A Stochastic Action Language $\mathcal{A}$

Hiroaki Watanabe and **Stephen Muggleton**
Department of Computing,
Imperial College of Science, Technology and Medicine,
University of London,
180 Queen's Gate, London SW7 2BZ, UK.
{hw3,shm}@doc.ic.ac.uk

Abstract

In this paper we present a new stochastic non-deterministic high-level action language $\mathcal{SAA}$ which is a stochastic extension of Action Language $\mathcal{A}$. We describe the syntax and semantics of $\mathcal{SAA}$ and show it has an equivalent expressive power to Hidden Markov Models (HMMs). The main advantage of $\mathcal{SAA}$ is its smooth conversion of propositions and probability, and use of a well-established stochastic model. We show two simple examples in the nuclear reactor domain and propose a normalisation technique for declarative probability assignments which match our intuition.

1 Introduction

We propose a new Stochastic non-deterministic Action Language $\mathcal{A}$ (SAA) which has an expressive power equivalent to Hidden Markov Models(HMMs). During the past 30 years Logic Based Artificial Intelligence (LBAI) has investigated a wide variety of representations and methods for reasoning about the world (Minker 2000). Large quantities of time-series data is available in various fields such as cognitive robotics, reliability engineerings, and bioinformatics. In order to meet the demands of analyzing such data in LBAI frameworks, there has been a need to develop a high-level action language which supports declarative descriptions of changing worlds in a simple way. Previous studies(Reiter 2001)(Shanahan 1997) have indicated several challenges. These include the 'frame problem' and the 'ramification problem'. Such difficulties increase the hard task of programming, and act as a bottleneck in real-world problems. In addition to issues related to description complexity, uncertainty is also a issue for handling the dynamic world.

Action Language $\mathcal{A}$ (Gelfond & Lifschitz 1993) has provided a basic framework for high-level action language research in terms of syntax and semantics. Nabeshima et al(Nabeshima & Inoue 1997) propose Automata Theory for analyzing action languages and showed the expressive power of Action Language $\mathcal{A}$ is equivalent to finite automata. In order to extend the expressive power of $\mathcal{A}$, many variants of high-level action languages have been designed(Baral & Gelfond 1997). One of the advantages of using Automata Theory is to provide the formal language theory view point for the action language design. For instance, Action Language $\mathcal{NA}$(Nabeshima, Inoue, & Haneda 1999) is designed based on nondeterministic finite automata. On the high-level action language under uncertainty, few investigations have been carried out except the language PAL (Baral & Tuan 2001). PAL is designed to represent Markov Decision Process (MDP). They assign probability to "unknown" variables which are unobserved fluents, and uniform probability for next possible state transitions. However, they have not investigated the relationship between PAL and HMMs.

This paper presents a stochastic extension of Action Language $\mathcal{A}$ and provides the syntax (Section 2) and semantics (Section 3) of the new action language $\mathcal{SAA}$. We also provide two examples in the nuclear reactor domain (Section 4) to show how to convert the domain description into HMMs.

2 Syntax of $\mathcal{SAA}$

The syntax of SAA is based on Action Language $\mathcal{A}$. In Language $\mathcal{A}$ situations and actions are represented by effect propositions and value propositions. In $\mathcal{SAA}$ stochastic versions of these statements are expressed the conditional probability notation of $Pr(|)$.

Let $A_{saa} = \{a_1, ..., a_m\}$ be a non-empty finite set of action names. We use a for simply denoting an action name. Let $F = \{F_1, ..., F_n\}$ be a non-empty finite set of fluent names. A positive fluent t_n is defined as a lower case version of a fluent name F_n. A negative fluent $\neg t_n$ is a logical negation of t_n. We simply use f for denoting a fluent that is either positive or negative. Let $T = \{T_1, ..., T_n\}$ be a family of sets $T_n = \{t_n, \neg t_n\}$. Let U be a set of the cross product (or Cartesian product) of $T_n \subset T$. A state name Q is a set of fluents appearing in a tuple from U. Let $Q_{saa} = \{Q_1, .., Q_k\}$ be a family of sets of every state name appearing in U.

