First-Order Stochastic Action Language

Hiroaki Watanabe and Stephen Muggleton

Department of Computing,
Imperial College
University of London
180 Queen's Gate, London SW7 2BZ, UK
Email: {hw3, shm}@doc.ic.ac.uk

Abstract. Action languages are high-level languages for describing changing worlds. We propose two new stochastic action languages for increasing robustness under uncertainties. First, we introduce a stochastic version of Action Language $\mathcal{A}$ at propositional level. Then it is upgraded to Datalog Stochastic Action Language (Datalog-SAL) for handling dynamic worlds at Datalog level. A model of Datalog-SAL is defined as a set of Hidden Markov Models (HMMs). Datalog-SAL assigns probability distributions over state exchanges, actions, and initial states. We also assign Bernoulli Distributions over Herbrand Base of fluent atoms in a domain in order to define probabilistic situations. Since a model of Datalog-SAL is given in a set of HMMs, learning Datalog-SAL would be possible via HMM's parameter learning algorithms such as Baum-Welch. A complexity analysis of learning Datalog-SAL is discussed with Baum-Welch. Datalog-SAL provides a simple and smooth conversion of Datalog and probability in the action and change framework.

1 Introduction

Challenging applications such as bioinformatics and cognitive robotics contain temporal aspects in their data. In order to meet the demands of handling such data under uncertainty, there is a need to combine temporal representations with probability. Logic Based Artificial Intelligence has investigated a wide variety of representations and methods for reasoning about the temporal changing world [12]. Situation calculus[11] is one of the well-studied frameworks for representing and reasoning about dynamically changing worlds. In situation calculus, time is captured in the ways of state space where actions are considered as mapping from one state to another rather than handling time explicitly. Stochastic enhancements of situation calculus have been studied since McCarthy et. al[11] remarked their importance. Most of the previous works [1][8][17][18] have discussed stochastic extensions of state transitions or actions, but little research has been studied on expressive powers of stochastic situation calculus.

Apart from the temporal representation, many investigations have been carried out on the convergence of logic and probability [4][5][6][9][15][13][21], however, few works have discussed explicitly in the action and change framework. Our purpose in this paper is to combine logic and probability smoothly within the action and change framework. For this objectives, we introduce a stochastic version of Action Language $\mathcal{A}$, called $\mathcal{A}$-SAL first. Then, it is upgraded to a first-order version, Datalog-SAL, in order to describe relational worlds under uncertainty. The semantics of $\mathcal{A}$-SAL and Datalog-SAL is equivalent with HMM and a set of HMMs respectively [19].

The advantage of declarative language has been discussed by McCarthy [11][10] since he applied Logic for AI [14]. In addition to the declarative effect, the usage of probability also increases an elaboration tolerance. Our two languages are well-suited for describing dynamically changing worlds from the elaboration tolerance point of view as well as for uncertainty.

This paper is organised as follows. In section 2 we propose a stochastic version of Action Language $\mathcal{A}$, A-SAL, at propositional level. We discuss its syntax, semantics and expressive power as well as an example from a nuclear reactor domain. Then A-SAL is upgraded to Datalog level in section 3. Some related works are shown in section 4. The discussion of future topics concludes this paper in section 5.

2 Stochastic Action Language: $\mathcal{A}$-SAL

2.1 Introduction

A new non-deterministic Stochastic Action Language $\mathcal{A}$ (A-SAL) is introduced in this section. A-SAL is a hi-level stochastic action language which is a stochastic extension of Action Language $\mathcal{A}$ [7]. A-SAL is designed for describing markov processes in the framework of action and change at propositional level. In addition to actions and changes, A-SAL also represents probability distributions over initial states, state transitions, and actions in a 'natural language like' simple syntax. Since the semantics of A-SAL is defined in HMM or stochastic finite automata, we can view A-SAL as a declarative language for HMMs.

Action Language $\mathcal{A}$ [7] has provided a basic framework for research on high-level action languages. Nabeshima et al [14] propose Automata Theory for analyzing action languages and show the expressive power of Action Language $\mathcal{A}$ is equivalent to Finite Automata. Automata Theory provides a formal language theory point of view for action language designs. In order to extend the expressive power of $\mathcal{A}$, many variants of high-level action languages have been designed [2], however, few investigations of stochastic extensions have been carried out except the language PAL[3]. Our motivation is to provide a stochastic action language based on Automata Theory point of view.

2.2 Syntax of A-SAL

Syntax of A-SAL is based on Action Language $\mathcal{A}$. In Action Language $\mathcal{A}$, actions and changes are represented by *effect propositions*, *initial value propositions*, and *value propositions*.

In A-SAL, stochastic versions of the first two propositions are expressed with the notion of conditional probability $Pr(|)$.

Let $A = \{a_1, ..., a_m\}$ be a non-empty finite set of propositional actions. Let $F = \{f_1, ..., f_n\}$ be a non-empty finite set of propositional fluents. A *literal* l is a fluent (called *positive* fluent) or a fluent proceeded by $\neg$ (called *negative* fluent). That is, $l_i = f_i$ or $\neg f_i$. Note that we define $\neg\neg f_i = f_i$. A *formula* is a conjunction of literals, $f_1 \wedge ... \wedge f_l \quad (1 \leq l)$. A set of literals defines a state of worlds. For defining all possible states of world, a family of sets of literals is introduced in $S = \{S_1, .., S_k\}$ where $S_i = \{l_1, ..., l_n\}$ and $S_i \neq S_j$ if $i \neq j$. Note that $k = 2^n$.

Example: Fluents and States

$$
\begin{aligned}
F =& \{alive, loaded\}.\\
S =& \{\{alive, loaded\}, \{alive, \neg loaded\},\\
& \{\neg alive, loaded\}, \{\neg alive, \neg loaded\}\}.
\end{aligned}
$$

In Action Language $\mathcal{A}$, effect propositions are defined to represent effects of actions in the form of

$$a \textbf{\textit{ causes }} l \textbf{\textit{ if }} P$$

where a is an action, l is a literal, and P is a formula. l and P is called *effect of action* and *precondition* respectively. We extend the above effect propositions to *Stochastic Effect Propositions* (SEP) in

$$Pr(a \textbf{\textit{ causes }} l \mid P) = c$$

where c is a constant number in the range $[0, 1]$. We also allow *unconditional* SEP in

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

which is a stochastic extension of

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

where ***if*** and the precondition is omitted since P is always true. In Action Language $\mathcal{A}$, value propositions (VP) are defined in order to represent observed evidences in the form of

$$l \textbf{\textit{ after }} D$$

where D is a conjunction of actions. In A-SAL, we use the same value propositions for representing observed facts. The statement

$$Pr(\textbf{initially } I) = c$$

is called *stochastic initial value proposition* (SIVP) where I is a formula. Note that I is called *initial fluents*. A program is a set of SEP, VP, and SIVP.

