5

An Extension of Finite-state Markov Decision Process and an Application of Grammatical Inference

Takeshi Shibata[1] and Ryo Yoshinaka[2]
[1] *the University of Tokyo*
[2] *Hokkaido University*
Japan

1. Introduction

In this chapter, we introduce the notion of **simple context-free decision processes**, which are an extension of episodic finite-state Markov decision processes (MDPs). Intuitively, a simple context-free decision process can be thought of as an episodic finite-state MDP with a stack. In fact, many reinforcement learning methods can be applied to the class of simple context-free decision processes with natural modification on their equations.

On the other hand, in grammatical inference area, some non-regular subclasses of simple grammars, such as very simple grammars and **right-unique simple grammars**, have been found to be efficiently identifiable in the limit from positive data. Especially, the class of right-unique simple decision processes, which are simple context-free processes based on right-unique simple grammars, is a superset of the class of episodic finite-state MDPs.

Because episodic states histories are regarded as positive data, one might expect that those positive results in grammatical inference area could be applied to reinforcement learning directly.

However, one should note that grammars generating the same language can generate different probabilistic languages. While it is enough to find a process representing the target language in the scheme of identification in the limit, in reinforcement learning, one has to find a process representing the target probabilistic language.

Therefore, we need to modify the results in grammatical inference area for applying them to reinforcement learning. Actually, a grammar can be more general than another in the sense that it generates all the probabilistic languages generated by the other. Hence, finding a most general grammar gives a solution to this problem. This chapter however shows that both classes of simple grammars and right-unique simple grammars do not admit most general grammars.

Besides, we show that there is an intermediate class between right-unique simple grammars and simple grammars that admits an algorithm computing a most general grammar from any two grammars whose languages coincide.

We present an algorithm that learns the optimal actions under right-unique simple context-free processes, by concatenating the algorithm learning right-unique simple grammars from

Source: Reinforcement Learning: Theory and Applications, Book edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer
ISBN 978-3-902613-14-1, pp.424, January 2008, I-Tech Education and Publishing, Vienna, Austria

positive data, the one computing a most general grammar and the modified update equations of some usual reinforcement learning methods.

2. Notation and definitions

Before we give the definition of simple context-free MDPs, we write some standard notation and definitions and introduce subclasses of simple grammars and probabilistic grammars.

A **context-free grammar** (CFG) is a quadruple denoted by $<V, \Sigma, R, S>$, where V is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols, $R \subset V \times (V \cup \Sigma)^*$ is a finite set of production rules, and $S \in V$ is the start symbol. Let $G=<V, \Sigma, R, S>$ be a CFG. We write $XAZ \Rightarrow_G XYZ$ if there is a rule $A \rightarrow Y$. When G is clearly identified, we write simply $\Rightarrow$ instead of $\Rightarrow_G$. $\Rightarrow^*$ denotes the reflective and transitive closure of $\Rightarrow$. G is said to be reduced if and only if, for all A in V, there are some x,y,z in Σ^* such that $S \Rightarrow^* xAz \Rightarrow^* xyz$.

$L(G, X)$ denotes a language derived from X, i.e., $\{x \in \Sigma^* \mid X \Rightarrow_G^* x\}$. $L(G)=L(G,S)$ is called the language of G. Let ε denote the empty sequence. If x is a sequence, let $|x|$ denote the length of x. Let $|A|$ for a set A denote the cardinality of A, and $|G|$ denote $\sum_{A \rightarrow X \in R} |A| + |X|$.

In order to be easy to read, terminal symbols and nonterminal symbols are denoted by $a,b,c,\ldots$ and $A,B,C,\ldots$ respectively, and finite sequence of terminal symbols and nonterminal symbols are denoted by $\ldots,x,y,z$ and $\alpha,\beta,\gamma,\ldots$ respectively.

CFGs $G=<V, \Sigma, R, S>$ and $H=<V', \Sigma, R', S'>$ are equivalent modulo renaming of nonterminal symbols when there is a bijection $\varphi: V \rightarrow V'$ such that $A \rightarrow X$ is in R if and only if $\varphi(A) \rightarrow \varphi^*(X)$ is in R', and $\varphi(S)=S'$. φ^* is a homomorphism $(V \cup \Sigma)^* \rightarrow (V \cup \Sigma)^*$ defined recursively: $\varphi^*(\varepsilon)=\varepsilon$, $\varphi^*(aX)=a\varphi^*(X)$ and $\varphi^*(AX)=\varphi(A)\varphi^*(X)$.

Definition 1. Let $G=<V, \Sigma, R, S>$ be a CFG. G is called a **simple grammar** (SG) if and only if

- G is in Greibach normal form, that is, for each rule of G is written as $A \rightarrow a\alpha$.
- $A \rightarrow a\alpha \in R$ and $A \rightarrow a\beta \in R$ imply $\alpha = \beta$.

The subclasses of SGs which will appear in this chapter are defined below.

Definition 2. Let $G=<V, \Sigma, R, S>$ be an SG. G is called a **right-unique simple grammar** (RSG) if and only if

- $A \rightarrow a\alpha \in R$ and $B \rightarrow a\beta \in R$ imply $\alpha = \beta$.

Definition 3. Let $G=<V, \Sigma, R, S>$ be an SG. G is called a **very simple grammar** (VSG) if and only if

- $A \rightarrow a\alpha \in R$ and $B \rightarrow a\beta \in R$ imply $A = B$ and $\alpha = \beta$.

From definitions, a VSG is an RSG, and an RSG is an SG.

Let $G=<V, \Sigma, R, S>$ be an SG. A **probability assignment** P on G is a map from R to [0,1] such that $\sum_{r \in R(A)} P(r) = 1$ for all A in V, where $R(A) = \{A \rightarrow X \in R\}$. A **probabilistic simple grammar** (PSG) is a pair of G and P, where P is probability assignment on an SG G. Let G(P) be a PSG. All of sequences of production rules that are used in the left-most derivation of $S \Rightarrow_G^* x$, where $x \in L(G)$, is called the **left Szilard language** of G. When G is an SG, every x in $L(G)$ has a unique sentence in the left Szilard language of it. Let us denote that sentence by $r(G,x,1)$, …, $r(G,x,|x|)$. Then, the **probabilistic language** of G(P), $\Pr[\cdot \mid G(P)] : Pow(\Sigma^*) \rightarrow [0,1]$ is defined as $\Pr[x \mid G(P), S]$, where

$$\Pr[x \mid G(P), X] = \begin{cases} \prod_{i=1}^{|x|} P(r(G,X,x,i)) & \text{if } x \in L(G,X), \\ 0 & \text{otherwize.} \end{cases}$$

$r(G,X,x,1),\ldots,r(G,X,x,|x|)$ are the sequence of rules used in the derivation $X \Rightarrow_G^* x$. $G(P)$ is called **consistent** if and only if $\sum_{x \in L(G)} \Pr[x \mid G(P)] = 1$. In order that $\Pr[\cdot \mid G(P)]$ is regarded as a probability on Σ^*, $G(P)$ is required to be consistent. A sufficient condition of consistency is known (Wetherell, C. S., 1980). Let $M(G(P))$ be a $(|V|, |V|)$ matrix whose element $m_{ij}(G(P))$ represents the expectation of the number of the occurrences of the nonterminal symbol A_j derivable in one step from A_i. $G(P)$ is consistent if $\rho(M(G(P))) < 1$, where $\rho(M)$ is the spectral radius of M.

3. An extension of finite-state Markov decision processes

3.1 A representation of episodic finite-state MDPs with grammatical formalism

In this section, first, we describe the notion of the simple context-free Markov decision processes, by using a simple example of an episodic finite-state MDP. After that, the definition of simple context-free Markov decision process will be given.

Fig.1 is an episodic finite-state MDP, which contains 5 states, {*a, b, c, d, e*}. '*S*' indicates the initial state, and double circles indicate end states ({*a, e*}). The actions the robot can take are '*L*' and '*R*'. Reward given to the robot is -1 for every step, and if the robot gets in the end state '*e*', 1 is given. In this case, it is obvious that, if the robot takes '*R*' action for every state, the best policy is acquired.

