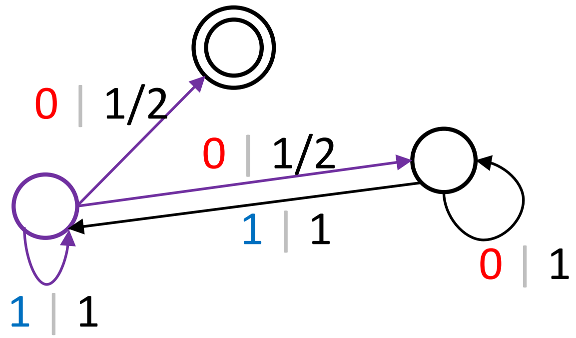
Automata Theoretic Approach for Model Checking Open Probabilistic Systems

A. Prasad Sistla

Joint work with Yue Ben, Rohit Chadha, Mahesh Viswanathan

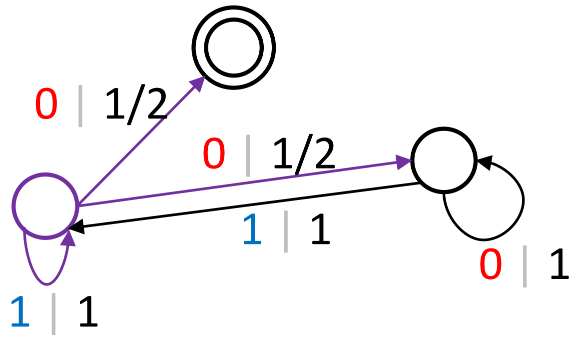
September 2015

Probabilistic Automata (PA)



- Like ordinary automata,
- Can go to multiple states with diff. probs. (Rabin ‘63)

Probabilistic Automata (PA)



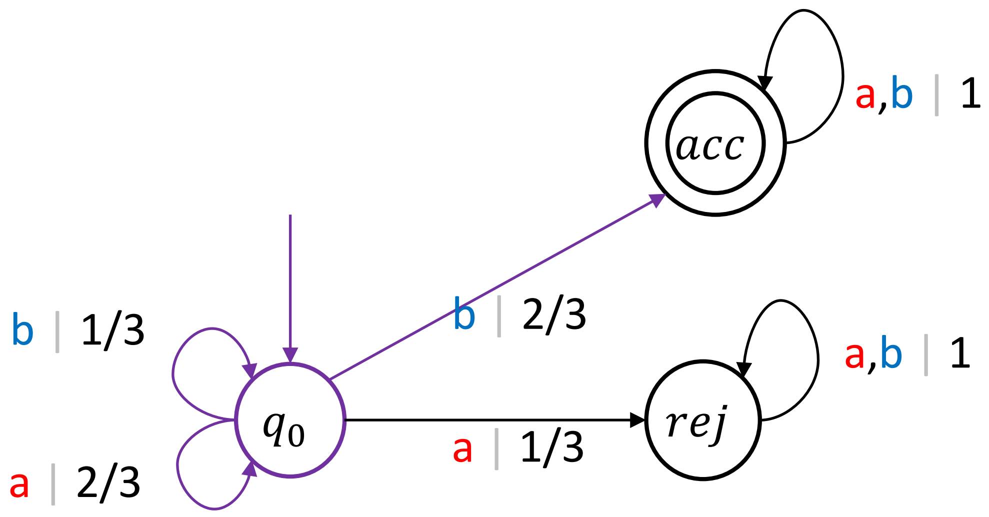
- Like ordinary automata,
- Can go to multiple states with diff. probs. (Rabin '63)

- Formally, a **PA**, on alphabet Σ, $\mathcal{A} = (Q, q_0, \delta, \mathrm{Acc})$,
  - q_0 - initial state.
  - $\mathrm{Acc} \subseteq Q$.
  - For $\mathrm{u} \in \Sigma^*, \delta_u(q_0, Acc)$ is the prob of reaching Acc on u

- PA on Finite strings (PFA)

PA Example

- PA $\mathcal{A} = (Q, q_0, \delta, Acc)$ on Σ :
  - $Q = \{q_0, acc, rej\}$
  - q_0- the initial state
  - $Acc = \{acc\}$
  - $\Sigma = \{a, b\}$
  - Transitions:
    - $\delta_a(q_0, q_0) = \frac{2}{3}$, $\delta_a(q_0, rej) = \frac{1}{3}$; $\delta_b(q_0, q_s) = \frac{1}{3}$, $\delta_b(q_0, acc\ ) = \frac{2}{3}$;
    - $\delta_a(acc, acc) = 1$, $\delta_b(acc, acc) = 1$;
    - $\delta_a(rej, rej) = 1$, $\delta_b(rej, rej) = 1$.



Given PA $\mathcal{A}$, and $x \in [0,1]$,

- $\boldsymbol{L_{>x}(\mathcal{A})}$
  - denotes input sequences accepted with prob$>x$;
  - can be non-regular. *(Rabin '63)*

Given PA $\mathcal{A}$, and $x \in [0,1]$,

- $\boldsymbol{L_{>x}(\mathcal{A})}$
  - denotes input sequences accepted with prob$>x$;
  - can be non-regular. *(Rabin ‘63)*

- Checking $\boldsymbol{L_{>x}(\mathcal{A})} \neq \emptyset$ is
  - undecidable for $x > 0$; *(Paz ‘71)*
  - trivially decidable for $x = 0$.

PA on Infinite Strings

- Prob Büchi Automata (PBA), Prob Muller Automata (PMA)
- For PA $\mathcal{A}$ and $\alpha \in \Sigma^{\omega}$,

$$\text{Prob}\{\mathcal{A} \text{ accepts } \alpha\} = \text{Prob}\{\mathcal{A}'\text{s accepting runs on } \alpha\}$$

PA on Infinite Strings

- Prob Büchi Automata (PBA), Prob Muller Automata (PMA)
- For PA $\mathcal{A}$ and $\alpha \in \Sigma^{\omega}$,

$$\text{Prob}\{\mathcal{A} \text{ accepts } \alpha\} = \text{Prob}\{\mathcal{A}'\text{s accepting runs on } \alpha\}$$

- Known results for PBA $\mathcal{A}$: checking if

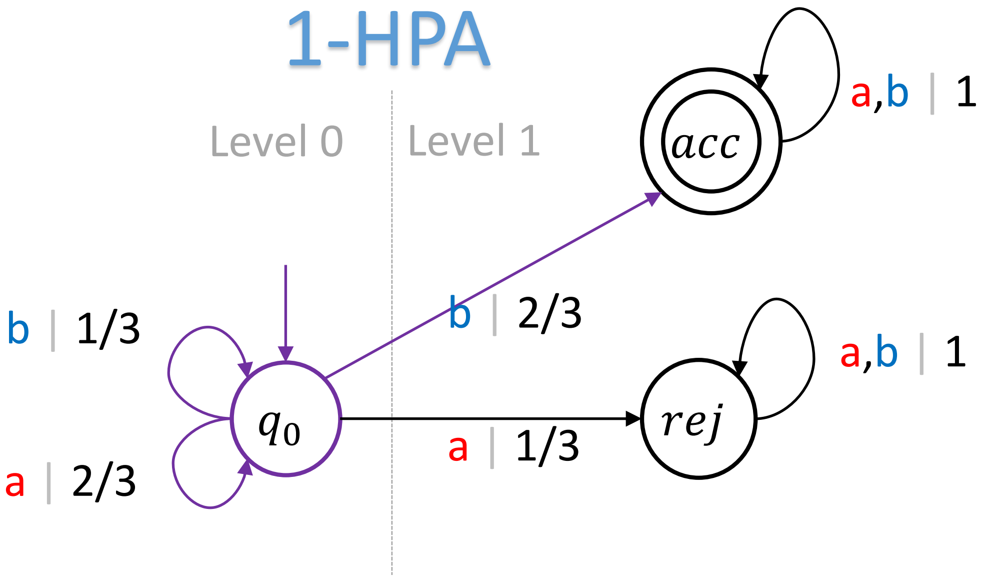
$$\begin{cases} \boldsymbol{L_{>0}(\mathcal{A}) \neq \emptyset} & \begin{cases}\text{is undecidable (Baier et al '09)} \\ \text{is } \Sigma_2^0\text{–complete. (C.S.V. '11)}\end{cases} \\ \boldsymbol{L_{>x}(\mathcal{A}) \neq \emptyset} & \text{is } \Sigma_2^0\text{–complete} \\ \begin{cases}\boldsymbol{L_{=1}(\mathcal{A}) \neq \emptyset} \\ \boldsymbol{L_{=1}(\mathcal{A}) = \Sigma^{\omega}}\end{cases} & \text{are PSPACE–complete(C.S.V. '11)} \end{cases}$$

Hierarchical PA (HPA)

- k-HPA $\mathcal{A} = (Q, q_0, \delta, \mathrm{A}cc)$ on alphabet Σ
  - States stratified into levels $0, 1, \dots, k$,
  - From a state at level i, on an input,
    - At most one transition goes to level i, all others go to higher levels.

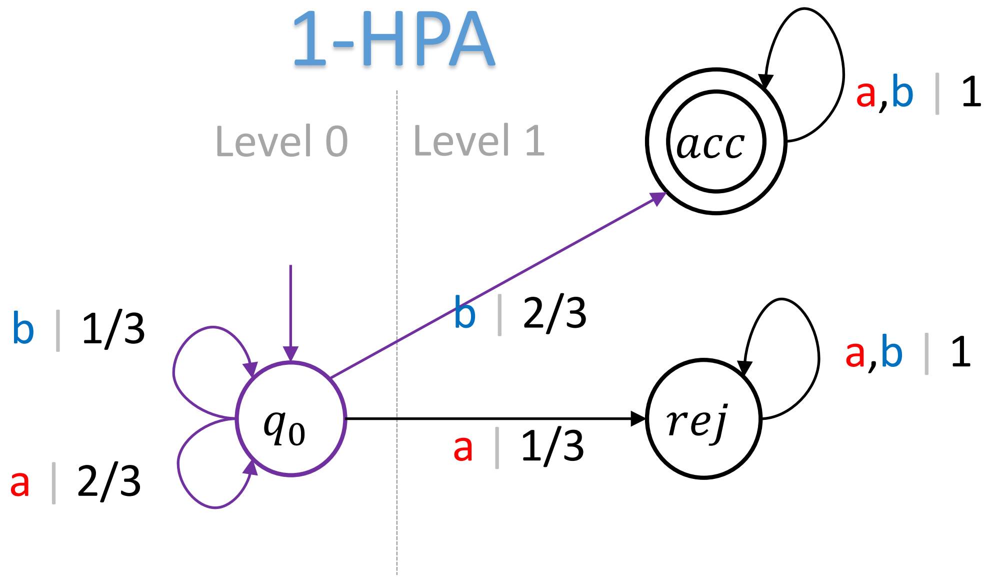
Hierarchical PA (HPA)

