Can Machines Learn Logics?

Chiaki Sakama[1] and Katsumi Inoue[2]

[1] Department of Computer and Communication Sciences
Wakayama University, Sakaedani, Wakayama 640-8510, Japan
sakama@sys.wakayama-u.ac.jp
[2] National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
inoue@nii.ac.jp

Abstract. This paper argues the possibility of designing AI that can learn logics from data. We provide an abstract framework for learning logics. In this framework, an agent $\mathcal{A}$ provides training examples that consist of formulas S and their logical consequences T. Then a machine $\mathcal{M}$ builds an axiomatic system that underlies between S and T. Alternatively, in the absence of an agent $\mathcal{A}$, the machine $\mathcal{M}$ seeks an unknown logic underlying given data. We next provide two cases of learning logics: the first case considers learning deductive inference rules in propositional logic, and the second case considers learning transition rules in cellular automata. Each case study uses machine learning techniques together with metalogic programming.

1 Introduction

Logic-based AI systems perform logical inferences to get solutions given input formulas. Such systems have been developed in the field of automated theorem proving or logic programming [10]. In those systems, however, a logic used in the system is specified and built-in by human engineers. Our question in this paper is whether it is possible to develop artificial (general) intelligence that automatically produces a logic underlying any given data set.

In his argument on "learning machines" in [14], Alan Turing wrote:

> Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain [14, p. 456].

According to Piaget's theory of cognitive development, children begin to understand logical or rational thought at age around seven [12]. If one can develop AI that automatically acquires a logic of human reasoning, it verifies Turing's assumption that a child's brain can grow into an adult's one by learning an appropriate logic. Recent advances in robotics argue possibilities of robots' recognizing objects in the world, categorizing concepts, and associating names to them (*physical symbol grounding*) [3]. Once robots successfully learn concepts

and associate symbols to them, the next step is to learn relations between concepts and logical or physical rules governing the world.

In this study, we will capture learning logics as a problem of *inductive learning*. According to [9], "(t)he goal of (inductive) inference is to formulate plausible general assertions that explain the given facts and are able to predict new facts. In other words, inductive inference attempts to derive a complete and correct description of a given phenomenon from specific observations of that phenomenon or of parts of it" [9, p. 88]. A logic provides a set of axioms and inference rules that underlie sentences representing the world. Then given a set of sentences representing the world, one could inductively construct a logic governing the world. This is in fact a work for mathematicians who try to find an axiomatic system that is sound and complete with respect to a given set of theorems. Induction has been used as an inference mechanism of machine learning, while little study has been devoted to the challenging topic of learning logics.

In this paper, we first describe an abstract framework for learning logics based on inductive learning. Next we provide two simple case studies: learning deductive inference rules and learning cellular automata (CAs) rules. In the former case, the problem of producing deductive inference rules from formulas and their logical consequences is considered. In the second case, the problem of producing transition rules from CA configurations is considered. In each case, we use machine learning techniques together with metalogic programming. The rest of this paper is organized as follows. Section 2 introduces an abstract framework for learning logics. Section 3 presents a case of learning deductive inference rules and Section 4 presents a case of learning CA rules. Section 5 discusses further issues and Section 6 summarizes the paper.

2 Learning Logics

To consider the question "*Can machines learn logics?*", suppose the following problem. There is an agent $\mathcal{A}$ and a machine $\mathcal{M}$. The agent $\mathcal{A}$, which could be a human or a computer, is capable of deductive reasoning: it has a set $\mathcal{L}$ of axioms and inference rules in classical logic. Given a (finite) set S of formulas as an input, the agent $\mathcal{A}$ produces a (finite) set of formulas T such that $T \subset Th(S)$ where $Th(S)$ is the set of logical consequences of S. On the other hand, the machine $\mathcal{M}$ has no axiomatic system for deduction, while it is equipped with a machine learning algorithm $\mathcal{C}$. Given input-output pairs $(S_1, T_1), \ldots, (S_i, T_i), \ldots$ (where $T_i \subset Th(S_i)$) of $\mathcal{A}$ as an input to $\mathcal{M}$, the problem is whether one can develop an algorithm $\mathcal{C}$ which successfully produces an axiomatic system $\mathcal{K}$ for deduction. An algorithm $\mathcal{C}$ is *sound* wrt $\mathcal{L}$ if it produces an axiomatic system $\mathcal{K}$ such that $\mathcal{K} \subseteq \mathcal{L}$. An algorithm $\mathcal{C}$ is *complete* wrt $\mathcal{L}$ if it produces an axiomatic system $\mathcal{K}$ such that $\mathcal{L} \subseteq \mathcal{K}$. Designing a sound and complete algorithm $\mathcal{C}$ is called a problem of *learning logics* (Figure 1). In this framework, an agent $\mathcal{A}$ plays the role of a teacher who provides training examples representing premises along with entailed consequences. The output $\mathcal{K}$ is refined by incrementally providing examples. We consider a deduction system $\mathcal{L}$ while it could be a system of