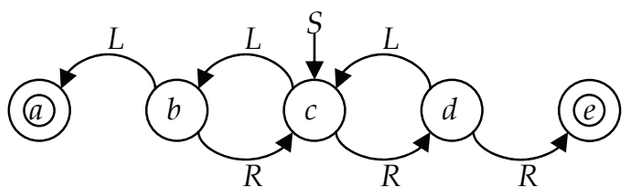

Fig. 1. An episodic finite-state MDP

A possible history of states is a sequence of states representing the transition of the robot from the initial state to an end state, such as {*cba, cde, cbdba*, … }. Let us call the set of possible histories the language generated by the finite-sate MDP in Fig. 1. While this language is a regular language, some regular languages cannot be generated by any finite-state MDPs. For example a singleton {*aae*} cannot be the language of any MDP, because each letter identifies one state. The fact that *aae* is a possible history implies that the robot may translate from the state a to a. Thus, $a^n e$ is also a possible history for any $n > 0$.

Therefore, the possible histories can be also described as a language for some regular grammar *G*, and its language class is not required to include the class of regular languages.

The class of simple grammars is one of the subclasses of CFGs such that all languages generated by finite-state MDPs are generated by them. For example, a CFG whose rules are {$S \to cC$, $C \to bB$, $B \to a$, $B \to cC$, $C \to dD$, $D \to e$, $D \to cC$} generates the language of the MDP in Fig.1. The derivation of '*cbcde*' is written as $S \Rightarrow cC \Rightarrow cbB \Rightarrow cbcC \Rightarrow cbcdD \Rightarrow cbcde$. That CFG is

a simple grammar, and a regular grammar, if nonterminal symbols are regarded as states (Fig. 2).

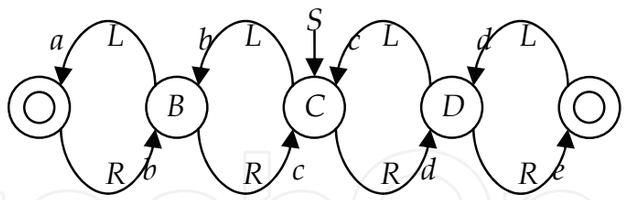

Fig. 2. Expressing the MDP in fig.1 by a regular grammar or simple grammar

A probabilistic CFG is defined by assigning a non-negative real number to every rule, where $\sum_{r\in R(A)} P(r) = 1$ and $R(A) = \{A \to X \in R\}$. In a finite-state MDP, if a policy is decided, the probabilities of histories (the measure on the language) are determined. On the MDP in Fig. 1, let us suppose that the policy is chosen as the probability of choice of '*R*' or '*L*' is assigned to 0.5 for every state. In that case, the probability of a sentence '*w*' in the language is $2^{-|w|}$. The pair of a language and a measure on it is called a probabilistic language. When a policy which is taken by a robot is changed, the probabilistic language of MDP changes correspondingly.

Every context-free grammar generates various probabilistic languages by assigning various probabilities to each rule of it. The set of all probabilistic languages generated from a CFG *G* by assigning a probability to each rule of it is called the probabilistic generality of *G*. The probabilistic generality of *G* is a subset of $\{L(G) \to [0,1]\}$. For instance, suppose that *G* is a CFG whose rules are {*S*→*aS*, *S*→*b*}. The probabilistic generality of it is written as follows:

$$\left\{P : \{a^* b\} \to [0,1] \mid P(a^n b) = q^n (1-q), q \in [0,1]\right\}.$$

It is clear that the fact that grammars *A* and *B* generate the same language does not imply its probabilistic generalities of them are the same. Suppose that *H* is a CFG that has rules {*S*→*aA*, *S*→*b*, *A*→*aA*, *A*→*b*}. Obviously, *L*(*G*) = *L*(*H*). But the generality of *H* is

$$\left\{P : \{a^* b\} \to [0,1] \mid P(b) = 1-q, P(aa^n b) = qr^n (1-r), q \in [0,1]\right\}.$$

So generalities of them are different from each other.

3.2 Simple context-free Markov decision processes

Simple context-free MDP is formally defined as follows

Definition 4. Let *G* be an simple grammar. *G*(*U*,*P*,*C*) =<*V*,Σ,*R*,*S*,*U*,*P*,*C*> is a **simple context-free decision process** if and only if *U*, *P* and *C* are the following set and functions.

- *U* is a finite set of actions.
- $P : R \times U \to [0,1]$ is a probabilistic assignment. For all (*A*,*u*) in *V*×*U*, $\sum_{r\in R(A)} P(r,u) = 1$
- $C : \Sigma \to (-\infty,\infty)$ is a reward function.

In the following, when *G* is in a subclass of SGs, we call the simple context-free decision process *G*(*U*,*P*,*C*) [the name of the subclass]-DP.

Corresponding to a given SG-DP, a sequence of discrete random variables *X*(1), *Y*(1), *X*(2), *Y*(2), … is given as follows. *X*(1) =*S*, the domain of *X*(*i*) is Σ**V** and the domain of *Y*(*i*) is *U*.

$$\begin{aligned}
&\Pr[X(t) = x_t\alpha_t \mid X(1) = S, Y(1) = u_1, ..., X(t-1) = x_{t-1}\alpha_{t-1}, Y(t-1) = u_{t-1}] \\
&= \Pr[X(t) = x_t\alpha_t \mid X(t-1) = x_{t-1}\alpha_{t-1}, Y(t-1) = u_{t-1}] \\
&= \begin{cases} P(r, u_{t-1}) & \text{if} \quad x_{t-1}\alpha_{t-1} \Rightarrow x_t\alpha_t \ \text{ with the rule } r, \\ 1 & \text{if} \quad x_{t-1}\alpha_{t-1} = x_t\alpha_t = x_{t-1} = x_t, \\ 0 & \textit{otherwise.} \end{cases}
\end{aligned}$$

Clearly, X and Y are an infinite-state MDP. Every episodic finite-state MDP is equivalent to some SG-DP as we have discussed in the beginning of this section. In fact, an episodic finite-state MDP whose states are $a_1, \cdots, a_{n+k}$ and end states are $a_{n+1}, \cdots, a_{n+k}$ for some n >0 and k >=0, can be represented by the form of SG-DP <*V*,Σ,*R*,*S*,*U*,*P*,*C*>:

$$\begin{aligned}
&V = \{A_1 (= S), ..., A_n\}, \\
&\Sigma = \{a_1, ..., a_{n+k}\}, \\
&R = \{A \to a_j A_j \mid A \in V, 1 \le j \le n\} \cup \{A \to a_j \mid A \in V, n+1 \le j \le n+k\}
\end{aligned}$$

$P(A_i \to a_j A_j)$ equals to the transition probability from *i* to *j*, and $P(A_i \to a_{n+j})$ equals to the transition probability from *i* to *n+j*, where *n+j* is an end state. *U* is the same set of actions as the MDP, and *C* is also the same.

Policies, value-function, optimal value function, etc. are introduced below in analogues to those of MDPs. Let *G*(*U*,*P*,*C*) = <*V*, Σ, *R*, *S*,*U*,*P*,*C*> be an SG-DP.

Definition 5. A **policy** of *G*(*U*,*P*,*C*) is a map $V \to U$.

One of the main purposes of reinforcement learning is to determine a policy μ so as to maximise the expectation of the total reward from *S*.

Definition 6. A **value function** of *G*(*U*,*P*,*C*) under a policy μ, $J_\mu : V \to (-\infty, \infty)$, is defined as

$$J_\mu(A) = \sum_{x \in L(G,A)} \Pr[x \mid G(P_\mu), A] \sum_{i=1}^{|x|} C(a_i),$$

where $x = a_1 a_2 \cdots a_{|x|}$ and P_μ is the probability assignment of *G* under μ, namely, for $A \to a\alpha \in R$, $P_\mu(A \to a\alpha) = P(A \to a\alpha, \mu(A))$.

When $\rho(M(G(P_\mu))) < 1$ for any policy μ, all value functions of G(U,P,C) are finite.