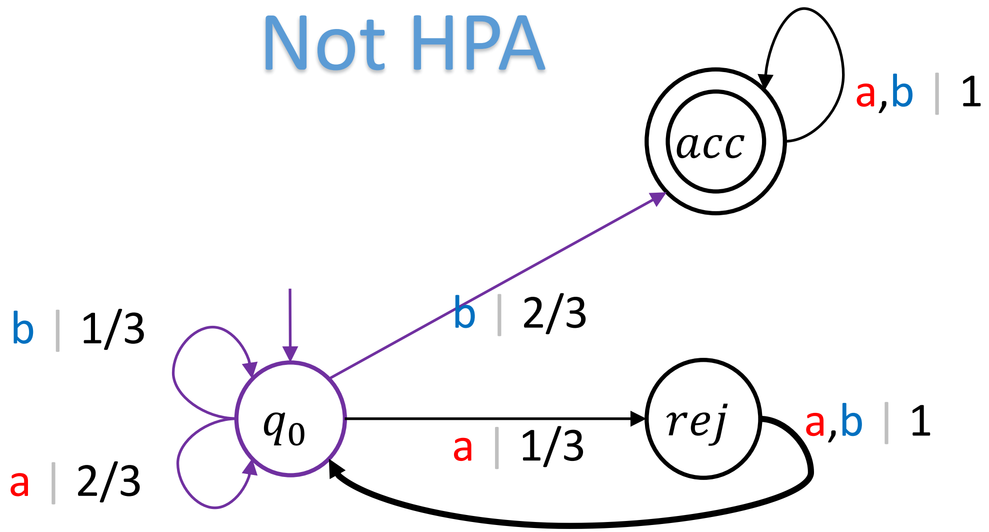
- k-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ on alphabet Σ
  - States stratified into levels $0, 1, \ldots, k$,
  - From a state at level i, on an input,
    - At most one transition goes to level i, all others go to higher levels.



Hierarchical PA (HPA)

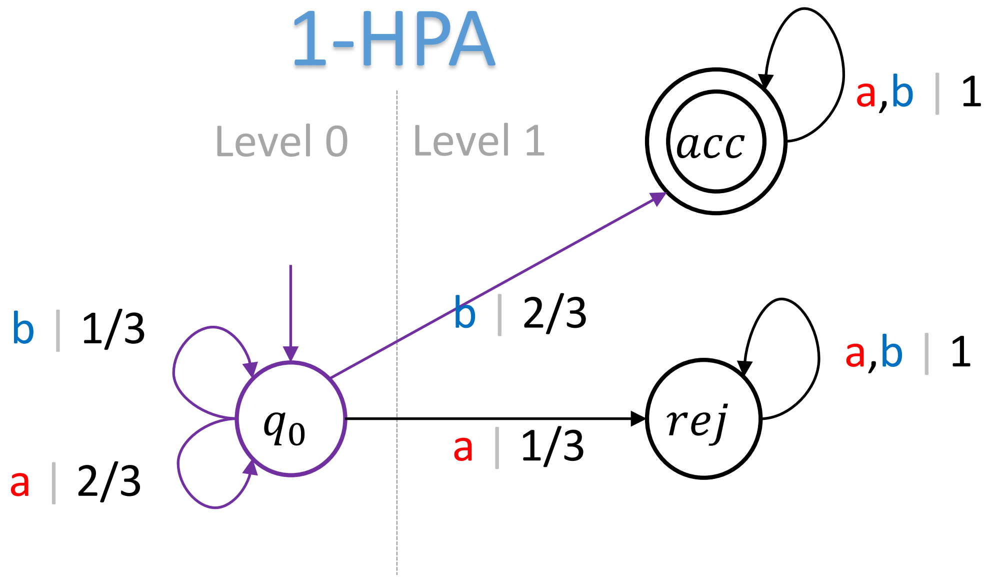
- k-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ on alphabet Σ
  - States stratified into levels $0, 1, \ldots, k$,
  - From a state at level i, on an input,
    - At most one transition goes to level i, all others go to higher levels.





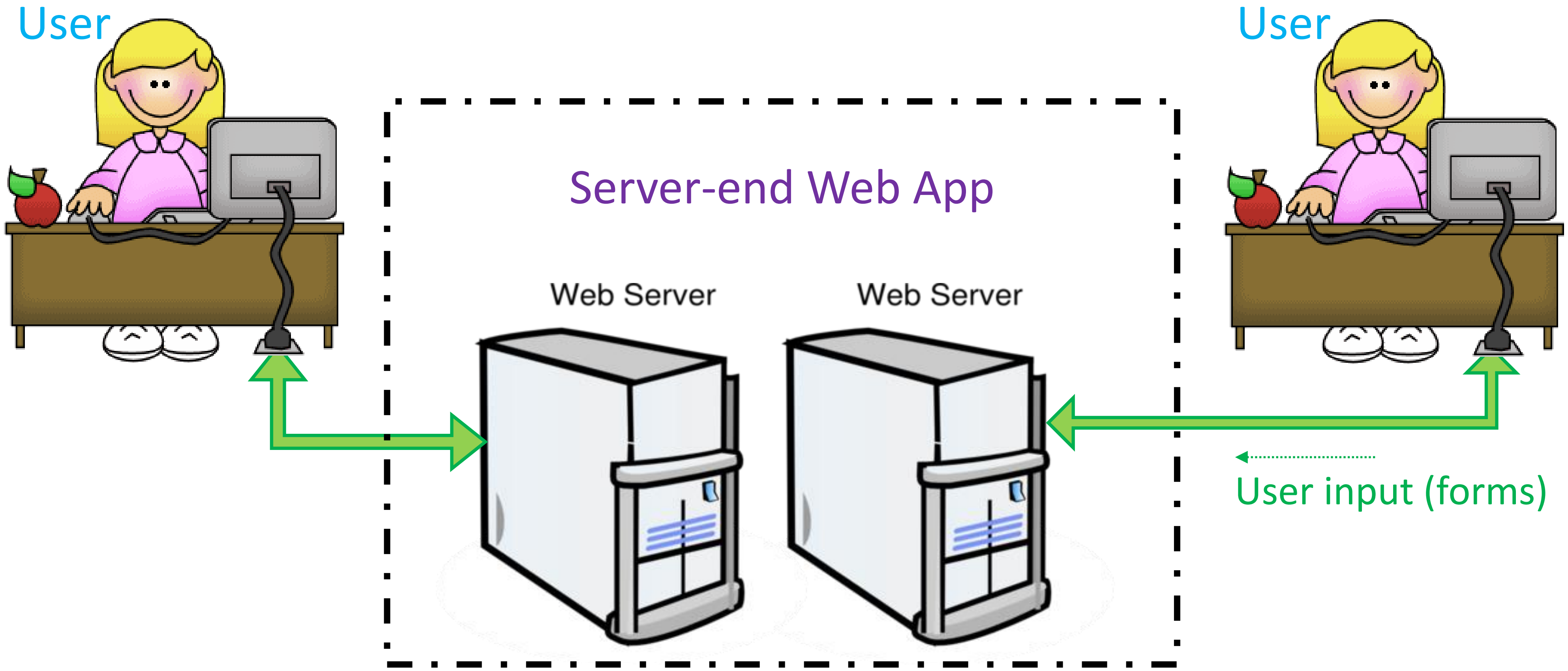
Hierarchical PA (HPA)

- k-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ on alphabet Σ
  - States stratified into levels $0, 1, \ldots, k$,
  - From a state at level i, on an input,
    - At most one transition goes to level i, all others go to higher levels.

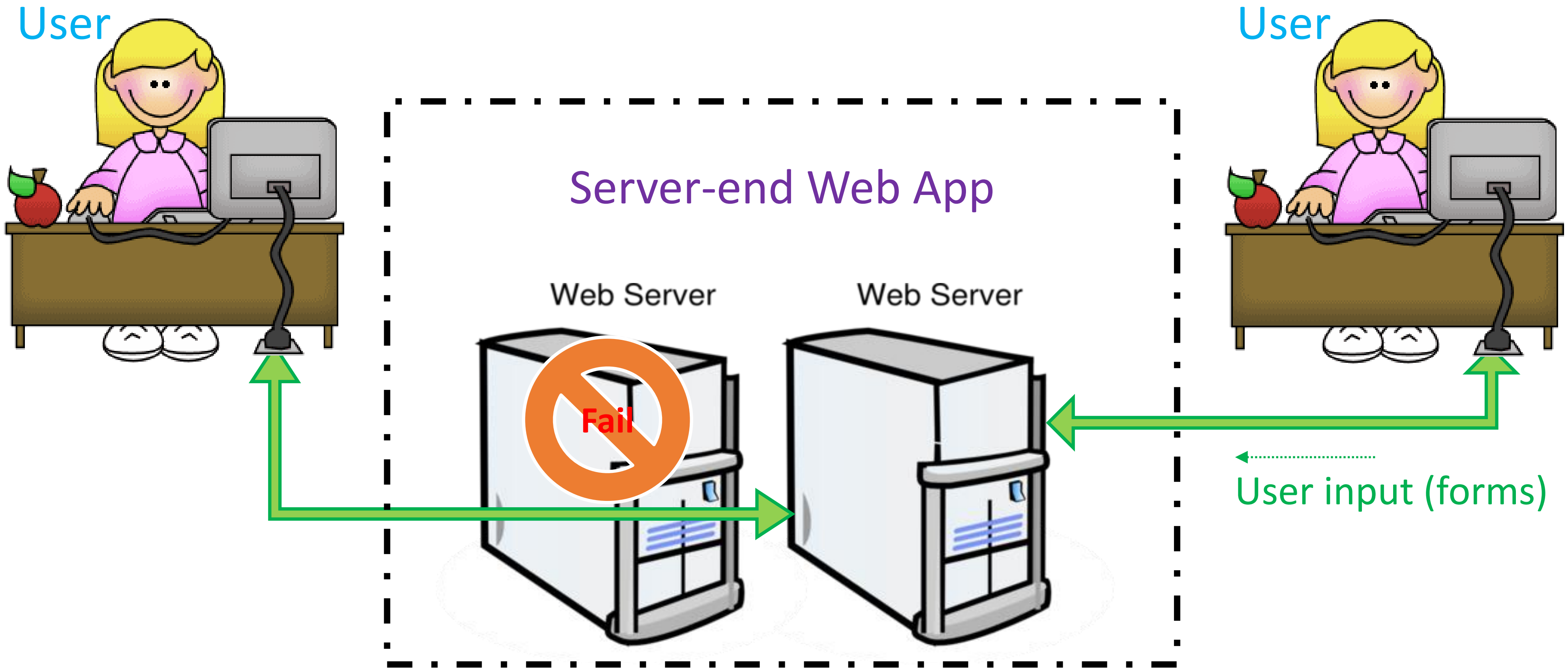


For 1-HPA, all transitions on Level 1 are deterministic with prob 1.