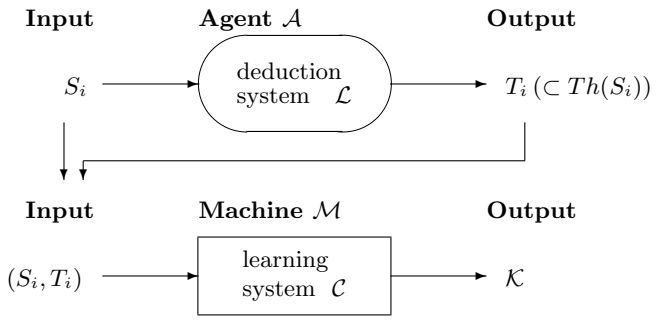

Fig. 1. Learning Logics

arbitrary logic, e.g. nonmonotonic logic, modal logic, fuzzy logic, as far as it has a formal system of inference. Alternatively, we can consider a framework in which a teacher agent $\mathcal{A}$ is absent. In this case, given input-output pairs (S_i, T_i) as data, the problem is whether a machine $\mathcal{M}$ can find an unknown logic (or axiomatic system) that produces a consequence T_i from a premise S_i.

The abstract framework provided in this section has challenging issues of AI including the questions:

1. *Can we develop a sound and complete algorithm $\mathcal{C}$ for learning a classical or non-classical logic $\mathcal{L}$?*
2. *Is there any difference between learning axioms and learning inference rules?*
3. *Does a machine $\mathcal{M}$ discover a new axiomatic system $\mathcal{K}$ such that $\mathcal{K} \vdash F$ iff $\mathcal{L} \vdash F$ for any formula F?*

The first question concerns the possibility of designing machine learning algorithms that can learn existing logics from given formulas. The second question concerns differences between learning Gentzen-style logics and Hilbert-style logics. The third question is more ambitious: it asks the possibility of AI's discovering new logics that are unknown to human mathematicians.

In this paper, we provide simple case studies concerning the first question. To this end, we represent a formal system $\mathcal{L}$ using *metalogic programming* which allows object-level and meta-level representation to be amalgamated [2].

3 Learning Deductive Inference Rules

The preceding section provided an abstract framework for learning logics. This section considers a simple case of the problem. Suppose a set S of atomic formulas which contains atoms with the predicate *hold*. Each atom in S is in the form $hold(F)$ where F is a formula in propositional logic. Hence, *hold* is a *meta-predicate*, $hold(F)$ is a *meta-atom*, while F is an *object-level* formula. A *rule* has the form:

$$A \leftarrow \Gamma \tag{1}$$

where A is a meta-atom and Γ is a conjunction of meta-atoms. Given a rule R of the form (1), A is called the *head* of R and Γ is called the *body* of R. The atom A is also represented by $head(R)$ and the set of atoms in Γ is represented by $body(R)$. In what follows, a meta-predicate or a meta-atom is simply called a predicate or an atom, and an object-level formula is called a formula as far as no confusion arises. We consider an agent $\mathcal{A}$ with an inference system $\mathcal{L}$ that performs the following inference:

$$\text{from } \ hold(p) \ \text{ and } \ hold(p \supset q) \ \text{ infer } \ hold(q)$$

where p and q are propositional variables. In this case, given a finite set S of atoms as an input, $\mathcal{A}$ outputs the set:

$$T = S \cup \{\, hold(q) \mid hold(p) \in S \ \text{ and } \ hold(p \supset q) \in S \,\}.$$

