Online appendix for the paper

A Probabilistic Extension of Action Language $\mathcal{BC}$+

published in Theory and Practice of Logic Programming

Joohyung Lee and Yi Wang
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
(*e-mail:* {`joolee, ywang485`}`@asu.edu`)

Appendix A Proofs

A.1 Proof of Proposition 1

Proposition 1 *For any transition* $\langle s, e, s' \rangle$, s *and* s' *are states.*

Proof. To show that s and s' are states, we show that $0 : s$ and $0 : s'$ are stable models of D_0. We use I^0 as an abbreviation of $0 : s \cup 0 : e \cup 1 : s' \cup 0 : pf$. By definition of a transition, we know that $0 : s \cup 0 : e \cup 1 : s'$ is a residual stable model of D_1, i.e., there exists an assignment pf to σ^{pf} such that $0 : s \cup 0 : e \cup 1 : s' \cup 0 : pf$ is a stable model of D_1. By definition of a probabilistic stable model, this means $0 : s \cup 0 : e \cup 1 : s' \cup 0 : pf$ is a (deterministic) stable model of

$$SD(0) \cup FD(0) \cup \overline{PF(0)_{I^0}} \cup UEC \cup EXG. \tag{A1}$$

To prove $0 : s$ **is a stable model of** D_0, we split (A1) into two disjoint subsets

$$SD(0)$$

and

$$SD(1) \cup FD(0) \cup \overline{PF(0)_{I^0}} \cup UEC \cup EXG.$$

It can be seen that $SD(0)$ is negative on $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$ and $SD(1) \cup FD(0) \cup \overline{PF(0)_{I^0}} \cup UEC \cup EXG$ is negative on $0 : \sigma^{fl}$. Every strongly connected components of (A1) is either a subset of $0 : \sigma^{fl}$ or a subset of $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$. By the splitting theorem, we have that $0 : s$ is stable model of $SD(0)$ w.r.t. $0 : \sigma^{fl}$ and $0 : e \cup 1 : s' \cup 0 : pf$ is a stable model of $SD(1) \cup FD(0) \cup \overline{PF(0)_{I^0}} \cup UEC \cup EXG$ w.r.t. $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$. Since $D_0 = SD(0)$, s' is a stable model of D_0.

To prove $0 : s'$ **is a stable model of** D_0, we further divide the set of fluents into the set σ^r of regular fluents and the set σ^{sd} of statically determined fluents. From the above reasoning, we know that $0 : e \cup 1 : s' \cup 0 : pf$ is a stable model of

$$SD(1) \cup FD(0) \cup \overline{PF(0)_{I^0}} \cup UEC \cup EXG$$

w.r.t. $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$, i.e. $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{r} \cup 1 : \sigma^{sd}$, . By Theorem 2 in (Ferraris et al. 2009), we have that $0 : e \cup 1 : s' \cup 0 : pf$ is a stable model of

$$SD(1) \cup FD(0) \cup \overline{PF(0)_{I^0}} \cup UEC \cup EXG$$

w.r.t. $1 : \sigma^{sd}$. Since $FD(0)$, $PF(0)$, UEC and EXG are negative on $1 : \sigma^{sd}$, this implies $0 : e \cup 1 : s' \cup 0 : pf$ is a stable model of $SD(1)$ w.r.t. $1 : \sigma^{sd}$. Since $SD(1)$ does not mention any atom in $0 : e \cup 0 : pf$, we have that $1 : s'$ is a stable model of $SD(1)$ w.r.t. $1 : \sigma^{sd}$. Let $(1 : \sigma^r)^{\text{ch}}$ denote the set of rules of the form (10) for each $c \in \sigma^r$. The above implies that $1 : s'$ is a stable model of $SD(1) \cup (1 : \sigma^r)^{\text{ch}}$ w.r.t. $1 : \sigma^{sd} \cup 1 : \sigma^r = 1 : \sigma^{fl}$. Changing all the timesteps from 1 to 0, we obtain that $0 : s'$ is a stable model of $SD(0) \cup (0 : \sigma^r)^{\text{ch}} = D_0$ w.r.t. $0 : \sigma^{fl}$. $\square$

A.2 Proof of Theorem 1

Proposition 2

For any multi-valued probabilistic program Π for which every total choice leads to $n (n > 0)$ stable models and any interpretation I, we have

$$P''_{\Pi}(I) = \frac{1}{n} W''_{\Pi}(I).$$

Proof. We show that the normalization factor is constant n, i.e.,

$$\sum_{I \text{ is an interpretation of } \Pi} W''_{\Pi}(I) = n.$$

Let $pf_1, \dots, pf_m$ be the probabilistic constants in Π, and $v_{i,1}, \dots, v_{i,k_i}$, each associated with probability $p_{i,1}, \dots, p_{i,k_i}$ resp. be the values of pf_i ($i \in \{1, \dots, m\}$). Let TC_{Π} be the set of all assignments to probabilistic constants in Π.

$$\begin{aligned}
&\sum_{I \text{ is an interpretation of } \Pi} W''_{\Pi}(I) \\
=&\sum_{I \in SM''[\Pi]} W''_{\Pi}(I) \\
=&\sum_{tc \in TC_{\Pi}} n \cdot \prod_{c=v \in tc} M_{\Pi}(c=v) \\
=&n \sum_{tc \in TC_{\Pi}} \prod_{c=v \in tc} M_{\Pi}(c=v) \\
=&n \cdot (p_{1,1} \sum_{\substack{tc \in TC_{\Pi} \\ tc(pf_1)=v_{1,1}}} \prod_{\substack{c=v \in tc \\ c \neq pf_1}} M_{\Pi}(c=v) + \dots + p_{1,k_1} \sum_{\substack{tc \in TC_{\Pi} \\ tc(pf_1)=v_{1,k_1}}} \prod_{\substack{c=v \in tc \\ c \neq pf_1}} M_{\Pi}(c=v)) \\
=&n \cdot (\sum_{i_1 \in \{1,\dots,k_1\}} p_{1,i_1} \sum_{\substack{tc \in TC_{\Pi} \\ tc(pf_1)=v_{1,i_1}}} \prod_{\substack{c=v \in tc \\ c \neq pf_1}} M_{\Pi}(c=v)) \\
=&n \cdot (\sum_{i_2 \in \{1,\dots,k_2\}} p_{2,i_2} \sum_{i_1 \in \{1,\dots,k_1\}} p_{1,i_1} \sum_{\substack{tc \in TC_{\Pi} \\ tc(pf_1)=v_{1,i_1} \\ tcpf_2=v_{2,i_2}}} \prod_{\substack{c=v \in tc \\ c \neq pf_1 \\ c \neq pf_2}} M_{\Pi}(c=v))
\end{aligned}$$