Example 1: state name.
Let $T_1 = \{t_1, \neg t_1\}, T_2 = \{t_2, \neg t_2\}$. $U = T_1 \times T_2 =$

$\{(t_1,t_2),(t_1,\neg t_2),(\neg t_1,t_2),(\neg t_1,\neg t_2)\}$. $Q_1 = \{t_1,t_2\}$. $Q_2 = \{t_1,\neg t_2\}$. $Q_3 = \{\neg t_1,t_2\}$. $Q_4 = \{\neg t_1,\neg t_2\}$. $Q_{saa} = \{\{t_1,t_2\},\{t_1,\neg t_2\},\{\neg t_1,t_2\},\{\neg t_1,\neg t_2\}\}$.

Let $\mathcal{D} = \{D_1,...,D_l\}$ be a family of sets of action names $D_l = \{d_1,...,d_x\}$ where x is a finite number. Let c be a constant number in the range $[0,1]$.

In Action Language $\mathcal{A}$, effect propositions are defined to represent effects of actions as follows.

$$a\ \textbf{\textit{causes}}\ f\ \textbf{\textit{if}}\ p_1,..,p_m \quad (m \geq 0)$$

where ***causes*** and ***if*** are connectives. $f,p_1,..,p_m$ are fluents. We call f an *effect of action*. We say $p_1,..,p_m$ is a precondition. For convenience, we use a notation of set for preconditions. Let $\mathcal{P} = \{P_1,...P_n\}$ be a family of sets $P_n = \{p_1,..,p_m\}$. We simply denote a precondition as P and rewrite the effect proposition as

$$a\ \textbf{\textit{causes}}\ f\ \textbf{\textit{if}}\ P.$$

We extend the effect proposition to a stochastic conditional proposition so called *Stochastic Effect Proposition* (SEP) in

$$Pr(a\ \textbf{\textit{causes}}\ f \mid p_1,..,p_m) = c \quad (m \geq 0)$$

$$\text{or} \quad Pr(a\ \textbf{\textit{causes}}\ f \mid P) = c.$$

We also allow an *unconditional* SEP (uSEP) in

$$Pr(a\ \textbf{\textit{causes}}\ f) = c$$

which is a stochastic extension of

$$a\ \textbf{\textit{causes}}\ f$$

where the connective ***if*** and the precondition is omitted since $p_1,..,p_m$ is always true.

In Action Language $\mathcal{A}$, value propositions are defined to represent observed evidences as follows.

$$f\ \textbf{\textit{after}}\ d_1,...,d_x \quad (x \geq 0)$$

$$\text{or} \quad f\ \textbf{\textit{after}}\ D.$$

We extend this to a *Stochastic Value Proposition* (SVP) in

$$Pr(f \mid D) = c$$

The statement $Pr(\textbf{initially}\ f_1,..,f_o) = c$ is called a *stochastic initial value proposition*. $f_1,..,f_o$ is called an *initial fluent*. For convenience, set notation is used for initial fluents. Let $\mathcal{I} = \{I_1,..,I_p\}$ be a family of sets $I_p = \{f_1,..,f_o\}$.

A $\mathcal{SAA}$ S is a set of SEPs and SVPs. More formally, $\mathcal{SAA}$ is defined as the 4-tuple $S = < A,F,E,V >$ where

$A = \{a_1,...,a_m\}$: A non-empty finite set of action names,
$F = \{F_1,...,F_n\}$: A non-empty finite set of fluent names,
$E = \{(a_m,f_j,P_n,c)\}$: a non-empty finite set of SEP,
$V = \{(f_m,D_l,c)\}$: a non-empty finite set of SVP.

$\mathcal{S}_{P_n}$, the subset of E with the precondition P_n, is called the *definition of precondition* P_n. If every *initial fluents* I_p in SVPs is an element of Q_{saa}, I is said to be pure. I is impure otherwise.

Definition 1: Complete $\mathcal{SAA}$
$\mathcal{SAA}$ is *complete* if all of the following conditions are satisfied, otherwise *incomplete*.