2.3 Semantics of A-SAL

This section discusses a semantics of A-SAL. Intuitively, a domain description written in an A-SAL program specifies stochastic state transition diagrams. A state can be viewed as a node that contains literals. An action is a label attached on an arrow between two nodes.

A state q_i is an interpretation of $S_i \in S$. An action act_j is an interpretation of $a_j \in A$. At any state q_i, a formula $l_1 \wedge ... \wedge l_r$ is called *true* if and only if $l_j \in q_i$ for any i $(1 \leq j \leq r)$. In A-SAL, SEP represent state transitions and probability distributions over state transitions ψ and actions φ. Intuitively, in SEP : $Pr(a$ ***causes*** $l \mid P) = c$, a semantics of c is a part of probability of a state exchange from q_i to q_j by action a where P is *true* at q_i and $(q_i \backslash \{\neg l\}) \cup l$ is true at q_j. More precisely, when a program is given, probability of stochastic state exchanges from q_i to q_j, is defined in the following steps.

1. For each state q_i, generate a maximum subset SEP_{q_i} of SEP where every SEP_{q_i}'s *precondition* is true at q_i.
2. For each state q_j, create a maximum subset $SEP_{q_i q_j}$ of SEP_{q_i} where every l in $SEP_{q_i q_j}$ satisfies $(q_i \backslash \{\neg l\}) \cup l$ at q_j.
3. Let $c_{q_i q_j}$ be the summation of c in $SEP_{q_i q_j}$. Finally, $\psi_{q_i q_j}$ is defined as $\psi_{q_i q_j} = c_{q_i q_j} / \sum_j c_{q_i q_j}$. Note that $\sum_j c_{q_i q_j}$ is a normalization factor.

Example: Probability Distribution over State Exchanges
Assume a domain description is given by the following A-SAL program. Let $q_1 = \{f_1, f_2\},\ q_2 = \{f_1, \neg f_2\}$.

$$Pr(a_1 \ \boldsymbol{causes} \ f_1 \mid f_1, f_2) = 0.2 \quad (1)$$
$$Pr(a_2 \ \boldsymbol{causes} \ f_2 \mid f_1, f_2) = 0.1 \quad (2)$$
$$Pr(a_1 \ \boldsymbol{causes} \ \neg f_2 \mid f_1, f_2) = 0.7 \quad (3)$$
$$Pr(a_1 \ \boldsymbol{causes} \ f_1 \mid f_1, \neg f_2) = 0.1 \quad (4)$$
$$Pr(a_2 \ \boldsymbol{causes} \ f_2 \mid f_1, \neg f_2) = 0.9 \quad (5)$$
$$Pr(\textbf{Initially} \ f_1, f_2) = 0.3 \quad (6)$$
$$Pr(\textbf{Initially} \ f_1, \neg f_2) = 0.7 \quad (7)$$

$$SEP_{q_1} = \{(1),(2),(3)\}, SEP_{q_1 q_1} = \{(1),(2)\}, SEP_{q_1 q_2} = \{(3)\},$$
$$SEP_{q_2 q_2} = \{(4)\}, SEP_{q_2 q_1} = \{(5)\}$$

$$\psi_{q_1 q_1} = (0.2 + 0.1)/(0.3 + 0.7) = 0.3$$
$$\psi_{q_1 q_2} = 0.7/(0.3 + 0.7) = 0.7$$
$$\psi_{q_2 q_2} = 0.1/(0.1 + 0.9) = 0.1$$
$$\psi_{q_2 q_2} = 0.9/(0.1 + 0.9) = 0.9$$

SEP also define probability distribution over stochastic actions. We calculate $\varphi = \{\varphi_{q_i q_j a_k}\}$ which is assigned for action a_k taken from state q_i to q_j as follows:

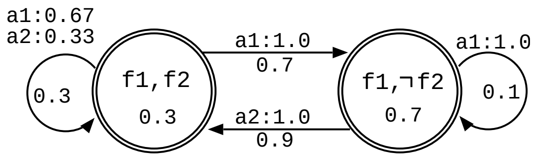

Fig. 1. An interpretation of example

1. For each action a_k, generate a maximum subset $SEP_{q_i q_j a_k}$ of $SEP_{q_i q_j}$ where $a = a_k$.
2. Let $c_{q_i q_j a_k}$ be the summation of c in $SEP_{q_i q_j a_k}$. We set $\varphi_{q_i q_j a_k} = c_{q_i q_j a_k} \ / \ c_{q_i q_j}$ where $c_{q_i q_j}$ is a normalization factor.

A probability distribution π_{q_i} over initial states is specified by SIVPs: $Pr(\textbf{initially } I) = c$ as follows:

1. For each state q_i, create a maximum subset $SIVP_{q_i}$ of $SIVPs$ where I is true at q_i.
2. Let c_{q_i} be the summation of c in $SIVP_{q_i}$. We set $\pi_{q_i} = c_{q_i} \ / \ \sum_i c_{q_i}$.

