Resolving Nondeterminism by Chance

Soumyajit Paul

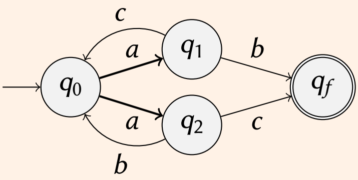
IARCS Verification Seminar Series
16 Sep, 2025

Joint work with
David Purser, Sven Schewe, Qiyi Tang, Patrick Totze, Di-de Yen

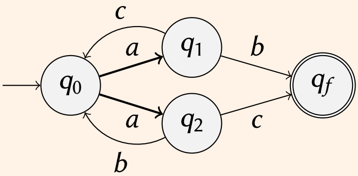
UNIVERSITY OF LIVERPOOL

Resolving Nondeterminism

Resolving Nondeterminism

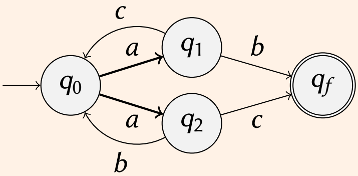


Resolving Nondeterminism



Resolver : strategy to choose next transition based on history

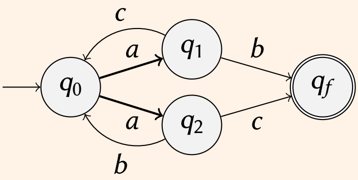
Resolving Nondeterminism



Resolver : strategy to choose next transition based on history

$$\mathcal{R} : \Sigma^* \times Q \times \Sigma \mapsto Q$$

Resolving Nondeterminism

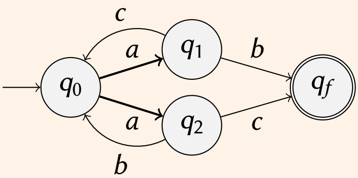


Resolver : strategy to choose next transition based on history

$$\mathcal{R} : \Sigma^* \times Q \times \Sigma \mapsto Q$$

$\mathcal{R}(\epsilon, q_0, a) = q_1 \mid \mathcal{R}(\Sigma^* c, q_0, a) = q_2 \mid \mathcal{R}(\Sigma^* b, q_0, a) = q_1$

Resolving Nondeterminism



Resolver : strategy to choose next transition based on history

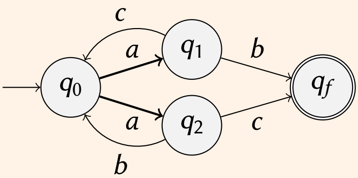
$$\mathcal{R} : \Sigma^* \times Q \times \Sigma \mapsto Q$$

$\mathcal{R}(\epsilon, q_0, a) = q_1 \mid \mathcal{R}(\Sigma^* c, q_0, a) = q_2 \mid \mathcal{R}(\Sigma^* b, q_0, a) = q_1$

Run on *acabab*

$$q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_0 \xrightarrow{a} q_2 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_f$$

Resolving Nondeterminism



Resolver : strategy to choose next transition based on history

$$\mathcal{R} : \Sigma^* \times Q \times \Sigma \mapsto Q$$

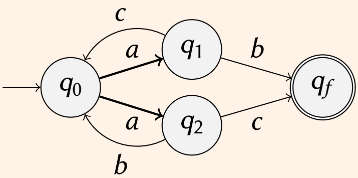
$\mathcal{R}(\epsilon, q_0, a) = q_1 \mid \mathcal{R}(\Sigma^* c, q_0, a) = q_2 \mid \mathcal{R}(\Sigma^* b, q_0, a) = q_1$

Run on *acabab*

$$q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_0 \xrightarrow{a} q_2 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_f$$

Doesn't give accepting run for *acacac*

Resolving Nondeterminism



Resolver : strategy to choose next transition based on history

$$\mathcal{R} : \Sigma^* \times Q \times \Sigma \mapsto Q$$

$\mathcal{R}(\epsilon, q_0, a) = q_1 \mid \mathcal{R}(\Sigma^* c, q_0, a) = q_2 \mid \mathcal{R}(\Sigma^* b, q_0, a) = q_1$

Run on *acabab*

$$q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_0 \xrightarrow{a} q_2 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_f$$

Doesn't give accepting run for *acacac*

No uniform strategy for accepting all words

Resolvability

Resolvability

Resolvable : If a resolver accepts all words in language

Resolvability

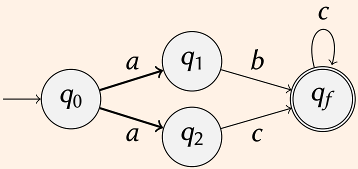
Resolvable : If a resolver accepts all words in language

$\exists \mathcal{R}, \forall w \in \mathcal{L}(\mathcal{A}), \mathcal{R}$ produces accepting run of w

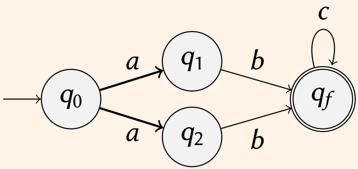
Resolvability

Resolvable : If a resolver accepts all words in language

$\exists \mathcal{R}, \forall w \in \mathcal{L}(\mathcal{A}), \mathcal{R}$ produces accepting run of w



