# Learning Stochastic Logical Automaton

Hiroaki Watanabe and Stephen Muggleton

Imperial College London, 180 Queen's Gate, London SW7 2AZ, UK
Tel.: +44 (0)20 7594 8287; Fax: +44 (0)20 7581 8024
{hw3, shm}@doc.ic.ac.uk

**Abstract.** This paper is concerned with algorithms for the logical generalisation of probabilistic temporal models from examples. The algorithms combine logic and probabilistic models through inductive generalisation. The inductive generalisation algorithms consist of three parts. The first part describes the graphical generalisation of state transition models. State transition models are generalised by applying state mergers. The second part involves symbolic generalisation of logic programs which are embedded in each states. Plotkin's LGG is used for symbolic generalisation of logic programs. The third part covers learning of parameters using statistics derived from the input sequences. The state transitions are unobservable in our settings. The probability distributions over the state transitions and actions are estimated using the EM algorithm. As an application of these algorithms, we learn chemical reaction rules from *StochSim*, the stochastic software simulator of biochemical reactions.

## 1 Introduction

Logical Induction from temporal observations is a challenging problem since the observations could contain uncertainties in many cases. One way to tackle the uncertainties is to derive statistics from the observations. If we express the statistical knowledge explicitly, we would need to combine stochastic and logical knowledge representations.

Logic-based AI has been studying logical representations of dynamic aspects of temporal data since McCarthy and Hayes proposed Situation Calculus [1] in first-order logic where dynamic changing world is expressed in *time-sliced* declarative representations. In the Inductive Logic Programming (ILP) literatures, a few studies have been reported from the time-sliced representation point of view [5, 6].

*Automata-based* representations have also been studied in computer science. One of the merits of the automata-based representations under uncertainties would be the applicability of the well-studied statistical learning algorithms such as EM-algorithm for Hidden Markov Models [2]. In this paper, we combine logic and probability model from the automata-based representation point of view. More precisely, we introduce a logical extension of stochastic non-deterministic finite automata.

We also present its induction algorithm. Dynamic aspects of the model are generalised by the state merging technique [10] whereas static (or symbolic)

T. Washio et al. (Eds.): JSAI 2005 Workshops, LNAI 4012, pp. 201–211, 2006.

knowledge are generalised by Plotkin's LGG [12]. Regarding learning automata, inductive inferences of automata are one of the well studied area in the research of computational complexity theory. Most of the previous works are negative (that is, non polynomial time learnable) [13, 14, 15] except [16]. These previous works suggest us to introduce a relevant constraint over the hypotheses space of our inductive algorithm in order to obtain an efficiency.

The paper is organised as follows. Section2 introduces our logical automata. Section3 explains the overview of the inductive algorithms of our model. Section4 contains our initial attempt to learn chemical reaction rules from the biochemical simulator *StochSim*[9]. We discuss some related works in Section5. Discussions and future work conclude the paper in Section6.

## 2 Logical Automata

### 2.1 Definitions

Let us start from the definition of a non-deterministic finite automaton (NFA) that provides the basic idea of state transitions for our model.

**Definition 1.** ***(NFA):*** *A non-deterministic finite automaton is a 5-tuple* $NFA = (S, \Sigma, T, S_0, G)$ *where* $S$ *is a finite set of states,* $\Sigma$ *is a finite alphabet,* $T$ *is a transition function,* $S_0$ *is a set of initial states, and* $G$ *is a set of accept states.*

We extend NFA logically next. Assume a first-order definite clausal language $L$ is given. Let $f$ be a definite clause in $L$. Then $F$, a finite set of clauses, is called *theory*. We represent an observation of a dynamic world as a sequence of ground theories as follows:

**Definition 2.** ***(Logical Sequence):*** *logical sequence is defined as*

$$O_0 A_0 O_1 ... O_{n-1} A_{n-1} O_n$$

*where* $O_i$ *is the ground theory in* $L$ *that describes the observed facts at time* $i$. $A_i$ *is also the ground theory in* $L$ *for the action (or input) at time* $i$.

