Finite Models for the Theory of Concatenation

Dominik D. Freydenberger

Loughborough University

Design goals

A logic that can be used to query words, like one queries a database,

Design goals

A logic that can be used to query words, like one queries a database,

that can express

- concatenation $x = yz$
- and equality $x = y$

Towards a new logic on words (with word equations)

Design goals

A logic that can be used to query words, like one queries a database,

that can express

- concatenation $x = yz$
- and equality $x = y$

and that has fragments that are

- tractable and expressive.

Towards a new logic on words (with word equations)

Design goals

A logic that can be used to query words, like one queries a database,

that can express

- concatenation $x = yz$
- and equality $x = y$

and that has fragments that are

- tractable and expressive.

Motivating application

Study inexpressibility and tractability of document spanners.

- FC can be used for this, but I think it is useful beyond document spanners, and interesting in itself.

On the way to FC: Other logics on words

Structures for MSO

- Universe: $\{1, \ldots, n\}$ for some n
- total order $<$
- predicate P_a for every $a \in \Sigma$

Monadic second-order logic (MSO)

- captures the regular languages

On the way to FC: Other logics on words

Structures for MSO

- Universe: $\{1, \ldots, n\}$ for some n
- total order $<$
- predicate P_a for every $a \in \Sigma$

Monadic second-order logic (MSO)

- captures the regular languages

Yes, but...

- regular is not enough (for me)
- SO-variables are expensive
- MSO does not behave like FO in the relational database setting

On the way to FC: Other logics on words

Structures for MSO

- Universe: $\{1, \ldots, n\}$ for some n
- total order $<$
- predicate P_a for every $a \in \Sigma$

Monadic second-order logic (MSO)

- captures the regular languages

Yes, but...

- regular is not enough (for me)
- SO-variables are expensive
- MSO does not behave like FO in the relational database setting

Structure for C

- Universe: Σ^*
- concatenation $\cdot$
- constants for ε and each $a \in \Sigma$

The theory of concatention (C)

- FO on this structure

- $\varphi(x, y) := (x\mathtt{aby} \doteq y\mathtt{ba}x)$
- $\psi(w) := \neg\exists x\colon (w \doteq xx)$

On the way to FC: Other logics on words

Structures for MSO

- Universe: $\{1, \ldots, n\}$ for some n
- total order $<$
- predicate P_a for every $a \in \Sigma$

Monadic second-order logic (MSO)

- captures the regular languages

Yes, but…

- regular is not enough (for me)
- SO-variables are expensive
- MSO does not behave like FO in the relational database setting

Structure for C

- Universe: Σ^*
- concatenation $\cdot$
- constants for ε and each $a \in \Sigma$

The theory of concatention (C)

- FO on this structure

- $\varphi(x, y) := (x\mathtt{aby} \doteq y\mathtt{ba}x)$
- $\psi(w) := \neg\exists x\colon (w \doteq xx)$

One small problem

- Satisfiability is undecidable (beyond existential-positive)

Main idea: FC combines FO and C

FO on finite models

- first-order logic
- **Universe:** finite set
- relations, functions, constants

Properties

Satisfiability: undecidable
Model checking:

- PSPACE-complete
- NP-complete for ex.-pos. fragment

Main idea: FC combines FO and C

FO on finite models

- first-order logic
- **Universe:** finite set
- relations, functions, constants

Properties

Satisfiability: undecidable
Model checking:

- PSPACE-complete
- NP-complete for ex.-pos. fragment

C (theory of concatenation)

- first-order logic
- **Universe:** Σ^*
- concatenation, ε, Σ

Properties

Satisfiability: undecidable
Model checking:
same problem, undecidable

Main idea: FC combines FO and C

Define the **finite model version of the theory of concatenation**:

FO on finite models

- first-order logic
- **Universe:** finite set
- relations, functions, constants

Properties

Satisfiability: undecidable
Model checking:

- PSPACE-complete
- NP-complete for ex.-pos. fragment

FC

- first-order logic
- **Universe:** a word and all its factors
- concatenation, ε, Σ

F., Peterfreund (ICALP 2021)

C (theory of concatenation)

- first-order logic
- **Universe:** Σ^*
- concatenation, ε, Σ

Properties

Satisfiability: undecidable
Model checking: same problem, undecidable

FC and FC[REG]

The logic FC

Universe

A word w and all its factors.

Atoms

word equations $x \dot{=} \alpha$

Constant for w

w represents w

Connectives

$\wedge$, $\vee$, $\neg$, $\exists$, $\forall$

FC and FC[REG]

The logic FC

Universe

A word w and all its factors.