$\mathcal{L} = abc^* + ac^+$

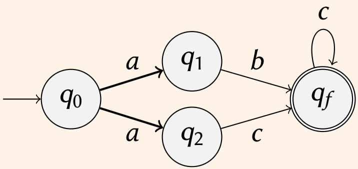


$\mathcal{L} = abc^*$

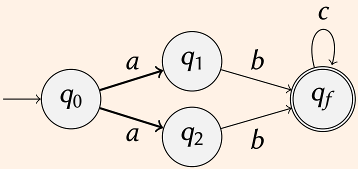
Resolvability

Resolvable : If a resolver accepts all words in language

$\exists \mathcal{R}, \forall w \in \mathcal{L}(\mathcal{A}), \mathcal{R}$ produces accepting run of w



$\mathcal{L} = abc^* + ac^+$



$\mathcal{L} = abc^*$

Commonly known as History Deterministic (HD) or Good for Games (GFG) automata

What are they good for?

Good for games automata studied for reactive synthesis
[Henzinger, Piterman'06]

What are they good for?

Good for games automata studied for reactive synthesis
[Henzinger, Piterman'06]

Has been studied for several models

- ω-regular automata
- Pushdown systems
- Timed automata
- VASS, etc

What are they good for?

Good for games automata studied for reactive synthesis
[Henzinger, Piterman'06]

Has been studied for several models

- ω-regular automata
- Pushdown systems
- Timed automata
- VASS, etc

This work : Generalise resolver strategies

Generalising Resolver Strategy

Generalising Resolver Strategy

Stochastic Resolver

Resolve using randomised strategy

Generalising Resolver Strategy

Stochastic Resolver

Resolve using randomised strategy

Produces probabilistic finite automaton (PFA) from NFA

Generalising Resolver Strategy

Stochastic Resolver

Resolve using randomised strategy

Produces probabilistic finite automaton (PFA) from NFA

Accept all words in language with some threshold probability

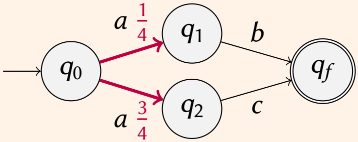
Overview

- Stochastic resolvers
- Classification of resolvable automata
- Complexity of recognising stochastic resolvability

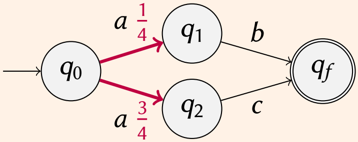
Focus on automata over finite words

Probabilistic automata

Probabilistic automata



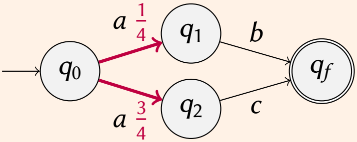
Probabilistic automata



$Pr(ab \text{ is accepted}) = \frac{1}{4}$

$Pr(ac \text{ is accepted}) = \frac{3}{4}$

Probabilistic automata

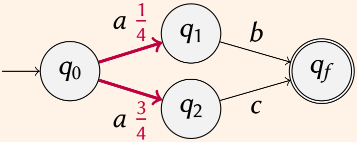


$Pr(ab \text{ is accepted}) = \frac{1}{4}$

$Pr(ac \text{ is accepted}) = \frac{3}{4}$

$$\mathcal{L}(\mathcal{P}_\lambda) \quad = \quad \{w \mid Pr(w \text{ is accepted}) \quad \geq \quad \lambda\}$$

Probabilistic automata



$Pr(ab \text{ is accepted}) = \frac{1}{4}$

$Pr(ac \text{ is accepted}) = \frac{3}{4}$

$$\mathcal{L}(\mathcal{P}_\lambda) = \{w \mid Pr(w \text{ is accepted}) \geq \lambda\}$$

$\mathcal{L}(\mathcal{P}_{\frac{1}{4}}) = \{ab, ac\}$
$\mathcal{L}(\mathcal{P}_{\frac{3}{4}}) = \{ac\}$

Stochastic Resolver

Stochastic Resolver

Resolve using randomised (memoryless) strategy

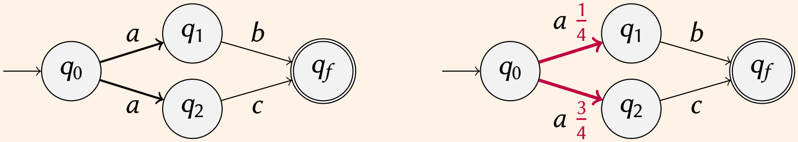
Resolver $\mathcal{R} : Q \times \Sigma \mapsto \Delta(Q)$

Stochastic Resolver

Resolve using randomised (memoryless) strategy

Resolver $\mathcal{R} : Q \times \Sigma \mapsto \Delta(Q)$

$\mathcal{R}$ produces probabilistic finite automaton (PFA) from NFA $\mathcal{A}$



$$\mathcal{R}(q_0, a, q_1) \quad = \quad \frac{1}{4} \mid \mathcal{R}(q_0, a, q_2) \quad = \quad \frac{3}{4}$$

Stochastic resolvability

Stochastic resolvability

Resolver accepts all words in $\mathcal{L}(\mathcal{A})$ with probability above a threshold

Stochastic resolvability

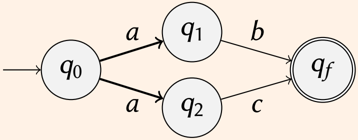
Resolver accepts all words in $\mathcal{L}(\mathcal{A})$ with probability above a threshold