We call $O_i A_i O_{i+1}$ a *unit* of the logical sequence. A graphical representation of the logical sequence is shown in Fig.1.

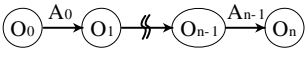


**Fig. 1.** A logical sequence

In a state transition system, a *state* is treated as a snapshot of the dynamic world where the state gives an *interpretation* (true or false) of each ground atom in the world.

**Definition 3.** ***(Logical State):*** *A logical state $q$ is a pair $(n, F)$ where $n \in \mathbb{N}$ is a label for a theory $F$ in $L$.*

We introduce a logical edge and action as follows:

**Definition 4.** ***(Logical Edge and Action):*** *A logical edge is an edge between two logical states. A logical action (or input) is the set of ground theories embedded in the logical edges. All of the logical actions are denoted by $\mathcal{E}$.*

Next, we consider relations in the state transition function. In the literatures of automata theory, *equivalence relation* is usually employed between each input alphabet and the members of $\Sigma$. For example, let $q$ and $q' = \sigma(q, a)$ be a current state and a state transition function from the state $q$ to $q'$ caused by the input $a$ respectively. Now let us assume we receive an input alphabet at $q$. If the input *is equals to* $a$, the resulted state is $q'$. That is, we check the equivalence relation between the input alphabet and $a$.

In this paper, we introduce *generality relations* instead of the equality between the logical states (respectively logical edges) and the observed facts (respectively observed actions) from a logical point of view. For measuring the generality relations between theories, logical entailment would be a candidate, however, we employ the subsumption order to avoid the semi-decidability of logical entailment.

**Definition 5.** ***(Clause Subsumption):*** *Clause $f_1$ subsumes clause $f_2$, $f_1 \succeq f_2$, iff there exists a substitution $\theta$ such that $f_1\theta \subseteq f_2$.*

**Definition 6.** ***(Theory Subsumption [12]):*** *Theory $F_1$ subsumes theory $F_2$, $F_1 \succeq_T F_2$, iff $\forall C_2 \in F_2 \;\; \exists C_1 \in F_1 \;\; C_1 \succeq C_2$.*

Under the above theory subsumption order, we define the *logically special* state transition function between the logical states.

**Definition 7.** ***(Special State Transition):*** *For a given unit of logical sequence $O_i \;\; A_i \;\; O_{i+1}$, the state transition $q' = \sigma(q, E_i)$ between the logical states $q = (n, F)$ and $q' = (n', F')$ occurs iff $F \succeq_T O_i$, $E_i \succeq_T A_i$ and $F' \succeq_T O_{i+1}$.*

Intuitively Special State Transition accepts the unit if the unit is more *special* than the related logical states and logical edge. Here $F$ and $E_i$ could be viewed as the *prior* conditions, and $F'$ as the *posterior* condition.

The scope of the first-order variables is expanded to the adjacent logical states:

**Definition 8.** ***(Scope of Variables):*** *For the state transition from the logical states $q = (n, F)$ to $q' = (n', F')$ by the logical action $E_i$, let $V_F$, $V_E$, and $V_{F'}$ be the sets of the first-order variables that appear in $F$, $E_i$, and $F'$ respectively. Then the scope of the variables are within $V_F \cup V_E \cup V_{F'}$.*

This extension allows us to represent the relations between the first-order variables associated with the special state transition.

A Probabilistic Logical Automaton for Special inputs (PLAS) is a nondeterministic finite automaton whose states and state transitions are defined as logical states and probabilistic version of special state transitions respectively as follows:

**Definition 9.** ***(PLAS):*** *PLAS is a quintuplet* $\langle Q, L, \sigma, I, G \rangle$ *where* $Q$ *is a finite set of logical states,* $L$ *is a definite language,* $\sigma$ *is a mapping defining the probabilistic special transition function,* $\sigma : Q \times L \times Q \rightarrow [0, 1]$*,* $I$ *is a mapping defining the initial probability of each state,* $I : Q \rightarrow [0, 1]$*, and* $G$ *is a mapping defining the final probability of each state,* $G : Q \rightarrow [0, 1]$*.*