Example: Probability Distribution over Actions and Initial States
From the previous example, we can define $\varphi_{q_i q_j}(a_k)$ and π_{q_i} as follows:

$$SEP_{q_1 q_1 a_1} = \{(1)\}$$
$$SEP_{q_1 q_1 a_2} = \{(2)\}$$
$$SIVP_{q_1} = \{(6)\}$$
$$SIVP_{q_2} = \{(7)\}$$
$$\varphi_{q_1 q_1}(a_1) = 0.2/(0.1 + 0.2) = 0.67$$
$$\varphi_{q_1 q_1}(a_2) = 0.1/(0.1 + 0.2) = 0.33$$
$$\pi_{q_1} = 0.3 \ / \ (0.3 + 0.7) = 0.3$$
$$\pi_{q_2} = 0.7 \ / \ (0.3 + 0.7) = 0.7$$

2.4 Model

Formally, we give an interpretation of A-SAL in the form of $M = (S, A, \psi, \varphi, \pi)$ where

- $S = \{q_1, .., q_m\}$ is a non-empty finite set of states which are reachable from at least one initial state by taking sequences of actions.
- $A = \{act_1, .., act_n\}$ is a non-empty finite set of actions
- $\psi = \{\psi_{q_i q_j}\}$ is a probability distribution over the state transition from state q_i to q_j where $\sum_j \psi_{q_i q_j} = 1$
- $\varphi = \{\varphi_{q_i q_j}(a_k)\}$ is a probability distribution over action act_k taken from state q_i to q_j where $\sum_k \varphi_{q_i q_j}(act_k) = 1$

– $\pi = \{\pi_{q_i}\}$ is a probability distribution over the initial state q_i where $\sum_i \pi_{q_i} = 1$.

We show an interpretation of D, $M(D)$, is equivalent with HMM. Let $M(D, q_{ini})$ be a state transition diagram whose initial state is q_{ini}. When D is given, for different initial states, each $M(D, q_{ini})$ can have different topologies, however, we can merge them into a single state transition diagram by simply overlapping each other. Intuitively, this is because: (1) Each $M(D, q_{ini})$ use the same ψ and φ. (2) The calculation of ψ and φ depends on only outgoing arrows. (3) When we execute the overlapping, each node only gets new incoming arrows instead of outgoing arrows. So the overlapping operation does not change ψ and φ.

Theorem 1. *$M(D)$ is equivalent with HMM.*

We generate a interpretation by following steps.

1. Choose an initial state q_{ini} by π.
2. Create a stochastic state transition diagram by ψ and φ with the rule of S. The diagram only contains reachable states from the initial state.
3. Iterate the above two steps for every initial state.
4. Overlap every diagram one by one.

If the overlapping operation does not change ψ and φ at all, the resulted stochastic state transition diagram is clearly a stochastic finite automata. We proof the following lemma in the Appendix.

Lemma 1. *The overlapping operations do not change ψ and φ.*

In our setting, the stochastic finite automata can be seen as HMM since value propositions only provide sequence of actions and resulted literals where the internal state transitions are hidden.

$M(D)$ is called model when every value proposition can be represented by $M(D)$.

2.5 Examples: Reactor Domain

We give an example of a *complete* stochastic domain description. Fig.2 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 used 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.3), 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 the reactor pressure will 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 occur. Once the pressure is over the threshold level, the valve opens automatically leading to the spread of nuclear pollution. The valve closes automatically when the pressure level becomes lower than the threshold level. Fig.2 and Fig.3 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 domain description is used to show the procedure of

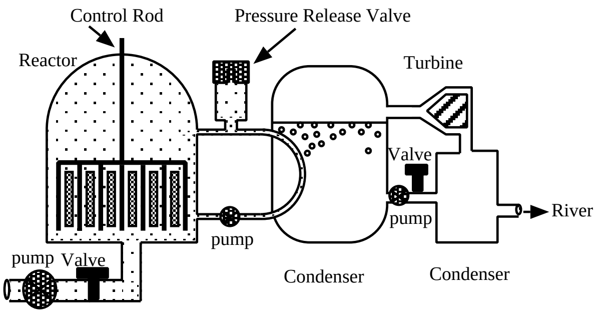

Fig. 2. Three Mile Island Nuclear Power Station Model

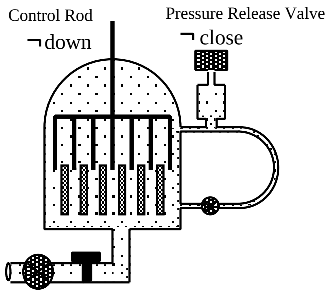

Fig. 3. A Nuclear Reactor Model

creating a model.

$$Pr(\textbf{Initially } close, down) = 0.6$$

$$Pr(\textbf{Initially } close, \neg down) = 0.4$$

$$Pr(rod\_up \textbf{ causes } \neg down | close, down) = 0.3 \tag{8}$$

$$Pr(rod\_down \textbf{ causes } down | close, down) = 0.1 \tag{9}$$

$$Pr(sleep \textbf{ causes } down | close, down) = 0.6 \tag{10}$$

$$Pr(rod\_up \textbf{ causes } \neg down | close, \neg down) = 0.1$$

$$Pr(rod\_down\ \textbf{causes}\ down|close, \neg down) = 0.8$$

$$Pr(sleep\ \textbf{causes}\ \neg close|close, \neg down) = 0.1$$

$$Pr(rod\_up\ \textbf{causes}\ \neg down|\neg close, \neg down) = 0.07$$

$$Pr(rod\_down\ \textbf{causes}\ down|\neg close, \neg down) = 0.9$$

$$Pr(sleep\ \textbf{causes}\ \neg close|\neg close, \neg down) = 0.03$$

$$Pr(rod\_up\ \textbf{causes}\ \neg down|\neg close, down) = 0.05$$

$$Pr(rod\_down\ \textbf{causes}\ \neg close|\neg close, down) = 0.05$$

$$Pr(sleep\ \textbf{causes}\ close|\neg close, down) = 0.9$$

We can convert above domain description into the stochastic state transition diagram (Fig.4) in the following three steps.

Step1: Create states.
Let $q_1 = \{close, down\}, q_2 = \{close, \neg down\}, q_3 = \{\neg close, \neg down\}, q_4 = \{\neg close, down\}$.

Step2: Generate a stochastic state transition function.
For example, $SEP_{q_1,q_1} = \{$
$Pr(rod\_down\ \textbf{causes}\ down|close, down) = 0.1,$
$Pr(sleep\ \textbf{causes}\ down|close, down) = 0.6\}$. $\psi_{q_1 q_1} = 0.7$. $\varphi_{q_1 q_1}(rod\_down) = 0.1/0.7 = 0.142$. $\varphi_{q_1 q_1}(sleep) = 0.6/0.7 = 0.858$.