(A2)

$$
\begin{aligned}
=& n \cdot \Big(\sum_{i_{m-1} \in \{1,\dots,k_{m-1}\}} p_{m,i_m} \cdots \sum_{i_2 \in \{1,\dots,k_2\}} p_{2,i_2} \sum_{i_1 \in \{1,\dots,k_1\}} p_{1,i_1} \\
& \qquad \sum_{\substack{tc \in TC_\Pi \\ tc(pf_1)=v_{1,i_1} \\ tc(pf_2)=v_{2,i_2} \\ \dots \\ tc(pf_{m-1})=v_{m-1,i_m}}} \prod_{\substack{c=v \in tc \\ c \neq pf_1 \\ c \neq pf_2 \\ \dots \\ c \neq pf_{m-1}}} M_\Pi(c=v) \Big) \\
=& n \cdot \Big(\sum_{i_{m-1} \in \{1,\dots,k_{m-1}\}} p_{m,i_m} \cdots \sum_{i_2 \in \{1,\dots,k_2\}} p_{2,i_2} \sum_{i_1 \in \{1,\dots,k_1\}} \\
& \qquad p_{1,i_1} (M_\Pi(pf_m = v_{m,1}) + \dots + M_\Pi(pf_m = v_{m,k_m})) \Big) \\
=& n \cdot \Big(\sum_{i_{m-1} \in \{1,\dots,k_{m-1}\}} p_{m,i_m} \cdots \sum_{i_2 \in \{1,\dots,k_2\}} p_{2,i_2} \sum_{i_1 \in \{1,\dots,k_1\}} p_{1,i_1} 1 \Big) \\
=& n \cdot (1) \\
=& n.
\end{aligned}
$$

□

For $i' \in \{0,\dots,m\}$, we use $SD(i')$ to denote the set of all rules of the form (8) in D_m where $i = i'$, and for $i' \in \{0,\dots,m-1\}$, we use $FD(i')$ to denote the set of all rules of the form (9) in D_m where $i = i'$. Furthermore, according to (Lee and Wang 2016), a pf constant declaration (12) is translated into

$$ln(p_j) \quad : \quad (i : pf) = v_j \tag{A3}$$

for each $j \in \{1,\dots,n\}$ and $i \in \{1,\dots,m-1\}$. For any $i \in \{0,\dots,m-1\}$, and any assignment to σ^{pf}, we use $PF(i)$ to denote the set of weighted rules (A3) in D_m where pf is an pf constant, and $PF(i)_{TC}$ to denote the subset of $PF(i)$ that TC satisfies.

Also according to (Lee and Wang 2016), D_m contains

$$\alpha : \quad \bot \leftarrow c = v_1 \wedge c = v_2 \tag{A4}$$

and

$$\alpha : \quad \bot \leftarrow \neg \bigvee_{v \in Dom(c)} c = v \tag{A5}$$

for all constants c and $v_1, v_2 \in Dom(c)$ such that $v_1 \neq v_2$. We use UEC to denote the set of rules of the form (A4) or (A5), and EXG to denote the set of rules of the form (11) and (10), disregarding their weights.

Lemma 1
Let $(\langle s, a, s' \rangle, pf)$ be a pf-transition of D. We have that $0 : s \cup 0 : a \cup 0 : pf \cup 1 : s'$ is a (deterministic) stable model of $SD(1) \cup FD(0) \cup \overline{PF(0)}_{0:pf} \cup UEC$ w.r.t. $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$.

Proof. By definition of pf-transition, we have that I is a deterministic stable model of

$$SD(0) \cup SD(1) \cup FD(0) \cup \overline{PF(0)_I} \cup UEC. \tag{A6}$$

We split (A6) into $SD(0)$ and the rest

$$SD(1) \cup FD(0) \cup \overline{PF(0)_I} \cup UEC.$$

It can be seen that $SD(0)$ is negative on $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$ and $SD(1) \cup FD(0) \cup$

$\overline{PF(0)_I} \cup UEC$ is negative on $0 : \sigma^{fl}$. Each strongly connected components of (A6) is either a subset of $0 : \sigma^{fl}$ or a subset of $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$.

By the splitting theorem, we have that $0 : s \cup 0 : a \cup 0 : pf \cup 1 : s'$ is a (deterministic) stable model of $SD(1) \cup FD(0) \cup \overline{PF(0)}_{0:pf} \cup UEC$ w.r.t. $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$.

□

For any set of constants $\mathcal{C}$, a of $\mathcal{C}$ is a function that maps each element c in $\mathcal{C}$ to a unique element in $Dom(c)$. We say an interpretation I of the propositional signature constructed from $\mathcal{C}$ (as described in Section 2.2) *satisfies* a value assignment V of $\mathcal{C}$ if for all $c \in \mathcal{C}$, $(c = v)^I = \mathbf{t}$ if and only if $V(c) = v$.

Theorem 1 *Given any value assignment TC of constants in $\sigma_m^{pf} \cup 0 : \sigma^{initpf}$ and a value assignment A of constants of σ_m^{act}, $I_{TC \cup A}$ is the only stable model of $Tr(D, m)$ that satisfies $TC \cup A$, and the probability of $I_{TC \cup A}$ is*

$$Pr_{Tr(D,m)}(I_{TC \cup A}) = \frac{\prod\limits_{c=v \in TC} M(c = v)}{(|\sigma^{act}| + 1)^m}.$$

Proof. We first show that $I_{TC \cup A}$ is the only stable model of $Tr(D, m)$ that satisfies $TC \cup A$. Clearly $I_{TC \cup A} \models TC \cup A$. We use $I^i_{TC \cup A}$ for $i \in \{0, 1, \dots, m - 1\}$ to denote the following subset of $I_{TC \cup A}$:

$$(I_{TC \cup A})|_{i:\sigma^{fl}} \cup (I_{TC \cup A})|_{i:\sigma^{act}} \cup (I_{TC \cup A})|_{i+1:\sigma^{fl}} \cup (I_{TC \cup A})|_{i:\sigma^{pf}}.$$

For any $i, j \in \{0, \dots, m\}$ such that $i < j$ and any signature σ', we use $i..j : \sigma'$ to denote $i : \sigma' \cup \dots \cup j : \sigma'$.

- **We show that $I_{TC \cup A}$, i.e., $(I_{TC \cup A})|_{0:\sigma^{initpf}} \cup I^0_{TC \cup A} \cup \dots \cup I^{m-1}_{TC \cup A}$ is a probabilistic stable model of $Tr(D, m)$ by induction:** Let $I_{TC \cup A}(n)$ denote $(I_{TC \cup A})|_{0:\sigma^{initpf}} \cup I^0_{TC \cup A} \cup \dots \cup I^{n-1}_{TC \cup A}$
Base Case: when $m = 1$, consider $I_{TC \cup A}(1)$, i.e, $(I_{TC \cup A})|_{0:\sigma^{initpf}} \cup I^0_{TC \cup A}$.

$$\overline{Tr(D, 1)_{I_{TC \cup A}(1)}}$$

is the ASP program

$$\begin{aligned} &\overline{(D_{init})_{I_{TC \cup A}(1)}} \cup \\ &SD(0) \cup SD(1) \cup \\ &FD(0) \cup \\ &\overline{PF(0)_{TC}} \cup UEC. \end{aligned}$$