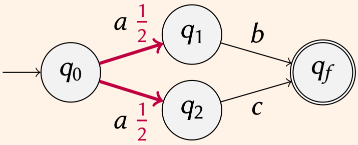
$\mathcal{A}$ is λ-resolvable if
$\exists$ resolver $\mathcal{R}$ s.t. $\forall w \in \mathcal{L}(\mathcal{A})$, $Pr_{\mathcal{R}}(w \text{ is accepted}) \geq \lambda$

Stochastic resolvability

Resolver accepts all words in $\mathcal{L}(\mathcal{A})$ with probability above a threshold

$\mathcal{A}$ is λ-resolvable if
$\exists$ resolver $\mathcal{R}$ s.t. $\forall\, w \in \mathcal{L}(\mathcal{A})$, $Pr_{\mathcal{R}}(w \text{ is accepted}) \geq \lambda$

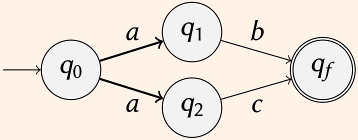


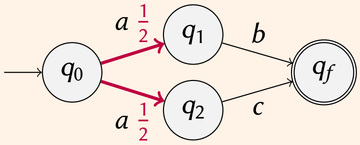


Stochastic resolvability

Resolver accepts all words in $\mathcal{L}(\mathcal{A})$ with probability above a threshold

> $\mathcal{A}$ is λ-resolvable if
> $\exists$ resolver $\mathcal{R}$ s.t. $\forall w \in \mathcal{L}(\mathcal{A})$, $Pr_{\mathcal{R}}(w \text{ is accepted}) \geq \lambda$





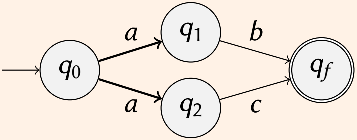
$\mathcal{R}(q_0, a, q_1) = p \mid \mathcal{R}(q_0, a, q_2) = 1 - p$

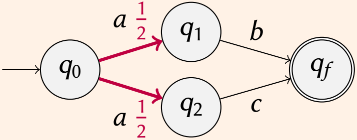
> λ-resolvable with $\lambda = \min(p, 1 - p)$

Stochastic resolvability

Resolver accepts all words in $\mathcal{L}(\mathcal{A})$ with probability above a threshold

$\mathcal{A}$ is λ-resolvable if
$\exists$ resolver $\mathcal{R}$ s.t. $\forall w \in \mathcal{L}(\mathcal{A})$, $Pr_{\mathcal{R}}(w \text{ is accepted}) \geq \lambda$





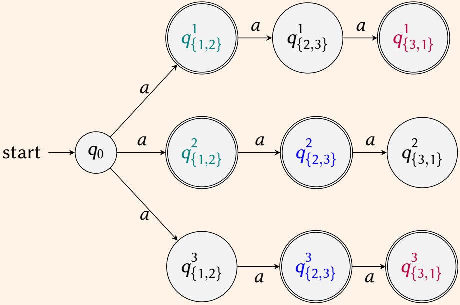
$\mathcal{R}(q_0, a, q_1) = p \mid \mathcal{R}(q_0, a, q_2) = 1 - p$

λ-resolvable with $\lambda = \min(p, 1-p)$

Next : Importance of threshold

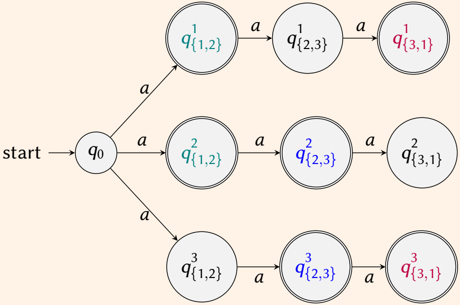
Resolvability with specific threshold

Resolvability with specific threshold



$\mathcal{L} = \{a, aa, aaa\}$

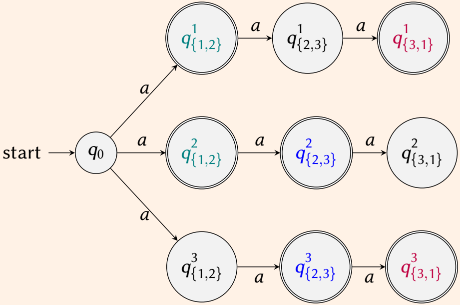
Resolvability with specific threshold



$\mathcal{L} = \{a, aa, aaa\}$

Uniform resolver $\mathcal{R}$, resolves with $\lambda = \frac{2}{3}$

Resolvability with specific threshold



$\mathcal{L} = \{a, aa, aaa\}$

Uniform resolver $\mathcal{R}$, resolves with $\lambda = \frac{2}{3}$

Cannot do better that $\frac{2}{3}$

Hierarchy of λ-resolvability

Hierarchy of λ-resolvability

Theorem

For each $\lambda \in \mathcal{Q}$ in $(0, 1)$, there is unary $\mathcal{A}_\lambda$ s.t.

Hierarchy of λ-resolvability

Theorem

For each $\lambda \in \mathcal{Q}$ in $(0, 1)$, there is unary $\mathcal{A}_\lambda$ s.t.