Step3: Assign the initial state probability.
$\pi_{q_1} = 0.6$ and $\pi_{q_4} = 0.4$.

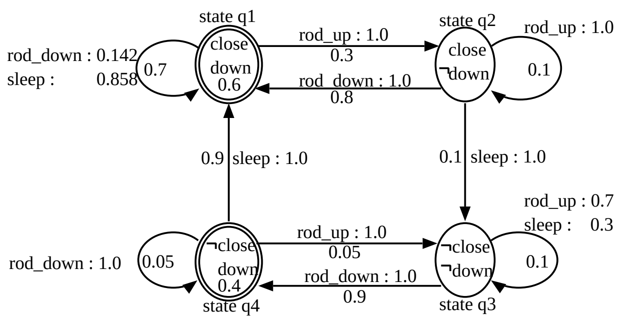

Fig. 4. A deterministic Stochastic Transition Diagram

Next we consider the *non-deterministic* version of the reactor domain description. Stochastic action failures [1] can be represented in A-SAL. (8), (9) and (10) are modified as follows in order to represent stochastic action failures.

$$Pr(rod\_up \textbf{ causes } \neg down \mid close, down) = 0.29$$
$$Pr(rod\_up \textbf{ causes } down \mid close, down) = 0.01$$
$$Pr(rod\_down \textbf{ causes } down \mid close, down) = 0.09$$
$$Pr(rod\_down \textbf{ causes } \neg down \mid close, down) = 0.01$$
$$Pr(sleep \textbf{ causes } down \mid close, down) = 0.59$$
$$Pr(sleep \textbf{ causes } \neg close \mid close, down) = 0.01$$

Figure 5 shows the transformed HMM of the non-deterministic domain description.

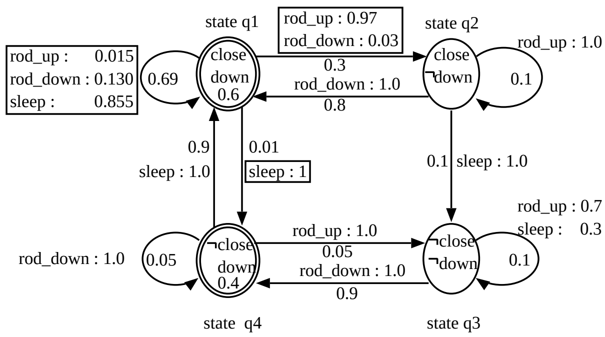

Fig. 5. A non-deterministic Stochastic Transition Diagram

3 Datalog Stochastic Action Language

In this section, we introduce a first-order stochastic action language, Datalog-SAL, which allows us to describe facts at Datalog level. After defining a syntax and semantics, we show that a model of Datalog-SAL is defined in a set of Stochastic Finite Automata or Hidden Markov Models.

3.1 Introduction

Stochastic Finite Automata and HMMs provide compact representations, however, they are usually managed at propositional level. Since our target domains such as bioinformatics require relational representations, there exists a requirement for developing a first-order action language under uncertainty. Our new first-order stochastic action language combines Datalog and probability in the action and change framework. Intuitively, a state is represented as a possible world and stochastic actions map one state to another. The first-order representation and the law of inertia provides a compact representation in Datalog-SAL.

3.2 Syntax of Datalog-SAL

In Datalog a term is defined as a variable or a constant. Datalog does not use function symbols.

Definition 1. *Datalog Atom*
A Datalog atom is of the form $p(t_1, ..., t_n)$ *where* p *is a predicate symbol and each* t_i *is a term. We denote* $p(t_1, ..., t_n)$ *as* p *for shorthand. Datalog atoms belong to either fluent atoms* fl *or action atoms* act.

Definition 2. *Datalog Literal*
A Datalog literal dl *is a fluent atom or a fluent atom proceeded by* $\neg$, *that is,* $dl = fl$ *or* $\neg fl$.

Definition 3. *Formula*
A formula is a conjunction of Datalog literals.

Definition 4. *Quantifier*
Quantifiers are $\forall$ *and* $\exists$.

Definition 5. *Connectives*
Connectives are $\neg$, $\wedge$, ***caused***, ***if***, $|$, *and* $=$.

We also use "," for $\wedge$.

Definition 6. *Stochastic Dynamic Law*
Stochastic dynamic law is defined in

$$Pr(\textbf{\textit{caused}}\ \ F\ \ \textbf{\textit{if}}\ \ G\ \ |\ \ Act,\ B) = c$$

where F *is a Datalog literal,* G *and* B *is a formula, and* Act *is a action atom. Quantifiers are placed between* $Pr($ *and* ***caused***. *A scope of quantifiers is* F, G, Act, *and* B. c *is a constant number in the range of* $[0, 1]$.

Example: *Stochastic Dynamic Law*

$$\begin{aligned} Pr(\forall_X \exists_Y\ \ & \textbf{caused}\ enzyme(e1) \\ & \textbf{if}\ adjacent(e0, X),\ adjacent(X, Y) \\ & |\ action(stir),\ enzyme(e0),\ substrate(Y, e1),\ coenzyme(X, e1)) \\ & = 0.7 \end{aligned}$$

Definition 7. *Stochastic Initial Law*
Stochastic initial law is defined in

$$Pr(\textbf{\textit{initially}}\ \ F) = c$$

where F *is a ground formula.*

In addition to the above three stochastic laws, we also use non-stochastic value propositions for describing observed facts. Knowledge Base of a target domain is represented in value propositions in Datalog-SAL.

3.3 Semantics of Datalog-SAL

A domain description, D , is a finite set of stochastic dynamic laws, stochastic initial laws, and value propositions. A constant belongs to either a *fluent constant* or a *action constant*. An *action* is an interpretation of a ground action atom. A *fluent* is an interpretation of a ground fluent atom.