Since

$$(\langle (I_{TC \cup A})|_{0:\sigma^{fl}}, (I_{TC \cup A})|_{0:\sigma^{act}}, (I_{TC \cup A})|_{1:\sigma^{fl}} \rangle, (I_{TC \cup A})|_{0:\sigma^{pf}})$$

is a pf-transition, by Lemma 1, $I_{TC \cup A}(1)$ is a deterministic stable model of $SD(1) \cup FD(0) \cup \overline{PF(0)_{TC}} \cup UEC$ w.r.t. $0 : \sigma^{act} \cup 0 : \sigma^{pf} \cup 1 : \sigma^{fl}$.
On the other hand, from the construction of $I_{TC \cup A}$, $I_{TC \cup A}(1)$ is a deterministic stable model of $\overline{(D_{init})_{(I_{TC \cup A})}} \cup SD(0)$ w.r.t. $0 : \sigma^{fl} \cup 0 : \sigma^{initpf}$.
It can be seen that $SD(1) \cup FD(0) \cup \overline{PF(0)_{TC}} \cup UEC$ is negative on $0 : \sigma^{fl} \cup 0 :$

σ^{initpf} and $\overline{(D_{init})_{(I_{TC\cup A})}} \cup SD(0)$ is negative on $0:\sigma^{act} \cup 0:\sigma^{pf} \cup 1:\sigma^{fl}$. Each strongly component of the dependency graph of $(Tr(D,m))_{I_{TC\cup A}(1)}$ is either a subset of $0:\sigma^{fl} \cup 0:\sigma^{initpf}$ or a subset of $0:\sigma^{act} \cup 0:\sigma^{pf} \cup 1:\sigma^{fl}$.
Applying the splitting theorem, we have that $I_{TC\cup A}(1)$ is a deterministic stable model of $\overline{(Tr(D,m))_{I_{TC\cup A}(1)}}$ and thus is a probabilistic stable model of $Tr(D,m)$, since it does not violate any hard rules.

For $m > 1$, consider $I_{TC\cup A}(m)$.

$$\begin{aligned}\overline{(Tr(D,m))_{I_{TC\cup A}(m)}} = \ & \overline{(D_{init})_{I_{TC\cup A}(m)}} \cup \\ & SD(0) \cup \cdots \cup SD(m) \cup \\ & FD(0) \cup \cdots \cup FD(m-1) \cup \\ & \overline{PF(0)_{TC}} \cup \cdots \cup \overline{PF(m-1)_{TC}} \cup UEC\end{aligned} \tag{A7}$$

Clearly we have

— each strongly connected component of the dependency graph of (A7) is either a subset of $0:\sigma^{initpf} \cup 0..(m-1):\sigma^{fl} \cup 0..(m-2):\sigma^{act} \cup 0..(m-2):\sigma^{pf}$ or a subset of $m:\sigma^{fl} \cup m-1:\sigma^{act} \cup m-1:\sigma^{pf}$;

—
$$\begin{aligned}\overline{(Tr(D,m-1))_{I_{TC\cup A}(m)}} = \ & \overline{(D_{init})_{I_{TC\cup A}(m)}} \cup \\ & SD(0) \cup \cdots \cup SD(m-1) \cup \\ & FD(0) \cup \cdots \cup FD(m-2) \cup \\ & \overline{PF(0)_{TC}} \cup \cdots \cup \overline{PF(m-2)_{TC}} \cup UEC \cup EXG\end{aligned}$$

is negative on $m:\sigma^{fl} \cup m-1:\sigma^{act} \cup m-1:\sigma^{pf}$;

—
$$\begin{aligned}& \overline{(Tr(D,m)\backslash Tr(D,m-1))_{I_{TC\cup A}(m)}} = \\ & SD(m) \cup \\ & FD(m-1) \cup \\ & \overline{PF(m-1)_{TC}}\end{aligned}$$

is negative on $0:\sigma^{initpf} \cup 0..m-1:\sigma^{fl} \cup 0..m-2:\sigma^{act} \cup 0..m-2:\sigma^{pf}$.

By I.H., $I_{TC\cup A}(m-1)$ is a probabilistic stable model of $Tr(D,m-1)$, which implies $I_{TC\cup A}(m)$ is a (deterministic) stable model of

$$\overline{(Tr(D,m-1))_{I_{TC\cup A}(m)}}$$

w.r.t. $0:\sigma^{initpf} \cup 0..m-1:\sigma^{fl} \cup 0..m-2:\sigma^{act} \cup 0..m-2:\sigma^{pf}$. The fact that $I_{TC\cup A}(m)$ is a (deterministic) stable model of

$$\overline{(Tr(D,m)\backslash Tr(D,m-1))_{I_{TC\cup A}(m)}}$$

w.r.t. $m:\sigma^{fl} \cup m-1:\sigma^{act} \cup m-1:\sigma^{pf}$ can be seen from Lemma 1 and replacing timesteps m and $m-1$ with 1 and 0 resp.
Thus, $I_{TC\cup A}(m)$ is a stable model of $Tr(D,m)$.

- **There does not exist more than one stable models of $Tr(D, m)$ that satisfies $TC \cup A$.** Suppose, to the contrary, that there exists $I \neq I_{TC\cup A}$ that satisfies $TC \cup A$ and I is also a stable model of $Tr(D, m)$. Since I and $I_{TC\cup A}$ agree on $TC \cup A$, they can differ only on the value assignment of constants in σ^{fl}. Let $i : fl$ be any one of the constants such that $I(i : fl) \neq I_{TC\cup A}(i : fl)$ and there does not exist any $j : fl'$ with $j \leqslant i$ and $I(j : fl') \neq I_{TC\cup A}(j : fl')$. By definition, the assumption that I is a probabilistic stable model of $Tr(D, m)$ means I is a (deterministic) stable model of

$$\begin{aligned}\overline{Tr(D,m)_I} = {} & \overline{D_{init\,I}} \cup \\ & SD(0) \cup \cdots \cup SD(m) \cup \\ & FD(0) \cup \cdots \cup FD(m-1) \cup \\ & \overline{PF(0)_{TC}} \cup \cdots \cup \overline{PF(m-1)_{TC}} \cup \\ & UEC \cup EXG\end{aligned}$$

which, by the splitting theorem, implies that I is a stable model of

$$\begin{aligned}& \overline{(D_{init})_I} \cup \\ & SD(0) \cup \cdots \cup SD(i-1) \cup SD(i+1) \cup \cdots \cup SD(m) \cup \\ & FD(0) \cup \cdots \cup FD(i-2) \cup FD(i) \cup \ldots FD(m-1) \cup \\ & \overline{PF(0)_{TC}} \cup \cdots \cup \overline{PF(i-2)_{TC}} \cup \overline{PF(i)_{TC}} \cup \cdots \cup \overline{PF(m-1)_{TC}} \cup \\ & \overline{INIT_{TC}} \cup UEC \cup EXG\end{aligned}$$

w.r.t. $\sigma^{initpf} \cup 0..(i-1) : \sigma^{fl} \cup (i+1)..m : \sigma^{fl} \cup 0..(i-2) : \sigma^{act} \cup i..(m-1) : \sigma^{act} \cup 0..(i-2) : \sigma^{pf} \cup i..(m-1) : \sigma^{pf}$, and I is a stable model of