**Definition 10.** ***(Acceptance):*** *Let* $M$ *be an PLAS such that* $\langle Q, L, \sigma, I, G \rangle$*, and* $X$ *be a logical sequence in* $L$*.* $M$ *accepts the logical sequence* $X$ *if there exist a sequence of the logical states* $q_0, ..., q_n$ $(q_i \in Q)$ *such that:* $I(q_0) = (0, 1]$*,* $q_i = \sigma(q_{i-1}, x_i)$ *for i=1,...,n, and* $G(q_n) = (0, 1]$*. Then we call the sequence of the logical states* $q_0...q_n$ *an acceptance.*

The brief comparisons of Probabilistic NFA (PNFA) and PLAS are given in Table 1.

**Table 1.** Comparisons of PNFA and PLAS

| | **PNFA** | **PLAS** |
|---|---|---|
| **Language** | alphabet | definite theory language |
| **Input** | word | logical sequence |
| **Relation** | equality | speciality |
| **Acceptance** | sequence of states | sequence of logical states |

### 2.2 Semantics of Logical States

In the previous section, we introduce the first-order logical extension of NFA under the theory subsumption order. This extension would bring many benefits to graphical temporal knowledge representations since we could combine the automata-based algorithms and logical knowledge representation smoothly. Let us investigate a semantic aspect of PLAS by focusing on the generality orders. Let LM($F$) be the least Herbrand Model of theory $F$. We present the following theorem in order to clarify an essential difference between propositional approach and our logical approach.

**Theorem 1.** *A PLAS accepts a logical sequence* $O_0, A_0, ..., O_n$ *such that its acceptance is* $q_0...q_n$ *where* $q_i = (n_i, F_i)$ $(i = 0, ..., n)$*, then*

$$LM(O_i) \subseteq LM(F_i).$$

*Proof.* $F \succeq_T O_i$ implies $F \models O_i$. Thus $LM(O_i) \subseteq LM(F)$ from the definition of the entailment relation.

Fig.2 illustrates an example of the inclusion relations between the logical states and logical sequences in PLAS such that the acceptance for the observation $O_0A_0O_1A_1O_2$ is $S_0S_1S_3$. Recall that the least Herbrand Model of the theory is defined by a set of ground atoms. Therefore the inclusion relations explain the semantic aspect of PLAS at the ground atom level. Note that the inclusion relations

$$\mathrm{LM}(A_j) \subseteq \mathrm{LM}(E_i)$$

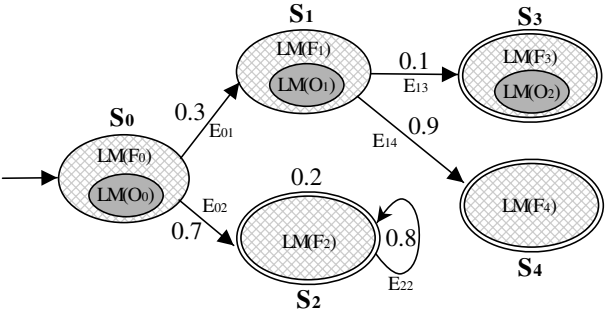


**Fig. 2.** Inclusion Relations in PLAS

also hold between the observed action $A_j$ and the logical action $E_i \in \mathcal{E}$ where $E_i$ is the logical action embedded in the logical edge between $q_j$ and $q_{j+1}$ of the acceptance $q_0 ... q_n$.

### 2.3 Inference in PLAS

PLAS could be applied for the probabilistic inference tasks such as *filtering*, *prediction*, *smoothing*, and *most likely explanation* [22]. However, we should recall that the inferences in PLAS would need the overheads of computing for the theory subsumption checks at each logical node and logical edge comparing with the standard inference algorithms [22]. Subsumption is decidable, however, it is an NP-complete problem [23]. Therefore the theories embedded in the logical nodes and edges must be restricted [24] for designing tractable inference algorithms.