Stochastic states are introduced in order to handle uncertainties in fluent literals. In uncertain worlds, some fluent values might be unknown since they cannot observe or unimportant from the cost-effectiveness point of view. *Stochastic states* are defined as follows. Let HU be a maximum set of *fluent constants* in D. Let C be a a maximum set of fluent atoms in D. Let $HB = \{b_1, ..., b_n\}$ be a Herbrand Base formed by C with ground terms from HU as arguments. We define $b_i \in HB$ as a stochastic variable and assign following Bernoulli Distributions P over $b_i \in HB$ for defining their truth values:

$$\begin{aligned} P(b_i) &= p_i & (b_i \ is \ true) \\ & \quad 1 - p_i & (b_i \ is \ false) \end{aligned}$$

where p_i is in the range of [0,1].
Then a *state* q_x is defined as a stochastic finite Herbrand Interpretation in the form of:

$$q_x = \{s_1, ..., s_n\}$$

where $s_i = b_i$ if b_i is assigned *true*, or $s_i = \neg b_i$ if b_i is *false*. Another representation of the above stochastic states is to describe all the possible cases with a set of P. The latter one is not compact, however, it fits for defining semantics of Datalog-SAL since the law of inertia is suite for the definite representation of the ground fluent literals. Let Q be a maximum set of states which correspond to all the possible stochastic Herbrand Interpretations. We view Q as possible worlds and assign a probability distribution μ over Q as:

$$\mu \ (q_x) = P(b_1) \times ... \times P(b_n)$$

where $\sum_x \mu \ (q_x) = 1$. The μ is used for determining initial states later.

Example: *States*
Assume $H = \{a, b\}$ and $C = \{f_1(X)\}$. Then HB is defined as $HB = \{ \ f_1(a), \ f_1(b) \ \}$. We assign $P(f_1(a)) = 0.8$, $P(\neg f_1(a)) = 0.2$, $P(f_1(b)) = 0.5$ and $P(\neg f_1(b)) = 0.5$. A set of states, Q, is defined as follows.

$$\begin{aligned} Q &= \{q_1, q_2, q_3, q_4\} \\ &= \{\{f_1(a), f_1(b)\}, \{f_1(a), \neg f_1(b)\}, \{\neg f_1(a), f_1(b)\}, \{\neg f_1(a), \neg f_1(b)\}\} \end{aligned}$$

$\mu(q_1) = 0.4$, $\mu(q_2) = 0.4$, $\mu(q_3) = 0.1$, and $\mu(q_4) = 0.1$.

Next we define truth values of literals in order to determine which stochastic laws are true at each state.

Definition 8. *Truth Value of Literals*
At state q_x, a ground Datalog literal dl is true if and only if $dl \in q_x$. Otherwise false. We denote $q_x \models_\theta DL$ when a Datalog literal DL is true by a ground substitution θ at q_x.

A truth table of the connective "|" and "**if**" shall be defined since the scope of quantifiers straddles them.

Definition 9. *$A \mid B$ is false if A is false and B is true. Otherwise $A \mid B$ is true. A **if** B is false if A is false and B is true. Otherwise A **if** B is true.*

Definition 10. *Stochastic Dynamic Law*
Stochastic dynamic law

$$Pr(\textbf{\textit{caused}}\ \ F\ \ \textbf{\textit{if}}\ \ G\ \ |\ \ Act,\ B) = c$$

is true when B, Act, G, and F is true. If there exists a substitution θ such as $q \models_\theta B$ and $q' \models_\theta ((q \backslash \neg F \backslash \neg G) \cup F \cup G)$, the stochastic dynamic law represents stochastic state transitions from state q to q' by action Act.

In the above definition, the law of inertia is applied in: $((q \backslash \neg F \backslash \neg G) \cup F \cup G)$ where truth values of fluents atoms held at q keep the values unless a change is explicitely declared in stochastic dynamic laws.

Example: *Stochastic Dynamic Law*
Let $q = \{f_1(a),\ f_2(a,b)\}$ and $q' = \{\neg f_1(a),\ f_2(a,b)\}$. The following stochastic dynamic law represents a stochastic state exchange from q to q' by an action$act(jump)$.

$$\begin{aligned} &Pr(\forall X \forall Y(\textbf{caused}\ \ \neg f_1(X)\ \ \textbf{if}\ \ f_2(X,Y) \\ &\quad |\ \ act(jump),\ f_1(X),\ f_2(X,Y))) = c \end{aligned}$$

The semantics of c in a set of stochastic dynamic laws (SDL) is defined as follows.

1. For each state q_i, generate a maximum subset SDL_{q_i} of SDL where there exist θ_1 that satisfy $q_i \models_{\theta_1} B$ for every B of SDL_{q_i}.
2. For each state q_j, create a maximum subset $SDL_{q_i q_j}$ of SDL_{q_i} where the θ_1 satisfy $q_j \models_{\theta_1} ((q_i \backslash \neg F \backslash \neg G) \cup F \cup G)$ for every element of $SDL_{q_i q_j}$. Create a maximum subset $SDL_{q_i q_j Act \theta_1}$ of $SDL_{q_i q_j}$.
3. Let $c_{q_i q_j}$ be the summation of c in $SDL_{q_i q_j}$. Let $c_{q_i q_j Act \theta_1}$ be the summation of c in $SDL_{q_i q_j Act \theta_1}$. Finally, a probability distribution over state exchanges $\psi_{q_i q_j}$ is defined as $\psi_{q_i q_j} = c_{q_i q_j} / \sum_j c_{q_i q_j}$. Note that $\sum_j c_{q_i q_j}$ is a normalization factor.

We set a probability distribution over actions taken from q_i to q_j as

$$\varphi_{q_i q_j Act \theta_1} = c_{q_i q_j Act \theta_1}\ /\ c_{q_i q_j}$$

where $c_{q_i q_j}$ is a normalization factor.

A set of stochastic initial laws defines a probability distribution over possible initial states.

$$Pr(\textbf{initially} \ \ F) = c$$

is true at states q where $q \models F$. Note that F is a ground formula. We define a probability distribution π over possible initial states in the following steps.

