathsf{x} \vee \mathsf{z}) \Longmapsto \mathsf{x} \vee (\mathsf{y} \wedge \mathsf{z}))$

Negation:

A57. $(\mathsf{Taut}, \neg\neg\mathsf{x} \Longmapsto \mathsf{x})$
A58. $(\mathsf{Taut}, \mathsf{x} \to \neg\mathsf{x} \Longmapsto \neg\mathsf{x})$
A59. $(\mathsf{Taut}, \neg\mathsf{x} \to \mathsf{x} \Longmapsto \mathsf{x})$
A60. $(\mathsf{Taut}, \neg\mathsf{x} \vee \mathsf{y} \Longmapsto \mathsf{x} \to \mathsf{y})$
A61. $(\mathsf{Taut}, \mathsf{x} \to \mathsf{y} \Longmapsto \neg\mathsf{x} \vee \mathsf{y})$
A62. $(\mathsf{Taut}, \neg(\mathsf{x} \to \mathsf{y}) \Longmapsto \mathsf{x} \wedge \neg\mathsf{y})$
A63. $(\mathsf{Taut}, \mathsf{x} \wedge \neg\mathsf{y} \Longmapsto \neg(\mathsf{x} \to \mathsf{y})$
A64. $(\mathsf{Taut}, \neg\mathsf{y} \to \neg\mathsf{x} \Longmapsto \mathsf{x} \to \mathsf{y})$
A65. $(\mathsf{Taut}, \mathsf{x} \to \mathsf{y} \Longmapsto \neg\mathsf{y} \to \neg\mathsf{x}$

De Morgan:

A66. $(\mathsf{Taut}, \neg\mathsf{x} \wedge \neg\mathsf{y} \Longmapsto \neg(\mathsf{x} \vee \mathsf{y}))$
A67. $(\mathsf{Taut}, \neg(\mathsf{x} \vee \mathsf{y}) \Longmapsto \neg\mathsf{x} \wedge \neg\mathsf{y})$
A68. $(\mathsf{Taut}, \neg(\neg\mathsf{x} \wedge \neg\mathsf{y}) \Longmapsto \mathsf{x} \vee \mathsf{y})$
A69. $(\mathsf{Taut}, \mathsf{x} \vee \mathsf{y} \Longmapsto \neg(\neg\mathsf{x} \wedge \neg\mathsf{y}))$
A70. $(\mathsf{Taut}, \neg(\mathsf{x} \wedge \neg\mathsf{y}) \Longmapsto \neg\mathsf{x} \vee \mathsf{y})$
A71. $(\mathsf{Taut}, \neg\mathsf{x} \vee \mathsf{y} \Longmapsto \neg(\mathsf{x} \wedge \neg\mathsf{y}))$

A72. $(\mathtt{Taut}, \neg(\neg x \wedge y) \Longmapsto x \vee \neg y)$
A73. $(\mathtt{Taut}, x \vee \neg y \Longmapsto \neg(\neg x \wedge y))$
A74. $(\mathtt{Taut}, \neg x \vee \neg y \Longmapsto \neg(x \wedge y))$
A75. $(\mathtt{Taut}, \neg(x \wedge y) \Longmapsto \neg x \vee \neg y)$
A76. $(\mathtt{Taut}, \neg(\neg x \vee \neg y) \Longmapsto x \wedge y)$
A77. $(\mathtt{Taut}, x \wedge y \Longmapsto \neg(\neg x \vee \neg y))$
A78. $(\mathtt{Taut}, \neg(x \vee \neg y) \Longmapsto \neg x \wedge y)$
A79. $(\mathtt{Taut}, \neg x \wedge y \Longmapsto \neg(x \vee \neg y))$
A80. $(\mathtt{Taut}, \neg(\neg x \vee y) \Longmapsto x \wedge \neg y)$
A81. $(\mathtt{Taut}, x \wedge \neg y \Longmapsto \neg(\neg x \vee y))$

Axioms for shallow rewrite:

A82. $(\mathtt{Taut}, x \vee y \longmapsto x)$
A83. $(\mathtt{Taut}, x \vee y \longmapsto y)$
A84. $(\mathtt{Taut}, x \to y \longmapsto y)$

B Tautologies

Below is the list of 40 tautologies that were used for testing agent LA. The same test items were used for testing human performance [16]. Column L shows the minimum length of bounded computations generated by LA. The symbol * indicates that agent LA failed to find a computation. LA successfully identified the tautologies in 34 of 40 cases. Accuracy and latency measures of these tautologies are listed in [16]. Trial numbers are non-consecutive as non-tautologies have been omitted from the list.