1. For a family of sets of preconditions $\mathcal{P} = \{P_1,..,P_n\}$, the summation of constants c in the *definition of the precondition* P_n equals 1 for each n.
$$\sum_{c\in\mathcal{S}_{P_1}} c = 1, ..., \sum_{c\in\mathcal{S}_{P_n}} c = 1.$$
2. The summation of all constants c of the stochastic initial value proposition in SVP equals 1.
3. E does not contain any uSEP,
4. Every preconditions P_n in SEPs is an element of Q_{saa},
5. Every initial fluents I_p in $\mathcal{I}$ is an element of Q_{saa},

Definition 2: Deterministic and Non-deterministic
If there exist two or more occurrences of an action name in the definition of P_n, we say the P_n and $\mathcal{SAA}$ $\mathcal{S}$ is non-deterministic and deterministic otherwise.

Example 2: Assume the following SEPs define $\mathcal{S}_{P_2}$:

$$Pr(a_1\ \textbf{\textit{causes}}\ f_1 \mid P_2) = 0.1$$
$$Pr(a_2\ \textbf{\textit{causes}}\ f_1 \mid P_2) = 0.2$$
$$Pr(a_4\ \textbf{\textit{causes}}\ f_1 \mid P_2) = 0.3$$
$$Pr(a_4\ \textbf{\textit{causes}}\ f_3 \mid P_2) = 0.7$$

$\mathcal{S}$ is incomplete because $\sum_{c\in S_{P_2}} c = 0.1+0.2+0.3+0.7 \neq 1$, and non-deterministic since there exist two a_4 in $\mathcal{S}_{P_2}$.

3 Semantics of $\mathcal{SAA}$

This section provides the semantics of $\mathcal{SAA}$. A domain description written in $\mathcal{SAA}$ specifies a stochastic state transition diagram. E (or SEPs) defines a probability distribution of the state transitions and a probability distribution of actions. We show a complete $\mathcal{SAA}$ has a stochastic non-deterministic state transition diagram equivalent to HMMs. We consider $S =< A_{saa}, Q_{saa}, E, V >$ as a $\mathcal{SAA}$ instead of $S =< A_{saa}, F, E, V >$. Note that the transformation between Q_{saa} and F is straightforward.

Formally, we give an interpretation of $\mathcal{SAA}$ in the stochastic state transition diagram $M = (Q_M, A_M, \psi, \varphi, \pi)$ where $Q_M = \{q_1,..,q_m\}$ is a non-empty finite set of states, $A_M = \{a_1,..,a_n\}$ is a non-empty finite set of actions, $\psi = \{\psi_{q_iq_j}\}$ is probability distribution of the state transition from state q_i to q_j where $\sum_j \psi_{q_iq_j} = 1$, $\varphi = \{\varphi_{q_iq_j}(a_n)\}$ is probability distribution of action a_n taken between state q_i to q_j where $\sum_n \varphi_{q_iq_j}(a_n) = 1$, and $\pi = \{\pi_{q_i}\}$ is probability distribution of the initial state q_i where $\sum_i \pi_{q_i} = 1$.

A function ϕ from A to B is called *one-to-one* if whenever $a1 \neq a2$ $(a1, a2 \in A)$ then ϕ $(a1) \neq \phi$ $(a2)$. A function ϕ from A to B is called *onto* if for all $b \in B$ there is an $a \in A$ such that $f(a) = b$. Let $name(f)$ be a function which returns the fluent name of f. We define a function $replace(g, P)$ which replaces the fluent f in the precondition P with the fluent g whenever $name(f) = name(g)$ if $name(g) \in F$. $replace(g, P)$ is based on the *inertia* low.

Example 3: $replace(g, P)$**.** For the precondition $P = \{f_1, .., f_k, .., f_i\}$, $replace(\neg f_k, P)$ returns the set of fluents $P^{'} = \{f_1, .., \neg f_k, .., f_i\}$.