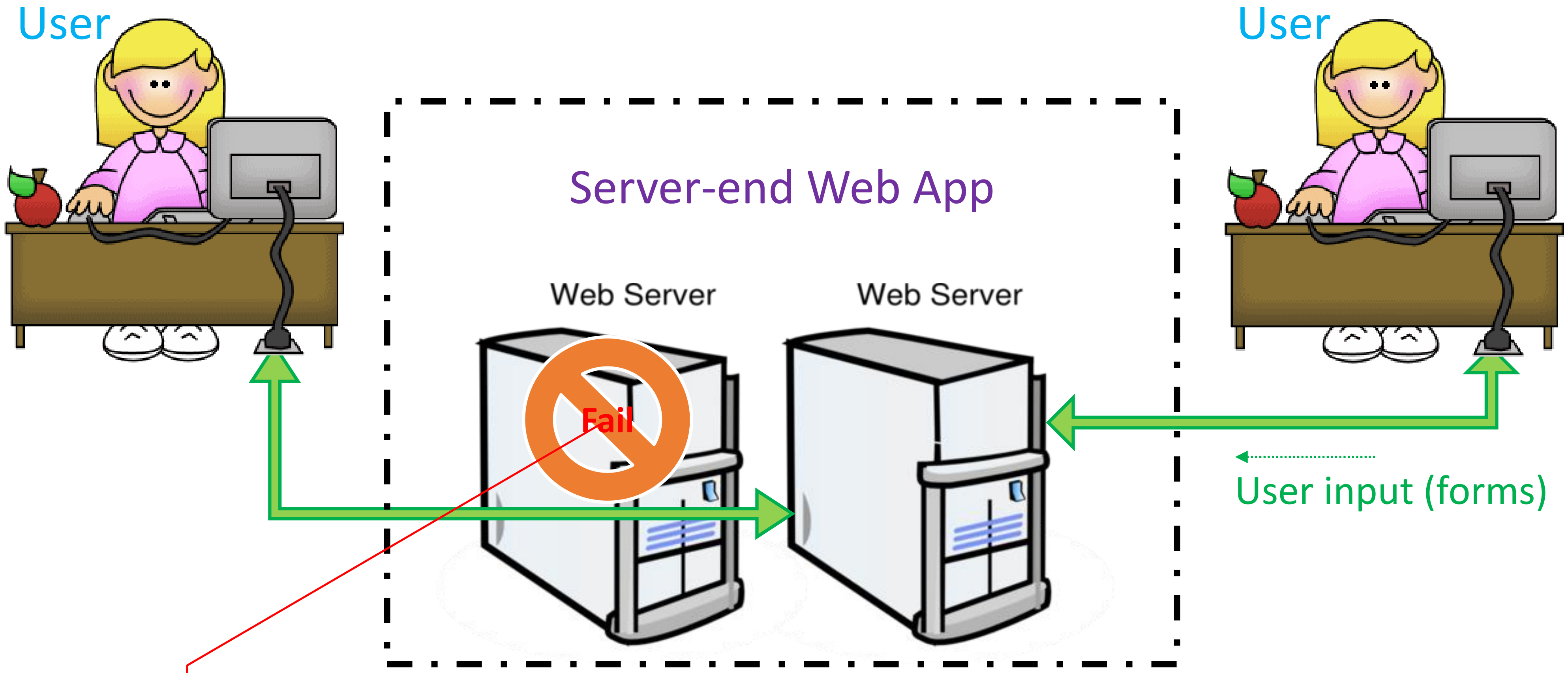
Failure-Prone Open System



Failure-Prone Open System



Failure-Prone Open System



Model Failure Probabilistically,
e.g. at state q_1 on input a, 0.4 fail, 0.6 not fail

Model Checking
Failure-Prone Open System

Decide if an open concurrent system ($g_1 \parallel g_2$)

under failure specification ($g_{1f} \parallel g_2$)

satisfies an incorrectness property (g_{1p})

with low prob $\leq x$

Model Checking
Failure-Prone Open System

Decide if an open concurrent system ($g_1 \parallel g_2$)

under failure specification ($g_{1f} \parallel g_2$)

satisfies an incorrectness property (g_{1p})

with low prob $\leq x$

$g_1 \parallel g_2$

$g_{1f} \parallel g_2$

$g_{1f} \parallel g_2 \parallel g_{1p}$
$= \boldsymbol{\mathcal{A}}$

Model Checking
Failure-Prone Open System

Decide if an open concurrent system $(g_1 \parallel g_2)$

under failure specification $(g_{1f} \parallel g_2)$

satisfies an incorrectness property (g_{1p})

with low prob $\leq x$

$\Leftrightarrow$ Check if $\boldsymbol{L}_{>\boldsymbol{x}}(\boldsymbol{\mathcal{A}}) \neq \emptyset$

$g_1 \parallel g_2$ → $g_{1f} \parallel g_2$ → $g_{1f} \parallel g_2 \parallel g_{1p} = \boldsymbol{\mathcal{A}}$

Contributions

Consider 1-HPA,

- **Expressiveness**
  - Accept non-regular languages.

Contributions

Consider 1-HPA,

- **Expressiveness**
  - Accept non-regular languages.
- **Decidability**
  - Checking if $\begin{cases} L_{>x}(\mathcal{A}) \neq \emptyset \\ L_{\geq x}(\mathcal{A}) \neq \emptyset \\ L_{>x}(\mathcal{A}) = \Sigma^* \end{cases}$ are all decidable in EXPTIME and are PSPACE-hard.
  - Results hold for PFA, PBA, and PMA.

Contributions

Consider 1-HPA,

- **Expressiveness**
  - Accept non-regular languages.
- **Decidability**
  - Checking if $\begin{cases} L_{>x}(\mathcal{A}) \neq \emptyset \\ L_{\geq x}(\mathcal{A}) \neq \emptyset \\ L_{>x}(\mathcal{A}) = \Sigma^* \end{cases}$ are all decidable in EXPTIME and are PSPACE-hard.
  - Results hold for PFA, PBA, and PMA.

1st decidability result for non-extreme thresholds.

Contributions (Cont.)

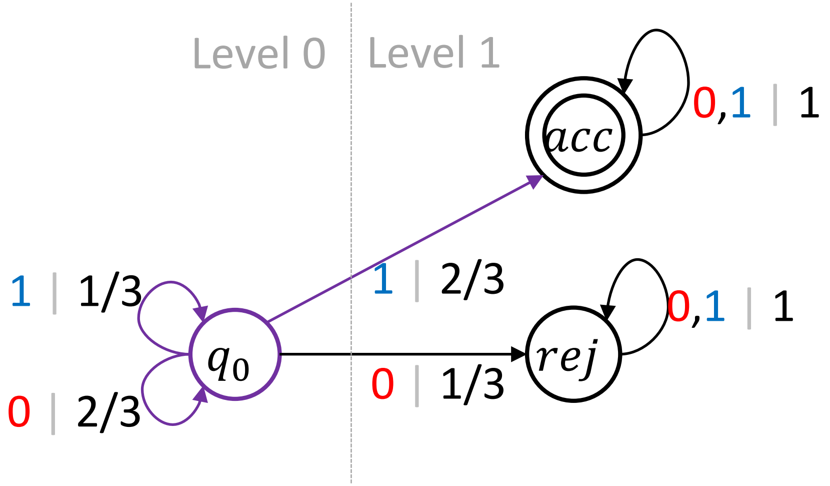
- **Decidability**
  - Algorithms for 1-HPA
    - Backward Alg
    - Forward Alg (faster in practice)
  - Tool: HPAMC – an HPA Model Checker
  - For a 2-HPA $\mathcal{A}$, checking if $L_{>x}(\mathcal{A}) \neq \emptyset$ is undecidable for $x > 0$.

Contributions (Cont.)

- **Decidability**
  - Algorithms for 1-HPA
    - Backward Alg
    - Forward Alg (faster in practice)
  - Tool: HPAMC – an HPA Model Checker
  - For a 2-HPA $\mathcal{A}$, checking if $L_{>x}(\mathcal{A}) \neq \emptyset$ is undecidable for $x > 0$.
- **Restricted Classes**
  - Integer Automata - $L_{>x}(\mathcal{A})$ is regular.

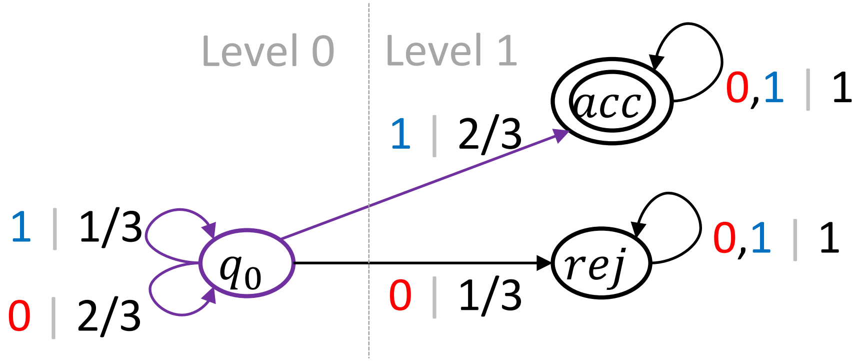
Expressiveness Results

- Consider the 1-HPA $\mathcal{A}$:



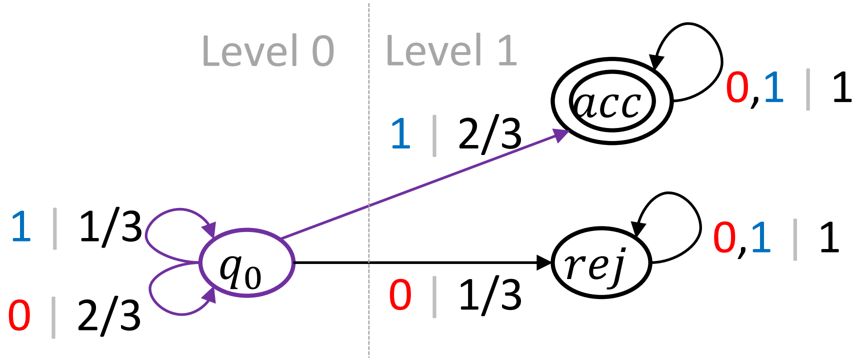
- **Theorem: $L_{>\frac{1}{2}}(\mathcal{A})$ is not regular.**

Proof



- For $u \in \{0,1\}^*$, define $Val(u) = \dfrac{\frac{1}{2} - \delta_u(q_0, Acc)}{\delta_u(q_0, q_0)}$,
  - $Val(u)$ denotes the % of the prob remaining at level 0 that needs to move to Acc to reach $\frac{1}{2}$.
  - $Val(u0) = \frac{3}{2} Val(u); Val(u1) = 3Val(u) - 2.$

Proof

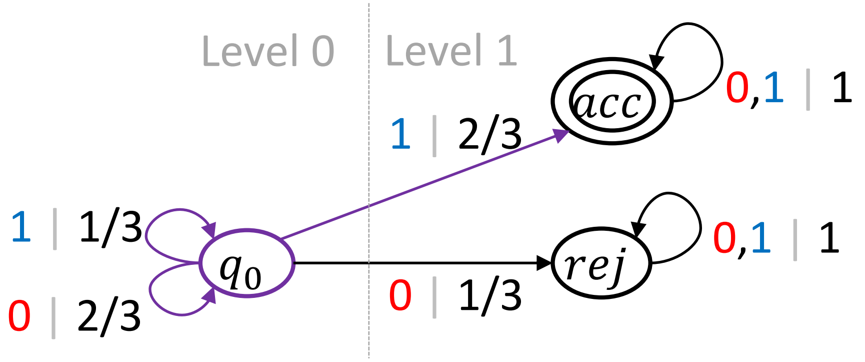


- For $u \in \{0,1\}^*$, define $Val(u) = \dfrac{\frac{1}{2} - \delta_u(q_0, Acc)}{\delta_u(q_0, q_0)}$,
  - $Val(u)$ denotes the % of the prob remaining at level 0 that needs to move to Acc to reach $\frac{1}{2}$.
  - $Val(u0) = \frac{3}{2} Val(u); Val(u1) = 3Val(u) - 2$.