Definition 7. The **optimal value function** of *G*(*U*,*P*,*C*) denoted as $J_* : V \to (-\infty, \infty)$ is defined as

$$J_*(A) = \sup_{\mu \in \pi} J_\mu(A),$$

where π is the set of all policies.

There exists some policy μ_* such that $J_{\mu_*}(A) = J_*(A)$.

Definition 8. μ_* is called an **optimal policy**.

Definition 9. The **optimal action-value function** $Q_* : V \times U \to (-\infty, \infty)$ is defined as

$$Q_*(A,u) = \sum_{A \to aB_1 \cdots B_k \in R(A)} P(A \to aB_1 \cdots B_k, u) \left(C(a) + \sum_{i=1}^{k} J_*(B_i) \right).$$

Definitions 5 - 9 are a natural extension of the usual definitions on reinforcement learning for finite-state MDPs, whose discounting factor equals 1. Most of well known reinforcement learning methods, such as Q-learning, TD(λ) can be applied to SG-DPs corresponding to the above definitions. In the following theorem, an extended Q-learning for SG-DPs is introduced and its convergence to the optimal action-value is established. A proof is in (Shibata et al., 2006).

Theorem 1. Assume that $\rho(M(G(P_\mu)))<1$ for any policy μ. A sequence of V×U×[0,1], $(A_1,u_1,k_1),(A_2,u_2,k_2),\cdots$ is supposed to satisfy the following conditions for all (A,u) in V×U.

$$\sum_{(A_t,u_t)=(A,u)} k_t = \infty \text{ and } \sum_{(A_t,u_t)=(A,u)} {k_t}^2 < \infty .$$

The sequence of random variables $Q_1,Q_2,\cdots$ defined by the following iteration (the **extended Q-Learning**) converges to the optimal action-value function of $G(U,P,C)$ as $t\to\infty$ with probability 1.

$$Q_{t+1}(A_t,u_t) = (1-k_t)Q_t(A_t,u_t) + k_t\left(C(a) + \sum_{i=1}^{m} \max_{v\in U} Q_t(B_i,v)\right),$$

where the sequence $B_1\cdots B_m$ is randomly chosen with probability $P(A_t \to aB_1\cdots B_m,u_t)$.

4. Identification in the limit of right-unique simple grammars from positive data

4.1 Learning simple grammars

A most rigid theoretical model for learning concepts would be identification in the limit proposed by Gold (1967). Because episodic states histories are regarded as positive data, one might expect that fruits of grammatical inference on identification in the limit from positive data could be directly applied to reinforcement learning. As simple context-free decision processes are a generalization of finite Markov decision processes, we should refer to results on grammatical inference of simple grammars. It is known that however simple languages are not identifiable in the limit from positive data, because every regular language with an endmarker is a simple language and the class of whole regular languages is not identifiable in the limit from positive data (Gold 1967).

On the other hand, Yokomori (2003, 2007) has shown that very simple grammars, which form a small subset of simple grammars, are polynomial-time identifiable in the limit from positive data. A very simple grammar is a simple grammar such that each terminal symbol has exactly one production rule in which it occurs. Therefore, the grammar has exactly the same number of production rules as terminal symbols. Decision processes constructed on very simple grammars are, however, too restricted and they are no longer able to cover finite Markov decision processes.

Thus we need another richer subclass of simple grammars that should give an extension of finite Markov decision processes and at the same time it should be efficiently identifiable in the limit from positive data. Here we introduce a new class of grammars, called right-unique simple grammars, which is located between simple grammars and very simple grammars. This class still defines a small proper subclass of simple languages, but actually it satisfies the two desired properties mentioned above.

In this section, we show the efficient learnability of right-unique simple grammars.

4.2 Identification in the limit from positive data

First we let the reader recall the notion of identification in the limit from positive data established by Gold (1967). A **positive presentation** of a language L_* is an infinite sequence of strings where all and only elements of L_* appear. Each string appearing in a positive presentation is called a positive example of L_*. A **learning algorithm** $\mathcal{A}$ is an algorithm which takes a positive presentation $w_1,w_2,\ldots$ as input, and outputs some infinite sequence of grammars $G_1,G_2,\ldots$, i.e., $\mathcal{A}$ infinitely repeats the cycle where $\mathcal{A}$ receives w_i and outputs G_i for $i = 1, 2,\ldots$. A learning algorithm $\mathcal{A}$ **converges** to G on a presentation $w_1,w_2,\ldots$ if for all but finitely many i, $G_i = G$ holds. $\mathcal{A}$ **identifies** a class $\mathcal{L}$ of languages in the limit from positive data if for every positive presentation of every $L_* \in \mathcal{L}$, $\mathcal{A}$ converges to a grammar G generating the exact language L_*.

Usually learning algorithms are supposed to output a grammar consistent with the given positive examples, i.e., the conjectured grammar generates all the examples. Moreover, they do not change the conjecture unless the current conjecture is inconsistent with the newly given example. Our learning algorithm, which will be presented in Sec. 3.4, also has this standard property.

4.3 Right-unique simple grammars

Our learning target here is **right-unique simple languages** defined by right-unique simple grammars. A simple grammar $G = \langle V, \Sigma, R, S \rangle$ is called a right-unique simple grammar (RSG) if whenever both $A \to a\alpha$ and $B \to a\beta$ are rules of the grammar with $a \in \Sigma$ and $\alpha, \beta \in V^*$, $\alpha = \beta$ holds. We note that G is a very simple grammar if moreover we have $A = B$ in addition to $\alpha = \beta$.

Let us see an example of an RSG. The grammar consisting of the following rules is an RSG:

$$S \to \neg S \mid \vee SS \mid \exists US \mid pT \mid qTT,\ T \to fT \mid gTT \mid a \mid b \mid x \mid y,\ U \to x \mid y,$$

where S,T,U are nonterminal symbols and $\neg, \vee, \exists, p, q, f, g, a, b$ are terminal symbols. The generated language is a set of formulae of first-order logic in Polish notation.

The definition of RSGs allows us to define the function $\sharp_G$ for each RSG G, called **the shape** of G, that assigns an integer to each terminal symbol as

$$\sharp_G(a) = |\alpha| - 1 \quad \text{if } G \text{ has a rule } A \to a\alpha \text{ for some } A.$$

This function is homomorphically extended so that $\sharp_G(xy) = \sharp_G(x) + \sharp_G(y)$ for any $x, y \in \Sigma^*$ ($\sharp_G(\varepsilon) = 0$ for the empty string ε). It is easy to see that whenever $\alpha \Rightarrow_G^* x\beta$ with $x \in \Sigma^*$ and $\alpha, \beta \in V^*$, we have $|\beta| = |\alpha| + \sharp_G(x)$. Moreover we have $|\alpha| + \sharp_G(x') \geq 1$ for any proper prefix x' of x, because $\alpha \Rightarrow_G^* x'\gamma \Rightarrow_G^+ x\beta$ entails $\gamma \neq \varepsilon$. Particularly for $w \in L(G)$, we have $\sharp_G(w) = -1$ and $\sharp_G(w') \geq 0$ for any proper prefix w' of w. In this way, the function $\sharp_G$ strongly characterizes the derivations and the language of G. In general, we let us call any function $\sharp$ from Σ^* to $\mathbb{Z}$ a **shape** if it holds that $\sharp(a) \geq -1$ and $\sharp(x) + \sharp(y) = \sharp(xy)$ (homomorphism) for all $a \in \Sigma$ and $x, y \in \Sigma^*$.

We also say that a shape $\sharp$ is **compatible with** a language L if $\sharp(w) = -1$ and $\sharp(w') \geq 0$ for all $w \in L$ and any proper prefix w' of w. Consequently, $\sharp_G$ is always compatible with $L(G)$.

Here we note that any language L admits a finite number of compatible shapes. This is because, if $\sharp$ is compatible with L and $xay \in L$, then

$$\sharp(a) = \sharp(xay) - \sharp(x) - \sharp(y) \leq -1 - 0 + |y|,$$

because x is a proper prefix of xay and $\sharp(b) \geq -1$ for any $b \in \Sigma$. Therefore, a language L admits at most $\prod_{a \in \Sigma} \ell_a$ compatible shapes, where $\ell_a = \min\{\, |ay| \mid xay \in L \,\}$.