Atoms

word equations $x \doteq \alpha$

Constant for w

w represents w

Connectives

$\wedge$, $\vee$, $\neg$, $\exists$, $\forall$

FC[REG]

FC with regular constraints.

FC and FC[REG]

The logic FC

Universe

A word w and all its factors.

Atoms

word equations $x \doteq \alpha$

Constant for w

w represents w

Connectives

$\wedge$, $\vee$, $\neg$, $\exists$, $\forall$

FC[REG]

FC with regular constraints.

w contains papaya or banana

- $\exists x\colon (x \doteq \mathtt{papaya} \vee x \doteq \mathtt{banana})$
- $\exists x\colon x \mathrel{\dot\in} (\mathtt{papaya}|\mathtt{banana})$

w is a non-empty square

- $\exists x\colon (\mathsf{w} \doteq xx \wedge x \mathrel{\dot\in} \Sigma^+)$
- $\exists x\colon (\mathsf{w} \doteq xx \wedge \neg x \doteq \varepsilon)$

factors x that occur exactly once

$$\varphi(x) := \exists p_1, s_1\colon \big(\mathsf{w} \doteq p_1 x s_1 \\ \wedge \neg \exists p_2, s_2\colon (\mathsf{w} \doteq p_2 x s_2 \wedge \neg p_2 \doteq p_1)\big)$$

From Regex to Logic

- regular expressions with back-references directly translate to FC[REG]
- unless variables occur under a star (let's ignore this case for now)

Main idea

Sub-regex without variables become regular constraints

From Regex to Logic

- regular expressions with back-references directly translate to FC[REG]
- unless variables occur under a star (let's ignore this case for now)

Main idea

Sub-regex without variables become regular constraints

$$x\{(\mathtt{a}|\mathtt{b})^*\} \cdot y\{\mathtt{c}^*\} \cdot \mathtt{blah} \cdot \&x \cdot (x|y)$$

is converted to

$$\mathtt{w} \doteq x \cdot y \cdot \mathtt{blah} \cdot x \cdot z \quad \land x \mathrel{\dot\in} (\mathtt{a}|\mathtt{b})^* \land y \mathrel{\dot\in} \mathtt{c}^* \land (z \doteq x \lor z \doteq y)$$

From Regex to Logic

- regular expressions with back-references directly translate to FC[REG]
- unless variables occur under a star (let's ignore this case for now)

Main idea

Sub-regex without variables become regular constraints

$$x\{(\mathtt{a}|\mathtt{b})^*\} \cdot y\{\mathtt{c}^*\} \cdot \mathtt{blah} \cdot \&x \cdot (x|y)$$

is converted to

$$\mathtt{w} \doteq x \cdot y \cdot \mathtt{blah} \cdot x \cdot z \quad \wedge x \mathrel{\dot\in} (\mathtt{a}|\mathtt{b})^* \wedge y \mathrel{\dot\in} \mathtt{c}^* \wedge (z \doteq x \vee z \doteq y)$$

- This allows us to treat regex as FO-formulas...
- and to optimize them like FO-formulas (and relational algebra).

Next goal:
tractable fragments

Efficient brute-force, through bounded width

Width

- Highest number of free variables in any subformula.
- bounds the number of columns in intermediary tables
  (when using bottom-up evaluation)

Efficient brute-force, through bounded width

Width

- Highest number of free variables in any subformula.
- bounds the number of columns in intermediary tables
  (when using bottom-up evaluation)

- $\exists x_1, \ldots, x_k \colon \mathsf{w} \doteq x_1 x_1 x_2 x_2 \cdots x_k x_k$ has width k

Efficient brute-force, through bounded width

Width

- Highest number of free variables in any subformula.
- bounds the number of columns in intermediary tables (when using bottom-up evaluation)

- $\exists x_1, \ldots, x_k \colon \mathsf{w} \dot{=} x_1 x_1 x_2 x_2 \cdots x_k x_k$ has width k
- but the following equivalent formula has width 3:

$$
\begin{aligned}
&\exists x_1, y_1 \colon \big(\mathsf{w} \dot{=} x_1 x_1 y_1 \wedge \\
&\quad \exists x_2, y_2 \colon \big(y_1 \dot{=} x_2 x_2 y_2 \wedge \\
&\qquad\quad \vdots \\
&\qquad \exists x_{k-1}, y_{k-1} \colon \big(y_{k-2} \dot{=} x_{k-1} x_{k-1} y_{k-1} \wedge \exists x_k \colon y_{k-1} \dot{=} x_k x_k\big)\big)\big)
\end{aligned}
$$

Next goal: sufficient criteria for bounded width

Splitting Atoms

We get some criteria for free from FO.