- $v \in \{0,1\}^* \cup \{0,1\}^\omega$ represents a number $bin(v) \in [0,1]$.

  e.g. $bin(011) = 0 + \frac{1}{2^2} + \frac{1}{2^3}$.

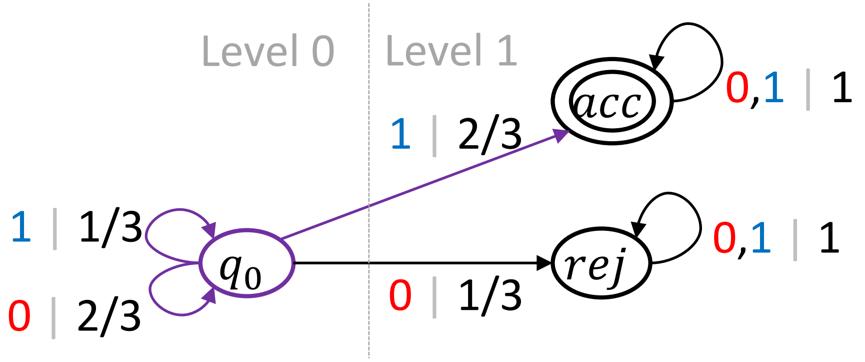
Proof (Cont.)



- Claim: $\beta \in \{0,1\}^{\omega}$ is ultimately periodic $\Rightarrow$

  $\mathbf{X}_{\boldsymbol{\beta}} = \{\boldsymbol{Val}(\boldsymbol{\beta}')|\boldsymbol{\beta}'\,\textbf{is a prefix of}\,\boldsymbol{\beta}\}$ is finite.
- X_β is infinite $\Rightarrow$ β is aperiodic $\Rightarrow$ $bin(\beta)$ is irrational.

Proof (Cont.)



- Claim: $\beta \in \{0,1\}^\omega$ is ultimately periodic $\Rightarrow$

  $\mathbf{X}_{\boldsymbol{\beta}} = \{\boldsymbol{Val(\beta')}|\boldsymbol{\beta'}\textbf{is a prefix of } \boldsymbol{\beta}\}$ is finite.

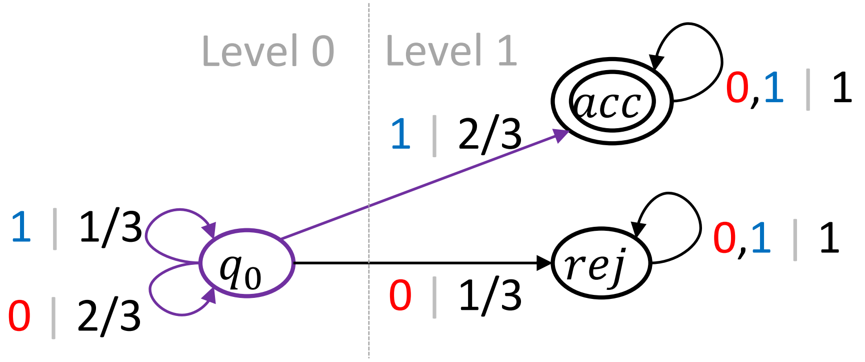
- X_β is infinite $\Rightarrow$ β is aperiodic $\Rightarrow$ $bin(\beta)$ is irrational.

- We exhibit a β s.t. X_β is infinite,

  s.t. $L_{>\frac{1}{2}}(\mathcal{A}) = \{u \in \{0,1\}^* | bin(u) > bin(\beta)\}$.

- Using Rabin's result, $L_{>\frac{1}{2}}(\mathcal{A})$ is non-regular.

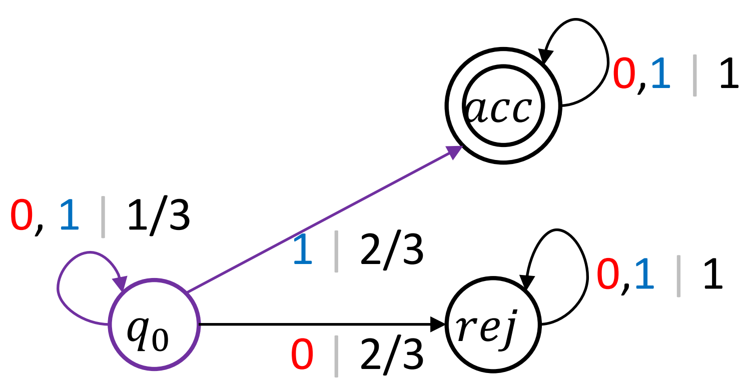
Proof (Cont.)

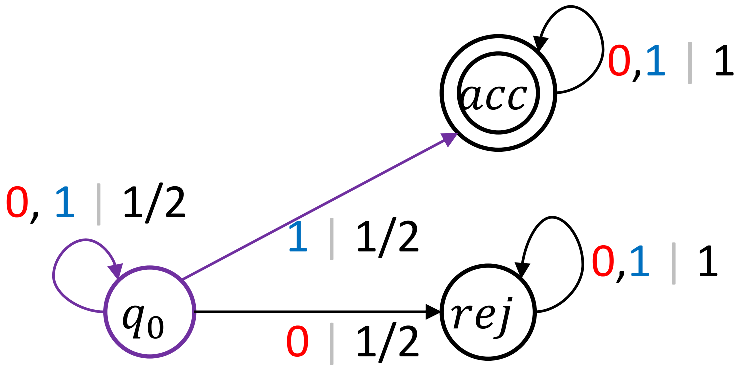


- $\beta = \lim_{i\to\infty} \beta_i$, and
- β_i is constructed as below:
  - β_0 is the empty string, thus $Val(\beta_0) = \frac{1}{2}$;
  - $\beta_{i+1} = \begin{cases} \beta_i 0 & if\ Val(\beta_i) < \frac{2}{3} \\ \beta_i 1 & else \end{cases}$.

Integer Automata (IA)

- A 1-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ on Σ is an IA, if
  - $\forall q \in Q_0, a \in \Sigma, r \in \mathrm{Q}_1$, $\delta_a(q, r)$ is an integer multiple of $\delta_a(q, Q_0)$.

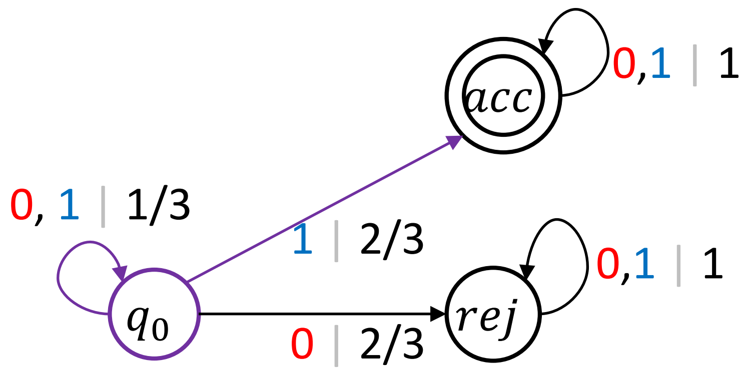


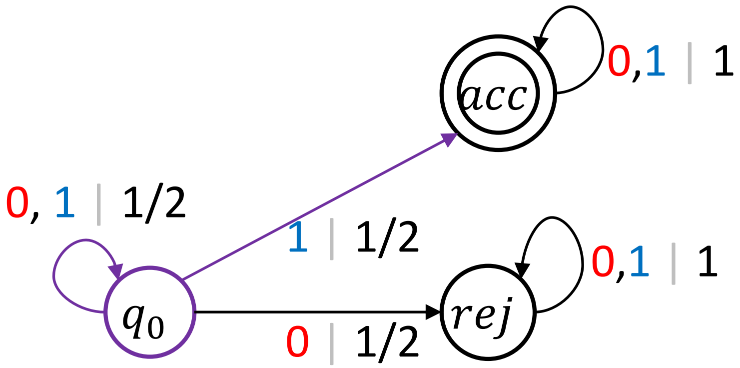


Rabin's example

Integer Automata (IA)

- A 1-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ on Σ is an IA, if
  - $\forall q \in Q_0, a \in \Sigma, r \in \mathrm{Q}_1$, $\delta_a(q, r)$ is an integer multiple of $\delta_a(q, Q_0)$.





Rabin's example

- **Theorem:**

  $L_{>x}(\mathcal{A})$ is regular for IA $\mathcal{A}$ and rational x.

Decidability Results for 1-HPA

Given 1-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ and Σ,

- let $|Q| = n$, $Q = Q_0 \cup Q_1$, where Q_0 and Q_1 denote level 0 and 1 states.
- A **witness set** W is a subset of Q with at most one state in Q_0.
  - q_W denotes the Q_0 state of W if exists.
  - W is "good" ~ if $\exists v \in \Sigma^*$ s.t. v is accepted with prob 1 from all $q \in W \cap Q_1$.