$$SD(i) \cup FD(i-1) \cup \overline{PF(i-1)_{TC}}$$

w.r.t. $i : \sigma^{fl} \cup i-1 : \sigma^{act} \cup i-1 : \sigma^{pf}$. Changing the timesteps, this means

$$0 : I|_{i-1:\sigma^{fl}}, 0 : I|_{i-1:\sigma^{act}}, 1 : I|_{i:\sigma^{fl}}, 0 : I|_{i-1:\sigma^{pf}}$$

is a stable model of

$$SD(1) \cup FD(0) \cup \overline{PF(0)_{TC}} \cup UEC \cup EXG$$

w.r.t. $1 : \sigma^{fl} \cup 0 : \sigma^{act} \cup 0 : \sigma^{pf}$. On the other hand, clearly, $0 : I|_{i-1:\sigma^{fl}}, 0 : I|_{i-1:\sigma^{act}}, 1 : I|_{i:\sigma^{fl}}, 0 : I|_{i-1:\sigma^{pf}}$ also satisfies $SD(0)$. Due to the existence of EXG, we have

$$0 : I|_{i-1:\sigma^{fl}}, 0 : I|_{i-1:\sigma^{act}}, 1 : I|_{i:\sigma^{fl}}, 0 : I|_{i-1:\sigma^{pf}}$$

is a stable model of

$$SD(0) \cup SD(1) \cup FD(0) \cup \overline{PF(0)_{TC}} \cup UEC \cup EXG = D_1$$

The above implies that $(\langle I|_{i-1:\sigma^{fl}}, I|_{i-1:\sigma^{act}}, I|_{i:\sigma^{fl}}\rangle, I|_{i-1:\sigma^{pf}})$ is also a pf-transition in addition to

$$(\langle (I_{TC\cup A})|_{i-1:\sigma^{fl}}, (I_{TC\cup A})|_{i-1:\sigma^{act}}, (I_{TC\cup A})|_{i:\sigma^{fl}}\rangle, (I_{TC\cup A})|_{i-1:\sigma^{pf}}),$$

which contradict our assumption 2.

Consequently, in $Tr(D, m)$, since there are $(|\sigma^{act}| + 1)^m$ different assignments of σ^{act} under

Assumption 1, every total choice leads to $(|\sigma^{act}|+1)^m$ stable models. By Theorem 2, we have

$$Pr_{Tr(D,m)}(I_{TC\cup A}) = \frac{\prod\limits_{c=v\in TC} M(c=v)}{(|\sigma^{act}|+1)^m}.$$

□

A.3 Proof of Theorem 2

For a multi-valued probabilistic program $\mathbf{\Pi}$, a *total choice* of $\mathbf{\Pi}$ is a value assignment to probabilistic constants in $\mathbf{\Pi}$. For any interpretation I, of a multi-valued probabilistic program, that satisfies uniqueness and existence constraints for all constants, the *total choice* of I, denoted $TC(I)$, is the function that maps each probabilistic constant c to the value v such that $c=v\in I$. We say a total choice tc *leads* to an interpretation I if I satisfies tc.

In the following proofs, we sometimes identify a value assignment A with the set

$$\{c=v \mid A(c)=v\}.$$

Proposition 3
For any multi-valued probabilistic program $\mathbf{\Pi}=\langle PF,\Pi\rangle$ such that every total choice leads to the same number of stable models, we have

$$Pr_{\mathbf{\Pi}}(c=v) = M_{\mathbf{\Pi}}(c=v)$$

for any probabilistic constant c and $v\in Dom(c)$.

Proof. Let n be the number of stable models that each total choices leads to. By Proposition 2 we have

$$\begin{aligned}
&Pr_{\mathbf{\Pi}}(c=v)\\
=&\sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi}\\ I\models c=v}} \frac{\prod\limits_{c'=v'\in TC(I)} M_{\mathbf{\Pi}}(c'=v')}{n}\\
=&M_{\mathbf{\Pi}}(c=v)\cdot\frac{1}{n}\cdot \sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi}\\ I\models c=v}} \prod_{\substack{c'=v'\in TC(I)\\ c'\neq c}} M_{\mathbf{\Pi}}(c'=v')\\
=&M_{\mathbf{\Pi}}(c=v)\cdot\frac{1}{n}\cdot n \sum_{v'\in Dom(c_1)} M_{\mathbf{\Pi}}(c_1=v')\dots \sum_{v'\in Dom(c_n)} M_{\mathbf{\Pi}}(c_n=v')\\
=&M_{\mathbf{\Pi}}(c=v)\cdot\frac{1}{n}\cdot n\cdot 1\\
=&M_{\mathbf{\Pi}}(c=v)
\end{aligned}$$

□

Proposition 4
For any multi-valued probabilistic program $\mathbf{\Pi}=\langle PF,\Pi\rangle$ such that every total choice leads to the same number of stable models, and any value assignment ppf of a subset P of probabilistic constants in $\mathbf{\Pi}$, we have

$$Pr_{\mathbf{\Pi}}(\bigwedge_{pf=v\in ppf} pf=v) = \prod_{pf=v\in ppf} Pr_{\mathbf{\Pi}}(pf=v).$$

Proof. Let n denote the number of stable models each total choice leads to. By Proposition 2 we have

$$\begin{aligned}
&Pr_{\mathbf{\Pi}}(\bigwedge_{ppf(pf)=v} pf = v) \\
= &\sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi} \\ I \models \bigwedge_{ppf(pf)=v} pf=v}} \frac{W''_{\mathbf{\Pi}}(I)}{n} \\
= &\sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi} \\ I \models \bigwedge_{ppf(pf)=v} pf=v}} \frac{\prod_{c=v \in TC(I)} M_{\mathbf{\Pi}}(c = v)}{n} \\
= &\sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi} \\ I \models \bigwedge_{ppf(pf)=v} pf=v}} \frac{\prod_{ppf(pf)=v} M_{\mathbf{\Pi}}(pf = v) \cdot \prod_{c=v \in TC(I) \setminus ppf} M_{\mathbf{\Pi}}(c = v)}{n} \\
= &\prod_{ppf(pf)=v} M_{\mathbf{\Pi}}(pf = v) \cdot \frac{1}{n} \sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi} \\ I \models \bigwedge_{ppf(pf)=v} pf=v}} \prod_{c=v \in TC(I) \setminus ppf} M_{\mathbf{\Pi}}(c = v)
\end{aligned}$$

We use C to denote the set of all constants in $\mathbf{\Pi}$. Let $C \setminus P = \{c_1, \dots, c_n\}$. Since every total choice leads to the same number of stable models, we have

$$\begin{aligned}
&\frac{1}{n} \sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi} \\ I \models \bigwedge_{pf=v \in ppf} pf=v}} \prod_{c=v \in TC(I) \setminus ppf} M_{\mathbf{\Pi}}(c = v) \\
= &\frac{1}{n} \sum_{TC \text{ is a value assignment of } C \setminus P} n \cdot \prod_{TC(c)=v} M_{\mathbf{\Pi}}(c = v) \\
= &\frac{1}{n} \cdot n \sum_{v \in Dom(c_1)} M_{\mathbf{\Pi}}(c_1 = v) \dots \sum_{v \in Dom(c_n)} M_{\mathbf{\Pi}}(c_n = v) \\
= &1.
\end{aligned}$$