We now consider the machine $\mathcal{M}$ that can produce deductive inference rules from S and T as follows. Given each pair (S, T) as an input, we first consider a learning system $\mathcal{C}$ which constructs a rule:

$$A \leftarrow \bigwedge_{B_i \in S} B_i \tag{2}$$

where $A \in T \setminus S$. The rule (2) represents that an atom A in $T \setminus S$ is derived using atoms in S. For example, given the set:

$$S = \{\, hold(p),\ hold(r),\ \ hold(p \supset q),\ \ hold(p \supset r),\ \ hold(r \supset s) \,\},$$

two atoms $hold(q)$ and $hold(s)$ are in $T \setminus S$. Then the following two rules are constructed by (2):

$$hold(q) \leftarrow hold(p) \wedge hold(r) \wedge hold(p \supset q) \wedge hold(p \supset r) \wedge hold(r \supset s).$$
$$hold(s) \leftarrow hold(p) \wedge hold(r) \wedge hold(p \supset q) \wedge hold(p \supset r) \wedge hold(r \supset s).$$

The body of each rule contains atoms which do not contribute to deriving the atom in the head. To distinguish atoms which contribute to deriving the consequence, the agent $\mathcal{A}$ is used as follows. For a pair (S, T) from $\mathcal{A}$ such that $T \setminus S \neq \emptyset$, assume that a rule R of the form (2) is constructed. Then, select a subset S_i of S and give it as an input to $\mathcal{A}$. If its output T_i still contains the atom A of $head(R)$, replace R with

$$A \leftarrow \bigwedge_{B_i \in S_i} B_i.$$

By continuing this process, find a minimal set S_j satisfying $A \in T_j$. Such S_j contains atoms that are necessary and sufficient for deriving atoms in $T_j \setminus S_j$. In the above example, there is the unique minimal set:

$$S_1 = \{\, hold(p),\ \ hold(p \supset q) \,\}$$

that satisfies $hold(q) \in T_1$, and there are two minimal sets that contain the atom $hold(s)$ in their outputs:

$$S_2 = \{\, hold(r),\ \ hold(r \supset s)\,\},$$
$$S_3 = \{\, hold(p),\ \ hold(p \supset r),\ \ hold(r \supset s)\,\}.$$

Then the following three rules are obtained by replacing S with S_i in (2):

$$hold(q) \leftarrow hold(p) \wedge hold(p \supset q). \tag{3}$$
$$hold(s) \leftarrow hold(r) \wedge hold(r \supset s). \tag{4}$$
$$hold(s) \leftarrow hold(p) \wedge hold(p \supset r) \wedge hold(r \supset s). \tag{5}$$

The rules (3) and (4) represent *Modus Ponens*, and (5) represents *Multiple Modus Ponens*. As such, unnecessary atoms in the body of a rule are eliminated by the *minimization* technique.

Unnecessary atoms in the bodies are also eliminated using the *generalization* technique developed in [6].[3] Suppose an agent $\mathcal{A}$ with an inference system $\mathcal{L}$ that performs the following inference:

$$\text{from}\ \ hold(p \vee q)\ \ \text{and}\ \ hold(\neg q)\ \ \text{infer}\ \ hold(p).$$

In this case, given a finite set S of atoms as an input, $\mathcal{A}$ outputs the set:

$$T = S \cup \{\, hold(p)\ \mid\ hold(p \vee q) \in S\ \ \text{and}\ \ hold(\neg q) \in S\,\}.$$

Given a sequence of input-output pairs from the agent $\mathcal{A}$, the machine $\mathcal{M}$ constructs a rule R of the form (2) each time it receives a new pair (S_i, T_i) from $\mathcal{A}$. Suppose two rules R and R' such that (i) $head(R) = head(R')$; (ii) there is a formula F such that $hold(F) \in body(R)$ and $hold(\neg F) \in body(R')$; and (iii) $(body(R') \setminus \{hold(\neg F)\}) \subseteq (body(R) \setminus \{hold(F)\})$. Then, a generalized rule of R and R' (upon F) is obtained as