- $\mathcal{A}_\lambda$ is λ-resolvable

Hierarchy of λ-resolvability

Theorem

For each $\lambda \in \mathcal{Q}$ in $(0, 1)$, there is unary $\mathcal{A}_\lambda$ s.t.

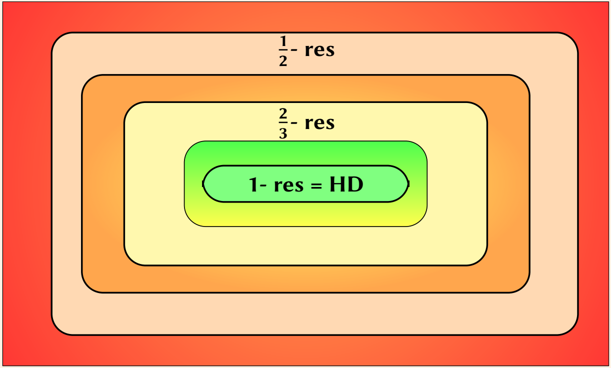
- $\mathcal{A}_\lambda$ is λ-resolvable
- $\forall \kappa > \lambda$, $\mathcal{A}_\lambda$ is not κ-resolvable

Hierarchy of λ-resolvability

Theorem

For each $\lambda \in \mathcal{Q}$ in $(0, 1)$, there is unary $\mathcal{A}_\lambda$ s.t.

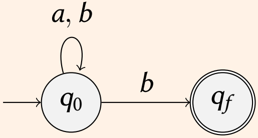
- $\mathcal{A}_\lambda$ is λ-resolvable
- $\forall \kappa > \lambda$, $\mathcal{A}_\lambda$ is not κ-resolvable



Set of all NFA λ-resolvable for some $\lambda > 0$

Resolvability with any threshold

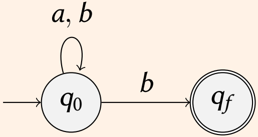
Resolvability with any threshold



$Pr(b^i \text{ is accepted}) \to 0$

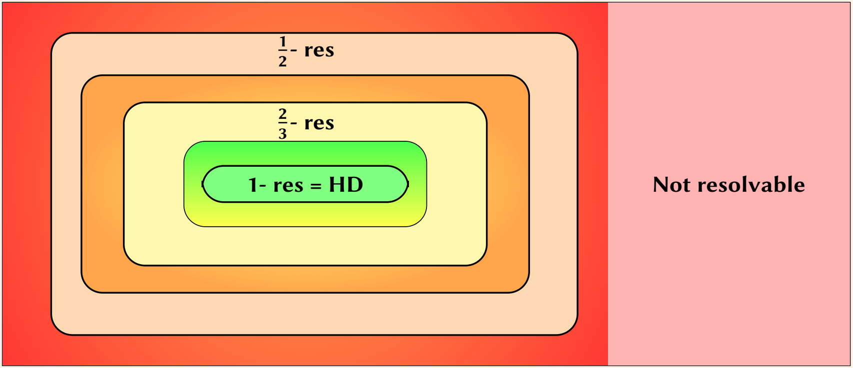
Not λ-resolvable for any $\lambda > 0$

Resolvability with any threshold



$Pr(b^i \text{ is accepted}) \to 0$

Not λ-resolvable for any $\lambda > 0$



Set of all NFA

Possible questions..

Possible questions..

For automata over finite words

Possible questions..

For automata over finite words

- Expressiveness (Same as regular languages)

Possible questions..

For automata over finite words

- Expressiveness (Same as regular languages)
- Succinctness

Possible questions..

For automata over finite words

- Expressiveness (Same as regular languages)
- Succinctness
- Memory needed for resolver (We focus on memoryless)

Possible questions..

For automata over finite words

- Expressiveness (Same as regular languages)
- Succinctness
- Memory needed for resolver (We focus on memoryless)
- Complexity of checking stochastic resolvability (Next)

Possible questions..

For automata over finite words

- Expressiveness (Same as regular languages)
- Succinctness
- Memory needed for resolver (We focus on memoryless)
- Complexity of checking stochastic resolvability (Next)

Questions of interest

Questions of interest

Given an NFA $\mathcal{A}$, is it stochastically resolvable with

Questions of interest

Given an NFA $\mathcal{A}$, is it stochastically resolvable with

- a specific threshold ?
- with any positive threshold ?

Questions of interest

Given an NFA $\mathcal{A}$, is it stochastically resolvable with

- a specific threshold ?
- with any positive threshold ?

Decision problems

Questions of interest

Given an NFA $\mathcal{A}$, is it stochastically resolvable with

- a specific threshold ?
- with any positive threshold ?

Decision problems

λ-RES : Given λ and $\mathcal{A}$, is NFA $\mathcal{A}$ λ-resolvable?

Questions of interest

Given an NFA $\mathcal{A}$, is it stochastically resolvable with

- a specific threshold ?
- with any positive threshold ?

Decision problems

λ-RES : Given λ and $\mathcal{A}$, is NFA $\mathcal{A}$ λ-resolvable?

Positive Resolvability
Given $\mathcal{A}$ $\exists \lambda \in (0, 1]$ s.t. $\mathcal{A}$ is λ-resolvable?