Trial	Formula	L
1	$P \to ((\neg\neg P \to P) \vee Q)$	4
2	$(\neg P \leftrightarrow P) \to (Q \leftrightarrow Q)$	2
3	$(P \vee P) \to \neg\neg P$	3
5	$(\neg P \wedge P) \to Q$	2
7	$P \to (Q \to P)$	2
9	$\neg(\neg P \leftrightarrow P)$	2
10	$(\neg(P \to Q) \wedge R) \to \neg Q$	*
11	$\neg\neg\neg(P \wedge \neg P)$	3
12	$(P \to P) \vee Q$	2
15	$(P \wedge P) \vee \neg P$	2
17	$P \vee (P \vee (\neg(Q \wedge P)))$	4
18	$(P \leftrightarrow \neg P) \to (Q \leftrightarrow P)$	2
21	$\neg((P \to \neg Q) \vee P) \to Q$	*
22	$((P \vee \neg Q) \vee Q) \vee P$	4
28	$(P \wedge P) \to P$	1
29	$(P \wedge P) \leftrightarrow P$	2
32	$\neg\neg(P \to Q) \vee \neg Q$	3
33	$(P \leftrightarrow P) \vee (P \wedge \neg Q)$	6
35	$(P \leftrightarrow P) \vee (P \wedge \neg Q)$	2
37	$(P \wedge Q) \to (Q \vee R)$	5
40	$\neg P \leftrightarrow \neg P$	1
42	$(\neg P \vee Q) \vee P$	3
43	$\neg(P \to P) \to Q$	3
44	$P \leftrightarrow \neg\neg P$	2
45	$P \to ((P \leftrightarrow Q) \vee (Q \to R))$	*
50	$(P \to Q) \to (P \to P)$	2
52	$\neg(P \wedge P) \vee (Q \to Q)$	2
56	$P \vee (P \to \neg Q)$	4
57	$P \vee ((P \to Q) \vee P)$	5
62	$(P \wedge ((Q \to P) \leftrightarrow P)) \leftrightarrow P$	*
63	$P \vee (Q \vee ((Q \wedge R) \leftrightarrow P))$	*
64	$P \vee ((Q \wedge Q) \to Q)$	2
66	$P \vee (P \to Q)$	4
67	$P \to (P \vee Q)$	1
69	$\neg(P \to Q) \vee (Q \to Q)$	2
70	$\neg\neg((P \to Q) \wedge Q) \wedge Q$	*
71	$P \vee (P \to \neg\neg(Q \leftrightarrow Q))$	4
72	$(P \to \neg Q) \vee P$	4
76	$(P \to \neg Q) \vee Q$	4
77	$\neg P \vee (Q \to (P \vee P))$	4

References

[1] John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

[2] Alan Baddeley. *Working Memory, Thought, and Action*. Oxford University Press, 2007.

[3] Nicholas L Cassimatis. Artificial Intelligence and Cognitive Modeling Have the Same Problem. In *Theoretical Foundations of Artificial General Intelligence*, pages 11–24. Springer, 2012.

[4] Fredrik Engström, Abdul Rahim Nizamani, and Claes Strannegård. Understandable Explanations in Description Logic. submitted.

[5] Fernand Gobet and Peter CR Lane. The CHREST Architecture of Cognition: The Role of Perception in General Intelligence. In *Procs 3rd Conf on Artificial General Intelligence*. Atlantis Press, 2010.

[6] Patrick C Kyllonen. Is Working Memory Capacity Spearman's g? *Human abilities: Their nature and measurement*, pages 49–75, 1996.

[7] John Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

[8] Peter Lane and Fernand Gobet. CHREST Models of Implicit Learning and Board Game Interpretation. In *Artificial General Intelligence*, pages 148–157. Springer, 2012.

[9] Ming Li and Paul MB Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2009.

[10] George A Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological review*, 63(2):81, 1956.

[11] Alessandro Oltramari and Christian Lebiere. Pursuing Artificial General Intelligence by Leveraging the Knowledge Capabilities of ACT-R. In *Artificial General Intelligence*, pages 199–208. Springer, 2012.

[12] David V Pynadath, Paul S Rosenbloom, Stacy C Marsella, and Lingshan Li. Modeling Two-Player Games in the Sigma Graphical Cognitive Architecture. In *Artificial General Intelligence*, pages 98–108. Springer, 2013.

[13] Claes Strannegård, Fredrik Engström, Abdul Rahim Nizamani, and Lance Rips. Reasoning About Truth in First-Order Logic. *Journal of Logic, Language and Information*, 22(1):115–137, 2013.

[14] Claes Strannegård, Abdul Rahim Nizamani, and Ulf Persson. Learning Symbolic Domains From Scratch. In press.

[15] Claes Strannegård, Abdul Rahim Nizamani, Anders Sjöberg, and Fredrik Engström. Bounded Kolmogorov Complexity Based on Cognitive Models. In *Proceedings of the 6th conference on Artificial General Intelligence*, pages 130–139. Springer, 2013.

[16] Claes Strannegård, Simon Ulfsbäcker, David Hedqvist, and Tommy Gärling. Reasoning Processes in Propositional Logic. *Journal of Logic, Language and Information*, 19(3):283–314, 2010.

[17] Claes Strannegård, Rickard von Haugwitz, Johan Wessberg, and Christian Balkenius. A Cognitive Architecture Based on Dual Process Theory. In *Artificial General Intelligence*, pages 140–149. Springer, 2013.

[18] Pei Wang and Ben Goertzel. Introduction: Aspects of Artificial General Intelligence. In *Proceedings of the 2007 conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006*, pages 1–16. IOS Press, 2007.