$$A \leftarrow \bigwedge_{B_i \in (body(R) \setminus \{hold(F)\})} B_i.$$

For example, given the two pairs, (S_1, T_1) and (S_2, T_2), where

$$S_1 = \{\, hold(p \vee q),\ hold(\neg q),\ hold(r)\,\},$$
$$T_1 = \{\, hold(p \vee q),\ hold(\neg q),\ hold(r),\ hold(p)\,\},$$
$$S_2 = \{\, hold(p \vee q),\ hold(\neg q),\ hold(\neg r)\,\},$$
$$T_2 = \{\, hold(p \vee q),\ hold(\neg q),\ hold(\neg r),\ hold(p)\,\},$$

the two rules are obtained as:

$$hold(p) \leftarrow hold(p \vee q) \wedge hold(\neg q) \wedge hold(r),$$
$$hold(p) \leftarrow hold(p \vee q) \wedge hold(\neg q) \wedge hold(\neg r).$$

[3] The technique is used for a logic with the law of excluded middle.

Then the generalization of them is:

$$hold(p) \leftarrow hold(p \vee q) \wedge hold(\neg q).$$

This rule represents *Disjunctive Syllogism*.

These procedures are applicable to learning one-step deduction rules such that

$$hold(\neg p) \leftarrow hold(\neg q) \wedge hold(p \supset q). \quad \text{(Modus Tollens)}$$
$$hold(p \supset r) \leftarrow hold(p \supset q) \wedge hold(q \supset r). \quad \text{(Hypothetical Syllogism)}$$

We can also obtain a rule for *abductive inference* [11] by this method. For example, given the pair $(S, T) = (\{hold(q), hold(p \supset q)\}, \{hold(q), hold(p \supset q), hold(p)\})$, we can construct *the Fallacy of Affirming the Consequent*:

$$hold(p) \leftarrow hold(q) \wedge hold(p \supset q).$$

In this way, the method in this section could be used for learning non-deductive inferences.

4 Learning CA Rules

In this section, we address another example of learning logics. *Cellular automata* (CAs) [15] are discrete and abstract computational models that have been used for simulating various complex systems in the real world. A CA consists of a regular grid of *cells*, each of which has a finite number of possible *states*. The state of each cell changes synchronously in discrete time steps (or *generations*) according to a local and identical *transition rule*. The state of a cell in the next time step is determined by its current state and the states of its surrounding cells (called *neighbors*). The collection of all cellular states in the grid at some time step is called a *configuration*. An *elementary* CA consists of a one-dimensional array of (possibly infinite) cells, and each cell has one of two possible states 0 or 1. A cell and its two adjacent cells form a neighbor of three cells, so there are $2^3 = 8$ possible patterns for neighbors. A transition rule describes for each pattern of a neighbor, whether the central cell will be 0 or 1 at the next time step. Then $2^8 = 256$ possible rules are considered and 256 elementary CAs are defined accordingly. Stephen Wolfram gave each rule a number 0 to 255 (called the *Wolfram code*), and analyzed their properties [15]. The evolution of an elementary CA is illustrated by starting with the initial configuration in the first row, the configuration at the next time step in the second row, and so on. Figure 2 shows the Rule 30 and one of its evolution where the black cell represents the state 1 and the white cell represents the state 0. The figure shows the first 16 generations of the Rule 30 starting with a single black cell. It is known that the Rule 30 displays aperiodic and random patterns in a chaotic manner.

Each transition rule is considered a logic of CA, that is, every pattern appearing in a configuration is governed by one transition rule. Then we consider

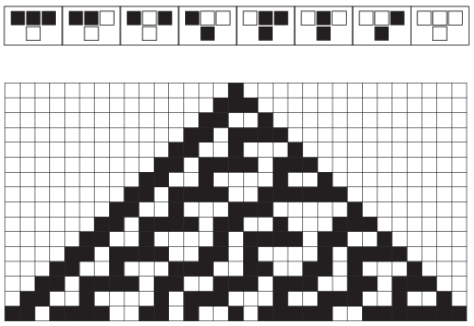

Fig. 2. Evolution of patterns by the Rule 30

the problem of producing a transition rule from input configurations. Such a problem is known as the *identification problem* of CAs [1]. In what follows, we consider the problem of learning the Wolfram's Rule 30 from a series of configurations. In an elementary CA, a configuration at a time step t is represented by a (possibly infinite) sequence of cells $\langle \cdots x_{i-1}^t\, x_i^t\, x_{i+1}^t \cdots \rangle$ where x_i^t represents a state of a cell x_i at a time step t. For example, the initial configuration of Figure 2 is represented by