Consequently by Proposition 3 we have

$$\begin{aligned}
&Pr_{\mathbf{\Pi}}(\bigwedge_{pf=v \in ppf} pf = v) \\
= &\prod_{pf=v \in ppf} M_{\mathbf{\Pi}}(pf = v) \cdot \frac{1}{n} \sum_{\substack{I \text{ is a stable model of } \mathbf{\Pi} \\ I \models \bigwedge_{pf=v \in ppf} pf=v}} \prod_{c=v \in TC(I) \setminus ppf} M_{\mathbf{\Pi}}(c = v) \\
= &\prod_{pf=v \in ppf} M_{\mathbf{\Pi}}(pf = v) \cdot 1 \\
= &\prod_{pf=v \in ppf} Pr_{\mathbf{\Pi}}(pf = v).
\end{aligned}$$

□

Theorem 2 *For any state s and s', and any interpretation e of σ^{act}, we have*

$$Pr_{Tr(D,m)}(i+1 : s' \mid i : s, i : e) = Pr_{Tr(D,m)}(j+1 : s' \mid j : s, j : e)$$

for any $i, j \in \{0, \dots, m-1\}$ such that $Pr_{Tr(D,m)}(i : s) \neq 0$ and $Pr_{Tr(D,m)}(j : s) \neq 0$.
Proof. For any $k \in \{0, \dots, m-1\}$ such that $Pr_{Tr(D,m)}(k : s) \neq 0$, we show that

$$Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e) = Pr_{D_m}(1 : s' \mid 0 : s, 0 : e).$$

Firstly, since $Tr(D,m)$ satisfies the condition that every total choice leads to the same number of stable models, by Proposition 3, we have

$$\begin{aligned} Pr_{Tr(D,m)}(i : pf = v) &= M_{Tr(D,m)}(i : pf = v) \\ &= M_{D_m}(i : pf = v) \end{aligned} \tag{A8}$$

for any pf constant pf and $v \in Dom(pf)$ and any $i \in \{0, \dots, m-1\}$.

Secondly, from Theorem 1, it can be seen that for any (probabilistic) stable model I of $Tr(D,m)$, $(\langle I|_{i:\sigma^{fl}}, I|_{i:\sigma^{act}}, I|_{i+1:\sigma^{fl}}\rangle, I|_{i:\sigma^{pf}})$ is always a pf-transition: the contrary would imply that for some stable model I of $Tr(D,m)$, there does not exist any assignment $TC \cup A$ on pf constants and action constants such that $I = I_{TC \cup A}$, which contradicts Theorem 1. Under Assumption 2, this implies

$$Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e, k : pf) = \begin{cases} 1 & \text{if } (\langle s, a, s'\rangle, pf) \text{ is a pf-transition} \\ 0 & \text{otherwise} \end{cases}$$

and thus

$$Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e, k : pf) = Pr_{D_m}(1 : s' \mid 0 : s, 0 : e, 0 : pf) \tag{A9}$$

for all assignments pf to σ^{pf}.

From (A8) and (A9), and by Proposition 4, we have

$$\begin{aligned}
& Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e) \\
&= \{\text{Law of Total Probability}\} \\
&= \sum_{pf \text{ is any value assignment to } \sigma^{pf}} Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e, k : pf) \cdot Pr_{Tr(D,m)}(k : pf) \\
&= \sum_{pf \text{ is any value assignment to } \sigma^{pf}} Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e, k : pf) \cdot \\
&\quad (\prod_{c \in \sigma^{pf}} Pr_{Tr(D,m)}(k : c = pf(c))) \\
&= \{\text{Proposition 4 and (A8)}\} \\
&= \sum_{pf \text{ is any value assignment to } \sigma^{pf}} Pr_{Tr(D,m)}(k+1 : s' \mid k : s, k : e, k : pf) \cdot \\
&\quad (\prod_{c \in \sigma^{pf}} M_{D_m}(k : c = pf(c))) \\
&= \{\text{From (A9)}\} \\
&= \sum_{pf \text{ is any value assignment to } \sigma^{pf}} Pr_{D_m}(1 : s' \mid 0 : s, 0 : e, 0 : pf) \cdot (\prod_{c \in \sigma^{pf}} M_{D_m}(0 : c = pf(c))) \\
&= Pr_{D_m}(1 : s' \mid 0 : s, 0 : e)
\end{aligned}$$

□

Corollary 1 *For every $m \geqslant 1$, X_m is a residual (probabilistic) stable model of $Tr(D,m)$ iff $X^0, \dots, X^{m-1}$ are transitions of D and $0\!:\!s_0$ is a residual stable model of D_{init}. Furthermore,*

$$Pr_{Tr(D,m)}(X_m \mid 0\!:\!e_0, \dots, m-1\!:\!e_{m-1}) = p(X^0) \times \dots \times p(X^m) \times Pr_{Tr(D,m)}(0\!:\!s_0).$$

Proof. By Theorem 1, an interpretation I is a (probabilistic) stable model of $Tr(D,m)$ iff $I^0, \dots, I^{m-1}$ are pf-transitions and $(I_{TC \cup A})|_{0:\sigma^{fl}} \cup (I_{TC \cup A})|_{\sigma^{initpf}}$ is a residual stable model of $D_{init} \cup PF_0(D)$. From the definition of transition and pf-transition, it follows that X_m is a residual (probabilistic) stable model of D_m iff $X^0, \dots, X^{m-1}$ are transitions of D and $0 : s_0$ is a residual stable model of D_{init}.

Furthermore, we have

$$\begin{aligned}
& Pr_{Tr(D,m)}(X_m \mid 0 : e_0, \dots, 0 : e_{m-1}) \\
= \; & Pr_{Tr(D,m)}(m : s_m \mid m-1 : s_{m-1}, m-1 : e_{m-1}) \cdot \\
& \dots \cdot Pr_{Tr(D,m)}(2 : s_2 \mid 1 : s_1, 1 : e_1) \cdot \\
& Pr_{Tr(D,m)}(1 : s_1 \mid 0 : s_0, 0 : e_0) \cup Pr_{Tr(D,m)}(0 : s_0)
\end{aligned}$$

We have

$$\begin{aligned}
& Pr_{Tr(D,m)}(X_m \mid s_0, e_0, \dots, e_{m-1}) \\
= \; & \{\text{By Theorem 2}\} \\
= \; & Pr_{D_m}(1 : s_m \mid 0 : s_{m-1}, 0 : e_{m-1}) \cdot \dots \cdot Pr_{D_m}(1 : s_2 \mid 0 : s_1, 0 : e_1) \cdot \\
& Pr_{D_m}(1 : s_1 \mid 0 : s_0, 0 : e_0) \cdot Pr_{Tr(D,m)}(0 : s_0) \\
= \; & Pr_{D_m}(1 : s_1 \mid 0 : s_0, 0 : e_0) \cdot Pr_{D_m}(1 : s_2 \mid 0 : s_1, 0 : e_1) \cdot \dots \cdot \\
& Pr_{D_m}(1 : s_m \mid 0 : s_{m-1}, 0 : e_{m-1}) \cdot Pr_{Tr(D,m)}(0 : s_0) \\
= \; & p(X^1) \times \dots \times p(X^m) \times Pr_{Tr(D,m)}(0 : s_0).
\end{aligned}$$