Decidability Results for 1-HPA

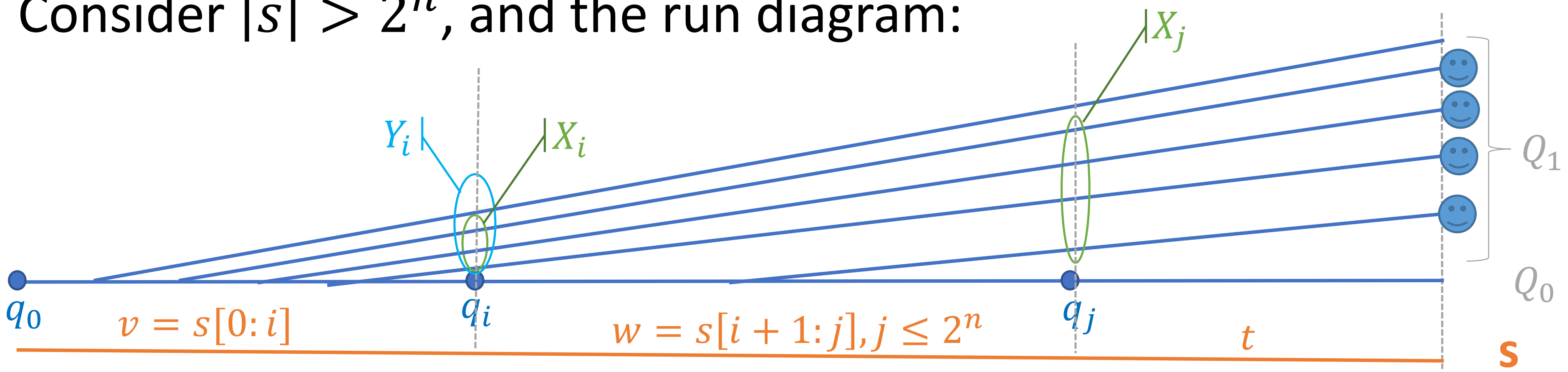
Given 1-HPA $\mathcal{A} = (Q, q_0, \delta, Acc)$ and Σ,

- let $|Q| = n$, $Q = Q_0 \cup Q_1$, where Q_0 and Q_1 denote level 0 and 1 states.
- A **witness set** W is a subset of Q with at most one state in Q_0.
  - q_W denotes the Q_0 state of W if exists.
  - W is "good" ~ if $\exists v \in \Sigma^*$ s.t. v is accepted with prob 1 from all $q \in W \cap Q_1$.
- **Theorem: $L_{>x}(\mathcal{A}) \neq \emptyset$ iff**

  $\exists u, |u| \leq 4rn8^n$ and a good witness set $H, \delta_u(q_0, H) > x$.

Proof (⇒)

- Assume $L_{>x}(\mathcal{A}) \neq \emptyset$, s is the shortest string in it.
- Consider $|s| > 2^n$, and the run diagram:



- $s = vwt$ as shown above, and $X_i = X_j$ and $q_i = q_j$.
- $\dfrac{\delta_w\,(q_i, X_j)}{\delta_w(q_i, Q_1)} > \delta_t(q_j, Acc)$.
- By repeating w sufficient number of times (m), we get desired $u = vw^m$.

Decidability Results (Cont.)

- **Theorem: Determining $L_{>x}(\mathcal{A}) \neq \emptyset$ is in EXPTIME.**
- Proof : Use last theorem and dynamic programming approaches.
  - Backward Algorithm (simpler)
  - Forward Algorithm (faster)

Backward Algorithm

- For $i = 1, \ldots, 4rn8^n$ and for each good witness set W,
- $\mathbf{Prob(W, i)}$ - the max prob of getting accepted from q_W using an input of length at most i.

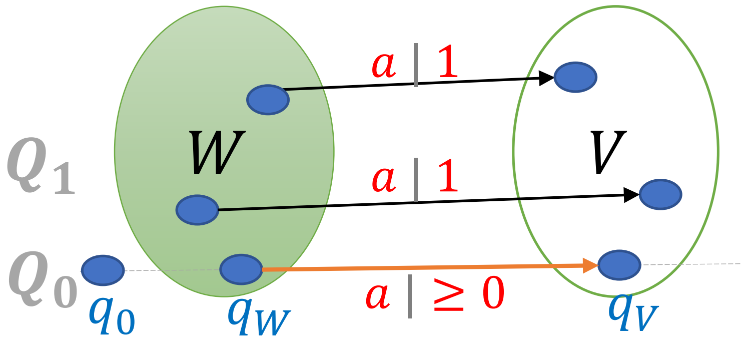
$$Prob(\mathrm{W}, 1) = \max\{\delta_a(q_{\mathrm{W}}, Acc)\};$$

$$Prob(\mathrm{W}, i+1) = \max\{$$

$$\{Prob(\mathrm{W}, i)\},$$

$$\{\delta_a(q_{\mathrm{W}}, q_V) \times Prob(V, i) + \delta_a(q_{\mathrm{W}}, V \cap Q_1) \mid$$

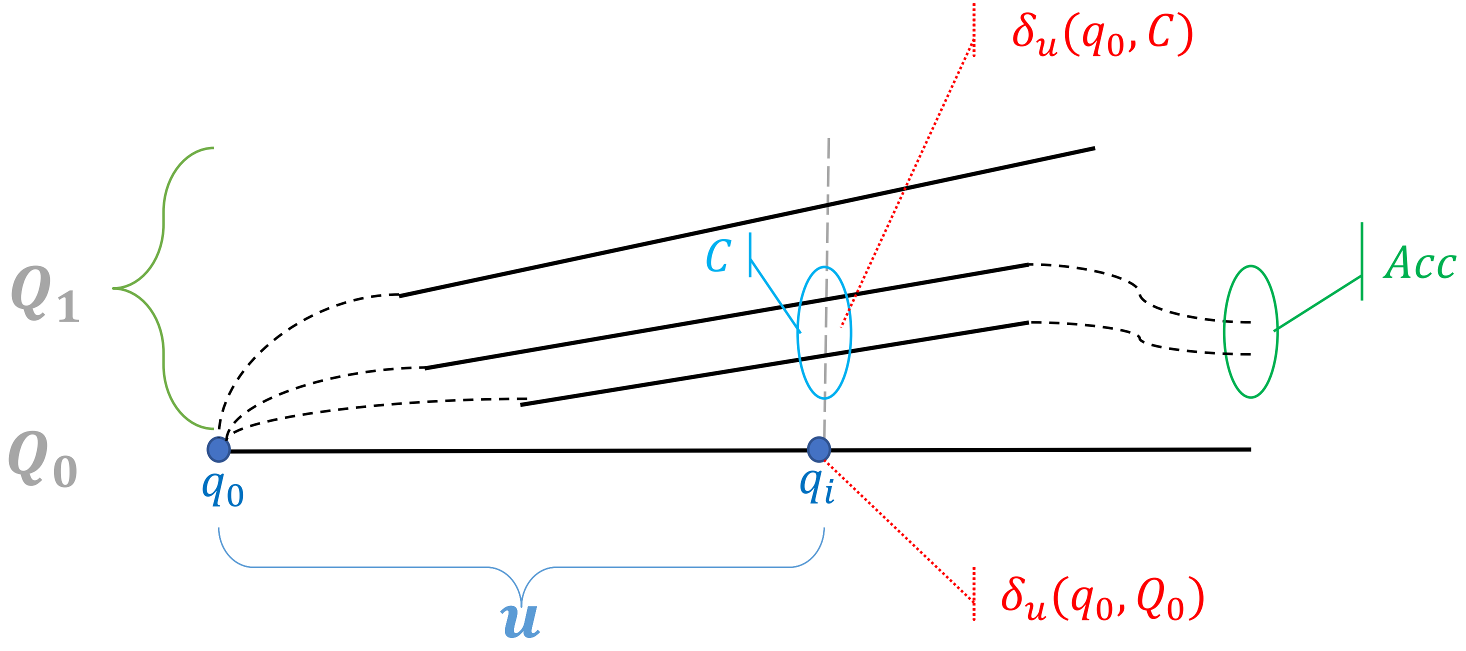
$$post(W \cap Q_1, a) \subseteq V\}\}.$$



- **Check if** $\boldsymbol{Prob(\{q_0\}, i) > x}$**.**

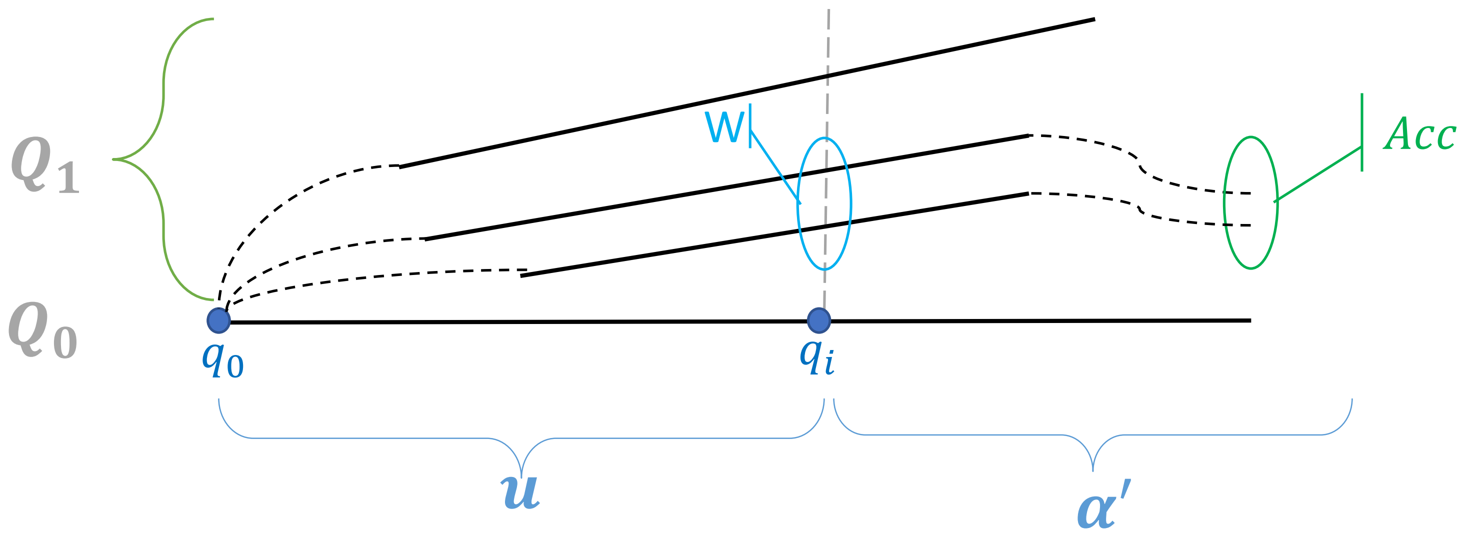
Forward Algorithm

- For $C \subseteq Q_1$ of HPA $\mathcal{A}$, $x \in (0,1)$, $u \in \Sigma^*$,
- $\mathbf{val(C, x, u)} = \frac{x - \delta_u(q_0, C)}{\delta_u(q_0, Q_0)}$ is the fraction of $\delta_u(q_0, Q_0)$ needed in C to exceed x.