To simplify our discussion, here we introduce a special form of RSGs, called **canonical form**. Every RSG can be transformed into canonical form with preserving the language and the shape. The set of nonterminal symbols of an RSG G in canonical form of shape $\sharp$ is exactly

$$V_\sharp = \{S_0\} \cup \{\, [a, i] \mid a \in \Sigma,\ 0 \leq i \leq \sharp(a) \,\}$$

and every rule has the form

$$A \to a[a, 0] \ldots [a, \sharp(a)]$$

for some $A \in V_\sharp$. For instance, an RSG G consisting of the rules

$$S \to aSA, \quad S \to bA, \quad A \to c$$

is converted into G' in canonical form with the rules

$$\begin{array}{llll} S_0 \to a[a,0][a,1], & S_0 \to b[b,0], & [a,1] \to c, & \\ [a,0] \to a[a,0][a,1], & [a,0] \to b[b,0], & [b,0] \to c. & \end{array} \tag{1}$$

Here the start symbol S of G is divided into S_0 and $[a,0]$ in G' and A is divided into $[a,1]$ and $[b,0]$. Therefore, it is allowed to consider only RSGs in canonical form. Actually our learning algorithm for RSGs computes its conjecture in canonical form.

4.4 Learning algorithm

By definition, each shape has a finite number of RSGs in canonical form. Together with the fact that any language admits a finite number of compatible shapes, we see that there is a finite number of RSLs consistent with the given positive examples. This property is known as **finite thickness**. Angluin (1980) has shown that every class of languages with finite thickness is identifiable in the limit from positive data. Thus RSGs are identifiable in the limit from positive data.

Our learning algorithm outputs an RSG that generates a minimal language among all the RSLs containing the given positive examples. This strategy ensures that the algorithm finally converges to a grammar representing the target language. If the current conjecture G does not generate the target language L_*, we never have $L_* \subsetneq L(G)$, because of the minimality of the conjecture. Thus there is $w \in L_* - L(G)$, which will appear in the positive presentation of L_*. Then the algorithm eventually abandons the conjecture G and the times changing the conjecture is finite by the finite thickness of the class of RSLs. Finally the conjecture converges to a grammar generating the target language.

To enable us to analyze the efficiency of the learning task, we present a concrete learning algorithm for RSGs. The learning method for RSGs is basically same as Yokomori's (2003, 2007) algorithm for very simple grammars. The first step of our algorithm for learning RSGs is to find shapes compatible with the given positive examples. Then the algorithm computes consistent RSGs in canonical form whose shapes are those found at the first step. At last a minimal (with respect to its language) grammar among those candidate RSGs is picked up. Fig. 3 represents this learning strategy.

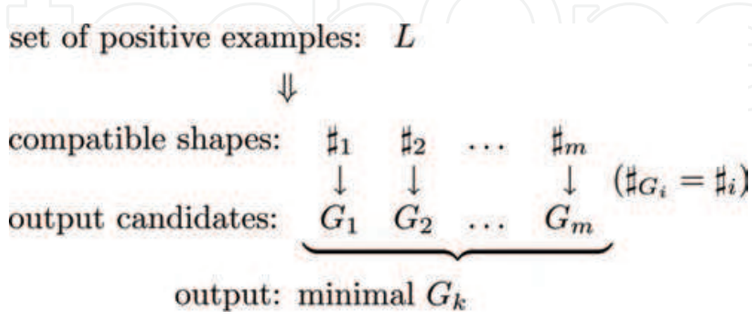

Fig. 3. Flow of our learning algorithm.

To enumerate all the compatible shapes for a given set L of positive examples, it is enough to check the compatibility of shapes $\sharp$ satisfying that $-1 \leq \sharp(a) < \ell_a - 1$ with $\ell_a = \min\{\, |ay| \mid xay \in L \,\}$. Deciding the compatibility of a shape with a finite language is trivially done in linear time. Therefore, the enumeration of compatible shapes is done in $O(\ell^{|\Sigma|})$ time simply by the brute-force search where $\ell = \max\{\, |w| \mid w \in L \,\}$. This upper bound is polynomial if we fix Σ. It would be natural to ask whether a more efficient algorithm that finds a compatible shape in polynomial time in $|\Sigma|$ is possible. Concerning this question, it is known that deciding whether or not a finite language admits a compatible shape is NP-complete if we regard $|\Sigma|$ as a variable.

Once we obtain a compatible shape, one can straightforwardly construct the minimum RSG in canonical form of that shape that is consistent with the given positive examples. For a given set L of examples and a shape $\sharp$ compatible with L, the algorithm picks up the least rules from the set

$$\{\, A \to a[a,0] \ldots [a,\sharp(a)] \mid A \in V_\sharp,\ a \in \Sigma \,\}$$

so that the resultant grammar generates all the elements of L. For instance, when two positive examples *aabccc*, *bc* are given, the only compatible shape $\sharp$ is such that $\sharp(a) = 1$, $\sharp(b) = 0$, $\sharp(c) = -1$. To derive those two strings, exactly the rules in (1) are needed. Thus, the output of the learning algorithm is G'. This procedure can be done in almost linear time in $\|L\|$ where $\|L\|$ is the sum of the lengths of all the positive examples.

In general, multiple compatible shapes would be computed. In that case, we will compute grammars as many as the compatible shapes and have to choose one among those as the conjecture. The criterion is to choose a minimal grammar with respect to the language. That is, we have to solve the inclusion problem of RSGs. This procedure would be rather purely an issue of formal language theory and thus we relegate this subroutine to (Yoshinaka 2006), where it is shown that inclusion of two RSGs computed as output candidates is decidable in

polynomial time in $\|L\|$. Let us write the upper bound of the running time of this subroutine as $p(\|L\|)$ for a polynomial p. In order to pick up a minimal RSG among m output candidates, we execute this subroutine $m-1$ times. Recall that we have $m \leq \ell^{|\Sigma|}$ where ℓ is the length of a longest positive example. Therefore, our algorithm updates its conjecture in $O(p(\|L\|)\ell^{|\Sigma|})$ steps.

Let us see an example. Suppose that the first positive example is *abbc*. There are two compatible shapes $\sharp_1$ and $\sharp_2$:

$$\sharp_1 = \{\, a \mapsto 0,\ b \mapsto 0,\quad c \mapsto -1 \,\},$$

$$\sharp_2 = \{\, a \mapsto 2,\ b \mapsto -1,\ c \mapsto -1 \,\}.$$

The minimum consistent RSGs G_1 and G_2 constructed on $\sharp_1$ and $\sharp_2$, respectively, have the following production rules:

$$G_1 : S_0 \to a[a,0],\ [a,0] \to b[b,0],\ [b,0] \to b[b,0],\ [b,0] \to c,$$

$$G_2 : S_0 \to a[a,0][a,1][a,2],\ [a,0] \to b,\ [a,1] \to b,\ [a,2] \to c.$$

We have $L(G_2) = \{abbc\} \subsetneq L(G_1) = \{\, ab^n c \mid n \geq 1 \,\}$. Therefore our algorithm outputs G_2.

5. Probabilistic generality of subclasses of simple grammars

5.1 Probabilistic generalities and unifiablity

Definition 10. The **Probabilistic generality** of an SG G is defined as

$$\Gamma(G) = \{\Pr[\cdot \mid G(P)] \mid P \text{ is a probability assignment on } G\}.$$

An SG H is more general than G if and only if $\Gamma(G) \subset \Gamma(H)$.

The following lemma establishes requirements for $\Gamma(G) \subset \Gamma(H)$.

