Isomorphism of regular trees and words

Markus Lohrey
Christian Mathissen

University of Leipzig

November 7, 2011

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega}$$

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab \cdots$$

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab \cdots$$

Example 2: Consider the equation $X = Xab$.

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab \cdots$$

Example 2: Consider the equation $X = Xab$.

What is its smallest solution?

$$X = (ab)^{\overline{\omega}}$$

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab \cdots$$

Example 2: Consider the equation $X = Xab$.

What is its smallest solution?

$$X = (ab)^{\overline{\omega}} = \cdots ababab$$

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab\cdots$$

Example 2: Consider the equation $X = Xab$.

What is its smallest solution?

$$X = (ab)^{\overline{\omega}} = \cdots ababab$$

Example 3: Consider the equation $X = XaXbX$.

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab \cdots$$

Example 2: Consider the equation $X = Xab$.

What is its smallest solution?

$$X = (ab)^{\overline{\omega}} = \cdots ababab$$

Example 3: Consider the equation $X = XaXbX$.

What is its smallest solution?

$$X = [a, b]^{\eta}$$

Regular words (3 examples)

Example 1: Consider the equation $X = abX$.

What is its smallest solution?

$$X = (ab)^{\omega} = ababab \cdots$$

Example 2: Consider the equation $X = Xab$.

What is its smallest solution?

$$X = (ab)^{\overline{\omega}} = \cdots ababab$$

Example 3: Consider the equation $X = XaXbX$.

What is its smallest solution?

$$X = [a, b]^{\eta} \ \text{ (dense shuffle of } a \text{ and } b\text{)}$$

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Concatenation of generalized words is defined in the natural way:
Put $(L_2, \leq_2, c_2)$ to the right of $(L_1, \leq_1, c_1)$.

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Concatenation of generalized words is defined in the natural way:
Put $(L_2, \leq_2, c_2)$ to the right of $(L_1, \leq_1, c_1)$.

A system of equations over the finite alphabet Σ consists of

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Concatenation of generalized words is defined in the natural way:
Put $(L_2, \leq_2, c_2)$ to the right of $(L_1, \leq_1, c_1)$.

A system of equations over the finite alphabet Σ consists of

- a finite set of variables $\{X_1, \ldots, X_n\}$ and

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Concatenation of generalized words is defined in the natural way:
Put $(L_2, \leq_2, c_2)$ to the right of $(L_1, \leq_1, c_1)$.

A system of equations over the finite alphabet Σ consists of

- a finite set of variables $\{X_1, \ldots, X_n\}$ and
- equations $X_1 = \alpha_1, \ldots, X_n = \alpha_n$, where $\alpha_i \in (\Sigma \cup \{X_1, \ldots, X_n\})^*$.

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Concatenation of generalized words is defined in the natural way:
Put $(L_2, \leq_2, c_2)$ to the right of $(L_1, \leq_1, c_1)$.

A system of equations over the finite alphabet Σ consists of

- a finite set of variables $\{X_1, \ldots, X_n\}$ and
- equations $X_1 = \alpha_1, \ldots, X_n = \alpha_n$, where $\alpha_i \in (\Sigma \cup \{X_1, \ldots, X_n\})^*$.

Such a system has (in a certain technical sense) a smallest solution, which (intuitively) can be obtained by unravelling the system starting from X_1.

Regular words (a bit more formal)

A generalized word over the finite alphabet Σ is a tuple $(L, \leq, c)$, where

- $(L, \leq)$ is a countable linear order and
- $c : L \to \Sigma$ is a coloring.

Concatenation of generalized words is defined in the natural way:
Put $(L_2, \leq_2, c_2)$ to the right of $(L_1, \leq_1, c_1)$.

A system of equations over the finite alphabet Σ consists of

- a finite set of variables $\{X_1, \ldots, X_n\}$ and
- equations $X_1 = \alpha_1, \ldots, X_n = \alpha_n$, where $\alpha_i \in (\Sigma \cup \{X_1, \ldots, X_n\})^*$.

Such a system has (in a certain technical sense) a smallest solution, which (intuitively) can be obtained by unravelling the system starting from X_1.

A generalized word is regular iff it is the smallest solution of a system of equations.

Regular words (1st alternative characterization)

Consider a tree T, which is:
(i) rooted, (ii) finitely branching, (iii) ordered, and (iv) Σ-labelled.

Regular words (1st alternative characterization)

Consider a tree T, which is:
(i) rooted, (ii) finitely branching, (iii) ordered, and (iv) Σ-labelled.

T defines a generalized word yield$(T) = (L, \leq, c)$ as follows:

- L is the set leaves of T
- $\leq$ is the natural left-to-right order on leaves, and
- $c(v) = a$ if the leaf v is labelled with a.

Regular words (1st alternative characterization)

Consider a tree T, which is:
(i) rooted, (ii) finitely branching, (iii) ordered, and (iv) Σ-labelled.

T defines a generalized word $\text{yield}(T) = (L, \leq, c)$ as follows:

- L is the set leaves of T
- $\leq$ is the natural left-to-right order on leaves, and
- $c(v) = a$ if the leaf v is labelled with a.

T is regular iff T has only finitely many subtrees up to isomorphism.

Regular words (1st alternative characterization)

Consider a tree T, which is:
(i) rooted, (ii) finitely branching, (iii) ordered, and (iv) Σ-labelled.

T defines a generalized word yield$(T) = (L, \leq, c)$ as follows:

- L is the set leaves of T
- $\leq$ is the natural left-to-right order on leaves, and
- $c(v) = a$ if the leaf v is labelled with a.

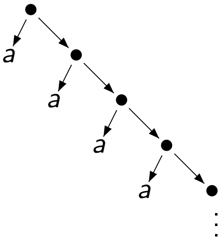
T is regular iff T has only finitely many subtrees up to isomorphism.

Observation

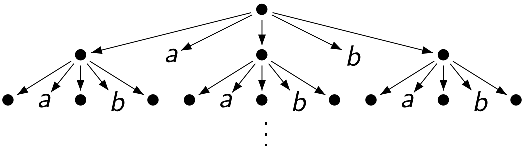
A generalized word w is regular $\iff w = \text{yield}(T)$ for a regular tree T.

Regular words (1st alternative characterization)

$X = aX$



$X = XaXbX$



Regular words (2nd alternative characterization)

A partitioned DFA is a tuple

$$A = (Q, \{1, \ldots, k\}, \delta, q_0, (F_a)_{a \in \Sigma}),$$

where Σ is a finite alphabet an $(Q, \{1, \ldots, k\}, \delta, q_0, \bigcup_{a \in \Sigma} F_a)$ is an ordinary DFA.

Regular words (2nd alternative characterization)

A partitioned DFA is a tuple

$$A = (Q, \{1, \dots, k\}, \delta, q_0, (F_a)_{a \in \Sigma}),$$

where Σ is a finite alphabet an $(Q, \{1, \dots, k\}, \delta, q_0, \bigcup_{a \in \Sigma} F_a)$ is an ordinary DFA.

A defines a generalized word $w(A) = (L, \leq, c)$, where

- L is the language accepted by $(Q, \{1, \dots, k\}, \delta, q_0, \bigcup_{a \in \Sigma} F_a)$,
- $\leq$ is the lexicographical order on L, and
- $c(v) = a$ if v is accepted by the DFA $(Q, \{1, \dots, k\}, \delta, q_0, F_a)$.

Regular words (2nd alternative characterization)

A partitioned DFA is a tuple

$$A = (Q, \{1, \dots, k\}, \delta, q_0, (F_a)_{a \in \Sigma}),$$

where Σ is a finite alphabet an $(Q, \{1, \dots, k\}, \delta, q_0, \bigcup_{a \in \Sigma} F_a)$ is an ordinary DFA.

A defines a generalized word $w(A) = (L, \leq, c)$, where

- L is the language accepted by $(Q, \{1, \dots, k\}, \delta, q_0, \bigcup_{a \in \Sigma} F_a)$,
- $\leq$ is the lexicographical order on L, and
- $c(v) = a$ if v is accepted by the DFA $(Q, \{1, \dots, k\}, \delta, q_0, F_a)$.

Observation

A generalized word w is regular $\iff$ $w = w(A)$ for a partitioned DFA A.

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation
- ω-power: $w^\omega = www\cdots$

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation
- ω-power: $w^{\omega} = www\cdots$
- $\overline{\omega}$-power: $w^{\overline{\omega}} = \cdots www$

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation
- ω-power: $w^\omega = www\cdots$
- $\overline{\omega}$-power: $w^{\overline{\omega}} = \cdots www$
- Dense shuffle $[w_1, w_2, \ldots, w_n]^\eta$:
    - Take a dense coloring of $(\mathbb{Q}, \leq)$ with colors $1, \ldots, n$ (every color i appears between any two rationals $x < y$).
    - Replace every i-colored point by a copy of w_i.

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation
- ω-power: $w^{\omega} = www\cdots$
- $\overline{\omega}$-power: $w^{\overline{\omega}} = \cdots www$
- Dense shuffle $[w_1, w_2, \ldots, w_n]^{\eta}$:
    - Take a dense coloring of $(\mathbb{Q}, \leq)$ with colors $1, \ldots, n$ (every color i appears between any two rationals $x < y$).
    - Replace every i-colored point by a copy of w_i.

A regular expression e over the finite alphabet Σ is built up from these operations and the symbols from Σ.

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation
- ω-power: $w^\omega = www\cdots$
- $\overline{\omega}$-power: $w^{\overline{\omega}} = \cdots www$
- Dense shuffle $[w_1, w_2, \ldots, w_n]^\eta$:
  - Take a dense coloring of $(\mathbb{Q}, \leq)$ with colors $1, \ldots, n$ (every color i appears between any two rationals $x < y$).
  - Replace every i-colored point by a copy of w_i.

A regular expression e over the finite alphabet Σ is built up from these operations and the symbols from Σ.

A regular expression e defines a generalized word $\mathrm{val}(e)$.

Regular words (3rd alternative characterization)

We define the following operations on generalized words:

- Concatenation
- ω-power: $w^\omega = www\cdots$
- $\overline{\omega}$-power: $w^{\overline{\omega}} = \cdots www$
- Dense shuffle $[w_1, w_2, \ldots, w_n]^\eta$:
  - Take a dense coloring of $(\mathbb{Q}, \leq)$ with colors $1, \ldots, n$ (every color i appears between any two rationals $x < y$).
  - Replace every i-colored point by a copy of w_i.

A regular expression e over the finite alphabet Σ is built up from these operations and the symbols from Σ.

A regular expression e defines a generalized word $\mathrm{val}(e)$.

Heilbrunner 1980

A generalized word w is regular $\iff$ $w = \mathrm{val}(e)$ for a regular expression e.

Isomorphism problem for a regular words I

Thomas 1986

It is decidable, whether $w_1 \cong w_2$ for two given regular words w_1, w_2 (given, e.g., by regular expressions or partitioned DFAs).

Isomorphism problem for a regular words I

Thomas 1986

It is decidable, whether $w_1 \cong w_2$ for two given regular words w_1, w_2 (given, e.g., by regular expressions or partitioned DFAs).

- Thomas proves this result by a reduction to the MSO theory of countable linear orders (decidable by Rabin 1969).

Isomorphism problem for a regular words I

Thomas 1986

It is decidable, whether $w_1 \cong w_2$ for two given regular words w_1, w_2 (given, e.g., by regular expressions or partitioned DFAs).

- Thomas proves this result by a reduction to the MSO theory of countable linear orders (decidable by Rabin 1969).
- No good complexity bound

Isomorphism problem for a regular words I

Thomas 1986

It is decidable, whether $w_1 \cong w_2$ for two given regular words w_1, w_2 (given, e.g., by regular expressions or partitioned DFAs).

- Thomas proves this result by a reduction to the MSO theory of countable linear orders (decidable by Rabin 1969).
- No good complexity bound

Bloom, Esik 2005

It can be checked in in polynomial time, whether $\mathrm{val}(e_1) \cong \mathrm{val}(e_2)$ for two given regular expressions e_1, e_2.

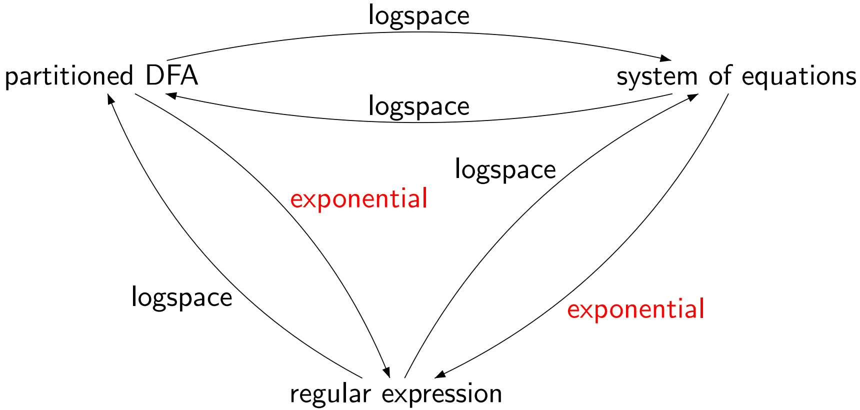
Isomorphism problem for a regular II

Open problem: What is the complexity of the isomorphism problem for regular words given by systems of equations or partitioned DFAs?

Isomorphism problem for a regular II

Open problem: What is the complexity of the isomorphism problem for regular words given by systems of equations or partitioned DFAs?

Complexity of transforming representations for regular words:



Our main result

Theorem 1

It can be checked in polynomial time whether $w(A_1) \cong w(A_2)$ for two given partitioned DFAs A_1, A_2 (in fact: P-complete).

Our main result

Theorem 1

It can be checked in polynomial time whether $w(A_1) \cong w(A_2)$ for two given partitioned DFAs A_1, A_2 (in fact: P-complete).

Analyzing Heilbrunner's algorithm yields:

Partitioned DFA $\longrightarrow$ DAG

From a given partitioned DFA A one can compute in logspace a DAG (directed acyclic graph), whose unfolding is a regular expression e with $\mathsf{val}(e) = w(A)$.

Our main result

Theorem 1

It can be checked in polynomial time whether $w(A_1) \cong w(A_2)$ for two given partitioned DFAs A_1, A_2 (in fact: P-complete).

Analyzing Heilbrunner's algorithm yields:

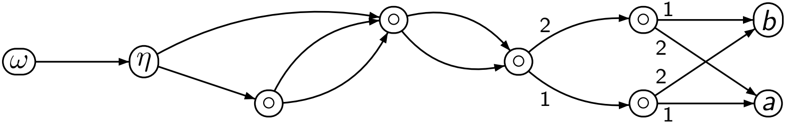
Partitioned DFA $\rightarrow$ DAG

From a given partitioned DFA A one can compute in logspace a DAG (directed acyclic graph), whose unfolding is a regular expression e with $\mathrm{val}(e) = w(A)$.

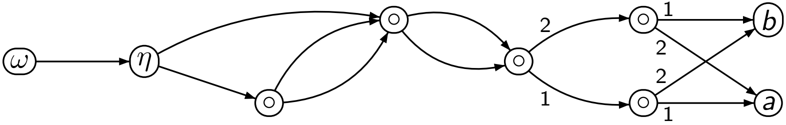
Proposition

It can be checked in polynomial time whether for two given DAGs, the corresponding regular expressions define isomorphic regular words.

DAGs: An example



DAGs: An example



This DAG produces the generalized word

$$[abbaabba, abbaabbaabbaabba]^{\eta})^{\omega}.$$

Its about data compression

Consider a DAG, in which the operations $()^{\omega}$, $()^{\overline{\omega}}$, and $[]^{\eta}$ are not used (only concatenation and constants from the alphabet).

Its about data compression

Consider a DAG, in which the operations $()^{\omega}$, $()^{\overline{\omega}}$, and $[]^{\eta}$ are not used (only concatenation and constants from the alphabet).

Such a DAG is also called a straight-line programm (SLP).

Its about data compression

Consider a DAG, in which the operations $()^{\omega}$, $()^{\overline{\omega}}$, and $[]^{\eta}$ are not used (only concatenation and constants from the alphabet).

Such a DAG is also called a straight-line programm (SLP).

An SLP with n nodes can define a finite word of length 2^n.

Its about data compression

Consider a DAG, in which the operations $()^{\omega}$, $()^{\overline{\omega}}$, and $[\,]^{\eta}$ are not used (only concatenation and constants from the alphabet).

Such a DAG is also called a straight-line programm (SLP).

An SLP with n nodes can define a finite word of length 2^n.

Plandowski 1994

It can be checked in polynomial time, whether two given SLPs produce the same string.

Its about data compression

Consider a DAG, in which the operations $()^\omega$, $()^{\overline{\omega}}$, and $[]^\eta$ are not used (only concatenation and constants from the alphabet).

Such a DAG is also called a straight-line programm (SLP).

An SLP with n nodes can define a finite word of length 2^n.

Plandowski 1994

It can be checked in polynomial time, whether two given SLPs produce the same string.

We have to extend Plandowskis algorithm from SLPs to general DAGs!

Its about data compression

Consider a DAG, in which the operations $()^{\omega}$, $()^{\overline{\omega}}$, and $[]^{\eta}$ are not used (only concatenation and constants from the alphabet).

Such a DAG is also called a straight-line programm (SLP).

An SLP with n nodes can define a finite word of length 2^n.

Plandowski 1994

It can be checked in polynomial time, whether two given SLPs produce the same string.

We have to extend Plandowskis algorithm from SLPs to general DAGs!

But: We reduce (by a polyomial time Turing reduction) the equivalence problem for general DAGs to those for SLPs, following the strategy of Bloom and Esik.

Related results

Theorem 2

It is PSPACE-hard (and in EXPTIME) to check for two given NFAs A_1, A_2, whether $(L(A_1), \leq_{\text{lex}}) \cong (L(A_2), \leq_{\text{lex}})$.

Related results

Theorem 2

It is PSPACE-hard (and in EXPTIME) to check for two given NFAs A_1, A_2, whether $(L(A_1), \leq_{\mathrm{lex}}) \cong (L(A_2), \leq_{\mathrm{lex}})$.

Theorem 3

It is EXPTIME-complete to check for two given NFAs A_1, A_2, whether $(L(A_1), \leq_{\mathrm{pref}}) \cong (L(A_2), \leq_{\mathrm{pref}})$.

Related results

Theorem 2

It is PSPACE-hard (and in EXPTIME) to check for two given NFAs A_1, A_2, whether $(L(A_1), \leq_{\text{lex}}) \cong (L(A_2), \leq_{\text{lex}})$.

Theorem 3

It is EXPTIME-complete to check for two given NFAs A_1, A_2, whether $(L(A_1), \leq_{\text{pref}}) \cong (L(A_2), \leq_{\text{pref}})$.

Theorem 4

It is P-complete to check for two given DFAs A_1, A_2, whether $(L(A_1), \leq_{\text{pref}}) \cong (L(A_2), \leq_{\text{pref}})$.

Related results

Kuske, Liu, L 2010

It is undecidable (even Σ^1_1-complete) to check for two given deterministic pushdown automata (even visibly pushdown automata) A_1, A_2, whether $(L(A_1), \leq_{\text{lex}}) \cong (L(A_2), \leq_{\text{lex}})$

Related results

Kuske, Liu, L 2010

It is undecidable (even Σ^1_1-complete) to check for two given deterministic pushdown automata (even visibly pushdown automata) A_1, A_2, whether $(L(A_1), \leq_{\mathrm{lex}}) \cong (L(A_2), \leq_{\mathrm{lex}})$

Kuske, Liu, L 2010

It is undecidable (even Σ^1_1-complete) to check for two given deterministic pushdown automata (even visibly pushdown automata) A_1, A_2, whether $(L(A_1), \leq_{\mathrm{pref}}) \cong (L(A_2), \leq_{\mathrm{pref}})$