Forward Algorithm

- Word α is accepted with prob$>x$, iff $\alpha = u\alpha'$ and $\exists$ a good witness set W s.t. $\boldsymbol{val(W,x,u) < 0}$.



Forward Algorithm (cont.)

- For each good witness set W and $i \geq 0$, $\mathbf{min}\boldsymbol{val}(\boldsymbol{W}, \boldsymbol{i})$ is
- the min of val over all strings of length at most i;
- $\mathrm{min}val(W, i) = \min\{val(W \cap \ Q_1, x, u) \mid |u| \leq i\}$.

Forward Algorithm (cont.)

- For each good witness set W and $i \geq 0$, $\mathbf{min}\boldsymbol{val}(\boldsymbol{W}, \boldsymbol{i})$ is
- the min of val over all strings of length at most i;
- $\min val(W, i) = \min\{val(W \cap \ Q_1, x, u) \mid |u| \leq i\}$.

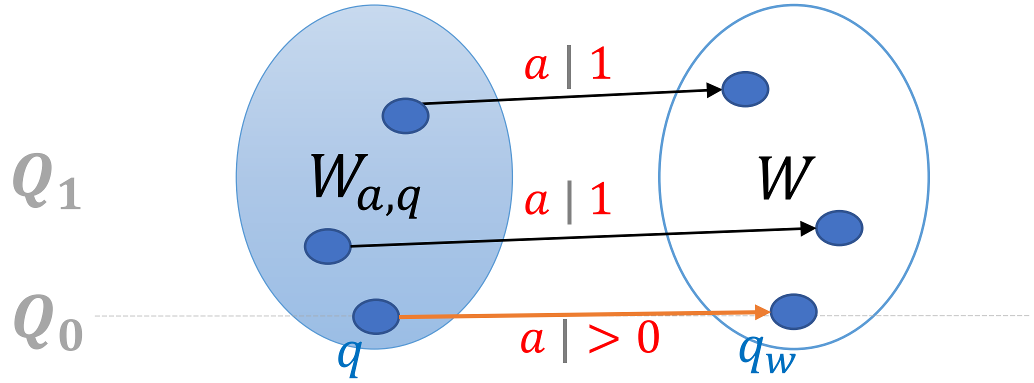
- For increasing i and each W, compute $\mathbf{min}\boldsymbol{val}(\boldsymbol{W}, \boldsymbol{i})$ incrementally.
- $\min val(W, 0) = \begin{cases} x & if\ q_0 \in W \\ +\infty & else \end{cases}$

Forward Algorithm (cont..)

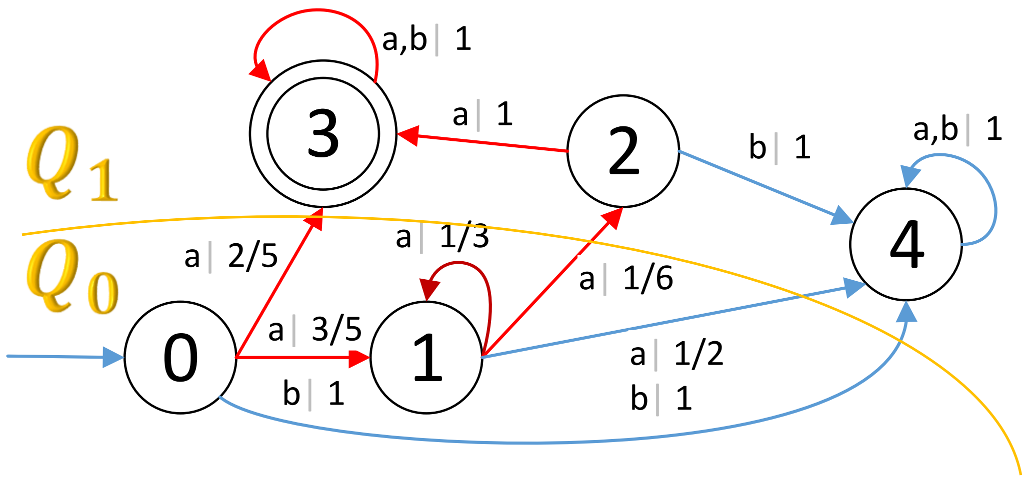
For $i > 0, a \in \Sigma, q \in Q_0,$

$$\boldsymbol{W}_{\mathbf{a},\boldsymbol{q}} = (\mathrm{pre}(W, a) \cap Q_1) \cup \{q\} \text{ for } W.$$

$$\boldsymbol{\mathrm{min}val}(\boldsymbol{W}, \boldsymbol{i}) = \min\left\{\mathrm{min}val(W, i-1), \left\{\frac{\mathrm{min}val\left(W_{a,q}, i-1\right) - \delta_a\,(q, W \cap\ Q_1)}{\delta_a\,(q, q_W)}\,\middle|\,\delta_a(q, q_W) > 0\right\}\right\}$$



Forward Example



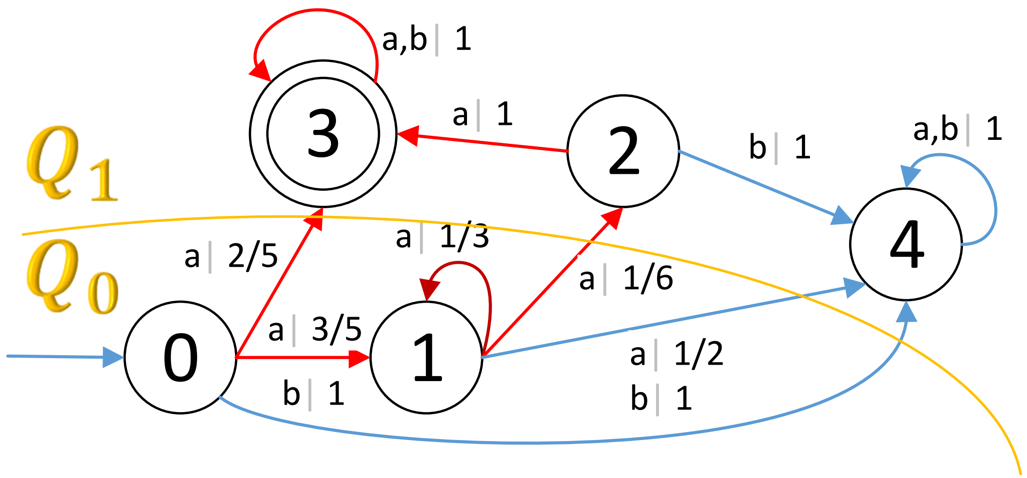
X=1/2

$W_{a,0}$ $W_{a,1}$

Witness Sets:	{1,3}	{0,2,3}	{1,2,3}
$i = 0$	$+\infty$	½	$+\infty$
$i = 1$	1/6	½	1/6
$i = 2$	1/6	½	0
$i = 3$	0	½	-1/2

$W_{a,0}$ $W_{a,1}$

Forward Example



X=1/2

Witness Sets:	∅	{3}	{2,3}	{0}	{1}	{0,3}	{1,3}	{0,2,3}	{1,2,3}
$i = 0$	+∞	+∞	+∞	½	+∞	½	+∞	½	+∞
$i = 1$	½	½	½	½	5/6	½	1/6	½	1/6
$i = 2$	½	½	½	½	5/6	½	1/6	½	0
$i = 3$	½	½	½	½	5/6	½	0	½	-1/2

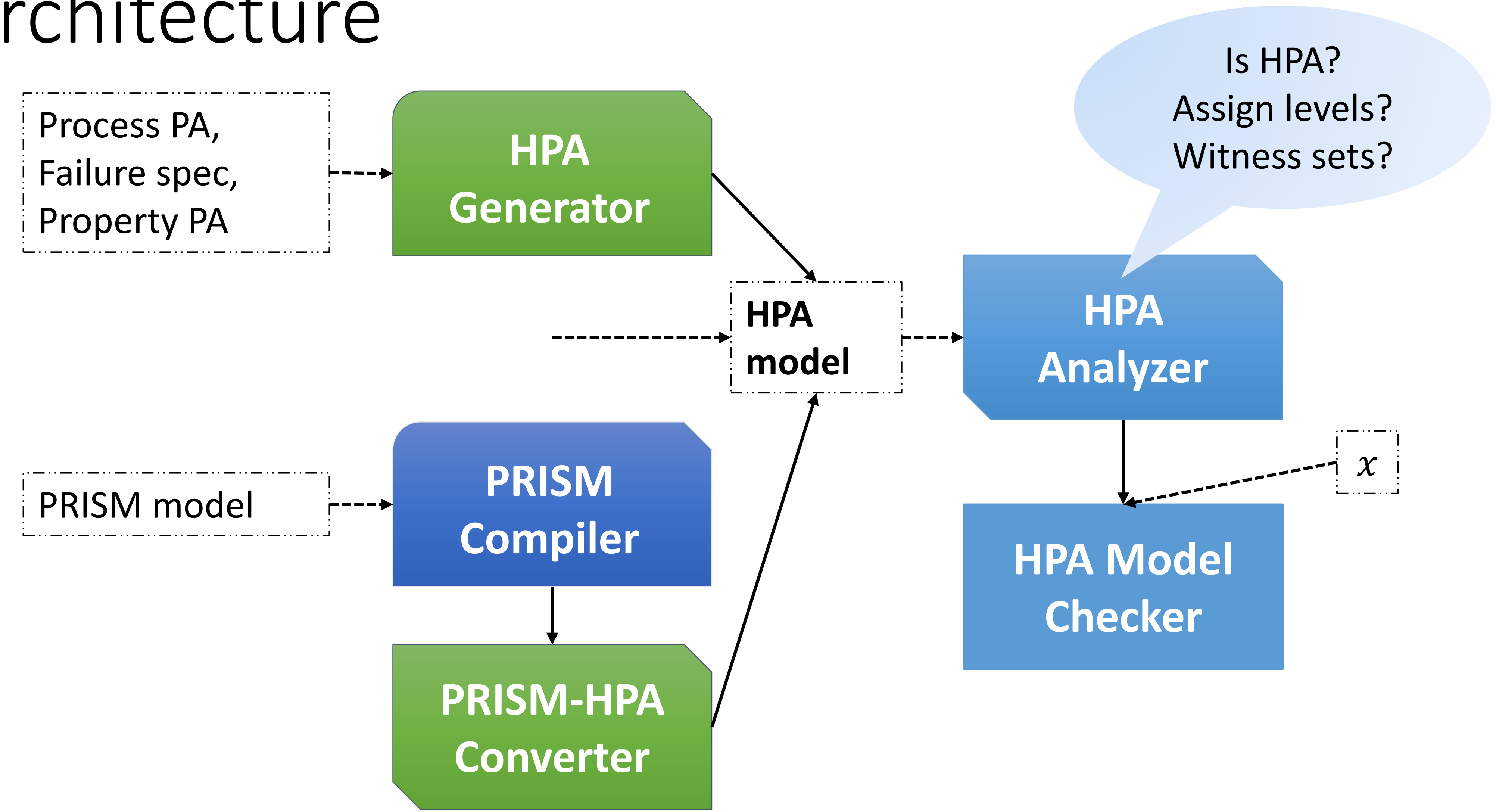
FWD vs. BKD

- Let $n =$|states|, $m =$|transitions|, w is # of good witness sets, and $s =|\Sigma|$, L_f and L_b are the # of iterations before FWD and BKD algs terminate respectively.
- Complexity
  - FWD - $\mathbf{O}\left((\mathbf{mn} + \boldsymbol{L_f m^2})\mathbf{w}\right)$ vs. BKD - $\mathbf{O}((\boldsymbol{L_b}\mathbf{m} + \mathbf{m} + \boldsymbol{n})\boldsymbol{s}\boldsymbol{w}^{\mathbf{2}})$
  - $L_f \leq w = O(2^n)$, $L_b \leq 4rn8^n$.
  - FWD is faster than BKD in practice.