For example of the inference tasks in PLAS, let us consider the filtering. The observations for PLAS could be partial observations as the inclusion relations show in the previous section. PLAS defines the probability distribution over *belief states* given a set of the logical sequences. In Fig.2, let us assume that the given logical sequence also has another acceptance $S_0 S_2 S_2$. Then the logical sequence gives

$$Pr(S3 \mid O_0 A_0 O_1 A_1 O_2) = 0.3 \times 0.1 = 0.03$$
$$Pr(S2 \mid O_0 A_0 O_1 A_1 O_2) = 0.7 \times 0.8 = 0.56.$$

## 3 Learning Stochastic Logical Automata

Our next interest is to learn the stochastic logical automata from observations. The above definitions indicate that PLAS could be learned by *generalising* the given inputs. Our generalisation algorithm combines logic and probabilistic models through inductive generalisation.

### 3.1 Setting

**Given**

- Positive Examples **E**: A set of Logical Sequences
- Background Knowledge **BK**: A set of ground atoms

**Learn**

- A PLAS that accepts **E** by the special state transitions related with **BK**.

### 3.2 Overview of the Algorithms

Our inductive generalisation algorithms consist of three parts. The first part describes the graphical generalisation of state transition models. Assume two logical sequences are given as positive examples (Fig.3). These sequences could be viewed as a state transition model. In our algorithm, the state transition models are generalised by applying state mergers. For example, in Fig.3 if state 2 and state 8 are merged, the new state 10 in Fig.4 is newly generated. The related transition functions are altered through the generalisation process as shown in Fig.4.

The second part involves symbolic generalisation of theories which are embedded in each states. Plotkin's LGG is used for symbolic generalisation of the theories. For example, the two theories in state 2 and state 8 (Fig.3) are generalised by LGG when the two states are merged (Fig.4).

The third part covers learning of parameters using statistics derived from the input sequences. The state transitions are unobservable in our settings. The probability distributions over the state transitions and actions are estimated using the EM algorithm. These three steps are iterated until the logical states converge to a logical state.

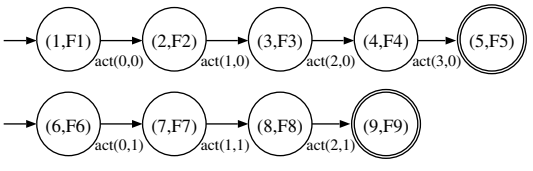


**Fig. 3.** Positive Examples

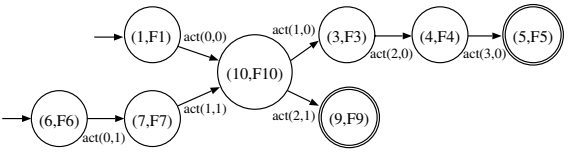


**Fig. 4.** Logical State Merging

## 4 Example

We implemented our inductive algorithm with two constraints: the depth bound for the graphical generalisation hypothesis space and the variable depth bound for LGG. In addition to the scope-extended LGG, our software has a function to invent $is(X, Y, int)$ atoms that represent $X = Y + int$ where $X$ and $Y$ are the variables restricted over integers.

### 4.1 Learning Chemical Reaction Rules from Biochemical Simulator

As an application of the system, we learn chemical reaction rules from StochSim that is a general purpose biochemical simulator in which individual molecules or molecular complexes are represented as individual software objects. Chemical reactions between the molecules occur stochastically according to probabilities derived from the given rate constants.

To run the simulator, a user needs to specify some definitions such as (1)kinds of molecules, (2)the initial numbers of molecules, and (3)possible chemical reaction rules. We assume the following simple chemical reactions in our experiment:

$$\text{2H2} + \text{O2} \Longleftrightarrow \text{2H2O}$$
$$\text{N2} + \text{3H2} \Longleftrightarrow \text{2NH3}$$

