Monadic Decomposability of Regular Relations

Pablo Barceló[1], Chih-Duo Hong[2], Xuan-Bach Le[2], Anthony W. Lin[3] and **Reino Niskanen**[2]

[1]Department of Computer Science, University of Chile & IMFD, Chile

[2]Department of Computer Science, University of Oxford, UK

[3]Department of Computer Science, Technische Universität Kaiserslautern, Germany

Reachability Problems 2019

What is monadic decomposability?

- For a relation, monadic decomposability captures the notion of sufficient independence of the components of the relation.
- Components of $X \times Y$ are completely independent. While in $\{(x, x) \mid x \in X\}$ the components are tightly coupled.

What is monadic decomposability?

- For a relation, monadic decomposability captures the notion of sufficient independence of the components of the relation.
- Components of $X \times Y$ are completely independent. While in $\{(x, x) \mid x \in X\}$ the components are tightly coupled.
- Monadic decomposable relation R is expressible as a finite union of direct products of unary predicates.

What is monadic decomposability?

- For a relation, monadic decomposability captures the notion of sufficient independence of the components of the relation.
- Components of $X \times Y$ are completely independent. While in $\{(x, x) \mid x \in X\}$ the components are tightly coupled.
- Monadic decomposable relation R is expressible as a finite union of direct products of unary predicates.
- It is a powerful tool for decision procedures in logical theories.
- Restricting analysis to monadic decomposable relations can turn an undecidable problem into a decidable one.
- They also provide a large and robust class that can be implemented in the context of SMT solvers.

Main problem

Main problem

Given a relation R. Is R monadic decomposable?

Main problem

Main problem

Given a relation R. Is R monadic decomposable?

We focus on relations defined by multitape automata.

Automata with multiple tapes

- Introduced in the 50s.
- Consists of a finite state-control and one-way read-only heads.
- Operates on finite inputs.
- An input is accepted, if a state reached after reading the whole input is final.
- An automaton *recognizes* a relation.

Automata with multiple tapes

- Introduced in the 50s.
- Consists of a finite state-control and one-way read-only heads.
- Operates on finite inputs.
- An input is accepted, if a state reached after reading the whole input is final.
- An automaton *recognizes* a relation.
- Represented as a labelled directed graph.

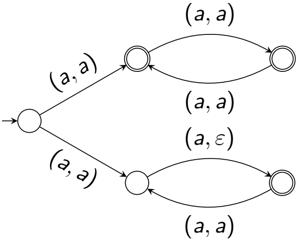
Syntactic restrictions

We study classes based on restrictions on the underlying graph.

Automata with multiple tapes

Variant 1

Nondeterministic behaviour with no restriction on tuples being read.

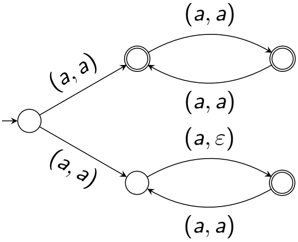


Automata with multiple tapes

Variant 1

Nondeterministic behaviour with no restriction on tuples being read.

We call such relations *rational* and denote by **Rat**.



Automata with multiple tapes

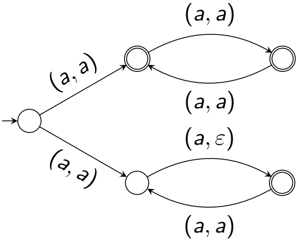
Variant 1

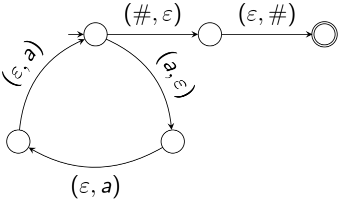
Nondeterministic behaviour with no restriction on tuples being read.

We call such relations *rational* and denote by **Rat**.

Variant 2

Deterministic behaviour with no restriction on tuples being read.





Automata with multiple tapes

Variant 1

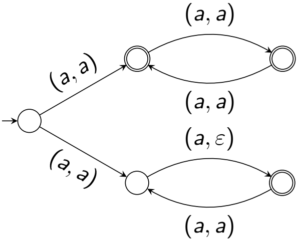
Nondeterministic behaviour with no restriction on tuples being read.

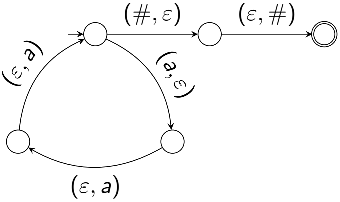
We call such relations *rational* and denote by **Rat**.

Variant 2

Deterministic behaviour with no restriction on tuples being read.

We call such relations *deterministic rational* and denote by **DRat**.





Automata with multiple tapes

Variant 3

No ε components in transitions allowed.

In other words, the automaton operates on k-tuples of symbols. So we need to *pad* the input words.

Automata with multiple tapes

Variant 3

No ε components in transitions allowed.

In other words, the automaton operates on k-tuples of symbols. So we need to *pad* the input words.

Let us call the relations $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ recognized by such automata regular and denote the class by **Reg**.

Variant 3

No ε components in transitions allowed.

In other words, the automaton operates on k-tuples of symbols. So we need to *pad* the input words.

Let us call the relations $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ recognized by such automata regular and denote the class by **Reg**.

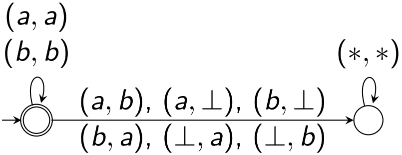
Also known as *synchronous* or *automatic* relations.

Regular relation example

Consider $R_{\mathsf{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

Regular relation example

Consider $R_{\mathrm{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.



Automata with multiple tapes

Variant 4

The components of the tuples are completely independent.

Essentially, the automaton has k-tuples of states and ith tape affects only ith component.

Automata with multiple tapes

Variant 4

The components of the tuples are completely independent.

Essentially, the automaton has k-tuples of states and ith tape affects only ith component.

Let us call the relations $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ recognized by such automata *recognizable* and denote the class by **Rec**.

Automata with multiple tapes

Rec can be equivalently seen as a finite union of direct products of regular languages.

$$R \in \textbf{Rec} \text{ iff } R = \bigcup_{i=1}^{n} L_{i,1} \times L_{i,2} \times \cdots \times L_{i,k},$$

for some regular languages $L_{i,j}$.

Rec can be equivalently seen as a finite union of direct products of regular languages.

$$R \in \mathbf{Rec} \text{ iff } R = \bigcup_{i=1}^{n} L_{i,1} \times L_{i,2} \times \cdots \times L_{i,k},$$

for some regular languages $L_{i,j}$.

Recognizable relations are also known as monadic decomposable relations.

Monadic decomposable relation example

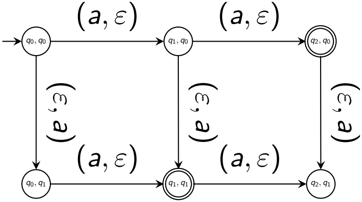
Consider

$$R_{\mathrm{fin}} = \{(a, a), (a^2, \varepsilon)\}.$$

Monadic decomposable relation example

Consider

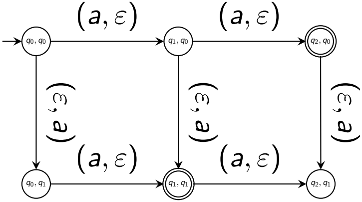
$$R_{\text{fin}} = \{(a, a), (a^2, \varepsilon)\}.$$



Monadic decomposable relation example

Consider

$$R_{\text{fin}} = \{(a, a), (a^2, \varepsilon)\}.$$



$$R_{\text{fin}} = (\{a\} \times \{a\}) \cup (\{a^2\} \times \{\varepsilon\})$$

Are the classes really different?

$k = 1$

For languages, the classes define the same family of languages.

Are the classes really different?

$k = 1$

For languages, the classes define the same family of languages.

For $k > 1$:

Theorem

There exists a strict hierarchy of families:

$$\mathbf{Rec} \subsetneq \mathbf{Reg} \subsetneq \mathbf{DRat} \subsetneq \mathbf{Rat}$$

Relative decidability

Given a relation R in $X \in \{\mathbf{Reg}, \mathbf{DRat}, \mathbf{Rat}\}$. Is R in $Y \in \{\mathbf{Rec}, \mathbf{Reg}, \mathbf{DRat}\}$ as well? (wlog $Y \subsetneq X$)

That is, can R be expressed by a simpler family?

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat			
Reg			
Rec			

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat	undecidable [Fisher, Rosenberg '67]		
Reg	undecidable		
Rec	undecidable [Lisovik '79]		

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat	undecidable [Fisher, Rosenberg '67]		
Reg	undecidable	open	
Rec	undecidable [Lisovik '79]	decidable [Carton, Choffrut, Grigorieff '06] 2-EXPTIME (for $k = 2$) [Valiant '75]	

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat	undecidable [Fisher, Rosenberg '67]		
Reg	undecidable	open	
Rec	undecidable [Lisovik '79]	decidable [Carton, Choffrut, Grigorieff '06] 2-EXPTIME (for $k = 2$) [Valiant '75]	decidable[1] [Libkin '00]

[1] Also independently by Carton et al.

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat	undecidable [Fisher, Rosenberg '67]		
Reg	undecidable	open	
Rec	undecidable [Lisovik '79]	decidable [Carton, Choffrut, Grigorieff '06] 2-EXPTIME (for $k = 2$) [Valiant '75]	decidable[1] [Libkin '00] EXPTIME (for $k = 2$) [Löding, Spinrath '17]

[1] Also independently by Carton et al.

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat	undecidable [Fisher, Rosenberg '67]		
Reg	undecidable	open	
Rec	undecidable [Lisovik '79]	decidable [Carton, Choffrut, Grigorieff '06] 2-EXPTIME (for $k = 2$) [Valiant '75]	decidable[1] [Libkin '00] EXPTIME (for $k = 2$) [Löding, Spinrath '17]

Theorem (BHLLN'19)

Let R be a regular k-ary relation given by a DFA (resp. NFA). Deciding whether R is monadic decomposable is in NL *(resp.* PSPACE*).*

[1] Also independently by Carton et al.

In literature

Given a relation in X. Is it also in Y?

Y \ X	**Rat**	**DRat**	**Reg**
DRat	undecidable [Fisher, Rosenberg '67]		
Reg	undecidable	open	
Rec	undecidable [Lisovik '79]	decidable [Carton, Choffrut, Grigorieff '06] 2-EXPTIME (for $k = 2$) [Valiant '75]	decidable[1] [Libkin '00] EXPTIME (for $k = 2$) [Löding, Spinrath '17]

Theorem (BHLLN'19)

Let R be a regular k-ary relation given by a DFA (resp. NFA). Deciding whether R is monadic decomposable is in NL *(resp.* PSPACE*). Matching lower bounds also hold.*

[1] Also independently by Carton et al.

Equivalence relation

Let R be a regular binary relation. Define equivalence relation $\sim$ as

$$u \sim v \text{ iff } \forall w : ((u, w) \in R \Leftrightarrow (v, w) \in R) \land ((w, u) \in R \Leftrightarrow (w, v) \in R)$$

Equivalence relation

Let R be a regular binary relation. Define equivalence relation $\sim$ as

$$u \sim v \text{ iff } \forall w : ((u, w) \in R \Leftrightarrow (v, w) \in R) \wedge ((w, u) \in R \Leftrightarrow (w, v) \in R)$$

Lemma (folklore)

Let R be a regular binary relation. Then the equivalence relation $\sim$ has infinite index iff R is not monadic decomposable.

Equivalence relation

Consider $R_{\mathrm{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

Equivalence relation

Consider $R_{\mathsf{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

It is clear that $u \not\sim v$ iff $u \neq v$.

Equivalence relation

Consider $R_{\mathsf{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

It is clear that $u \not\sim v$ iff $u \neq v$.

Hence there are infinitely many equivalence classes and the relation is not monadic decomposable.

Equivalence relation

Consider $R_{\mathsf{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

It is clear that $u \not\sim v$ iff $u \neq v$.

Hence there are infinitely many equivalence classes and the relation is not monadic decomposable.

Consider $R_{\mathsf{fin}} = \{(a, a), (a^2, \varepsilon)\}$.

Equivalence relation

Consider $R_{\text{eq}} = \{(u, u) \mid u \in \Sigma^*\}$, where $\Sigma = \{a, b\}$.

It is clear that $u \not\sim v$ iff $u \neq v$.

Hence there are infinitely many equivalence classes and the relation is not monadic decomposable.

Consider $R_{\text{fin}} = \{(a, a), (a^2, \varepsilon)\}$.

There are four equivalence classes, $\{\varepsilon\}$, $\{a\}$, $\{a^2\}$ and "everything else". Hence, the relation is monadic decomposable.

Key techniques

- Previous works studied whether a relation R is monadic decomposable.
- We focus on being not decomposable.
- In a way, we are reasoning on infinite objects rather than finite.

Key techniques

- Previous works studied whether a relation R is monadic decomposable.
- We focus on being not decomposable.
- In a way, we are reasoning on infinite objects rather than finite.
- We will show
  - R is not **Rec** iff there exists a sequence of representatives with a *nice* structure — utilizing combinatorial arguments such as Infinite Ramsey Theorem; and
  - how to generate such a sequence — utilizing pumping argument.

Goal

Given regular relation R. We need to

Goal

Given regular relation R. We need to

- construct regular automaton for $R^{\not\sim}$;

Goal

Given regular relation R. We need to

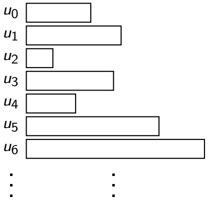
- construct regular automaton for $R^{\not\sim}$;
- verify that $R^{\not\sim}$ recognizes infinite number of pairs of distinct representatives;

Goal

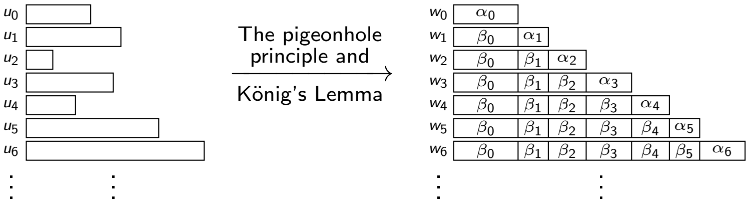
Given regular relation R. We need to

- construct regular automaton for $R^{\not\sim}$;
- verify that $R^{\not\sim}$ recognizes infinite number of pairs of distinct representatives;
- *and be efficient*.

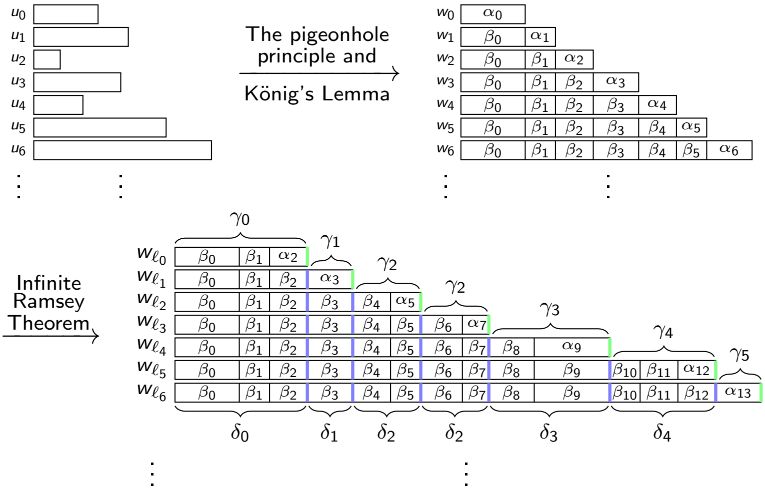
An infinite sequence of representatives



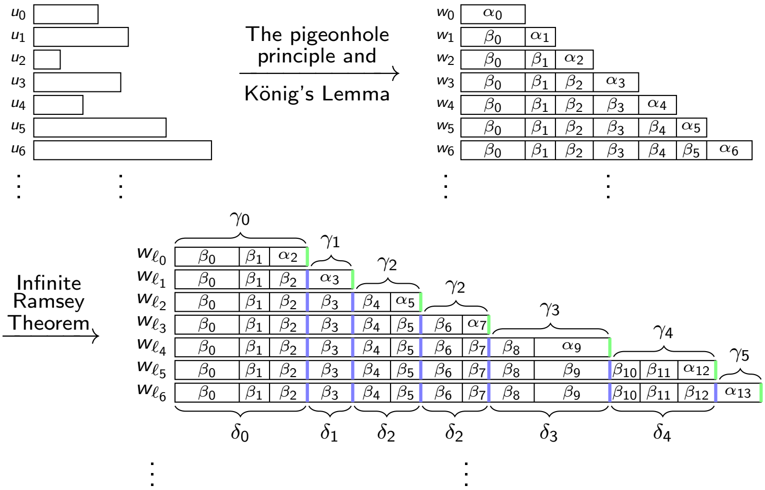
An infinite sequence of representatives



An infinite sequence of representatives



An infinite sequence of representatives



In this sequence, no matter which pair of words is being read, the states are the same after reading (δ_i, δ_i)!

Properties of the sequence

We have an infinite sequence of words $\{w_i\}_{i \geq 0}$, where

- $w_j \not\sim w_\ell$ for all $j \neq \ell$;
- $w_i = \delta_0 \cdots \delta_{i-1} \gamma_i$;
- $|\gamma_i| = |\delta_i| > 0$;
- No matter which pair of words is read, the automaton visits the same states in particular points of computation in $R^{\not\sim}$.

This sequence exists if and only if R is not monadic decomposable.

Synchronizing states and witnesses

We can further prove that R is not monadic decomposable iff there are synchronizing states with pumping property in $R^{\not\sim}$.

Synchronizing states and witnesses

We can further prove that R is not monadic decomposable iff there are synchronizing states with pumping property in $R^{\not\sim}$.

"Algorithm"

Given binary regular relation R, construct automaton for $R^{\not\sim}$. Guess these pumping states and check the reachability (in NL).

Synchronizing states and witnesses

We can further prove that R is not monadic decomposable iff there are synchronizing states with pumping property in $R^{\not\sim}$.

"Algorithm"

Given binary regular relation R, construct automaton for $R^{\not\sim}$. Guess these pumping states and check the reachability (in NL).

- If R is given by a DFA, then $R^{\not\sim}$ can be constructed in L.
- If R is given by an NFA, then $R^{\not\sim}$ can be constructed in PSPACE.

Main result

Theorem (BHLLN'19)

Let R be a regular binary relation given by a DFA (resp. NFA). Deciding whether R is monadic decomposable is in NL *(resp.* PSPACE*).*

Main result

Theorem (BHLLN'19)

Let R be a regular k-ary relation given by a DFA (resp. NFA). Deciding whether R is monadic decomposable is NL*-complete (resp.* PSPACE*-complete).*

Main result

Theorem (BHLLN'19)

Let R be a regular k-ary relation given by a DFA (resp. NFA). Deciding whether R is monadic decomposable is NL*-complete (resp.* PSPACE*-complete).*

P. Barceló, C-D. Hong, X-B. Le, A. Lin, R. Niskanen.
Monadic Decomposability of Regular Relations.
ICALP 2019, LIPIcs 132: 103:1–103:14, 2019.

Thank you for your attention!