- we can directly use formula parameters (like treewidth of a formula)
- but there are also some that are specific for word equations

Atom splitting

Some word equations can be decomposed into formulas with lower width.

Splitting Atoms

We get some criteria for free from FO.

- we can directly use formula parameters (like treewidth of a formula)
- but there are also some that are specific for word equations

Atom splitting

Some word equations can be decomposed into formulas with lower width.

Treewidth

- Reidenbach, Schmid (LATA 2012): introduced treewidth of a pattern

- F., Peterfreund (ICALP 2021): patterns of bounded treewidth become FC-formulas of bounded width.

(formula directly from nice tree decomposition)

Splitting Atoms

We get some criteria for free from FO.

- we can directly use formula parameters (like treewidth of a formula)
- but there are also some that are specific for word equations

Atom splitting

Some word equations can be decomposed into formulas with lower width.

Treewidth

- Reidenbach, Schmid (LATA 2012): introduced treewidth of a pattern

- F., Peterfreund (ICALP 2021): patterns of bounded treewidth become FC-formulas of bounded width.

(formula directly from nice tree decomposition)

Acyclic patterns (F., Thompson, ICDT 2022)

can be decomposed into

- formulas of the form $x \doteq yz$
- that are combined with semi-joins

Stars and Recursion

How to do star in FC

Add (deterministic) transitive closure or least/ partial fixed points.

- these logics capture L, NL, P, PSPACE
- formulas get painful to read and write

Stars and Recursion

How to do star in FC

Add (deterministic) transitive closure or least/ partial fixed points.

- these logics capture L, NL, P, PSPACE
- formulas get painful to read and write

Alternative model: FC-Datalog

Build relations through recursive rules.

- $R(x, y) \leftarrow x \doteq yy,$
- $R(x, z) \leftarrow x \doteq yy, R(y, z).$

Stars and Recursion

How to do star in FC

Add (deterministic) transitive closure or least/ partial fixed points.

- these logics capture L, NL, P, PSPACE
- formulas get painful to read and write

Alternative model: FC-Datalog

Build relations through recursive rules.

- $R(x,y) \leftarrow x \doteq yy,$
- $R(x,z) \leftarrow x \doteq yy, R(y,z).$

F., Peterfreund (ICALP 2021):

- FC-Datalog captures P

Bell, Day, F. (ICDT 2025):

- linear FC-Datalog captures NL
- det. lin. FC-Datalog captures L

- that's data complexity, but...
- good combined complexity is possible
- even deterministic linear FC-Datalog can express deterministic regex

What about satisfiability?

Recall

Satisfiability for FC is undecidable.

But that doesn't need to stop you.

What about satisfiability?

Recall

Satisfiability for FC is undecidable.

But that doesn't need to stop you.

Hopeful wish

Perhaps some approach from FO can be combined with word equation techniques

- part of this is wishful thinking, but we had some surprising success for inexpressibility:
- Thompson, F. (PODS 2024) managed to use EF-games for FC and FC[REG]

Papers on FC

Special thanks to EPSRC grant *Foundations of the Finite Model Theory of Concatenation*.

- F., Peterfreund: *The Theory of Concatenation over Finite Models*. ICALP 2021
- F., Thompson: *Splitting Spanner Atoms: A Tool for Acyclic Core Spanners*. ICDT 2022
- F., Thompson: *Languages Generated by Conjunctive Query Fragments of FC[REG]*. DLT 2023
- Thompson, F.: *Generalized Core Spanner Inexpressibility via Ehrenfeucht-Fraïssé Games for FC*. PODS 2024
- Bell, Day, F.: *FC-Datalog as a Framework for Efficient String Querying*. ICDT 2025
- Bell, Thompson, F.: *Parsing with the logic FC*. LangSec 2025
- Thompson, Schweikardt, F.: *Characterization and Decidability of FC-Definable Regular Languages*. LICS 2025

Wrapping things up

- String equality quickly causes problems.
- But FC allows us to use various approaches to contain this:
  - From algorithms that are impressive (but will never be used),
  - to ignoring the theory and just plugging queries together,
  - and more measured approaches in between.

Wrapping things up

- String equality quickly causes problems.
- But FC allows us to use various approaches to contain this:
  - From algorithms that are impressive (but will never be used),
  - to ignoring the theory and just plugging queries together,
  - and more measured approaches in between.

Not covered in this talk (but also fun)

- data structures and enumeration
- inexpressibility results
- characterization of FC-expressible regular languages
  Thompson, Schweikardt, F. (LICS 2025)
- engineering tricks when actually implementing this