Lemma 1. Let $G = \langle V,\Sigma,R,S \rangle$ and $H = \langle V',\Sigma,R',S' \rangle$ be reduced SGs. $\Gamma(G) \subset \Gamma(H)$ if and only if $L(G) = L(H)$ and there is some map $\psi : V'_{\geq 2} \to V_{\geq 2}$ that satisfies the following condition, where $V_{\geq 2} = \{A \in V \mid R(A) \geq 2\}$. For all A in $V'_{\geq 2}$ and all x in Σ^*, $S' \Rightarrow^*_H xA\alpha$ implies $S \Rightarrow^*_G x\psi(A)\beta$.

Suppose that C is a subclass of SGs. In the following, we will discuss whether C has a more general grammar than arbitrary two grammars that generate the same language.

Definition 11. Let C and D be subclasses of SGs. C is **unifiable within** D if and only if, for all G, H in C such that $L(G) = L(H)$, there is I in D such that $\Gamma(G) \cup \Gamma(H) \subset \Gamma(I)$.

The main result of this section is construction of an SG G^* that is more general than a finite number of given RSGs whose languages are equivalent. However, neither the class of SGs nor the class of RSGs is unifiable within itself, as we see in what follows. The proofs for all of them use Lem. 1. In the following, we say that C is **unifiable** when C is unifiable within C itself.

Theorem 2. The class of SGs is not unifiable.

A proof of Theorem 2 is written in (Shibata et al., 2006). The class of RSGs is also not unifiable. This fact is showed by considering the finite language L= $(a \mid b)(c \mid d)(e \mid f)$ = {*ace,acf,ade,adf,bce,bcf,bde,bdf*}. On the other hand, the class of VSGs is unifiable. For any VSGs G and H, $L(G)=L(H)$ implies $\Gamma(G)=\Gamma(H)$. Suppose that $S \Rightarrow_G^* xA\alpha$ and $S \Rightarrow_G^* yB\beta$. $A=B$ if and only if $xA\alpha \Rightarrow_G xa\alpha'$ and $yB\beta \Rightarrow_G ya\beta'$ from the definition of a VSG. If H is a VSG such that $L(G)=L(H)$, $S \Rightarrow_G^* xA\alpha \Rightarrow xa\alpha'$ if and only if $S' \Rightarrow_H^* xC\gamma \Rightarrow xa\gamma'$. Thus, there is a bijection $\psi : V \to V'$, and $S \Rightarrow_G^* xA\alpha$ if and only if $S' \Rightarrow_H^* x\psi(A)\gamma$. From Lemma 1, $\Gamma(G)=\Gamma(H)$.

5.2 A unifiable subclass of simple grammars

In this subsection, we will introduce **unifiable simple grammars**. The class of USGs is unifiable and is a superclass of the class of RSGs. This implies that the class of RSGs is unifiable within the class of USGs. Because the proof of unifiability for the class of USGs is constructive, the algorithm that unifies a finite number of RSGs whose languages are equal is also given.

Let G = <V,Σ,R,S > be an SG. Let $\sigma_G(A)=\{a \in \Sigma \mid A \to a\alpha \in R\}$. The relation between A and B given by $\sigma_G(A)=\sigma_G(B)$ is an equivalence relation, thus let $\overline{A}$ denote the equivalence class containing A. $\overline{A}=\{A' \in V \mid \sigma(A)=\sigma(A')\}$. We also introduce the notation $\overline{U}=\{A' \in V \mid \exists A \in U, \sigma(A)=\sigma(A')\}$ and $\overline{A_1 \cdots A_m}=\overline{A_1} \cdots \overline{A_m}$.

Definition 12. An SG G is a **unifiable simple grammar** (USG) if and only if

- $\overline{A}=\overline{B}$, $A \to a\alpha \in R$ and $B \to a\beta \in R$ imply $\overline{\alpha}=\overline{\beta}$.

Neighbourhood pairs introduced below play the most important role in constructing the proof of the unifiability of the class of USGs. Before we define neighbourhood pairs, it is necessary that two new notions are introduced.

Definition 13. For a subset U of V, $\mathrm{up}(U)=\{A \in V \mid \exists B \in U, A \Rightarrow^* xB\}$ is called the **upstream** of U.

Definition 14. Let T and U be subsets of V. W(T,U) denotes (V-$T \mid TU$)*.

The upstream of U is easy to compute from R. Using the above definitions, we define a neighbourhood pair as follows.

Definition 15. Let <T,U> be a pair of subsets of V. <T,U> is called a **neighbourhood pair** if and only if the following conditions hold.

1. $T \cap U = \phi$.
2. For some A in V, $U=\overline{A}$.
3. For some B in V, $T=\mathrm{up}(\overline{B})$.
4. For all x, $S \Rightarrow^* x\alpha$ implies $\alpha \in \mathrm{W}(T,U)$.

An intuitive meaning of a neighbourhood pair can be seen in the fourth condition of Definition 15. If a nonterminal symbol $A \in T$ appears in α such that $S \Rightarrow^* x\alpha$, some $B \in U$ is adjoining on the right side of A. Fig. 4 is an algorithm to find a neighbourhood pair. The computational cost required to find a neighbourhood pair from G is $O(|G|\,|V|)$.

```
NeighbourhoodPair find(USG G=<V,Σ,R,S >) {
  for(each A ∈ V and B ∈ V − {S} ){
    T = up(B̄);
    U = Ā;
    for (each B → aα ∈ R ) {
      if ((B ∈ T and αC ∉ W(T,U) for some C ∈ U )
          or ( B ∉ T and α ∉ W(T,U))) {
        < T,U > is not a neighbourhood pair, so break this and continue the 1st loop;
      }
    }
    return < T,U >;
  }
  return no_neighbourhood_pair ;
}
```

Fig. 4. Finding a neighbourhood pair

We explain only the abstract of the proof, which is constructing a unified USG from two arbitrary USGs. For the complete proof of it, refer to (Shibata et al., 2006).
First, we find and eliminate all neighbourhood pairs from both USGs. While some neighbourhood pair is found, we eliminate it keeping the generality of the grammar.

```
USG transform(USG G) {
   while ( There exists an neighbourhood pair <T,U> in G ) {
      G = Φ(G, <T,U>) ;
   }
   return G;
}
```

Fig. 5. Transformation of USGs

In Fig. 5, $\Phi(G, \langle T,U\rangle)$ denotes the USG obtained by eliminating a neighbourhood pair keeping the generality. $\Phi(G, \langle T,U\rangle)$ is a USG obtained by eliminating useless nonterminal symbols and rules from a USG $\langle V',\Sigma,R',S\rangle$ where

$$V' = (V - T) \cup (T \times U),$$

$$R' = \{A \to a\varphi(\alpha) \mid A \to a\alpha \in R,\ A \in V - T\} \cup \{(A,B) \to a\varphi(\alpha B) \mid A \to a\alpha \in R,\ (A,B) \in T \times U\}.$$

The above map φ from V^* to V'^* is defined recursively as the following.

- $\varphi(\varepsilon) = \varepsilon$.
- $\varphi(A\beta) = \begin{cases} (A,B)\varphi(\gamma) & \text{if } A \in T \text{ and } \beta = B\gamma, \\ \mathrm{A}\varphi(\beta) & \text{othewize.} \end{cases}$

$\Phi(G, \langle T,U\rangle)$ is more general than G.

Let G/σ denote a USG $<V/\sigma,\Sigma,R/\sigma,S>$, where

$V/\sigma = \{\overline{A} \mid A \in V\}$,

$R/\sigma = \{\overline{A} \to a\overline{\alpha} \mid A \to a\alpha \in R\}$.

Definition 16. USGs G and H are ***σ*-isomorphic** if and only if G/σ and H/σ are equivalent modulo renaming of nonterminal symbols.

When neither a USG G nor a USG H has neighbourhood pair, $L(G) = L(H)$ implies that G is σ-isomorphic to H. If G and H are σ-isomorphic, it is easy to unify them. In fact, let $G=<V,\Sigma,R,S>$ and $H=<V',\Sigma,R',S'>$ be σ-isomorphic. A USG defined as $G \otimes H$ is obtained by eliminating useless nonterminal symbols and rules from a USG $<V \otimes V',\Sigma,R \otimes R',(S,S')>$, where

$V \otimes V' = \{(A,B) \in V \times V' \mid \sigma(A) = \sigma(B)\}$,

$R \otimes R' = \{(A,B) \to a\alpha \otimes \beta \mid A \to a\alpha \in R \text{ and } B \to a\beta \in R'\}$,

$\mathrm{A}_1 \cdots \mathrm{A}_m \otimes \mathrm{B}_1 \cdots \mathrm{B}_m = (\mathrm{A}_1,\mathrm{B}_1) \cdots (\mathrm{A}_m,\mathrm{B}_m)$.