1. When D is given, define a maximum set of stochastic initial laws as $INI = \{ini_1, ..., ini_k\}$ where ini_j is $Pr(\textbf{initially} \ \ F_j) = c_j$.
2. Define $Q_j = \{q_1, ..., q_l\}$ where $q_i \models F_j \ \ (1 \leq i \leq l)$.
3. Let $sum(Q_j)$ be a summation of $\mu\ (q_i)$ where $q_i \in Q_j$.
4. Then $c_j(q_i) = c_j \times \mu(q_i)/sum(Q_j)$.
5. A probability distribution over initial states q_i is defined as follows.

$$\pi_{q_i} = \sum_j c_j(q_i)$$

Example: *Probability Distribution* π
Assume $HB = \{f_1(a),\ f_1(b)\}$ and $P(f_1(a)) = 0.5, P(\neg f_1(a)) = 0.5, P(f_1(b)) = 0.2, andP(\neg f_1(b)) = 0.8$. Let

$$\begin{aligned} Q &= \{q_1,\ q_2,\ q_3,\ q_4\} \\ &= \{\{f_1(a), f_1(b)\}, \{f_1(a), \neg f_1(b)\}, \\ &\quad \{\neg f_1(a), f_1(b)\}, \{\neg f_1(a), \neg f_1(b)\}\}. \end{aligned}$$

Then assume $\mu(q_1) = 0.1, \mu(q_2) = 0.4, \mu(q_3) = 0.1$ and $\mu(q_4) = 0.4$. Now the following two stochastic initial laws:

$$Pr(\textbf{initially} \ \ f_1(a)) = 0.4 \tag{11}$$
$$Pr(\textbf{initially} \ \ f_1(a),\ \neg f_1(b)) = 0.6 \tag{12}$$

define a probability distribution π over q_1 and q_2 as follows.

1. Let $INI = \{ini_1, ini_2\} = \{(11), (12)\}$.
2. Then Q_1 and Q_2 is specified as $Q_1 = \{q_1,\ q_2\}$ and $Q_2 = \{q_2\}$.
3. $sum(Q_1) = \mu(q_1) + \mu(q_2) = 0.5, sum(Q_2) = \mu(q_2) = 0.4$.
4. Then we get

$$\begin{aligned} c_1(q_1) &= 0.4 \times 0.1/0.5 = 0.08 \\ c_1(q_2) &= 0.4 \times 0.4/0.5 = 0.32 \\ c_2(q_2) &= 0.6 \times 0.4/0.4 = 0.6 \end{aligned}$$

5. Finally we get $\pi_{q_1} = 0.08$ and $\pi_{q_2} = 0.92$.

3.4 Model

An interpretation of Datalog-SAL is given in the form of $M = (S, A, \psi, \varphi, \pi)$ where

- $S = \{q_1, .., q_m\}$ is a non-empty finite set of states which are reachable from at least one initial state by taking sequences of actions.
- $A = \{act(a_1), .., act(a_n)\}$ is a non-empty finite set of ground action atoms.
- $\psi = \{\psi_{q_i q_j}\}$ is a probability distribution over the state transition from state q_i to q_j where $\sum_j \psi_{q_i q_j} = 1$.
- $\varphi = \{\varphi_{q_i q_j}(act(a_k))\}$ is a probability distribution over action $act(a_k)$ taken from state q_i to q_j where $\sum_k \varphi_{q_i q_j}(act(a_k)) = 1$.
- $\pi = \{\pi_{q_i}\}$ is a probability distribution over the initial state q_i where $\sum_i \pi_{q_i} = 1$.

In Datalog-SAL, interpretations of states and procedures of calculating probability distributions are different from the definitions of A-SAL, however, both of their interpretations are defined as state transition diagrams at the ground level. We show M(D,BD) is defined in a set of HMMs where D is a domain description and BD is a set of the Bernoulli Distributions. When D and BD is given, the following procedure translates from Datalog-SAL to HMM.

1. Calculate π with D and BD.
2. Calculate an initial state.
3. Create a stochastic state transition diagram based on ψ and φ. The resulted diagram only contains reachable states from the initial state.
4. Iterate the above 2 and 3 for every initial state.
5. Overlap each diagram one by one.

Lemma 1 shows that the resulted diagram is HMM. Since BD defines various initial states, $M(D, BD)$ is equivalent with a set of HMMs. $M(D, BD)$ is called model when every value proposition can be represented by $M(D, BD)$.

3.5 Random Sampling ω

In [5], ω is introduced for representing random samplings in a domain. In Datalog-SAL, we use ω for representing statistical knowledge of states. The following stochastic dynamic law shows an example in a chemical reaction domain.

$$\begin{aligned} Pr(\forall_X \exists_Y \ & \textbf{caused } enzyme(e1) \\ & \textbf{if } (\omega_X \ (adjacent(e0, X)) \geq 0.9),\ (\omega_X \ (adjacent(X, Y)) \geq 0.6) \\ & \mid action(stir),\ enzyme(e0),\ substrate(Y, e1),\ coenzyme(X, e1)) \\ & = 0.7 \end{aligned}$$

In the above case, ω_X represents a random sampling of X from a maximum subset of HU where $coenzyme(X, e1)$ is true. More precisely, ω_X represents the following statistical knowledge in this example.

$$\frac{|adjacent(e0, X)|}{|coenzyme(X, e1)|} \geq 0.9$$

$$\frac{|adjacent(X, Y)|}{|coenzyme(X, e1)|} \geq 0.6$$

The knowledge representations with the ω quantifier have an advantage in the description complexity. Assume the following situation "*All members in a party must talk with somebody*". We can represent this situation in $\forall_X \exists_Y \;\; speak(X, Y)$. Let N be the number of members who attend the party. Then, the description complexity of the above representation is $O(0)$ because the above representation still valid even if N increases. The propositional (ground) level description complexity is given in $O(N^N)$.

Next assume the situation " *The probability that a randomly chosen member in a party must talk with somebody is 0.99*". We can represent this in

$$\omega_X \exists_Y \;\; speak(X, Y) = 0.99.$$

Then the description complexity of the above stochastic representation is $O(0)$ whereas the ground level description complexity is $O((n/2)!)$.

3.6 Learning Complexity of Datalog-SAL Parameter Learning

Since Datalog-SAL is equivalent with HMMs, learning stochastic state transitions is realized by well-known HMM learning algorithm such as Baum-Welch algorithm.