Let $sub(f, P)$ be a function of selecting every SEPs with the *effect of action* f from the *definition of precondition* P and returning them as a subset $S_{P,f}$. Let $\sum_{S_{P,f}} c$ be the summation of the constant in $S_{P,f}$. $norm(a, f, P) = c / \sum_{S_{P,f}}$ calculates a normalized constant c of $Pr(a$ ***causes*** $f \mid P) = c$.

Example 4: $sub(f, P)$**,** $\sum_{S_{P,f}} c$**, and** $norm(a, f, P)$. For the following $\mathcal{SAA}$:

$$Pr(a_1 \textbf{\textit{ causes }} f_2 \mid P_1) = c_1$$
$$Pr(a_1 \textbf{\textit{ causes }} f_3 \mid P_1) = c_2$$
$$Pr(a_2 \textbf{\textit{ causes }} f_2 \mid P_1) = c_3$$
$$Pr(a_1 \textbf{\textit{ causes }} f_2 \mid P_2) = c_4$$

$sub(f_2, P_1)$ generates the following subset S_{P_1,f_2}:

$$Pr(a_1 \textbf{\textit{ causes }} f_2 \mid P_1) = c_1$$
$$Pr(a_2 \textbf{\textit{ causes }} f_2 \mid P_1) = c_3$$

where $\sum_{S_{P_1,f_2}} c = c_1 + c_3$, $norm(a_1, f_2, P_1) = c_1/(c_1 + c_3)$.

Let $full\_state(P, Q)$ be a function of finding all state names Q_k where $P \subset Q_k$ and Q_k is an element of Q, and returning them as a family of sets. We denote an element of the family of sets as P_{full}. Note that $P_{full} = P$ if $P = Q_k$. We can view $full\_state(P, Q)$ is based on the *inertia* low.

Example 5: $full\_state(P)$.
Let $P_1 = \{t_1\}$. For $Q_1, Q_2, Q_3, Q_4, and Q_{saa}$ shown at Example 1, $full\_state(P_1, Q_{saa}) = \{Q_1, Q_2\}$.

The interpretation of Q_{saa} is given by Q_M. Note that Q_{saa} is a family of sets and Q_M is a set. We consider a state name as an state-object and Q_{saa} as a set of state-objects. Let $q_{saa} = \{f_1, .., f_s\}$ be a state-object and $q_{saa} \in Q_{saa}$. We give a one-to-one onto function ϕ_q from Q_{saa} to Q_M where the state $\phi(q_{saa}) \in Q_M$ contains $f_1, .., f_s$ - all the members of q_{saa}. Note that $|Q_{saa}| = |Q_M|$ is required for ϕ_q. The interpretation of A_{saa} are given by A_M. We define a one-to-one onto function ϕ_a from A_{saa} to A_M where $|A_{saa}| = |A_M|$ is required.

At the state q, a SEP $Pr(a$ ***causes*** $f \mid P) = c$ is said to be *true* if the precondition $P = \{p_1, .., p_m\} \subset q_{saa}$, that is, if the state q contains all the members of P. A SVP $Pr(f \mid d_1, ..., d_x) = c$ is said to be *true* if f is a member of q_{saa} and

$$c = \sum_{path} \Pi_{t=1}^{x} \psi_{q_{t-1}q_t} \varphi_{q_{t-1}q_t}(d_t)$$

is satisfied where $\psi_{q_0 q_1} = \pi_{q1}$. Note that $\sum_{path}$ sums up every transition path which generates the action sequence $d_1, ..., d_x$.