Thus we have the following theorem.

Theorem 3. The class of USGs is unifiable.

As a finite language admits a finite number of compatible shapes, any RSL has a finite number of compatible shapes. This entails that any RSL is generated by only a finite number of RSGs in canonical form. In fact, for any RSG G, we can compute all the RSGs in canonical form generating the same language. It is easy to see that if G is an RSG and H is the canonical form of G, i.e., $L(G)=L(H)$, $\#_G = \#_H$ and H is in canonical form, then H is more general than G. Since we have proven Theorem 3 in a constructive manner we obtain the following theorem.

Theorem 4. For every RSG G, we can construct a USG G^* which satisfies the following property. For any RSG H such that $L(H) = L(G)$, $\Gamma(H) \subset \Gamma(G^*)$.

Fig. 6 shows a unification algorithm of RSGs whose languages are equivalent.

```
USG unify(RSGs G_1,⋯,G_m) {
  Confirm that all languages of G_1,⋯,G_m equal;
  return transform(G_1) ⊗ ⋯ ⊗ transform(G_m);
}
```

Fig. 6. Unification of USGs

We finally describe only the result of the order of $|G^*|$. The time complexity of the algorithm is mainly dominated by $|G^*|$. $|G^*|$ is $O(m(G)^{2\,\mathrm{amb}(G)})$, where

$$m(G) = \max\{|H| \mid H \text{ is an RSG and } L(H) = L(G)\}$$

and amb(G) is the number of the equivalence classes with respect to σ-isomorphism whose languages equal to $L(G)$, that is,

$$\mathrm{amb}(G) = \left|\{H/\sigma \text{ modulo renaming nonterminals} \mid H \text{ is an RSG and } L(H) = L(G)\}\right|.$$

6. Implementation and experiments

6.1 Implementation of the learning algorithm for RSGs and the extension of QL

In the following, we assume that environments are RSG-DPs. The class of RSG-DPs is sufficiently large and includes the class of episodic finite-state MDPs, as we have described in Section 3. Those are the reasons why we assume that environments are RSG-DPs. For instance, let us consider other classes described in this chapter. The class of VSG-DPs does not include the class of episodic finite-state MDPs. If we assume the class of SG-DPs in stead of RSG-DPs as the class of environments, it is too large to learn, that is, the class of SGs is not learnable from positive data. The class of USGs is learnable theoretically, but efficient learning method is not known yet.

Although sequences of observations can be identified as positive data of grammars and we have an efficient learning algorithm for RSGs, we cannot apply a technique of reinforcement learning with assuming that the environment is constructed on the grammar obtained by the learning algorithm in Section 4. As discussed in Section 5, it is possible that any RSG-DP based on the grammar output by the learning algorithm does not generate the same probabilistic language as the environment, though both define the same (nonprobabilistic) language. In this case, actually the RSG that bases the environment must be one candidate computed in the learning algorithm, though it is not chosen as the output by the nondeterministic choice. Here, we modify the learning algorithm in Section 4 so that it outputs all the RSGs generating the same language as the grammar output by the original algorithm. By applying the unification algorithm in Section 5 to obtained RSGs, we get a USG that is more general than the RSG that bases the environment.

Combining the learning algorithm and the unification algorithm with an extended Q-learning, we obtain the algorithm in Fig. 7. Let us call that algorithm RSG-QL. RSG-QL does not require any grammar which bases the environment to be given. Only the set of actions and the set of observations (= terminal symbols) are given.

Fig. 7 shows update of the RSG-QL for one episode, or sentence. At first, a USG G, which intends to represent the environment, is set to an episodic finite-state MDP. This is just a tentative one, and when a USG is computed from the learning algorithm and the unification algorithm, it is substituted for G. If G cannot derive a prefix x given by the environment, the algorithm abandon updating Q until the episode ends. After the episode ends, it is added to the set of histories, and new USG is computed.

There are two new external functions in Fig. 7. Those are the same things in the ordinary reinforcement learning. environ(x,u) returns an observation (= terminal symbol) when a prefix of the sequence of observations is x and the action that the learning robot takes is u. Suppose that the environment is identified with an RSG-DP $H(U,P,C)$, then environ(x,u) randomly returns a with the probability $P(A \to a\alpha, u)$, where $S \Rightarrow_H^* xA\beta$. environ($x$,$u$) itself is not controlled by the learning algorithm directly. It is controlled by the selection of actions.

The other function is strategy(Q,G,x). strategy(Q,G,x) is the function that decides which action the learning robot takes. It can be arbitrarily chosen within the condition of Theorem 1. Actually, famous strategies such as eps-greedy and soft-max are often chosen (Sutton & Barto 1998).

For an implementation of the function enumRSGs(positive_data), i.e., the learning algorithm of RSGs from positive data, we introduce the way to save computational cost (Yokomori

2003, 2007). Suppose that input positive data is $w_1,\cdots,w_t$, and terminal symbols which occurs in positive data at least once are $a_1,\cdots,a_n$. Let M_t be a matrix whose element m_{ij} is the number of a_j in w_i. As we have seen in Section 4, each compatible shape $\vec{s}=(\#(a_1),\cdots,\#(a_n))^{\mathrm{T}}$ satisfies $M_t\vec{s}=\overrightarrow{-1}$. This implies that the number of independent variables is dimker(M_t) if other conditions to be compatible are ignored. In addition, any elements of $\vec{s}$ is not less than -1. Thus the number of candidates of compatible shapes is $O\left((\max_i|w_i|)^{\dim\ker(M_t)}\right)$.
$\lim_{t\to\infty}\dim\ker(M_t)$ is constant for any positive data of *G*, so we denote it as dim(*G*). If an upper bound of dim(G) is known, the algorithm can wait for enumeration of candidates until dimker(M_t) becomes less than it.

$G = <V,\Sigma,R,S>$ is a USG, initially, an episodic finite-state MDP.
Q = a map from $V\times U$ to $(-\infty,\infty)$.
RSG-QL () {
  $x=\varepsilon$;
  u = strategy (G,Q,ε);
  while $((a=\text{environ}\ (x,u))\neq\varepsilon)$ {
    if $(S\Rightarrow_G^* xA\alpha\Rightarrow xa\beta\alpha)$ {
      $Q(A,u)=(1-k)Q(A,u)+k\left(C(a)+\sum_{i=1}^{m}\max_{v\in U}Q(B_i,v)\right)$, where $\beta=B_1\cdots B_m$;
    }
    $x=xa$;
    u = strategy (G,Q,x);
  }
  history = history ∪ {x};
  if $(x\notin L(G))$ {
    minimals = enumRSGs (history);
    if(minimals != null) H = unifyRSGs(minimals);
  }
}

Fig. 7. RSG-QL for one episode

Now let us see how our learning algorithm works through an example. Let *G* be an RSG whose rules are

$S\to a[a,0][a,1][a,2]$, $S\to b[b,0][b,1]$, $\{[a,0],[b,0],[e,0]\}\to c[c,0]$,
$\{[a,2],[c,0]\}\to d$, $[c,0]\to e[e,0][e,1][e,2]$, $[e,0]\to f$,
$[a,1]\to g$, $[b,1]\to h[h,0]$, $\{[a,2],[e,1],[h,0]\}\to i$,
$[e,2]\to j$.

Positive data from *G* is, for instance, {*bcefijhi, bcecdijhi, bcdhi, acefijgi, bcdhi, acececefijijijgi, …*}. Fig. 8 shows the transition of dimker(M_t), the number of appeared terminal symbols $|\Sigma_t|$ and the number of compatible shapes for t sentences from some positive data.
We assumed that $\dim\ker(G) \le 5$. When dimker(M_t) is less than 5, candidates of possible shapes are enumerated.