After each simulation, we have a dump file that contains time-series measurements of the concentrations of the molecules. For this experiment, we modify the StochSim code to output the *one-chemical-reaction* time-series results. The following shows two artificial dump files of two simulations:

```
[First Run]
Time h2o nh3 o2  h2   n2
0    2   3   2   18   5
1    2   3   2   18   5
2    4   3   1   16   5
3    4   5   1   13   4
4    4   7   1   10   3
```

```
[Second Run]
Time h2o nh3 o2  h2   n2
0    2   3   2   18   5
1    2   5   2   15   4
2    4   5   1   13   4
3    6   5   0   11   4
```

where time units are defined as *iterations* at the one-chemical-reaction timescale. Since the reactions occur stochastically, the concentrations do not change sometimes. For example, in the above First Run file, the initial concentration (Time 0) does not change at the next iteration (Time 1). This means that the simulator applied a chemical reaction for the molecules during the iteration, but the reaction did not happen because of its stochastic nature. During Time 1 and Time 2 in the first run, we could conclude that the reaction

$$\text{2H2} + \text{O2} \Longrightarrow \text{2H2O}$$

happened, because this is only the reaction that could bring the change in our setting. The outputs are converted into two logical sequences as follows:

```
[{mol(h2o,2),mol(nh3,3),mol(o2,2),mol(h2,18),mol(n2,5)} act(0,0)
 {mol(h2o,2),mol(nh3,3),mol(o2,2),mol(h2,18),mol(n2,5)} act(1,0)
 {mol(h2o,4),mol(nh3,3),mol(o2,1),mol(h2,16),mol(n2,5)} act(2,0)
 {mol(h2o,4),mol(nh3,5),mol(o2,1),mol(h2,13),mol(n2,4)} act(3,0)
 {mol(h2o,4),mol(nh3,7),mol(o2,1),mol(h2,10),mol(n2,3)}]

[{mol(h2o,2),mol(nh3,3),mol(o2,2),mol(h2,18),mol(n2,5)} act(0,1)
 {mol(h2o,2),mol(nh3,5),mol(o2,2),mol(h2,15),mol(n2,4)} act(1,1)
 {mol(h2o,4),mol(nh3,5),mol(o2,1),mol(h2,13),mol(n2,4)} act(2,1)
 {mol(h2o,6),mol(nh3,5),mol(o2,0),mol(h2,11),mol(n2,4)}]
```

Our learning task is set as:

**Given**

– A set of outputs from StochSim in the form of logical sequences.

**Find**

– Chemical Reaction Rules happen during the simulations.

Note that learning probability distribution is omitted since the number of examples is too small in this example.

The graphical representation of the input is shown in Fig.3 where

```
F1 = {mol(h2o,2),mol(nh3,3),mol(o2,2),mol(h2,18),mol(n2,5)}
F2 = {mol(h2o,2),mol(nh3,3),mol(o2,2),mol(h2,18),mol(n2,5)}
F3 = {mol(h2o,4),mol(nh3,3),mol(o2,1),mol(h2,16),mol(n2,5)}
F4 = {mol(h2o,4),mol(nh3,5),mol(o2,1),mol(h2,13),mol(n2,4)}
F5 = {mol(h2o,4),mol(nh3,7),mol(o2,1),mol(h2,10),mol(n2,3)}
F6 = {mol(h2o,2),mol(nh3,3),mol(o2,2),mol(h2,18),mol(n2,5)}
F7 = {mol(h2o,2),mol(nh3,5),mol(o2,2),mol(h2,15),mol(n2,4)}
F8 = {mol(h2o,4),mol(nh3,5),mol(o2,1),mol(h2,13),mol(n2,4)}
F9 = {mol(h2o,6),mol(nh3,5),mol(o2,0),mol(h2,11),mol(n2,4)}.
```

Then our software starts to search the hypothesis space by applying the state merging with symbolic generalisation. Since our search space is the version space, the examples are consistent with all hypothesis learned by the software. Our program returns the state transition model shown in Fig.5 with the following is/3 atoms:

```
is(V5,V8,4), is(V3,V9,2), is(V2,V5,-3), is(V2,V8,1), is(V1,V6,-1),
is(V0,V1,-1), is(V0,V6,-2)
```

We can translate the learned rules as:

> *Through the reaction, the number of H2O increases by two, the number of O2 decreases by one, and the number of H2 decreases2. The number of N2 and NH3 do not change.*