$$\langle \cdots x_{i-1}^0\, x_i^0\, x_{i+1}^0 \cdots \rangle = \langle \cdots 010 \cdots \rangle$$

where the central black cell at the time step 0 is represented by $x_i^0 = 1$. Likewise, the configuration at the time step 2 is represented by

$$\langle \cdots x_{i-3}^2\, x_{i-2}^2\, x_{i-1}^2\, x_i^2\, x_{i+1}^2\, x_{i+2}^2\, x_{i+3}^2 \cdots \rangle = \langle \cdots 0110010 \cdots \rangle.$$

We represent the state of a cell at each time step by an atom as: $\mathit{hold}(x_i^t)$ if $x_i^t = 1$ and $\mathit{hold}(\neg x_i^t)$ if $x_i^t = 0$. Then the initial configuration of Figure 2 is represented by the (infinite) set of atoms:

$$\{\ldots, \mathit{hold}(\neg x_{i-1}^0),\ \mathit{hold}(x_i^0),\ \mathit{hold}(\neg x_{i+1}^0), \ldots\}.$$

To cope with the problem using a finite set, we consider the five cells:

$$S^t = \langle\, x_{i-2}^t\, x_{i-1}^t\, x_i^t\, x_{i+1}^t\, x_{i+2}^t \,\rangle$$

in each time step. Table 1 represents evolution of those five cells in the first four time steps.

Corresponding to the framework provided in Section 2, an agent $\mathcal{A}$ produces S^{t+1} from an input S^t. Given input-output pairs $(S^0, S^1), \ldots, (S^t, S^{t+1}), \ldots$ of $\mathcal{A}$ as an input to a machine $\mathcal{M}$, the problem is whether $\mathcal{M}$ can identify the transition rule of this CA. For a pair of configurations (S^0, S^1), the machine $\mathcal{M}$ produces a rule R that represents the states of the cell x_j^0 $(i-1 \leq j \leq i+1)$

Table 1. Evolution of $\langle\, x^t_{i-2}\, x^t_{i-1}\, x^t_i\, x^t_{i+1}\, x^t_{i+2}\,\rangle$

step	x^t_{i-2}	x^t_{i-1}	x^t_i	x^t_{i+1}	x^t_{i+2}
t=0	0	0	1	0	0
t=1	0	1	1	1	0
t=2	1	1	0	0	1
t=3	1	0	1	1	1

and its neighbors in the body of R and represents the state of the cell x^1_j in the head of R. There are three such rules:

$$hold(x^1_{i-1}) \leftarrow hold(\neg x^0_{i-2}) \wedge hold(\neg x^0_{i-1}) \wedge hold(x^0_i).$$
$$hold(x^1_i) \leftarrow hold(\neg x^0_{i-1}) \wedge hold(x^0_i) \wedge hold(\neg x^0_{i+1}).$$
$$hold(x^1_{i+1}) \leftarrow hold(x^0_i) \wedge hold(\neg x^0_{i+1}) \wedge hold(\neg x^0_{i+2}).$$

Similarly, given a pair of configurations (S^1, S^2), the machine $\mathcal{M}$ produces the following three rules:

$$hold(x^2_{i-1}) \leftarrow hold(\neg x^1_{i-2}) \wedge hold(x^1_{i-1}) \wedge hold(x^1_i).$$
$$hold(\neg x^2_i) \leftarrow hold(x^1_{i-1}) \wedge hold(x^1_i) \wedge hold(x^1_{i+1}).$$
$$hold(\neg x^2_{i+1}) \leftarrow hold(x^1_i) \wedge hold(x^1_{i+1}) \wedge hold(\neg x^1_{i+2}).$$

The following two rules are respectively obtained by (S^3, S^4) and (S^6, S^7):

$$hold(\neg x^4_{i-1}) \leftarrow hold(x^3_{i-2}) \wedge hold(\neg x^3_{i-1}) \wedge hold(x^3_i).$$
$$hold(\neg x^7_i) \leftarrow hold(\neg x^6_{i-1}) \wedge hold(\neg x^6_i) \wedge hold(\neg x^6_{i+1}).$$

Since a transition rule does not change during the evolution and it is equally applied to each cell, the above eight rules are rewritten as

$$hold(x^{t+1}_i) \leftarrow hold(\neg x^t_{i-1}) \wedge hold(\neg x^t_i) \wedge hold(x^t_{i+1}). \quad (6)$$
$$hold(x^{t+1}_i) \leftarrow hold(\neg x^t_{i-1}) \wedge hold(x^t_i) \wedge hold(\neg x^t_{i+1}). \quad (7)$$
$$hold(x^{t+1}_i) \leftarrow hold(x^t_{i-1}) \wedge hold(\neg x^t_i) \wedge hold(\neg x^t_{i+1}). \quad (8)$$
$$hold(x^{t+1}_i) \leftarrow hold(\neg x^t_{i-1}) \wedge hold(x^t_i) \wedge hold(x^t_{i+1}). \quad (9)$$
$$hold(\neg x^{t+1}_i) \leftarrow hold(x^t_{i-1}) \wedge hold(x^t_i) \wedge hold(x^t_{i+1}). \quad (10)$$
$$hold(\neg x^{t+1}_i) \leftarrow hold(x^t_{i-1}) \wedge hold(x^t_i) \wedge hold(\neg x^{t+1}_{i+1}). \quad (11)$$
$$hold(\neg x^{t+1}_i) \leftarrow hold(x^t_{i-1}) \wedge hold(\neg x^t_i) \wedge hold(x^t_{i+1}). \quad (12)$$
$$hold(\neg x^{t+1}_i) \leftarrow hold(\neg x^t_{i-1}) \wedge hold(\neg x^t_i) \wedge hold(\neg x^t_{i+1}). \quad (13)$$

The eight rules (6)–(13) represent the transition rule of the Rule 30. Further, we get the following rules:

$$\begin{aligned}
&hold(x_i^{t+1}) \leftarrow hold(\neg x_{i-1}^t) \wedge hold(x_i^t). && \text{(by (7) and (9))}\\
&hold(x_i^{t+1}) \leftarrow hold(\neg x_{i-1}^t) \wedge hold(x_{i+1}^t). && \text{(by (6) and (9))}\\
&hold(\neg x_i^{t+1}) \leftarrow hold(x_{i-1}^t) \wedge hold(x_i^t). && \text{(by (10) and (11))}\\
&hold(\neg x_i^{t+1}) \leftarrow hold(x_{i-1}^t) \wedge hold(x_{i+1}^t). && \text{(by (10) and (12))}
\end{aligned}$$

Those rules are finally summarized as:

$$\begin{aligned}
hold(x_i^{t+1}) \leftarrow\ & (hold(x_{i-1}^t) \wedge hold(\neg x_i^t) \wedge hold(\neg x_{i+1}^t))\\
& \vee (hold(\neg x_{i-1}^t) \wedge (hold(x_i^t) \vee hold(x_{i+1}^t))). \qquad (14)\\
hold(\neg x_i^{t+1}) \leftarrow\ & (hold(\neg x_{i-1}^t) \wedge hold(\neg x_i^t) \wedge hold(\neg x_{i+1}^t))\\
& \vee (hold(x_{i-1}^t) \wedge (hold(x_i^t) \vee hold(x_{i+1}^t))). \qquad (15)
\end{aligned}$$

The rules (14) and (15) represent the Wolfram's Rule 30.

Learning elementary CA rules is implemented in [6]. Learning elementary CA rules is simple because it is one-dimensional, two-state, and has the fixed neighborhood size. On the other hand, identifying CA rules in practice is difficult because configurations are observed phenomena in the real-world and there is no teacher agent $\mathcal{A}$ in general.

5 Discussion

This paper argues the possibility of discovering logics using AI. Logic is considered as meta-mathematics here, so the task is to find meta-laws given pairs of premises and consequences in mathematical or physical domain. On the other hand, discovering mathematical theorems or scientific laws in the objective theories has been studied in AI. For instance, Lenat [7] develops the *automated mathematician* (AM) that automatically produces mathematical theorems including Goldbach's Conjecture and the Unique Factorization Theorem. Schmidt and Lipson [13] develop AI that successfully deduces the laws of motion from a pendulum's swings without a shred of knowledge about physics or geometry. To the best of our knowledge, however, there are few studies that aim at discovering logics or meta-theorems.