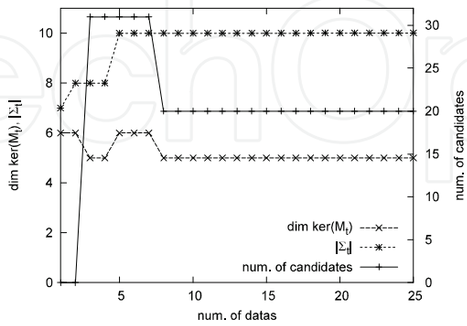

Fig. 8. dimker(M_t) and the number of candidates for possible shapes

After 100 sentences were input, the algorithm output the grammar *H* whose rules are

$$S \to a[a,0][a,1], \quad S \to b[b,0][b,1], \quad \{[a,0],[b,0],[e,0]\} \to c[c,0],$$
$$\{[a,2],[c,0]\} \to d, \quad [c,0] \to e[e,0][e,1][e,2], \quad [e,0] \to f,$$
$$[a,1] \to g[g,0], \quad [b,1] \to h[h,0], \quad \{[e,1],[g,0],[h,0]\} \to i,$$
$$[e,2] \to j.$$

L(G)=L(H) although *G* and *H* have different shapes. While there are 15 candidates for possible shape, recall that all of them do not give the same minimal language. The ambiguity of *G* is usually less than it. In this case, the ambiguity of *G* is 4.

6.2 An experiment for RSG-QL

Finally, as an experiment for RSG-QL, we consider the problem of maximizing total reword in a maze (Fig. 9).
A robot starts from the position st=(1,2) on the map of Fig. 9 and moves towards the goal gl=(9,2). The robot observes the position where it is. The robot can take four kinds of actions, left, right, up or down. The robot is given a reward -1 per single step.

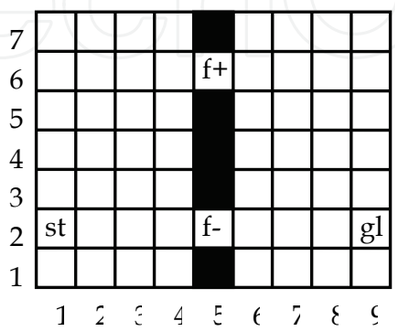

Fig. 9. The map of the maze.

The difference from ordinary maze problems is the way to give reward after the goal. The robot is allowed to occupy a location either f+=(5,6) or f-=(5,2) at most once. If the robot arrives at the goal, the robot is given two different kinds of reward that depend on its path. If the robot has passed through f+, it observes h+ with probability 0.9 and h- with 0.1. On the other hand, if the robot has passed through f-, it observes h+ with 0.1 and h- with 0.9. The reward corresponding to observation of h+ is 100, and of h- is 50. So, the best action of the robot is to pass through f+.

Formally, the RSG $G=<V,\Sigma,R,S>$ representing the environment is written as follows.

$$\Sigma = \mathrm{MAP} \cup \{\mathrm{h}+,\mathrm{h}-\}.$$

$$V = \{[a,0] \mid a \in \mathrm{MAP},\ a \neq \mathrm{gl}\} \cup \{[\mathrm{f}+,0],[\mathrm{f}-,0],S\}.$$

$$\begin{aligned} R = {} & \{[a,0] \to b[b,0] \mid a \triangleright b,\ b \neq \mathrm{gl},\mathrm{f}+,\mathrm{f}-\} \\ & \cup \{[a,0] \to \mathrm{f}+[\mathrm{f}+,0][\mathrm{f}+,1] \mid a \triangleright \mathrm{f}+\} \\ & \cup \{[a,0] \to \mathrm{f}-[\mathrm{f}-,0][\mathrm{f}-,1] \mid a \triangleright \mathrm{f}-\} \\ & \cup \{[a,0] \to \mathrm{gl} \mid a \triangleright \mathrm{gl}\} \\ & \cup \{[\mathrm{f}+,1] \to \mathrm{h}+,\ [\mathrm{f}+,1] \to \mathrm{h}-,\ [\mathrm{f}-,1] \to \mathrm{h}+,\ [\mathrm{f}-,1] \to \mathrm{h}-,\ S \to \mathrm{st}[\mathrm{st},0]\}, \end{aligned}$$

where MAP = {(*i*,*j*) | (*i*,*j*) is a reachable position on the map in Fig. 9}, and $a \triangleright b$ if and only if the robot can move from *a* to *b* in one step. $a \triangleright b \triangleright c$ denotes $a \triangleright b$ and $b \triangleright c$. For instance, $(4,6) \triangleright \mathrm{f}+ \triangleright (6,6)$, $(6,6) \not\triangleright \mathrm{f}+$ and $\mathrm{gl} \not\triangleright$ anywhere.

There is another $H=<V',\Sigma,R',S>$ such that $L(G) = L(H)$, where V' and R' are as follows.

$$V' = \{[a,0] \mid a \in \mathrm{MAP}\} \cup \{S\}.$$

$$\begin{aligned} R' = {} & \{[a,0] \to b[b,0] \mid a \triangleright b\} \\ & \cup \{[\mathrm{gl},0] \to \mathrm{h}+,\ [\mathrm{gl},0] \to \mathrm{h}-,\ S \to \mathrm{st}[\mathrm{st},0]\}. \end{aligned}$$

Fig.10 shows the shapes of *G* and *H* for gl, f+ and f-. the RSG-DP based on *G* cannot identified with any finite-state MDP, while the one on *H* is identified with an episodic finite-state MDP.

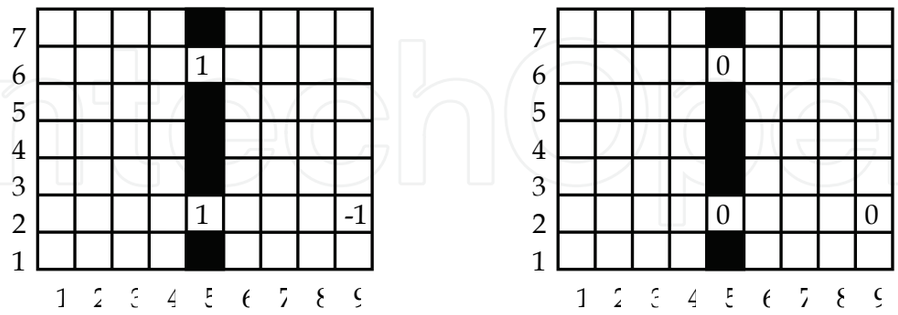

Fig. 10. (Left) The shape of *G* (Right) The shape of *H*

G and *H* are all the RSGs in canonical form whose languages are equivalent to *L*(*G*). Thus both *G* and *H*, and only *G* and *H* are enumerated by the learning algorithm for RSGs from positive data.

The USG $G^*=<V^*,\Sigma,R^*,S>$ which is the result of unifyRSG(*G*,*H*), i.e. the algorithm in Fig. 6, is as follows.

$$V^* = \{[a,0] \mid a \in \text{WEST}\} \cup \{[a,0]+,[a,0]- \mid a \in \text{EAST}\} \cup \{[\text{f}+,0]+,[\text{f}-,0]+,[\text{f}+,0]-,[\text{f}-,0]-,S\}.$$

$$\begin{aligned} R^* = & \{[a,0] \to b[b,0] \mid a \triangleright \text{b},\ b \in \text{WEST}\} \\ & \cup \{[a,0]+ \to b[b,0]+,\ [a,0]- \to b[b,0]- \mid a \triangleright \text{b}, b \in \text{EAST}\} \\ & \cup \{[a,0]+ \to \text{gl}[\text{f}+,\text{l}],\ [a,0]- \to \text{gl}[\text{f}-,\text{l}] \mid a \triangleright \text{gl}\} \\ & \cup \{(4,6) \to \text{f}+[\text{f}+,0]+,\ (4,2) \to \text{f}-[\text{f}-,0]-\} \\ & \cup \{[\text{f}+,\text{l}] \to \text{h}+,\ [\text{f}+,\text{l}] \to \text{h}-,\ [\text{f}-,\text{l}] \to \text{h}+,\ [\text{f}-,\text{l}] \to \text{h}-,\ S \to \text{st}[\text{st},0]\}, \end{aligned}$$

where WEST denotes the left half of MAP, that is, $\{(i,j) \in \text{MAP} \mid i \le 4\}$, and EAST denotes the right half of MAP, that is, $\{(i,j) \in \text{MAP} \mid i \ge 6\}$.