The interpretation of E is given by ψ (probability distribution of the state transitions) and φ (probability distribution of actions). Generally speaking, E is involving uSEPs and SEPs whose preconditions are not the element of Q_{saa}. The compactness of $\mathcal{P}$ is decompressed and extracted by $full\_state(P, Q_{saa})$ into a subset of Q_{saa}. uSEP, $Pr(a$ ***causes*** $f) = c$, is also modified to SEPs $Pr(a$ ***causes*** $f \mid Q_k) = c$ for every k of $Q_{saa} = \{Q_1, .., Q_k\}$. We assume P_n is an element of Q_{saa} in the following part. A constructive view of E provides a clear explanation for the interpretation of E. Recall another definition of E: $E = \{S_{P_1}, .., S_{P_n}\}$ is a family of sets of the *definitions-of-precondition-*P_n S_{P_n} where S_{P_n} is a set of SEPs with the precondition P_n. First at $Pr($ a **causes** $f \mid P_n) = c \in S_{P_n}$ level, the action $\phi_a(a) \in A_M$ causes a stochastic state transition from $\phi_q(P_n) \in Q_M$ to $\phi_q(replace(f, P_n)) \in Q_M$. Next at S_{P_n} level, S_{P_n} contains all stochastic state transitions originated from the state $\phi_q(P_n) \in Q_M$. Note that if the S_{P_n} is deterministic, S_{P_n} represents a stochastic state transition function from the state $\phi_q(P_n)$ and actions to states. In case S_{P_n} is non-deterministic, it maps from the state $\phi_q(P_n)$ and actions to a set of states. $S_{P_n,f}$ contains all stochastic state transitions from the state $\phi_q(P_n) \in Q_M$ to the state $\phi_q(replace(f, P_n)) \in Q_M$. We interpret

$$\sum_{S_{P_n,f}} c = \psi_{\phi_q(P_n)\phi_q(replace(f,P_n))} \quad \text{and}$$

$$norm(a, f, P_n) = \varphi_{\phi_q(P_n)\phi_q(replace(f,P_n))}(\phi_a(a)).$$

Finally at the top level, E contains all stochastic state transitions originated from every state in E and all probability distribution of actions.

The interpretation of the stochastic initial value propositions is $c = \pi_{\phi_q(I_p)}$. In case $\mathcal{I}$ is not *pure*, we use $full\_state(I_p, Q_{saa})$ and assign $c/|full\_state(I_p, Q_{saa})|$ for each new stochastic initial value proposition.

Next we discuss on the model of the domain description. Any domain description has a stochastic model under the interpretation M if following conditions are satisfied.

1. $|Q_{saa}| = |Q_M|$ and $|A_{saa}| = |A_M|$ are satisfied.
2. SVPs in the domain description are true in M.

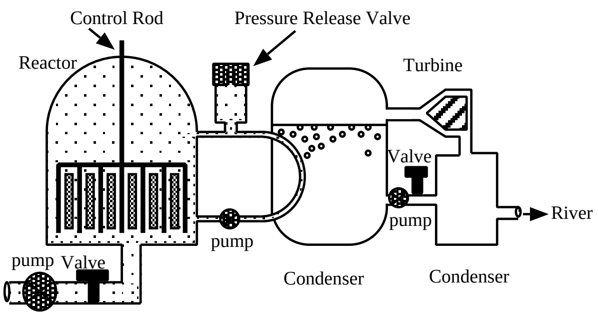

Figure 1: Three Mile Island Nuclear Power Station Model

3. $\sum_j \psi_{q_i q_j} = 1$, $\sum_n \varphi_{q_i q_j}(a_n) = 1$, and $\sum_i \pi_{q_i} = 1$ are satisfied.

The stochastic model is equivalent to HMMs. Every domain descriptions described in *complete* $\mathcal{SAA}$ has a HMMs as their models.

4 Examples: Reactor Domain

We give an example of a *complete* stochastic domain description. Fig.1 is a simple pressurized water reactor model based on the nuclear power plant at Three Mile Island. In the reactor, nuclear fuel is used to produce heat which is applied to turn the water into steam in the first condenser. The steam spins the turbine and turns back into water at the second condenser. In the nuclear reactor domain (Fig.2), we only consider two components, a control rod and a pressure release valve. The state of the control rod is explained by the *down* fluent. If *down* is true, the control rod is inserted into the nuclear fuel core and stops the nuclear reaction. If $\neg down$, the control rod does not moderate the reaction and reactor pressure would increase. The state of the pressure release valve is represented by the *close* fluent. If *close* is true, the reactor pressure is below a threshold level and nuclear pollution does not happen. Once the pressure is over the threshold level, the valve opens automatically and the nuclear pollution would be spreading. The valve closes automatically when the pressure level becomes lower than the threshold level. Fig.1 and Fig.2 show the state $(down, close)$ and $(\neg down, \neg close)$ respectively. All the possible actions for the system controller are rod_up, rod_down, and $sleep$. First, the following *deterministic* domain description is used to show the procedure of creating a model.

$Pr($**Initially** $close, down) = 0.6$
$Pr($**Initially** $close, \neg down) = 0.4$
$Pr(rod\_up$ **causes** $\neg down | close, down) = 0.3$
$Pr(rod\_down$ **causes** $down | close, down) = 0.1$
$Pr(sleep$ **causes** $down | close, down) = 0.6$
$Pr(rod\_up$ **causes** $\neg down | close, \neg down) = 0.1$
$Pr(rod\_down$ **causes** $down | close, \neg down) = 0.8$
$Pr(sleep$ **causes** $\neg close | close, \neg down) = 0.1$
$Pr(rod\_up$ **causes** $\neg down | \neg close, \neg down) = 0.07$
$Pr(rod\_down$ **causes** $down | \neg close, \neg down) = 0.9$
$Pr(sleep$ **causes** $\neg close | \neg close, \neg down) = 0.03$

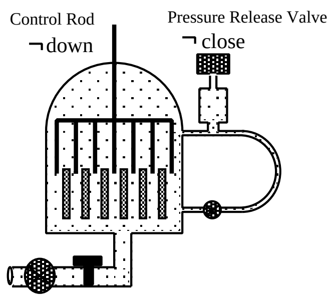

Figure 2: A Nuclear Reactor Model

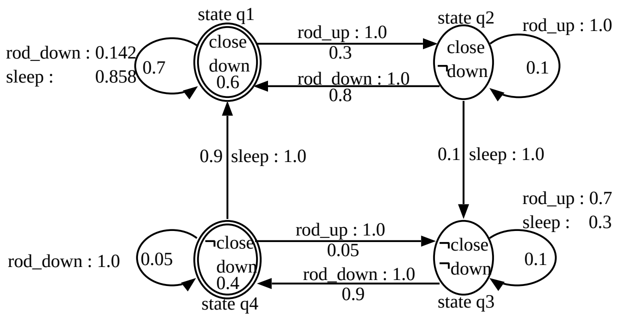

Figure 3: A deterministic Stochastic Transition Diagram

$Pr(rod\_up$ **causes** $\neg down | \neg close, down) = 0.05$
$Pr(rod\_down$ **causes** $\neg close | \neg close, down) = 0.05$
$Pr(sleep$ **causes** $close | \neg close, down) = 0.9$

We can convert above domain description into the stochastic state transition diagram in three steps.

Step1: Create a family of set of *definitions of precondition*. Let $P_1 = \{close, down\}, P_2 = \{\neg close, down\}, P_3 = \{close, \neg down\}, P_4 = \{\neg close, \neg down\}$.For instance, $S_{P_1} = \{Pr(rod\_up$ **causes** $\neg down | close, down) = 0.3,$
$Pr(rod\_down$ **causes** $down | close, down) = 0.1,$
$Pr(sleep$ **causes** $down | close, down) = 0.6\}$.

Step2: Generate a stochastic state transition function by defining ψ and φ. For example, $S_{P_1, down} = \{$
$Pr(rod\_down$ **causes** $down | close, down) = 0.1,$
$Pr(sleep$ **causes** $down | close, down) = 0.6\}$ represents stochastic state transitions from state $q_1 (= \phi_q(P_1))$ to q_1 in probability $\sum_{P_1, down} = 0.1 + 0.6 = 0.7$, that is, $\psi_{q_1 q_1} = 0.7$. We also get $\varphi_{q_1 q_1}(rod\_down) = 0.1/0.7 = 0.143$ and $\varphi_{q_1 q_1}(sleep) = 0.6/0.7 = 0.857$.

Step3: Assign the initial state probability π. For instance, $\pi_{q1} = 0.6$.

Next we consider the *non-deterministic* version of the reactor domain description. Only S_{P_1} is modified as follows in order to introduce non-determinacy.

$Pr(rod\_up$ **causes** $\neg down \mid close, down) = 0.29$

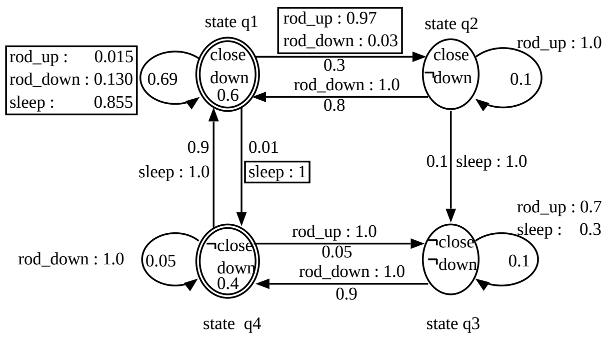

Figure 4: A non-deterministic Stochastic Transition Diagram

$Pr(rod\_up$ **causes** $down \mid close, down) = 0.01$
$Pr(rod\_down$ **causes** $down \mid close, down) = 0.09$
$Pr(rod\_down$ **causes** $\neg down \mid close, down) = 0.01$
$Pr(sleep$ **causes** $down \mid close, down) = 0.59$
$Pr(sleep$ **causes** $open \mid close, down) = 0.01$

5 Discussion and Conclusion

In the previous sections, we showed examples of the *complete* $\mathcal{SAA}$. The semantics of probability on HMMs is clear, however, at the high-level language level it sometimes does not meet our intuitions even in the *complete* domain description. One approach for assigning constants of SEPs based on our intuitions is to apply the normalization of probability. Consider failures of actions in probability setting (Bacchus, Halpern, & Levesque 1995). In the reactor domain, failures of actions can be represented as follows.

$Pr(rod\_up$ **causes** $\neg down \mid close, down) = 0.95$
$Pr(rod\_up$ **causes** $down \mid close, down) = 0.05$
$Pr(rod\_down$ **causes** $down \mid close, down) = 0.9$
$Pr(rod\_down$ **causes** $\neg down \mid close, down) = 0.1$
$Pr(sleep$ **causes** $down \mid close, down) = 0.99$
$Pr(sleep$ **causes** $open \mid close, down) = 0.01$

Each action name causes different results. Note that the summation of probability at the state name $\{close, down\}$ equals 1 for each action name. By using the normalization technique, it is possible to generate a complete model.

Since our language is equivalent to HMMs, we can use various algorithms developed in HMMs. Learning $\mathcal{SAA}$ would be possible with HMMs' parameter learning and topology learning algorithms.

In this paper, we described the new stochastic action language $\mathcal{SAA}$ as a stochastic extension of Action Language $\mathcal{A}$. This stochastic language provides stochastic extension for existing Action Language $\mathcal{A}$ programs and also gives a framework for designing new stochastic action languages based on current existing other high-level action languages.

References

Bacchus, F.; Halpern, J. Y.; and Levesque, H. J. 1995. Reasoning about noisy sensors in the situation calculus. In Mellish, C., ed., *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1933–1940. San Francisco: Morgan Kaufmann.

Baral, C., and Gelfond, M. 1997. Reasoning about effects of concurrent actions. *Journal of Logic Programming* 31(1-3):85–117.

Baral, C., and Tuan, L.-C. 2001. Reasoning about actions in a probabilistic setting. In *Symposium on Logical Formalization of Common Sense 2001*.

Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming* 17(2-4):301–321.

Minker, J., ed. 2000. *Logic-Based Artificial Intelligence*. Kluwer.

Nabeshima, H., and Inoue, K. 1997. Automata theory for action language a. (in japanese). *Transactions of Information Processing Society of Japan* 38(3):462–471.

Nabeshima, H.; Inoue, K.; and Haneda, H. 1999. A non-deterministic action language based on finite automata. (in japanese). *Transactions of Information Processing Society of Japan* 40(10):3661–3671.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge, Massachusetts: The MIT Press.

Shanahan, M. 1997. *Solving the Frame Problem*. Cambridge, Massachusetts: The MIT Press.