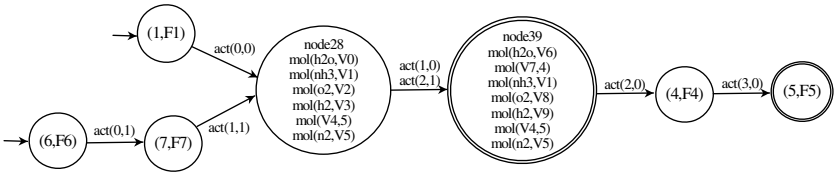


**Fig. 5.** Logical State Merging: 2H2+O2 $\Longrightarrow$ 2H2O

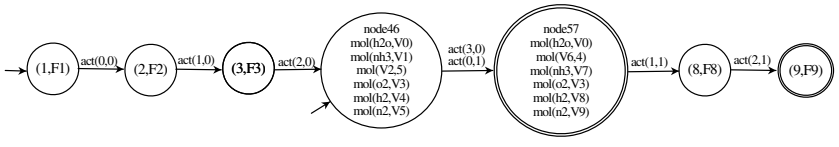


**Fig. 6.** Logical State Merging: N2 + 3H2 $\Longrightarrow$ 2NH3

Our program also returns the model shown in Fig.6 with the following is/3 atoms:

```
is(V0,V7,-3), is(V1,V0,1), is(V1,V7,-2), is(V3,V9,-2), is(V4,V8,3)
is(V5,V3,3),  is(V5,V9,1)
```

> *Through the reaction, the number of NH3 increases by two, the number of N2 decreases by one, and the number of H2 decreases3. The number of O2 and H2O do not change.*

The system learns some additional knowledge in the form of is/3.

## 5 Related Works

There exist many attempts for combining first-order logic and probability [17, 18, 19, 20]. Relational extensions of Bayesian Networks have also reported including [21]. Logical Hidden Markov Model [7] and Logical Markov Decision Programs [8] are closely related to our PLAS model. Logical Decision Program could embed a set of atoms in a state. Since PLAS could put a set of definite clauses in the logical state, it would be more expressive than Logical Decision Program.

PLAS is originally designed as a graphical representation of First-order Stochastic Action Language [3] which is a first-order logical version of dynamic Bayesian networks.

Our logical automata could be viewed as a graphical version of Situation Calculus. Successor state axioms are encoded in our each conditional state transitions. Since PLAS defines probability distributions over belief states, PLAS would be suitable for representing POMDPs.

## 6 Discussions and Conclusions

In this paper, we propose an extension of stochastic non-deterministic finite automata. By focusing on the generality order between the logical states and the partial inputs, the semantic aspect of our models becomes clear. PLAS defines probability distributions over belief states. We also proposed the inductive learning algorithms of PLAS by combining graphical and symbolic generalisations smoothly. Parameter learning of PLAS are implemented using EM algorithm.

Regarding the complexity of the state merging technique, the size of the hypotheses space of automata is known as exponential [10, 11]. Therefore heuristic searches would be mandatory to obtain the efficiency for the state merger.

One of the advantages of first-order logic is its compact representation. In PLAS, each snapshot of dynamic worlds is captured in a definite theory. The logical state could represent multiple states under the theory subsumption order. That is, the compactness has been realised at each state level.

Our logical sequence represents temporal changes of theories, that is, we could input a series of Logic Programs (LPs) to our model. If we learn a PLAS model from the observed series of LPs, the model should capture how LPs change proceeding with time. In [4], a STRIPS-like first-order stochastic operator is proposed in order to modify LPs temporally. The operator is expressed in the form of dynamic Bayesian networks with add/delete-lists functions. The development of the translation algorithms between the operator and PLAS would be useful for the distributed executions of LPs in the Multi-Agent research.

Since our models are based on Automata, further extensions would be possible by considering the existing extensions of Automata.

**Acknowledgements.** The first author would like to thank Katsumi Inoue for his advice in this research. The research was supported by European Union IST programme, contract no.FP6-508861, *Application of Probabilistic Inductive Logic Programming II.* The authors are also partially supported by National Institute of Informatics (NII), Joint Research Grant no.A-5.

## References

1. J. McCarthy and P. J. Hayes.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence 4, pages 463–502, Edinburgh University Press, 1969.
2. L. Rabiner.: A tutorial on hidden markov models and selected applications in speech recognition. In Proceedings of the IEEE, 77, 1989.
3. First-Order Stochastic Action Language, Hiroaki Watanabe and Stephen Muggleton.: Electronic Transactions in Artificial Intelligence, 7, 2002. http://www.doc.ic.ac.uk/~hw3/doc/watanabe02FirstSAL.ps
4. Towards Belief Propagation in Shared Logic Program.: Hiroaki Watanabe and Stephen Muggleton, BN2003, Kyoto, 2003. http://www.doc.ic.ac.uk/~hw3/doc/bn2003final2.pdf
5. S. Moyle and S.H. Muggleton.: Learning programs in the event calculus. In N. Lavrac and S. Dzeroski, editors, Proceedings of the Seventh Inductive Logic Programming Workshop (ILP97), LNAI 1297, pages 205–212, Berlin, 1997. Springer-Verlag.

6. R. P. Otero.: Induction of Stable Models, in Proceedings of 11th Int. Conference on Inductive Logic Programming, ILP-01, pages 193–205, LNAI 2157, Springer, Strasbourg 2001.
7. K. Kersting, T. Raiko, and L. De Raedt.: Logical Hidden Markov Models (Extended Abstract). In J. A. Ga'mez and A. Salmero'n, editors, Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM-02), pages 99–107, November 6-8, 2002, Cuenca, Spain.
8. K. Kersting, L. De Raedt.: Logical Markov Decision Programs and the Convergence of Logical TD($\lambda$). In A. Srinivasan, R. King, and R.Camacho, editors, Proceedings of the Fourteenth International Conference on Inductive Logic Programming (ILP-2004), pages 180–197. Porto, Portugal, September 6-8, 2004.
9. Morton-Firth, C. J. (1998) Stochastic simulation of cell signalling pathways Ph. D. Thesis, University of Cambridge.
10. P. Dupont, L. Miclet and E. Vidal.: What is the search space of Regular Inference?. Lecture Notes in Artificial Intelligence, No. 862, Springer-Verlag, Grammatical Inference and Applications, pages 25–37, 1994.
11. Coste, F., and Fredouille, D.: What is the search space for the inference of non deterministic, unambiguous and deterministic automata ?. technical report INRIA RR-4907, 2003.
12. G. Plotkin.: Automatic Methods of Inductive Inference. PhD thesis, Edinburgh University, UK, 1971.
13. E. M. Gold.: Complexity of automaton identification from given data. Information and Control, 37(3): pages 302–320, 1978.
14. D. Angluin.: Negative Results for Equivalence Queries. Machine Learning, 5, pages 121–150, 1990.
15. M. Kearns and L. G. Valiant.: Cryptographic limitations on learning boolean formulae and finite automata. In Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pages 433–444, New York. ACM, 1989.
16. D. Angluin.: Learning regular sets from queries and counterexamples. Information and Computation, 75: pages 87–106, 1987.
17. Joseph Y. Halpern.: An analysis of first-order logics of probability, Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence, pages 1375–1381, 1989.
18. S. H. Muggleton.: Stochastic logic programs. In L. de Raedt, editor, Advances in Inductive Logic Programming, pages 254–264. IOS Press, 1996.
19. Taisuke Sato.: A statistical learning method for logic programs with distribution semantics. Proc. ICLP'95, Syounan-village, pages 715–729, 1995.
20. Kristian Kersting and Luc De Raedt. Bayesian Logic Programs. In J. Cussens and A. Frisch, editors, Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming, pages 138–155, 2000.
21. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc, pages 1300–1309, 1999.
22. Stuart Russell and Peter Norvig.: Artificial Intelligence: A Modern Approach. 2nd Edition, Prentice Hall, 2003.
23. M.R.Garey and D.S.Johnson.: Computers and Intractability: *A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
24. J-U.Kietz and M.Lübbe.: An efficient subsumption algorithm for inductive logic programming. Proc. of the 4th International Workshop on Inductive Logic Programming (ILP-94), pages 97–105, 1994.