□

Appendix B LP$^{\text{MLN}}$ Translation of the Yale Shooting Example (Section 4) in the Input Language of LPMLN2ASP

```
astep(0).
step(0..1).
boolean(t; f).
turkey(slimTurkey; fatTurkey).

% ---------- INIT(D) ------------
@log(0.5) pf_initAlive(T, t) :- turkey(T).
@log(0.5) pf_initAlive(T, f) :- turkey(T).
@log(0.5) pf_initLoaded(t).
@log(0.5) pf_initLoaded(f).

:- not alive(T, B, 0), pf_initAlive(T, B).
:- not loaded(B, 0), pf_initLoaded(B).

:- pf_initAlive(T, t), pf_initAlive(T, f).
:- not pf_initAlive(T, t), not pf_initAlive(T, f), turkey(T).
:- pf_initLoaded(t), pf_initLoaded(f).
:- not pf_initLoaded(t), not pf_initLoaded(f).

% ---------- PF(D) ----------
%% Probability Distribution
@log(0.6) pf_turkeyKilled(slimTurkey, t, I) :- astep(I).
@log(0.4) pf_turkeyKilled(slimTurkey, f, I) :- astep(I).
@log(0.3) pf_turkeyKilled_Alert(slimTurkey, t, I) :- astep(I).
@log(0.7) pf_turkeyKilled_Alert(slimTurkey, f, I) :- astep(I).
@log(0.9) pf_turkeyKilled(fatTurkey, t, I) :- astep(I).
@log(0.1) pf_turkeyKilled(fatTurkey, f, I) :- astep(I).
@log(0.7) pf_turkeyKilled_Alert(fatTurkey, t, I) :- astep(I).
@log(0.3) pf_turkeyKilled_Alert(fatTurkey, f, I) :- astep(I).

%% Fluent Dynamic Laws
loaded(t, I+1) :- load(t, I), astep(I).
alive(T, f, I+1) :- loaded(t, I), fire(T, t, I), alert(T, f, I),
    pf_turkeyKilled(T, t, I), astep(I).
alive(T, f, I+1) :- loaded(t, I), fire(T, t, I), alert(T, t, I),
    pf_turkeyKilled_Alert(T, t, I), astep(I).
loaded(f, I+1) :- fire(T, t, I).

{alive(T, B, I+1)} :- alive(T, B, I), astep(I), boolean(B), turkey(T).
{loaded(B, I+1)} :- loaded(B, I), astep(I), boolean(B).

%% Static Laws
alert(T, f, I) :- not not alert(T, f, I), turkey(T), step(I).
alert(T, t, I) :- alive(T1, f, I), alive(T, t, I).

%% Initial State and Actions are Random
{alive(T, B, 0)} :- turkey(T), boolean(B).
{loaded(B, 0)} :- boolean(B).
{load(B, I)} :- boolean(B), astep(I).
{fire(T, B, I)} :- turkey(T), boolean(B), astep(I).

%% UEC
:- alive(T, t, I), alive(T, f, I).
:- not alive(T, t, I), not alive(T, f, I), step(I), turkey(T).
:- alert(T, t, I), alert(T, f, I).
:- not alert(T, t, I), not alert(T, f, I), step(I), turkey(T).
:- loaded(t, I), loaded(f, I).
:- not loaded(t, I), not loaded(f, I), step(I).
:- fire(T, t, I), fire(T, f, I).
:- not fire(T, t, I), not fire(T, f, I), astep(I), turkey(T).
:- load(t, I), load(f, I).
:- not load(t, I), not load(f, I), astep(I).
:- pf_turkeyKilled(T, t, I), pf_turkeyKilled(T, f, I).
:- not pf_turkeyKilled(T, t, I), not pf_turkeyKilled(T, f, I),
   astep(I), turkey(T).
:- pf_turkeyKilled_Alert(T, t, I), pf_turkeyKilled_Alert(T, f, I).
:- not pf_turkeyKilled_Alert(T, t, I), not pf_turkeyKilled_Alert(T, f, I),
   astep(I), turkey(T).

%% No Concurrency
:- fire(T1, t, I), fire(T2, t, I), astep(I), T1 != T2,
   turkey(T1), turkey(T2).
:- load(t, I), fire(T, t, I).
```

The prediction query "given that only the fat turkey is alive and the gun is loaded at the beginning, what is the probability that the turkey died after shooting is executed" is answered by adding the constraints

```
:- not alive(slimTurkey, f, 0).
:- not alive(fatTurkey, t, 0).
:- not loaded(t, 0).
:- not fire(fatTurkey, t, 0).
```

to the LP$^{\text{MLN}}$ program and executing the command line:

```
lpmln2asp -i yale-shooting.lpmln -q alive
```

The output is:

```
alive(fatTurkey, t, 0) 1.0
alive(fatTurkey, t, 1) 0.299999550682
alive(fatTurkey, f, 1) 0.700000449318
alive(slimTurkey, f, 0) 1.0
alive(slimTurkey, f, 1) 1.0
```

The postdiction query "given that the slim turkey was alive and the gun was loaded at the beginning, the person shot at the slim turkey and it died, what is the probability that the fat turkey was alive at the beginning" is answered by adding the constraints

```
:- not alive(slimTurkey, t, 0).
:- not loaded(t, 0).
:- not fire(slimTurkey, t, 0).
:- not alive(slimTurkey, f, 1).
```

to the LP$^{\text{MLN}}$ program and executing the command line:

```
lpmln2asp -i yale-shooting.lpmln -q alive
```

The output is

```
alive(fatTurkey, f, 0) 0.333338788027
alive(slimTurkey, t, 0) 1.0
alive(fatTurkey, t, 1) 0.666661211973
alive(slimTurkey, f, 1) 1.0
alive(fatTurkey, t, 0) 0.666661211973
alive(fatTurkey, f, 1) 0.333338788027
```

The planning problem "given that both the turkeys are alive and the gun is not loaded at the beginning, generate a plan that gives best chance to kill both the turkeys with 4 actions" is answered by adding the constraints, describing the initial state and the goal,

```
:- not alive(slimTurkey, t, 0).
:- not alive(fatTurkey, t, 0).
:- not loaded(f, 0).

:- not alive(slimTurkey, f, 4).
:- not alive(fatTurkey, f, 4).
```

to the LP$^{\text{MLN}}$ program, and executing the command line:

```
lpmln2asp -i yale-shooting.lpmln
```