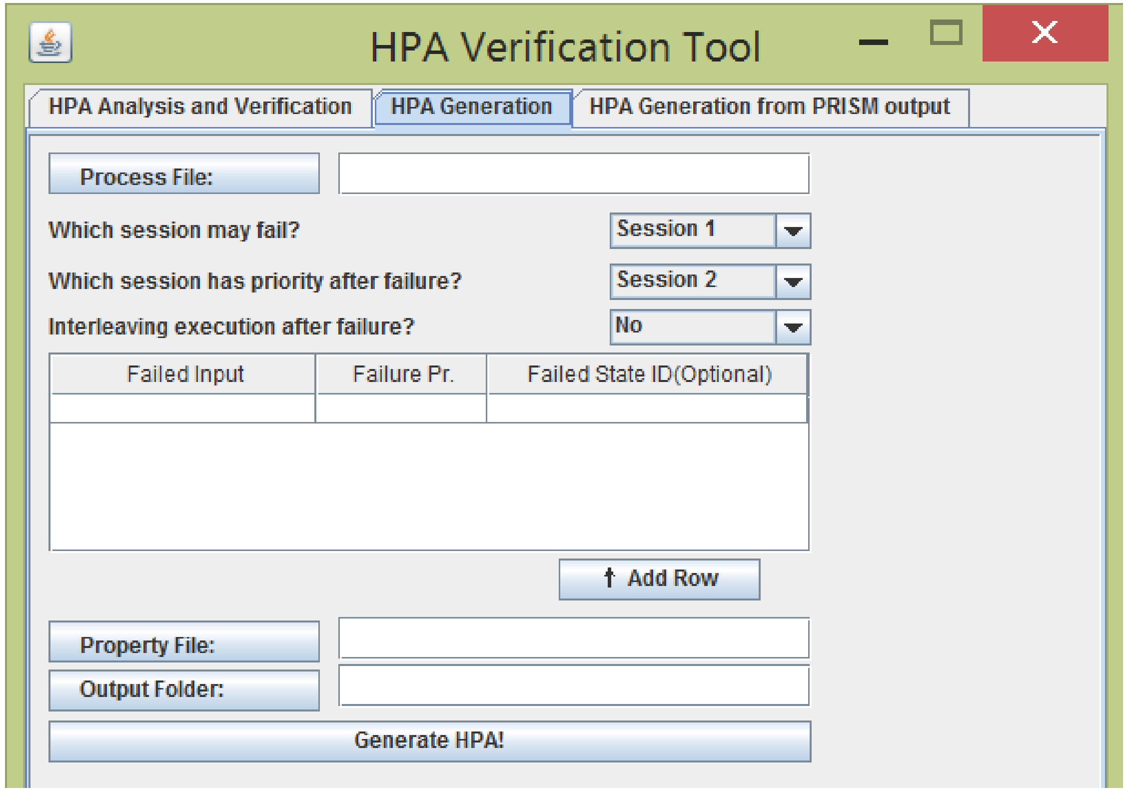
FWD vs. BKD

Web Application	n	m	s	w	**Result**	**(CPU Time in ms)**			L_f	L_b
						Forward	**Backward**	**BKD/FWD**		
eBay Auction	19	248	13	60	empty	16	140	**8.75**	4	2
					non-empty	0	125	**N/A**	2	2
On-line Shopping 1	86	1472	17	342	empty	78	4009	**51.40**	16	15
					non-empty	47	4165	**88.62**	13	14
On-line Shopping 2	80	1365	17	341	empty	47	4571	**97.26**	15	15
					non-empty	31	4228	**136.39**	13	14
On-line Shopping 3	87	1489	17	2051	empty	328	338273	**1031.32**	14	10
					non-empty	140	321362	**2295.44**	5	8
Medium HPA	191	4209	22	3402	empty	391	491719	**1257.59**	8	10
					non-empty	172	502984	**2924.33**	5	8
Larger HPA	399	797	12	3874	empty	47	210734	**4483.70**	1	5
					non-empty	156	221641	**1420.78**	2	7

HPAMC: an HPA Model Checker Architecture

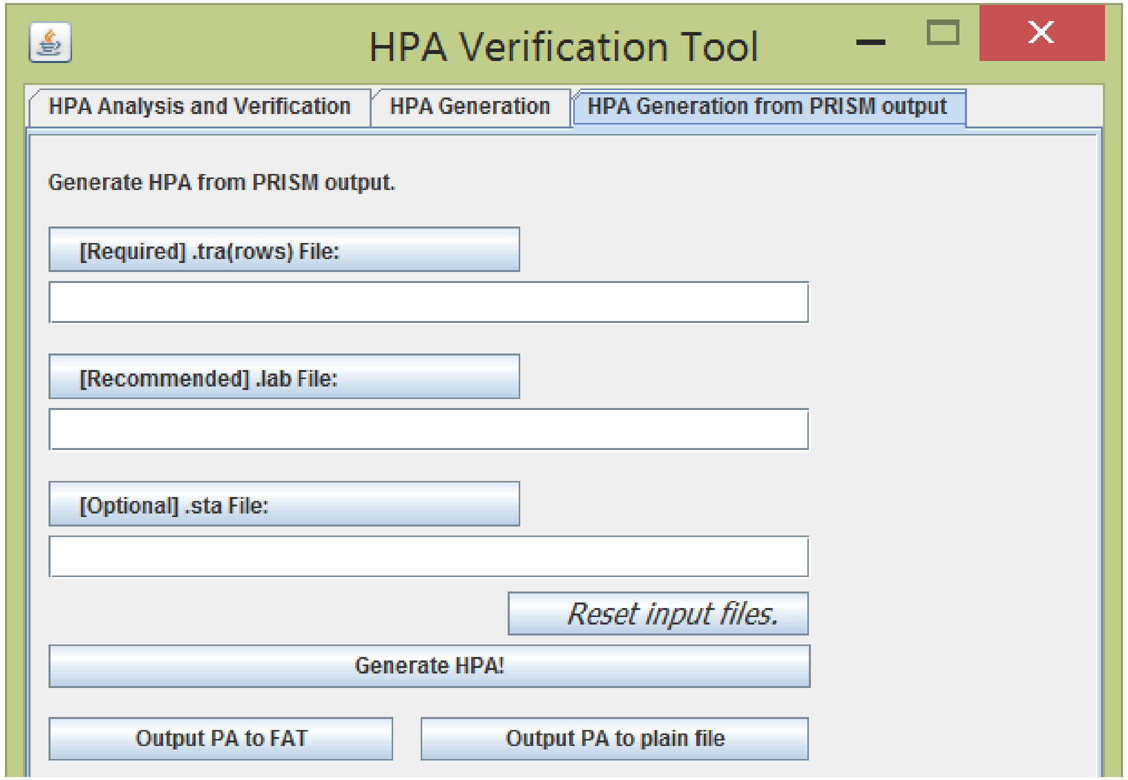


HPAMC: an HPA Model Checker



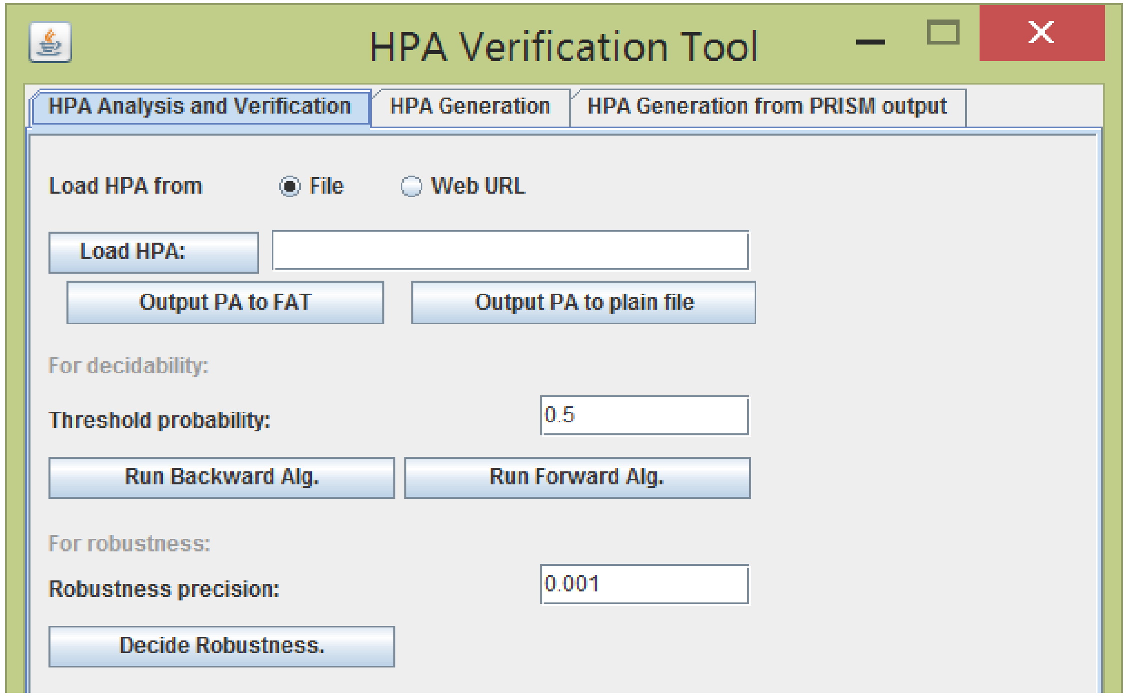
- HPA Generation
- Features
  - PA model specification.
  - PA model abstraction by synthesizing system, failure specification, and incorrectness property.

HPAMC: an HPA Model Checker



- HPA Generation from PRISM
- Features
  - Compatible with popular model checkers like PRISM.
  - Obtain PA from MDP.