Complexity

Complexity

		unambiguous	**finitely-ambiguous**	**general**
Positive-resolvability	unary	NL	coNP-hard Σ_2^P	
	non-unary	NL-complete	PSPACE-complete	open
λ-**resolvability**	unary	P	coNP-hard **decidable**	open
	non-unary	NL-hard P	PSPACE-hard **decidable**	undecidable

Undecidability of λ-resolvability

Reduction from universality of PFA

Undecidability of λ-resolvability

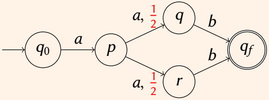
Reduction from universality of PFA

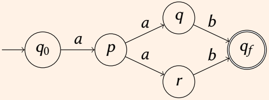
Given a PFA with probabilities in $\{0, 1, \frac{1}{2}\}$, it is universal?

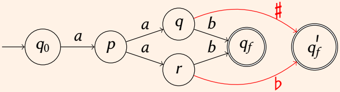
Undecidability of λ-resolvability

Reduction from universality of PFA

Given a PFA with probabilities in $\{0, 1, \frac{1}{2}\}$, it is universal?





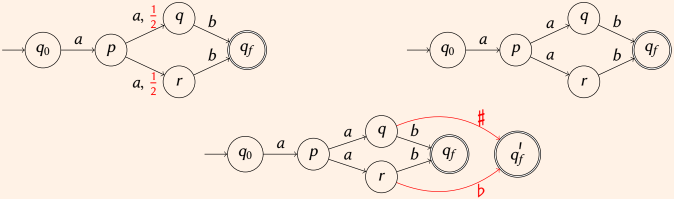


Optimal resolver $\mathcal{R}$ should have same support as PFA

Undecidability of λ-resolvability

Reduction from universality of PFA

Given a PFA with probabilities in $\{0, 1, \frac{1}{2}\}$, it is universal?



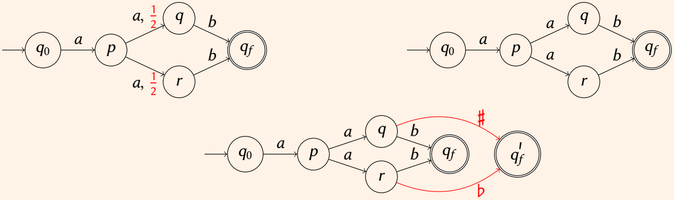
Optimal resolver $\mathcal{R}$ should have same support as PFA

Corollary : λ-RES is undecidable even for fixed λ. ($\lambda = \frac{1}{4}$)

Undecidability of λ-resolvability

Reduction from universality of PFA

Given a PFA with probabilities in $\{0, 1, \frac{1}{2}\}$, it is universal?



Optimal resolver $\mathcal{R}$ should have same support as PFA

Corollary : λ-RES is undecidable even for fixed λ. ($\lambda = \frac{1}{4}$)

Decidable for finitely ambiguous NFA

Positive resolvability

Positive resolvability

Decidability still open for general case

Positive resolvability

Decidability still open for general case

Theorem

Positive-Resolvability is decidable for

- unary NFA
- finitely ambiguous NFA

Finitely Ambiguous Automata

Finitely Ambiguous Automata

k-ambiguous automata : every word w has at most k accepting runs

Finitely Ambiguous Automata

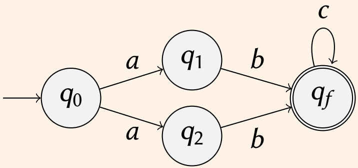
k-ambiguous automata : every word w has at most k accepting runs

Finitely ambiguous : k-ambiguous for some k
Unambiguous : 1-ambiguous

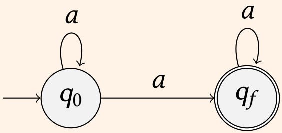
Finitely Ambiguous Automata

k-ambiguous automata : every word w has at most k accepting runs

Finitely ambiguous : k-ambiguous for some k
Unambiguous : 1-ambiguous



2-ambiguous



a^k has k runs

Complexity of positive resolvability

Theorem

The positive resolvability problem is

- PSPACE-complete for finitely ambiguous automata
- NL-complete for unambiguous automata

Complexity of positive resolvability

Theorem

The positive resolvability problem is

- PSPACE-complete for finitely ambiguous automata
- NL-complete for unambiguous automata

Next : Positive resolvability for finitely ambiguous automata

Positive resolvability

Positive resolvability

When is a resolver $\mathcal{R}$ good for positive resolvability of $\mathcal{A}$?

Positive resolvability

When is a resolver $\mathcal{R}$ good for positive resolvability of $\mathcal{A}$?

No diminishing sequence of words in resulting PFA

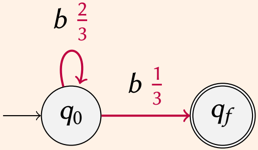
> There is no sequence of words $w_1, \ldots, w_i, \ldots$ in $\mathcal{L}(\mathcal{A})$ s.t. $\lim_{i\to\infty} Pr_{\mathcal{R}}(w) = 0$

Positive resolvability

When is a resolver $\mathcal{R}$ good for positive resolvability of $\mathcal{A}$?