We have proved the learning complexity of Datalog-SAL with Baum-Welch algorithm as follows. According to [19], the computational complexity of Baum-Welch algorithm is $O(N^2L)$ where N is the number of states and L is the length of outputs. In a stochastic action description written in Datalog SAL, the upper bound of the number of states N_d is

$$N_d \leq 2^P C^K$$

where P is the number of $atom^{fl}$, C is the maximum number of constants which is substituted to a parameter in $atom^{fl}$, and K is the maximum number of arity in $atom^{fl}$. Note that 2^N is from the number of Herbrand Interpretation, and C^K is from the maximum number of ground substitutions for the $atom^{fl}$ whose arity is K. Therefore the complexity of learning Datalog SAL by Baum-Welch algorithm is

$$O(4^P C^{2K} L).$$

4 Related Works

In this section, we review previous works in stochastic situation calculus. A definition of a language of the situation calculus can be found in [22] as: (1) A sort of *situation*, (2) A sort of *fluents*, (3) A sort of *actions*, (4) A function Result from actions and situations to situations, (5) A predicate Holds whose arguments are a fluent and a situation.

On the stochastic situation and fluents, stochastic initial situations are proposed in stGolog [20]. stGolog divides predicates as *stochastic predicates* and non-stochastic predicates and then assigns a probability distribution over the ground atoms of the stochastic predicate. Each different stochastic predicate is treated as stochastically independent each other. For example [20], assume that a mail delivery robot is ordered to collect a mail. The mail is currently put either Pat's office or Sue's office. Assume stGolog makes a set of mailPresent ground atoms as $\{\mathsf{mailPresent}(sue, s0), \mathsf{mailPresent}(pat, s0)\}$ where $s0$ represents an initial situation. The initial location of the robot also contains uncertainty and a set of ground atoms

$$\{\mathsf{location}(mailroom, s0), \mathsf{location}(sue\_office, s0)\}$$

represents the possible initial locations of the robot. Then stGolog assigns probability distributions over these sets such as 0.1 for $\mathsf{mailPresent}(\mathrm{sue, s0})$ and 0.9 for $\mathsf{mailPresent}(\mathrm{pat, s0})$, and 0.4 for $\mathsf{location}(\mathrm{mailroom, s0})$ and 0.6 for $\mathsf{location}(sue\_office, s0)$. For instance, the probability of being in the initial situation is calculated by 0.9×0.4 where the mail is at Pat's office and robot is in the mailroom.

In action languages, certain details might be left out, either because they are not known or because they are thought to be unimportant[16]. The omitted fluents could be filled by the inertia law, however, there exist uncertainties if the inertia law is not applicable for the fluents. PAL[3] assigns a probability distribution over non-inertial unknown variables.

On the stochastic action, Bacchus et. al [1] partitions an agent's action into two basic parts: an intended action and outcome actions[1]. The intended action is treated as a controllable action by an agent, but outcome actions are beyond the control for the agent. That is, the intended action can be executed, however, the result of the intended action can only be decided stochastically. [1][17][20] assign a probability distribution over actions.

On the stochastic results, Grosskreutz et. al proposes a stochastic version of Golog, called pGolog[8]. pGolog defines stochastic state transition of the form

$$transPr(prob(p, \sigma_1, \sigma_2), s, \delta, s')$$

where a state transition from a state s to s' is executed by a program σ_1 in the probability p or by a program σ_2 in $1 - p$. δ is the remaining program after executing the program. Since the choice of programs is the Bernoulli process, we can execute only one of two programs. This could become a limitation when we have multiple choices of programs.

The stochastic Holds is closely related with the Bernoulli distributions P. Little works have investigated on the stochastic Holds, however, a stochastic static relation between Datalog literals could be represented in stochastic static laws as follows.

$$Pr(\textbf{caused} \;\; F \;\; \textbf{if} \;\; G) = c$$

Intuitively we read this statement as "Probability of $F\theta$ caused true at state q is c if there exist θ such as $q \models_\theta G$. Note that the stochastic static laws define the Bernoulli distributions P.

5 Discussions and Conclusions

We have introduced two stochastic action languages for describing dynamic changing worlds. A-SAL provides a simple declarative stochastic language for describing actions and changes at propositional level. Since the model of A-SAL is equivalent with stochastic finite automata or HMM, A-SAL provides a basic framework for learning action languages. We expect that existing applications written in Action Language $\mathcal{A}$ can be modified into A-SAL by simply attaching probability values to effect propositions and initial value propositions.

Datalog-SAL is combining logic and probability in the action and change framework at Datalog level. One of the advantage of Datalog-SAL is to provide compact representations which increase importance when we apply action languages to real world problems. As a future study, apart from the logical issues, probabilistic aspects of situation calculus would be an interesting topic. Frame Problem is one of the most important issues in situation calculus. When we consider stochastic situation calculus, it is worth to consider frame axioms in probability. Assume a state of Datalog-SAL as a universe U of the probability domain which contains ground literals. Assume A, B, and C as events in U. One way to represent frame axiom in probability is as follows.

Unconditional Stochastic Frame Axiom

$$Pr(A \wedge B) = Pr(A)Pr(B)$$

Conditional Stochastic Frame Axiom

$$Pr(A, B \mid C) = Pr(A|C)Pr(B|C)$$

Further studies are required when we apply Datalog-SAL to multi-agents settings which could include concurrent executions of actions by multiple agents. Fluent values might be updated by multiple agents who take actions independently each other. Concurrent actions by single agent has studied by [2], however, little works have been reported on concurrent actions by multiple agents in action languages.

Our two languages provide a simple approach for converging logic and probability in the action and change framework. Probabilistic extensions of action languages might bring new research topics including probabilistic approaches for Frame Problem, Qualification Problem, and Ramification Problem.

Appendix

Lemma 1 The overlapping operations do not change ψ and φ.

Proof. We consider an overlapping operation which merges two diagrams (Net_A and Net_B) into one (Net_C). Net_A and Net_B has one initial state in their diagrams. Note that the overlapping operation merges same two states into one state. The operation causes following four cases.

Case 1: Imagine Net_A contains q_1 and q_2. Net_B also contains q_1 and q_2. Consider a state transition from q_1 to q_2. We denote the arrow from q_1 to q_2 as $arrow(q_1, q_2)$. Assume Net_A has $arrow(q_1, q_2)$, however, Net_B does not have $arrow(q_1, q_2)$. Net_A and Net_B is an interpretation of the same D, however, if

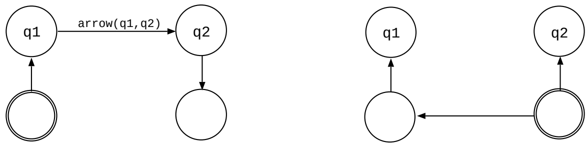

Fig. 6. Case1: Net_A and Net_B

Net_B does not have $arrow(q_1, q_2)$ it means q_2 is removed by the rule for S. That is, q_2 is not reachable from the initial state in Net_B. This is contradictory for the fact that q_2 is in Net_B. Therefore the case 1 should not happen.

Case 2: Imagine Net_A contains q_1 and q_2 and has $arrow(q_1, q_2)$. Net_B contains q_1, however, does not contain q_2. Thus Net_B does not have $arrow(q_1, q_2)$. Net_A and Net_B is an interpretation of the same D, however, if there is not $arrow(q_1, q_2)$ in Net_B, it is because q_2 is removed by the rule for S. This means q_1 is also not reachable from the initial state in Net_B otherwise we can get to q_2 by $arrow(q_1, q_2)$ in Net_B if q_1 is reachable. This is contradictory for the fact that B contains q_1. Therefore the case 2 should not happen.

Case 3: Imagine Net_A contains q_1 and q_2. Net_B does not contain both q_1

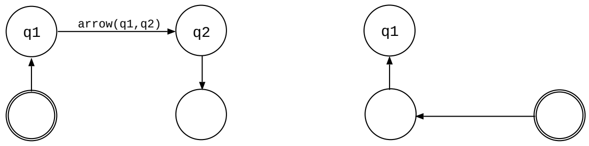

Fig. 7. Case2: Net_A and Net_B

and q_2. There exists $arrow(q_1, q_2)$ in Net_A, however, the arrow does not change $\psi_{q_1 q_2}$ and $\varphi_{q_1 q_2}$ during the merging because q_1 is not included in Net_B.

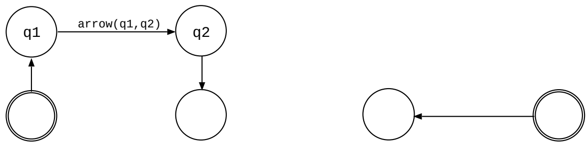

Fig. 8. Case3: Net_A and Net_B

Case 4: When $arrow(q_1, q_2)$ is overlapped between Net_A and Net_B, Net_A and Net_B has same $\psi_{q_1 q_2}$ and $\varphi_{q_1 q_2}$. Net_C can represent Net_A and Net_B if the resulted Net_C takes over the same $\psi_{q_1 q_2}$ and $\varphi_{q_1 q_2}$. The above cases show that

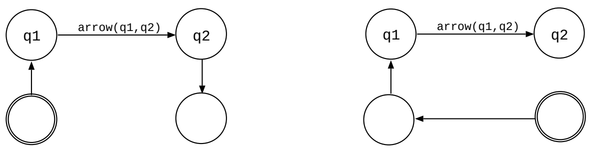

Fig. 9. Case4: Net_A and Net_B

overlapping operations do not change ψ and φ.

References

1. Fahiem Bacchus, Joseph Y. Halpern, Hector J. Levesque.: Reasoning about noisy sensors in the situation calculus. Artificial Intelligence, 111(1-2):171–208, 1999.
2. Chitta Baral, Michael Gelfond.: Reasoning about effects of concurrent actions. Journal of Logic Programming, 31(1-3):85–117, 1997.
3. Chitta Baral, Le-Chi Tuan.: Reasoning about actions in a probabilistic setting. In Symposium on Logical Formalization of Common Sense 2001, 2001.

4. R. Carnap.: The Logical Foundations of Probability. University of Chicago Press, Chicago, 1962.
5. Joseph Y. Halpern.: An analysis of first-order logics of probability, Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence, 1375–1381, 1989.
6. R. Fagin, J. Halpern.: Uncertainty, belief and probability. Computational Intelligence, 7:160–173, 1991.
7. M. Gelfond, V. Lifschitz.: Representing action and change by logic programs. Journal of Logic Programming, 17(2-4):301–321, 1993.
8. Henrik Grosskreutz, Gerhard Lakemeyer.: Turning High-Level Plans into Robot Programs in Uncertain Domains, ECAI2000, 2000.
9. Thomas Lukasiewicz, Probabilistic Logic Programming, European Conference on Artificial Intelligence, 388-392, 1998.
10. J. McCarthy.: Elaboration Tolerance, Workshop on Formalization of Commonsense Reasoning 1998.
11. John McCarthy, Patrick Hayes.: Some philosophical problems from the standpoint of artificial intelligence, Machine Intelligence, vol.4, 463-502, 1969.
12. J. Minker, editor.: Logic-Based Artificial Intelligence. Kluwer, 2000.
13. S.H. Muggleton.: Stochastic logic programs. In L. de Raedt, editor, Advances in Inductive Logic Programming, pages 254–264. IOS Press, 1996.
14. Hidetomo Nabeshima and Katsumi Inoue. Automata theory for action language a. (in Japanese). Transactions of Information Processing Society of Japan, 38(3):462–471, 1997.
15. Nil Nilsson.: Probabilistic logic. Artificial Intelligence Journal, 28, 1986.
16. Edwin Pednault.: Formulating multi-agent, dynamic world problems in the classical planning framework. In Michael Georgeff and Amy Lansky, editors, Reasoning about Actions and Plans, pages 47–82. Morgan Kaufmann, San Mateo, CA, 1987.
17. J. Pinto, A. Sernadas, P. Mateus, A. Pacheco, C. Sernadas.: Probabilistic situation calculus. Annals of Mathematics and Artificial Intelligence, In print, 2001.
18. David Poole.: Decision theory, the situation calculus, and conditional plans, Linkoping Electronic Articles in Computer and Information Science, 3(8), 1998.
19. L. Rabiner.: A tutorial on hidden markov models and selected applications in speech recognition. In Proceedings of the IEEE, 77, 1989.
20. Raymond Reiter.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, Cambridge, Massachusetts, 2001.
21. Taisuke Sato.: A statistical learning method for logic programs with distribution semantics. In Proceedings of ICLP 95, 1995.
22. Murray Shanahan.: Solving the Frame Problem. The MIT Press, Cambridge, Massachusetts, 1997.