HPAMC: an HPA Model Checker

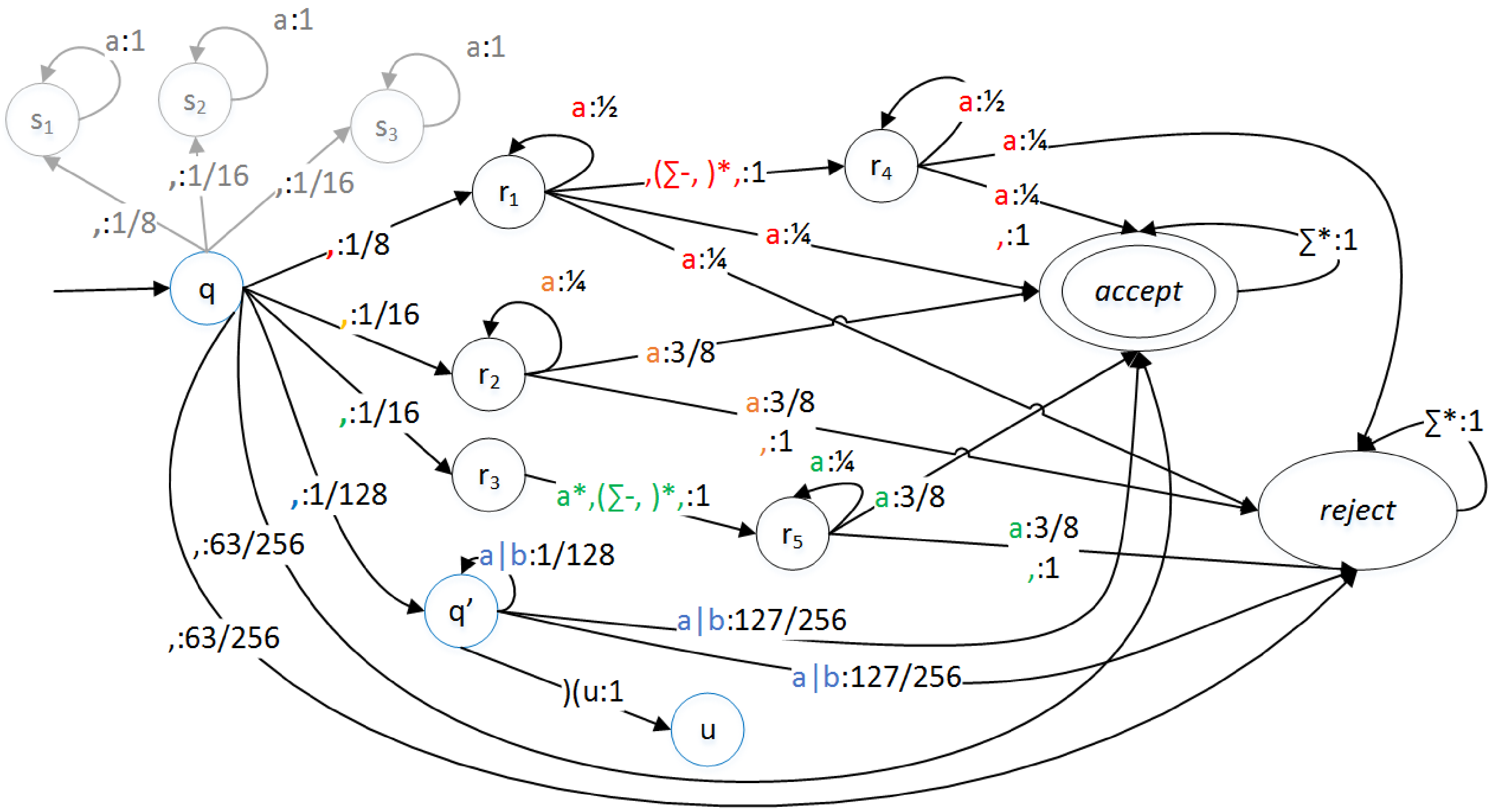


- HPA Analysis and Verification
- Features
  - HPA-based verification of PA models.
  - Targeting decidability and robustness problems.

Undecidability Results for 2-HPA

- **Theorem: Given a 2-HPA $\mathcal{A}$, a rational threshold $x \in [0, 1]$, the problem of determining if $L_{>x}(\mathcal{A}) \neq \emptyset$ is undecidable.**
- Proof: By construction.
  - Reduce the halting problem for counter automata.

Undecidability Results for 2-HPA



Conclusions

- For the 1st time, we identified a decidable and expressive class of PA.
  - The results hold for PFA, PBA, and PMA; also for $>, \geq$.
- Model checker for open concurrent probabilistic systems - HPAMC.
- Problem of checking $L_{>\frac{1}{2}}(\mathcal{A}) \neq \emptyset$ is PSPACE-hard.
- Problem of checking $L_{>x}(\mathcal{A}) \neq \emptyset$ is undecidable for 2-HPA.

Future Work

- Identify subclasses of PA whose non-emptiness can be checked in poly-time.
- Extend to infinite acceptance and liveness property.
- Support temporal logic property.
- Tool refinement.

Thanks! Questions?

Automata Theoretic Approach for Model Checking Open Probabilistic Systems

A. Prasad Sistla

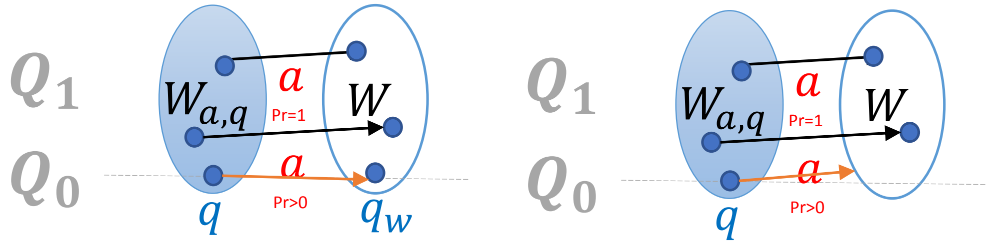
Joint work with Yue Ben, Rohit Chadha, Mahesh Viswanathan

Decidability Results (Cont.)

- **Theorem: Determining $L_{>x}(\mathcal{A}) \neq \emptyset$ is in EXPTIME.**
- Proof : Use last theorem and a dynamic programming approach.
  - Let $\mathcal{X}$ be the set of witness sets U such that $U \cap Q_0 \neq \emptyset$ and $U \cap Q_1$ is a good set, $\mathcal{Y}$ be the set of good witness sets; $\mathcal{Y} \subseteq \mathcal{X}$.
  - $Prob(U, 1) = \max\{\delta_a(q_U, W) | a \in \Sigma, W \in \mathcal{Y}, post(U \cap Q_1, a) \subseteq W\};$
  - $Prob(U, i+1) = \max\left(\begin{array}{c}\{Prob(U,i)\} \cup \\ \{\delta_a(q_U, q_V)Prob(V,i) + \delta_a(q_U, V \cap Q_1) | \\ a \in \Sigma, V \in \mathcal{X}, post(U \cap Q_1, a) \subseteq V\}\end{array}\right).$
- Check if $Prob(\{q_0\}, .) > x$.

-> Backward Alg

Forward Algorithm (cont..)



- For $i > 0, a \in \Sigma, q \in Q_0, \boldsymbol{W}_{\mathbf{a},\boldsymbol{q}} = (\mathrm{pre}(W, a) \cap Q_1) \cup \{q\}$ for each W.

- If $W \cap Q_0 \neq \emptyset, \boldsymbol{\mathrm{m}val(W, i)} = \min\{\mathrm{m}val(W, i-1), \{\frac{\mathrm{m}val(W_{a,q}, i-1) - \delta_a(q, W \cap Q_1)}{\delta_a(q, q_W)} | \delta_a(q, q_W) > 0\}\}$.

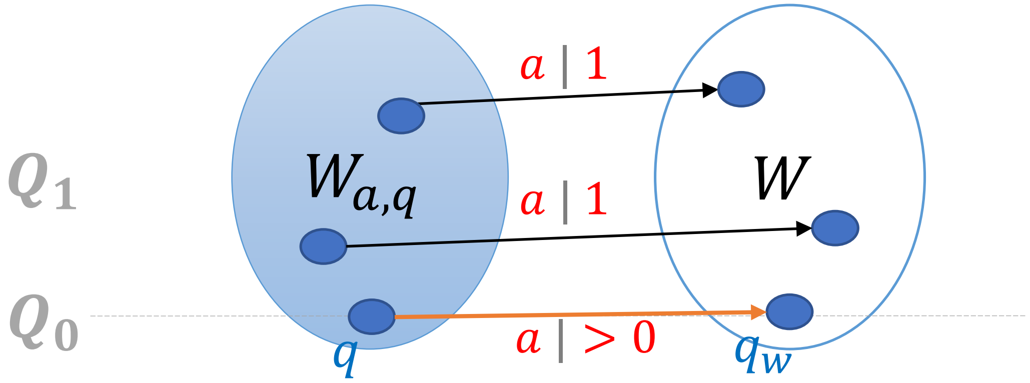
- If $W \cap Q_0 = \emptyset, \boldsymbol{\mathrm{m}val(W, i)} =$ $\begin{cases} -\infty & \text{if } \exists W_{a,q} \text{ of } W \text{ s.t. } \mathrm{m}val\left(W_{a,q}, i-1\right) < \delta_a(q, W \cap Q_1) \\ \mathrm{m}val(W, i-1) & else \end{cases}$

Forward Algorithm (cont..)

For $i > 0, a \in \Sigma, q \in Q_0$, $\boldsymbol{W_{\mathbf{a},q}} = (\mathrm{pre}(W, a) \cap Q_1) \cup \{q\}$ for each W.

If $W \cap Q_0 \neq \emptyset$, $\mathbf{mval}(\boldsymbol{W}, \boldsymbol{i}) = \min\{\mathrm{m}val(W, i-1),$

$$\left\{ \frac{\mathrm{m}val\left(W_{a,q}, i-1\right) - \delta_a\left(q, W \cap Q_1\right)}{\delta_a\left(q, q_W\right)} \middle| \delta_a(q, q_W) > 0 \right\}\}$$



Forward Algorithm (cont…)

For $i > 0, a \in \Sigma, q \in Q_0$, $\boldsymbol{W_{\mathbf{a},q}} = (\mathrm{pre}(W,a) \cap Q_1) \cup \{q\}$ for each W.

If $W \cap Q_0 = \emptyset$, $\boldsymbol{\mathrm{m}val(W,i)} =$

$$\begin{cases} -\infty & \text{if } \exists W_{a,q} \text{ of } W \text{ s.t. } \mathrm{m}val\left(W_{a,q}, i-1\right) < \delta_a(q, W \cap Q_1) \\ \mathrm{m}val(W, i-1) & else \end{cases}$$