The output is

```
Answer: 9
pf_initAlive(slimTurkey,t) pf_initAlive(fatTurkey,t)
unsat(2,''-0.693100'',slimTurkey)
unsat(2,''-0.693100'',fatTurkey) unsat(3,''-0.693100'')
pf_initLoaded(f) pf_turkeyKilled(slimTurkey,t,0)
pf_turkeyKilled(slimTurkey,t,1) pf_turkeyKilled(slimTurkey,t,2)
pf_turkeyKilled(slimTurkey,t,3) pf_turkeyKilled(fatTurkey,t,0)
pf_turkeyKilled(fatTurkey,t,1) pf_turkeyKilled(fatTurkey,t,2)
pf_turkeyKilled(fatTurkey,t,3)
alert(slimTurkey,f,0) alert(slimTurkey,f,1) alert(slimTurkey,f,2)
alert(slimTurkey,f,3) alert(slimTurkey,f,4) alert(fatTurkey,f,0)
alert(fatTurkey,f,1) alert(fatTurkey,f,4)
loaded(t,1) load(t,0) loaded(t,3) load(t,2) loaded(f,2)
fire(slimTurkey,t,1) loaded(f,4) fire(fatTurkey,t,3)
alive(slimTurkey,f,2) alive(slimTurkey,f,3) alive(slimTurkey,f,4)
alive(fatTurkey,f,4)
unsat(13,''-1.203900'',0) unsat(13,''-1.203900'',1)
unsat(13,''-1.203900'',2) unsat(13,''-1.203900'',3)
pf_turkeyKilled_Alert(fatTurkey,t,0)
pf_turkeyKilled_Alert(fatTurkey,t,1)
pf_turkeyKilled_Alert(fatTurkey,t,2)
pf_turkeyKilled_Alert(fatTurkey,t,3)
alive(slimTurkey,t,0) alive(fatTurkey,t,0) alive(slimTurkey,t,1)
alive(fatTurkey,t,1) alert(fatTurkey,t,2) alive(fatTurkey,t,2)
alert(fatTurkey,t,3) alive(fatTurkey,t,3) loaded(f,0)
```

```
unsat(12,''-0.916200'',0) unsat(12,''-0.916200'',1)
unsat(12,''-0.916200'',2) unsat(12,''-0.916200'',3)
pf_turkeyKilled_Alert(slimTurkey,f,0)
pf_turkeyKilled_Alert(slimTurkey,f,1)
pf_turkeyKilled_Alert(slimTurkey,f,2)
pf_turkeyKilled_Alert(slimTurkey,f,3) unsat(16,''-2.302500'',0)
unsat(16,''-2.302500'',1) unsat(16,''-2.302500'',2)
unsat(16,''-2.302500'',3) unsat(18,''-1.203900'',0)
unsat(18,''-1.203900'',1) unsat(18,''-1.203900'',2)
unsat(18,''-1.203900'',3) fire(slimTurkey,f,0) fire(slimTurkey,f,2)
fire(slimTurkey,f,3) fire(fatTurkey,f,0) fire(fatTurkey,f,1)
fire(fatTurkey,f,2) load(f,1) load(f,3)
Optimization: 7387
OPTIMUM FOUND
```

Appendix C LP$^{\text{MLN}}$ Translation of the Robot Example (Section 5) in the Input Language of LPMLN2ASP

```
astep(0..2).
step(0..3).
boolean(t; f).
room(r1; r2).

% ---------- D_Init ------------
%% Probability Distribution
%%% caused Init_LocRobot = {R1: 0.5, R2: 0.5}
@log(0.5) init_locRobot(r1).
@log(0.5) init_locRobot(r2).
%%% caused Init_LocBook = {R1: 0.5, R2: 0.5}
@log(0.5) init_locBook(r1).
@log(0.5) init_locBook(r2).
%%% caused Init_HasBook = {t: 0.5, f: 0.5}
@log(0.5) init_hasBook(t).
@log(0.5) init_hasBook(f).

%% Initial Static Laws
%%% initially LocBook = r if Init_LocBook = r
:- not locBook(R, 0), init_locBook(R).
%%% initially LocRobot = r if Init_LocRobot = r
:- not locRobot(R, 0), init_locRobot(R).
%%% initially hasBook = b if Init_HasBook = b
:- hasBook(B, 0), init_hasBook(B).
%%% initially \bot if EnterFailed
:- ab(enter_failed, t, 0).
%%% initially \bot if PickupFailed
```

```
:- ab(pickup_failed, t, 0).
%%% initially \bot if DropBook
:- ab(drop_book, t, 0).
%% UEC
:- init_locRobot(r1), init_locRobot(r2).
:- not init_locRobot(r1), not init_locRobot(r2).
:- init_locBook(r1), init_locBook(r2).
:- not init_locBook(r1), not init_locBook(r2).
:- init_hasBook(t), init_hasBook(f).
:- not init_hasBook(t), not init_hasBook(f).

% ---------- D_m ----------
%% Probability Distribution
%%% caused Pf_EnterFailed = {t: 0.1, f: 0.9}
@log(0.1) pf_enterFailed(t, I) :- astep(I).
@log(0.9) pf_enterFailed(f, I) :- astep(I).
%%% caused Pf_PickupFailed = {t: 0.3, f: 0.7}
@log(0.3) pf_pickupFailed(t, I) :- astep(I).
@log(0.7) pf_pickupFailed(f, I) :- astep(I).
%%% caused Pf_DropBook = {t: 0.2, f: 0.8}
@log(0.2) pf_dropBook(t, I) :- astep(I).
@log(0.8) pf_dropBook(f, I) :- astep(I).

%% Fluent Dynamic Laws
%%% caused LocRobot = r after Goto(r) & ~EnterFailed
locRobot(R, I+1) :- goto(R, t, I), not ab(enter_failed, t, I+1).
%%% caused HasBook if LocRobot = LocBook after PickupBook & ~PickupFailed
hasBook(t, I+1) :- pickupBook(t, I), locRobot(R, I+1), locBook(R, I+1),
    not ab(pickup_failed, t, I+1).
%%% caused ~HasBook after putdownBook
hasBook(f, I+1) :- putdownBook(t, I).

%%% caused_ab EnterFailed if \top after pf_EnterFailed & Goto(r)
ab(enter_failed, B, I+1) :- ab(t, I+1), goto(R, t, I), pf_enterFailed(B, I).
%%% caused_ab EnterFailed if \top after pf_EnterFailed & Goto(r)
ab(pickup_failed, B, I+1) :-
    ab(t, I+1), pickupBook(t, I), pf_pickupFailed(B, I).
%%% caused_ab DropBook if \top after hasBook & Pf_DropBook
ab(drop_book, B, I+1) :- ab(t, I+1), hasBook(t, I), pf_dropBook(B, I).

%%% caused {LocRobot = r}^{ch} after LocRobot = r
{locRobot(R, I+1)} :- locRobot(R, I), astep(I).
%%% caused {LocBook = r}^{ch} after LocBook = r
{locBook(R, I+1)} :- locBook(R, I), astep(I).
%%% caused {HasBook = b}^{ch} after HasBook = b
{hasBook(B, I+1)} :- hasBook(B, I), astep(I).
```

```
%% Static Laws
%%% caused LocBook = r if LocRobot = r & HasBook
locBook(R, I) :- locRobot(R, I), hasBook(t, I).
hasBook(f, I) :- ab(drop_book, t, I).
%%% caused {~EnterFailed}^{ch} if ~EnterFailed
ab(enter_failed, f, I) :- not not ab(enter_failed, f, I), step(I).
%%% caused {~PickupFailed}^{ch} if ~PickupFailed
ab(pickup_failed, f, I) :- not not ab(pickup_failed, f, I), step(I).
%%% caused {~DropBook}^{ch} if ~DropBook
ab(drop_book, f, I) :- not not ab(drop_book, f, I), step(I).

%% Initial State and Actions are Random
{locRobot(R, 0)} :- room(R).
{locBook(R, 0)} :- room(R).
{hasBook(B, 0) : boolean(B)}.
{ab(enter_failed, B, 0) : boolean(B)}.
{ab(pickup_failed, B, 0) : boolean(B)}.
{ab(drop_book, B, 0) : boolean(B)}.
{goto(R, B, I) : boolean(B)} :- room(R), astep(I).
{pickupBook(B, I) : boolean(B)} :- astep(I).
{putdownBook(B, I) : boolean(B)} :- astep(I).

%% UEC
:- locRobot(r1, I), locRobot(r2, I).
:- not locRobot(r1, I), not locRobot(r2, I), step(I).
:- locBook(r1, I), locBook(r2, I).
:- not locBook(r1, I), not locBook(r2, I), step(I).
:- hasBook(t, I), hasBook(f, I).
:- not hasBook(t, I), not hasBook(f, I), step(I).
:- ab(enter_failed, t, I), ab(enter_failed, f, I).
:- not ab(enter_failed, t, I), not ab(enter_failed, f, I), step(I).
:- ab(pickup_failed, t, I), ab(pickup_failed, f, I).
:- not ab(pickup_failed, t, I), not ab(pickup_failed, f, I), step(I).
:- ab(drop_book, t, I), ab(drop_book, f, I).
:- not ab(drop_book, t, I), not ab(drop_book, f, I), step(I).
:- goto(R, t, I), goto(R, f, I).
:- not goto(R, t, I), not goto(R, f, I), astep(I), room(R).
:- pickupBook(t, I), pickupBook(f, I).
:- not pickupBook(t, I), not pickupBook(f, I), astep(I).
:- putdownBook(t, I), putdownBook(f, I).
:- not putdownBook(t, I), not putdownBook(f, I), astep(I).
:- ab(t, I), ab(f, I).
:- not ab(t, I), not ab(f, I), step(I).

%% No Concurrency
```

```
:- goto(R1, t, I), goto(R2, t, I), astep(I), R1 != R2.
:- goto(R, t, I), pickupBook(t, I), room(R), astep(I).
:- goto(R, t, I), putdownBook(t, I), room(R), astep(I).
:- pickupBook(t, I), putdownBook(t, I), astep(I).

% Action and Observation History
:- not locRobot(r1, 0).
:- not locBook(r1, 0).
:- not hasBook(f, 0).
:- not pickupBook(t, 0).
:- not goto(r2, t, 1).
:- not putdownBook(t, 2).
:- locBook(r2, 3).

% Enable abnormality
%% caused ab
ab(t,I) :- step(I).

#show ab/3.
#show pickupBook/2.
#show goto/3.
#show putdownBook/2.
#show locRobot/2.
#show locBook/2.
#show hasBook/2.
```