In Section 2 we address an abstract framework of learning formal systems based on logics. An interesting question is whether the same or a similar framework can be applied for learning non-logical systems. In this case, a set of input-output pairs (or premise-consequence pairs) are not given from a teacher agent $\mathcal{A}$ in general, but can be implicitly hidden in log files of dynamic systems or in dialogues with unknown agents. The machine $\mathcal{M}$ has to identify those input-output relations automatically to output a set of meta-theoretical inference rules for the domain or inference patterns of those agents. Non-logical inferences are also used in *pragmatics* [8]. In conversation or dialogue, the notion of *conversational implicature* [4] is known as a pragmatic inference to an implicit meaning

of a sentence that is not actually uttered by a speaker. For instance, if a speaker utters the sentence "I have two children", it normally implicates "I do *not* have *more than* two children". This is called a *scalar implicature* which says that a speaker implicates the negation of a semantically stronger proposition than the one asserted. Given a collection of dialogues, a question is whether a machine can automatically acquire pragmatic rules of inference that interpret implicit meaning behind utterance. Once such a non-logical inference is learned, it must be refined or revised through a continuous, cyclic process between evidences and abduction on meta-theoretical relations [5]. The process would thus introduce a dynamics of incremental perfection of theories. To realize such a system, further extension and elaboration of the framework provided in this paper are needed and much work are left for future research.

6 Summary

Answering the question "can machines learn logics?" is one of the challenging topics in artificial general intelligence. We argued the possibility of realizing such AI and provided some case studies. A number of questions remain open, for instance, whether the goal is achieved using existing techniques of machine learning or AI; which logics are to be learned and which logics are not; whether non-logical rules are learned as well, etc. Exploring those issues would contribute to better understanding human intelligence and take us one step closer to realizing "strong AI." Although the abstract framework provided in this paper is conceptual and case studies are rather simple, the current study serves as a kind of base-level and would contribute to opening the topic.

References

1. Adamatzky, A.: Identification of Cellular Automata. Taylor & Francis, London (1994)
2. Bowen, K.A., Kowalski, R.A.: Amalgamating language and metalanguage in logic programming. In: Clark, K., Tarnlund, S.A. (eds.) Logic Programming, pp. 153–172. Academic Press (1983)
3. Coradeschi, S., Loutfi, A., Wrede, B.: A short review of symbol grounding in robotic and intelligent systems. KI - Kunstliche Intelligenz 27(2), 129–136 (2013)
4. Grice, H.P.: Logic and conversation. In: Cole, P., Morgan, J. (eds.) Syntax and Semantics, 3: Speech Acts, pp. 41–58. Academic Press (1975)
5. Inoue, K.: Meta-level abduction. IFCoLog Journal of Logic and their Applications (in print) (2015)
6. Inoue, K., Ribeiro, T., Sakama, C.: Learning from interpretation transition. Machine Learning 94(1), 51–79 (2014)
7. Lenat, D.B.: On automated scientific theory formation: a case study using the am program. In: Hayes, J.E., Michie, D., Mikulich, O.I. (eds.) Machine Intelligence, vol. 9, pp. 251–283. Ellis Horwood (1979)
8. Levinson, S.C.: Pragmatics. Cambridge University Press (1983)

9. Michalski, R.S.: A theory and methodology of inductive learning. In: Michalski, R.S., et al. (eds.) Machine Learning: An Artificial Intelligence Approach, pp. 83–134. Morgan Kaufmann (1983)
10. Minker, J. (ed.): Logic-based Artificial Intelligence. Kluwer Academic (2000)
11. Peirce, C.S.: Elements of logic. In: Hartshorne, C., Weiss, P. (eds.) Collected Papers of Charles Sanders Peirce, vol. II. Harvard University Press (1932)
12. Piaget, J.: Main Trends in Psychology. London: Allen & Unwin (1973)
13. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science 324 (2009)
14. Turing, A.M.: Computing machinery and intelligence. Mind 59, 433–460 (1950)
15. Wolfram, S.: Cellular Automata and Complexity. Westview Press (1994)