No diminishing sequence of words in resulting PFA

There is no sequence of words $w_1, \ldots, w_i, \ldots$ in $\mathcal{L}(\mathcal{A})$ s.t. $\lim_{i\to\infty} Pr_{\mathcal{R}}(w) = 0$



Diminishig sequence : $b, bb, \cdots, b^i, \cdots$

$Pr_{\mathcal{R}}(b^i \text{ is accepted }) = (\frac{2}{3})^{i-1}\frac{1}{3}$

Support is all that matters to stay positive

Support is all that matters to stay positive

Set of transitions assigned positive probability by resolver

Support of $\mathcal{R}$: $\{(q, a, q') \mid \mathcal{R}(q, a, q') > 0\}$

Support is all that matters to stay positive

Set of transitions assigned positive probability by resolver

Support of $\mathcal{R}$: $\{(q, a, q') \mid \mathcal{R}(q, a, q') > 0\}$

Observation : Probability values over a support do not matter for positive resolvability

Support is all that matters to stay positive

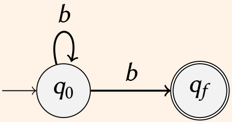
Set of transitions assigned positive probability by resolver

Support of $\mathcal{R}$: $\{(q, a, q') \mid \mathcal{R}(q, a, q') > 0\}$

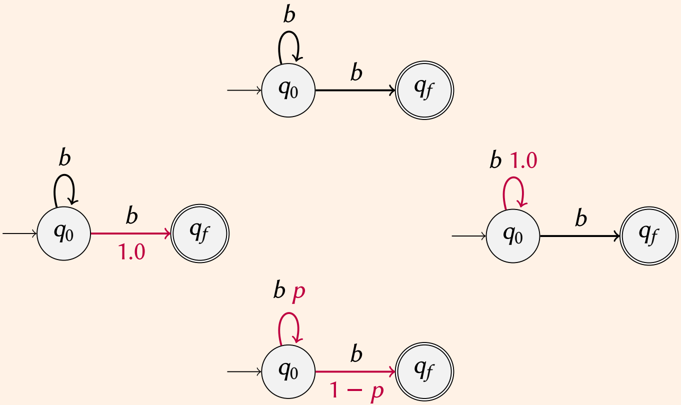
Observation : Probability values over a support do not matter for positive resolvability

Bad support : A support over which no probability assignments works

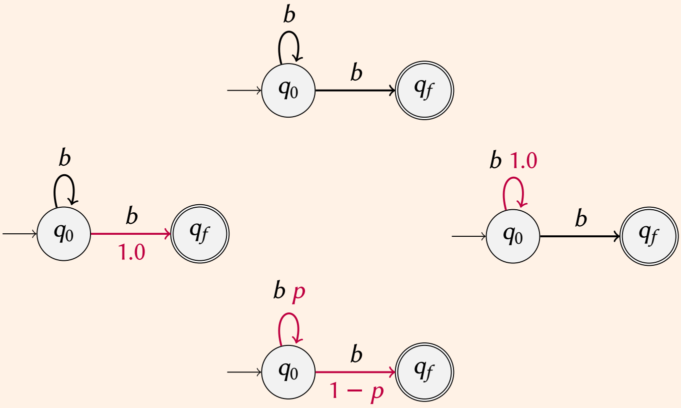
Bad support



Bad support



Bad support



- $\mathcal{L}(\mathcal{A}) \neq \mathcal{L}(\mathcal{A}_{\mathcal{S}})$
- Some condition equivalent to existence of diminishing sequence

Idea behind algorithm

Check if support is bad using
the two conditions

Unambiguous case : Positive resolvability

Unambiguous case : Positive resolvability

First step : trim the automata

Unambiguous case : Positive resolvability

First step : trim the automata

Now there is unique support S, with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\mathcal{S}})$

Unambiguous case : Positive resolvability

First step : trim the automata

Now there is unique support S, with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\mathcal{S}})$

Cond. equivalent to diminishing sequence

Support S is bad iff there is an SCC in $\mathcal{A}_{\mathcal{S}}$ with non-det transition

Unambiguous case : Positive resolvability

First step : trim the automata

Now there is unique support S, with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\mathcal{S}})$

Cond. equivalent to diminishing sequence

Support S is bad iff there is an SCC in $\mathcal{A}_{\mathcal{S}}$ with non-det transition

Can construct diminishing sequence by pumping loop containing non-det transition

Unambiguous case : Positive resolvability

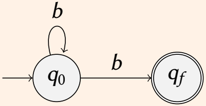
First step : trim the automata

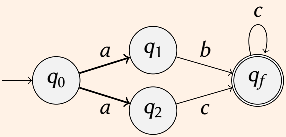
Now there is unique support S, with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\mathcal{S}})$

Cond. equivalent to diminishing sequence

Support S is bad iff there is an SCC in $\mathcal{A}_{\mathcal{S}}$ with non-det transition

Can construct diminishing sequence by pumping loop containing non-det transition





Unambiguous case : Positive resolvability

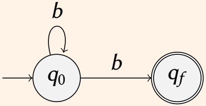
First step : trim the automata

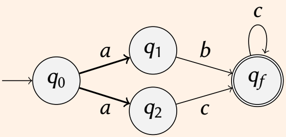
Now there is unique support S, with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\mathcal{S}})$

Cond. equivalent to diminishing sequence

Support S is bad iff there is an SCC in $\mathcal{A}_{\mathcal{S}}$ with non-det transition

Can construct diminishing sequence by pumping loop containing non-det transition





Positive resolvability for unambiguous automata is in NL

Finitely ambiguous case : Positive resolvability

Finitely ambiguous case : Positive resolvability

Generalise bad support from unambiguous to k-ambiguous

Finitely ambiguous case : Positive resolvability

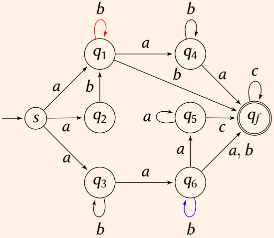
Generalise bad support from unambiguous to k-ambiguous

Bad support S : every run has non-det transition in some SCC of product $\mathcal{A}_S^k$

Finitely ambiguous case : Positive resolvability

Generalise bad support from unambiguous to k-ambiguous

> Bad support S : every run has non-det transition in some SCC of product $\mathcal{A}_S^k$



a b b a b b a c

$s \to q_1 \to q_1 \to q_1 \to q_4 \to q_4 \to q_4 \to q_f \to q_f$

$s \to q_2 \to q_1 \to q_1 \to q_4 \to q_4 \to q_4 \to q_f \to q_f$

$s \to q_3 \to q_3 \to q_3 \to q_6 \to q_6 \to q_6 \to q_5 \to q_f$

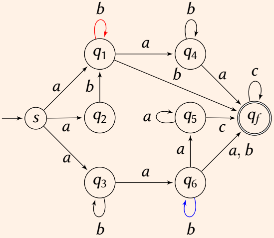
$s \to q_3 \to q_3 \to q_3 \to q_6 \to q_6 \to q_6 \to q_f \to q_f$

4-ambiguous automata

Finitely ambiguous case : Positive resolvability

Generalise bad support from unambiguous to k-ambiguous

Bad support S : every run has non-det transition in some SCC of product $\mathcal{A}_S^k$



	a	b	b	a	b	b	a	c
s	$\to q_1$	$\to q_1$	$\to q_1$	$\to q_4$	$\to q_4$	$\to q_4$	$\to q_f$	$\to q_f$
s	$\to q_2$	$\to q_1$	$\to q_1$	$\to q_4$	$\to q_4$	$\to q_4$	$\to q_f$	$\to q_f$
s	$\to q_3$	$\to q_3$	$\to q_3$	$\to q_6$	$\to q_6$	$\to q_6$	$\to q_5$	$\to q_f$
s	$\to q_3$	$\to q_3$	$\to q_3$	$\to q_6$	$\to q_6$	$\to q_6$	$\to q_f$	$\to q_f$

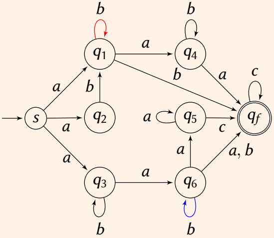
4-ambiguous automata

Diminishing sequence : $abb^i abb^i ac$

Finitely ambiguous case : Positive resolvability

Generalise bad support from unambiguous to k-ambiguous

Bad support S : every run has non-det transition in some SCC of product $\mathcal{A}_S^k$



	a	b	b	a	b	b	a	c
s	$\to q_1$	$\to q_1$	$\to q_1$	$\to q_4$	$\to q_4$	$\to q_4$	$\to q_f$	$\to q_f$
s	$\to q_2$	$\to q_1$	$\to q_1$	$\to q_4$	$\to q_4$	$\to q_4$	$\to q_f$	$\to q_f$
s	$\to q_3$	$\to q_3$	$\to q_3$	$\to q_6$	$\to q_6$	$\to q_6$	$\to q_5$	$\to q_f$
s	$\to q_3$	$\to q_3$	$\to q_3$	$\to q_6$	$\to q_6$	$\to q_6$	$\to q_f$	$\to q_f$

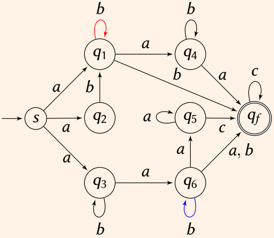
4-ambiguous automata

Diminishing sequence : abb^iabb^iac

Need to conserve number of runs in the pumped words

Need to store more

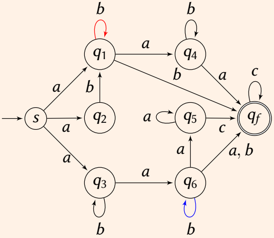
Need to store more



		a	b	b	a	b	b	a	c
	s	q_1	q_1	q_1	q_4	q_4	q_4	q_f	q_f
	s	q_2	q_1	q_1	q_4	q_4	q_4	q_f	q_f
	s	q_3	q_3	q_3	q_6	q_6	q_6	q_5	q_f
	s	q_3	q_3	q_3	q_6	q_6	q_6	q_f	q_f
R	$\varnothing$	$\varnothing$	$\{q_f\}$	$\{q_f\}$	$\varnothing$	$\{q_f\}$	$\{q_f\}$	$\varnothing$	$\varnothing$

R stores states from which there is no accepting run of suffix after reading prefix from start state

Need to store more



		a		b		b		a		b		b		a		c	
	s	→	q_1	→	q_1	→	q_1	→	q_4	→	q_4	→	q_4	→	q_f	→	q_f
	s	→	q_2	→	q_1	→	q_1	→	q_4	→	q_4	→	q_4	→	q_f	→	q_f
	s	→	q_3	→	q_3	→	q_3	→	q_6	→	q_6	→	q_6	→	q_5	→	q_f
	s	→	q_3	→	q_3	→	q_3	→	q_6	→	q_6	→	q_6	→	q_f	→	q_f
R	$\varnothing$		$\varnothing$		$\{q_f\}$		$\{q_f\}$		$\varnothing$		$\{q_f\}$		$\{q_f\}$		$\varnothing$		$\varnothing$

R stores states from which there is no accepting run of suffix after reading prefix from start state

> Bad support S : every run has non-det transition in some SCC of product $\mathcal{A}_S^k \times Q$ under this transition system

PSPACE algorithm

PSPACE algorithm

- Guess support
- Guess a short word in the transition system witnessing bad support

PSPACE algorithm

- Guess support
- Guess a short word in the transition system witnessing bad support

Ambiguity can be exponential

PSPACE algorithm

- Guess support
- Guess a short word in the transition system witnessing bad support

Ambiguity can be exponential

Store useful abstractions of the system and guess word on the fly

Automata on Infinite words ?

Automata on Infinite words ?

λ-resolvability is defined similarly for Parity automata

Automata on Infinite words ?

λ-resolvability is defined similarly for Parity automata

Some complexity results for stochastic resolvability extends

Automata on Infinite words ?

λ-resolvability is defined similarly for Parity automata

Some complexity results for stochastic resolvability extends

Theorem

- λ-Resolvability is undecidable
- λ-Resolvability is decidable for finitely ambiguous
- Positive-Resolvability is in PSPACE for finitely ambiguous

Automata on Infinite words ?

λ-resolvability is defined similarly for Parity automata

Some complexity results for stochastic resolvability extends

Theorem

- λ-Resolvability is undecidable
- λ-Resolvability is decidable for finitely ambiguous
- Positive-Resolvability is in PSPACE for finitely ambiguous

Independent work for 1-resolvability (almost sure acceptance) [Henzinger, Prakash, Thejaswini'25]

Automata on Infinite words ?

λ-resolvability is defined similarly for Parity automata

Some complexity results for stochastic resolvability extends

Theorem

- λ-Resolvability is undecidable
- λ-Resolvability is decidable for finitely ambiguous
- Positive-Resolvability is in PSPACE for finitely ambiguous

Independent work for 1-resolvability (almost sure acceptance)
[Henzinger, Prakash, Thejaswini'25]

Application : 1-resolvable Büchi automaton used in faster Markov Chain verification for UBA specifications
[Li, P, Schewe, Tang'25]

Open problems

Open problems

Given unary $\mathcal{A}$, is $\mathcal{A}$ λ-resolvable?

Open problems

Given unary $\mathcal{A}$, is $\mathcal{A}$ λ-resolvable?

The related problem for PFA is still open : Positivity of linear recurrence sequences

Open problems

Given unary $\mathcal{A}$, is $\mathcal{A}$ λ-resolvable?

The related problem for PFA is still open : Positivity of linear recurrence sequences

Given $\mathcal{A}$, is $\mathcal{A}$ positively resolvable?

Open problems

Given unary $\mathcal{A}$, is $\mathcal{A}$ λ-resolvable?

The related problem for PFA is still open : Positivity of linear recurrence sequences

Given $\mathcal{A}$, is $\mathcal{A}$ positively resolvable?

Decidable for unary : analysis of periodic behaviour of support matrix

Open problems

Given unary $\mathcal{A}$, is $\mathcal{A}$ λ-resolvable?

The related problem for PFA is still open : Positivity of linear recurrence sequences

Given $\mathcal{A}$, is $\mathcal{A}$ positively resolvable?

Decidable for unary : analysis of periodic behaviour of support matrix

Requires analysis of matrices obtained as product of support matrices of each letter

What's next?

What's next?

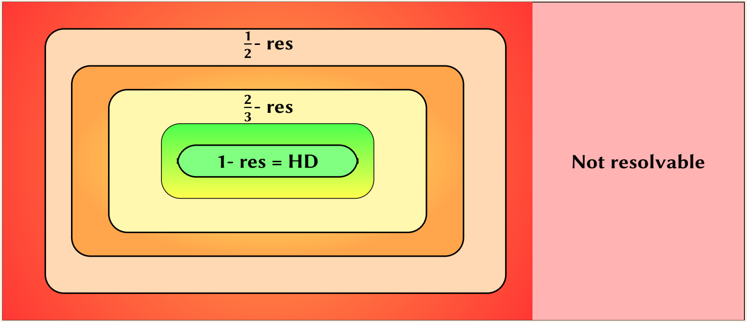
- Closing complexity gaps
- Other models : Pushdown, Timed Automata, VASS,...
- Applications in reactive synthesis

What's next?

- Closing complexity gaps
- Other models : Pushdown, Timed Automata, VASS,...
- Applications in reactive synthesis

Thank You

Summary : Automata on finite words



		unambiguous	finitely-ambiguous	general
Positive-resolvability	unary	NL	coNP-hard Σ_2^P	
	non-unary	NL-complete	PSPACE-complete	open
λ-resolvability	unary	P	coNP-hard **decidable**	open
	non-unary	NL-hard P	PSPACE-hard **decidable**	undecidable