For the above program,

```
lpmln2asp -i robot.lpmln
```

gives the output

```
Answer: 71
ab(enter_failed,f,1) ab(enter_failed,f,2)
goto(r2,t,1) ab(enter_failed,f,3)
locRobot(r1,1) locRobot(r2,2) locRobot(r2,3) ab(pickup_failed,t,1)
pickupBook(t,0) ab(pickup_failed,f,2) ab(pickup_failed,f,3)
hasBook(f,1) hasBook(f,2) hasBook(f,3) putdownBook(t,2)
ab(drop_book,f,1) hasBook(f,0) locBook(r1,0) locRobot(r1,0)
ab(drop_book,f,2) locBook(r1,1) ab(drop_book,f,3) locBook(r1,2)
locBook(r1,3) ab(enter_failed,f,0) ab(pickup_failed,f,0)
ab(drop_book,f,0) goto(r1,f,0) goto(r1,f,1) goto(r1,f,2) goto(r2,f,0)
goto(r2,f,2) pickupBook(f,1) pickupBook(f,2) putdownBook(f,0)
putdownBook(f,1)
Optimization: 3283
OPTIMUM FOUND
```

The observation that the book was in the robot's hand after it picked up the book is represented as the constraint

```
:- not hasBook(t, 1).
```

With the constraint in the program,

```
lpmln2asp -i robot.lpmln
```

gives the output

```
Answer: 50
ab(enter_failed,f,1) ab(enter_failed,f,2) goto(r2,t,1)
ab(enter_failed,f,3) locRobot(r1,1) locRobot(r2,2) locRobot(r2,3)
pickupBook(t,0) ab(pickup_failed,f,1) ab(pickup_failed,f,2)
ab(pickup_failed,f,3) hasBook(f,2) hasBook(f,3) putdownBook(t,2)
ab(drop_book,f,1) hasBook(f,0) locBook(r1,0) locRobot(r1,0)
ab(drop_book,t,2) hasBook(t,1) locBook(r1,1) ab(drop_book,f,3)
locBook(r1,2) locBook(r1,3) ab(enter_failed,f,0)
ab(pickup_failed,f,0) ab(drop_book,f,0) goto(r1,f,0) goto(r1,f,1)
goto(r1,f,2) goto(r2,f,0) goto(r2,f,2) pickupBook(f,1)
pickupBook(f,2) putdownBook(f,0) putdownBook(f,1)
Optimization: 3688
OPTIMUM FOUND
```

The observation that the robot itself was not at `r2` after the execution of the plan is represented as the constraint

```
:- locRobot(r2, 3).
```

With the constraint in the program,

```
lpmln2asp -i robot.lpmln
```

gives the output

```
Answer: 89
ab(enter_failed,f,1) ab(enter_failed,t,2) goto(r2,t,1)
ab(enter_failed,f,3) locRobot(r1,1) locRobot(r1,2) locRobot(r1,3)
pickupBook(t,0) ab(pickup_failed,f,1) ab(pickup_failed,f,2)
ab(pickup_failed,f,3) hasBook(f,3) putdownBook(t,2) ab(drop_book,f,1)
hasBook(f,0) locBook(r1,0) locRobot(r1,0) hasBook(t,1)
ab(drop_book,f,2) locBook(r1,1) hasBook(t,2) ab(drop_book,f,3)
locBook(r1,2) locBook(r1,3) ab(enter_failed,f,0)
ab(pickup_failed,f,0) ab(drop_book,f,0) goto(r1,f,0) goto(r1,f,1)
goto(r1,f,2) goto(r2,f,0) goto(r2,f,2) pickupBook(f,1)
pickupBook(f,2) putdownBook(f,0) putdownBook(f,1)
Optimization: 4381
OPTIMUM FOUND
```