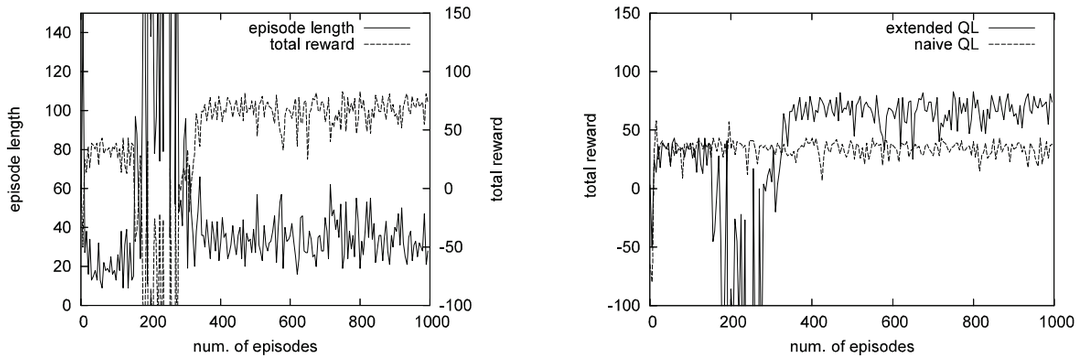

Fig.11. (Left) Total reward and episode length of RSG-QL, (Right) Comparison of naive QL and RSG-QL

Note that G^* is not an RSG. Because the SG-DP based on G^* is not an RSG-DP, it is not identified with any episodic finite-state MDP.

The optimal length of episode of this problem is 16 when the robot passes f+, and thus the maximum total reward is 79. The left side of Fig. 11 shows a result of the experiment of the SG-QL algorithm in Fig. 7 on the above maze problem. The algorithm converges to one USG as the basis of the environment at approximately the 200th episode. The result demonstrates that the robot approaches the optimal path and obtains maximum total reward after the completion of the inference. In the right side of Fig. 11, our method is compared to the naive Q-Learning method, in which the environment is assumed to be an episodic finite MDP (the same thing as H). The total reward obtained by the naive Q-Learning method is approximately 40, indicating that the robot is passing through f- and failing to maximize total reward.

7. Conclusion

In this chapter, we presented two new notions. One is an extension of episodic finite-state MDPs from the point of view of grammatical formalism. We can extend well-known methods of reinforcement learning and apply them to this extension easily. The other is the probabilistic generalities of grammars and unifiability of them. This notion plays an important role to apply the recent results of grammatical inference area. The difficulty with

applying them is the need of consideration for the probabilistic generality of grammars. The reason is that, if the languages of two grammars are equivalent, it is not necessary that the generalities of them are also equivalent. We presented the idea of unifiability and a method to unify grammars in some grammar class to overcome the difficulty.

Episodic finite-state MDPs can be extended by using the class of SGs and its subclasses; VSG, RSG, USG and SG itself. Although the class of SG-DPs is enough large to contain all episodic finite-state MDPs, the class of SGs is neither learnable from positive data nor unifiable. The class of VSGs is efficiently learnable in the limit from positive data, but the class of VSG-DPs does not include the class of episodic finite-state MDPs. The class of USG-DPs contains all episodic finite-state MDPs and the class of USGs is unifiable, but no efficient learning algorithm for it is known yet. In the four subclasses of simple grammars, the class of RSGs is the only class that satisfies efficient learnability and unifiablity and contains all episodic finite-state MDPs at the same time. The class of RSGs is not unifiable within itself, but it is unifiable within the class of USGs.

Finally, we presented the method RSG-QL for RSG-DPs, combining the extended Q-learning with the learning algorithm and the unification algorithm. Using a maximize-reward problem in a simple maze, we demonstrated that RSG-QL learns the best answer, but the naive QL does not, when the environment is regarded as an RSG-DP. The advantage of RSG-QL is that it is applicable for the wider class of environment with requiring no prior knowledge except that the environment is regarded as an RSG-DP. On the other hand, RSG-QL requires the environment to be precisely identified with some RSG-DP, otherwise the learning algorithm for RSGs from positive data does not work well. In future work, it is required to find algorithms that are stronger for errors. That might be established by using statistical learning methods for grammatical inferences.

8. References

Angluin, D. (1980). Inductive inference of formal languages from positive data, *Information and Control 45,* pp.117-135.

Angluin, D. (1982). Inference of reversible languages, *Journal of the Association for Computing Machinery 29*, pp. 741-765.

Barto, A. G. & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning, *Discrete Event Dynamic Systems: Theory and Applications 13,* pp.41-77.

Bertsekas, D. P. & Tsitsiklis, J. N. (1996). *Neuro-dynamic Programming*, Athena Scientific, Sec. 5.6.

Gold, E. M. (1967). Language identification in the limit, *Information and Control 10-5*, pp.447-474.

Hirshfeld, Y., Jerrum, M. & Moller, F. (1996). A polynomial algorithm for deciding bisimilarity of normed context-free processes, *Theoretical Computer Science 158,* pp. 143-159.

Kobayashi, S. (2000). Iterated transductions and efficient learning from positive data: A unifying view, *Proceedings of the 5th International Colloquium on Grammatical Inference, Lecture Notes in Computer Science 1891*, pp. 157-170.

Kaelbling, L. P. , Littman, M. L. & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains, *Artificial Intelligence 101* pp.99-134.

Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science 185,* pp. 15-45.

Shibata, T., Yoshinaka, R. & Chikayama, T. (2006). Probabilistic Generalization of Simple Grammars and Its Application to Reinforcement Learning, *Proceedings of the 17th International Conference on Algorithmic Learning Theory*, *Lecture Notes in Computer Science 4264*, pp.348-362.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction,* MIT Press

Wakatsuki, M., Teraguchi, K. & Tomita, E. (2004). Polynomial time identification of strict deterministic restricted one-counter automata in some class from positive data, *Proceedings of the 7th International Colloquium on Grammatical Inference, Lecture Notes in Computer Science 3264* pp.260-272.

Wetherell, C. S. (1980). Probabilistic languages: A review and some open questions, *Computing Surveys 12-4,* pp.361-379.

Yokomori, T. (2003). Polynomial-time identification of very simple grammars from positive data, *Theoretical Computer Science 298*, pp.179-206.

Yokomori, T. (2007). Polynomial-time identification of very simple grammars from positive data, Theoretical Computer Science 377(1-3), pp.282-283.

Yoshinaka, R. (2006). Polynomial-Time Identification of an Extension of Very Simple Grammars from Positive Data, *Proceedings of the 8th International Colloquium on Grammatical Inference, Lecture Notes in Computer Science 4201*, pp.45-58.

Reinforcement Learning

Edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer

ISBN 978-3-902613-14-1
Hard cover, 424 pages
Publisher I-Tech Education and Publishing
Published online 01, January, 2008
Published in print edition January, 2008

Brains rule the world, and brain-like computation is increasingly used in computers and electronic devices. Brain-like computation is about processing and interpreting data or directly putting forward and performing actions. Learning is a very important aspect. This book is on reinforcement learning which involves performing actions to achieve a goal. The first 11 chapters of this book describe and extend the scope of reinforcement learning. The remaining 11 chapters show that there is already wide usage in numerous fields. Reinforcement learning can tackle control tasks that are too complex for traditional, hand-designed, non-learning controllers. As learning computers can deal with technical complexities, the tasks of human operators remain to specify goals on increasingly higher levels. This book shows that reinforcement learning is a very dynamic area in terms of theory and applications and it shall stimulate and encourage new research in this field.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen