Deciding Context Unification with regular constraints

Artur Jeż
University of Wrocław
Developments in Language Theory
Warsaw 08.08.2019
Equations over terms (unification)
Equations over terms

- Equations over terms (unification)
- What can the variables represent?
Variables represent closed terms
Variables represent closed terms

\[
\begin{align*}
&f \\
& \quad f \\
& \quad \quad f \\
& x \quad c \quad c \\
& \quad \quad z
\end{align*}
\]

\[
\begin{align*}
&f \\
& \quad f \\
& \quad \quad f \\
& f \quad z \\
& \quad \quad x \\
& \quad \quad \quad y \quad c
\end{align*}
\]
Variables represent closed terms: \textit{polynomial}

\begin{itemize}
  \item iterative decomposition
\end{itemize}
(First order) term unification

Variables represent closed terms: polynomial

- iterative decomposition

Not covered: “functions”

- open terms
- $\lambda$-terms
- ...
Variables have arguments that can be used. Terms with ‘holes’.
Variables have arguments that can be used. Terms with ‘holes’.

\[
X = \begin{array}{c}
h \end{array} \quad f \quad \begin{array}{c}
c \quad c \quad f \end{array} \\
\begin{array}{c}
\triangle \quad \triangle \quad \square 
\end{array}
\]

\[
X(t) = \begin{array}{c}
h \end{array} \quad f \quad \begin{array}{c}
h \quad f \end{array} \\
\begin{array}{c}
\triangle \quad \triangle \quad \triangle 
\end{array}
\]

\[
\begin{array}{c}
t \quad t \quad t 
\end{array} \quad \begin{array}{c}
c \quad c \quad c 
\end{array} \\
\begin{array}{c}
\triangle \quad t 
\end{array}
\]
Example

Equation

\[
\begin{align*}
X & \quad X \\
\quad c & \quad c
\end{align*}
= 
\begin{align*}
X & \quad f \\
\quad c & \quad c
\end{align*}
\]
Equation

Solutions: full binary trees
Problems

Undecidable.
Undecidable.

In very restricted cases:
- one argument
- one binary symbol
- ...

Subtlety
- unbounded number of usages (this causes hardness)
- can the argument be ignored (easy)
Problems

Undecidable.

In very restricted cases:
- one argument
- one binary symbol
- ...

Subtlety
- unbounded number of usages (this causes hardness)
- can the argument be ignored (easy)
Word equations

- Unary signature: word equations.
- Variables represent words $\in \Sigma^*$.
- $aXbXYbbb = XabaabYbY$ ($X = aa, Y = bb$)
Restrictions of second order unification

Word equations

- Unary signature: word equations.
- Variables represent words $\in \Sigma^*$.
- $aaabaabbbbbb = aabaabbbbbb$ ($X = aa, Y = bb$)

Decidable. [Makanin → Plandowski (PSPACE)]
Restrictions of second order unification

Word equations

- Unary signature: word equations.
- Variables represent words $\in \Sigma^*$.
- $aXbXYbbb = XabaabYbY$ $(X = aa, Y = bb)$

Decidable. [Makanin $\rightarrow$ Plandowski (PSPACE)]
In words equation arguments are used once.
In words equation arguments are used once.

Second order unification
Substitution for $X$ uses argument exactly once.
In words, equation arguments are used once.

**Context unification**

Second order unification
Substitution for \( X \) uses argument exactly once.

\[
X = h \circ c \circ c \circ f \circ c \circ f \circ X(t) = h \circ c \circ c \circ f \circ c \circ f \circ t
\]

Easy reduction: \( k \)-arguments \( \rightarrow \) one argument (NP-reduction)
In words equation arguments are used once.

Second order unification
Substitution for $X$ uses argument exactly once.

Easy reduction: $k$-arguments $\rightarrow$ one argument (NP-reduction)

Regular constraints: only substitution from a regular (tree) language.
More formally

**Definition (Context unification)**

- signature (fixed arities) \([f/2, a/1, c/0]\)
- context variables denoting terms with one ‘hole’ \([X/1]\)
- term variables denoting closed terms \([x/0]\)
- equations built with them
More formally

**Definition (Context unification)**

- **signature** (fixed arities) \([f/2, a/1, c/0]\)
- **context variables** denoting terms with one ‘hole’ \([X/1]\)
- **term variables** denoting closed terms \([x/0]\)
- equations built with them

**Definition (Substitution)**

A **substitution** \(S\) assigns

- to a term variable: a closed term over a signature
- to a context variable: a term over a signature plus a unique occurrence of ‘hole’ (●)
- extend in a natural way
  \(S(X(t))\): in \(S(X)\) we replace ● with \(S(t)\).
Example

- $f$: arity 2, $c, c'$: arity 0
- $X(c) = Y(c')$
Example

- $f$: arity 2, $c, c'$: arity 0
- $X(c) = Y(c')$
Larger example
Larger example

\[ S(X) \]
\[ S(Y) \]
\[ S(Z) \]
\[ S(x) \]
Regular constraints

Additional requirements on $S$:

- $S(x) \in L_x$ (tree regular language)
- $S(u) \in L_u$ (tree regular language)
## Regular constraints

**Additional requirements on** $S$:

- $S(x) \in L_x$ (tree regular language)
- $S(u) \in L_u$ (tree regular language)
- $S(X) \in L_X$ (??? regular language...—technicalities)
Additional requirements on $S$:
- $S(x) \in L_x$ (tree regular language)
- $S(u) \in L_u$ (tree regular language)
- $S(X) \in L_X$ (??? regular language...—technicalities)

Why regular constraints?
- because we can
- more expressive ones $\implies$ undecidability
- equivalent to linear second order unification
- regular constraints are somehow expressive
- very successful for word equations
Regular constraints

Additional requirements on $S$:

- $S(x) \in L_x$ (tree regular language)
- $S(u) \in L_u$ (tree regular language)
- $S(X) \in L_X$ (???.regular language... technicalities

Why regular constraints?

- because we can
- more expressive ones $\iff$ undecidability
- equivalent to linear second order unification
- regular constraints are somehow expressive
- very successful for word equations

- representation of $L$s matters for computational complexity.
- some representations are easier to handle
In between important problems:

- word equations, first order-unification
- second-order unification
In between important problems:

- word equations in PSPACE, first order-unification in P
- second-order unification **undecidable**
Why

In between important problems:

- word equations in PSPACE, first order-unification in P
- second-order unification undecidable
- Unknown status (20 years)
- Unique with this property
  (almost: linear second order unification)
In between important problems:

- word equations in \text{PSPACE}, first order-unification in \text{P}
- second-order unification \text{undecidable}
- Unknown status (20 years)
- Unique with this property
  (almost: linear second order unification)

Connections

- one-step term rewriting
- natural language parsing
- linear second-order unification =
  context unification + regular constraints
What was known

- special cases
  - one context variable
  - two context variables
  - stratified context unification
  - context variable always applied on the same term

We will not use any of that.
What was known

- special cases
  - one context variable
  - two context variables
  - stratified context unification
  - context variable always applied on the same term
- undecidability of generalisations
  - $\forall \exists^*$ theory of one-step term rewriting
  - $\forall \exists^*$ theory of word equations

We will not use any of that.
What was known

- special cases
  - one context variable
  - two context variables
  - stratified context unification
  - context variable always applied on the same term
- undecidability of generalisations
  - $\forall \exists^*$ theory of one-step term rewriting
  - $\forall \exists^*$ theory of word equations
- simplifications
  - only one binary symbol and constants

We will not use any of that.
What was known

- special cases
  - one context variable
  - two context variables
  - stratified context unification
  - context variable always applied on the same term
- undecidability of generalisations
  - $\forall \exists^*$ theory of one-step term rewriting
  - $\forall \exists^*$ theory of word equations
- simplifications
  - only one binary symbol and constants

We will not use any of that.
Context unification is in \textbf{PSPACE}.
Context unification is in PSPACE.

Trivially NP-hard
Context unification is in PSPACE.

Trivially NP-hard

- generalises earlier (simpler) solution for word equations
- applies compression to the equation
Context unification is in PSPACE.

Trivially NP-hard

- generalises earlier (simpler) solution for word equations
- applies compression to the equation
- quite natural
- proof reasonable
Context unification is in \textbf{PSPACE}.

Trivially \textbf{NP-hard}

- generalises earlier (simpler) solution for word equations
- applies compression to the equation
- quite natural
- proof reasonable

With regular constraints: in \textbf{EXPTIME}

Easily \textbf{EXPTIME}-hard.
The solution may have elements not present in the equation:

- \( \Sigma = \{f/2, c/0, c'/0\} \)
- \( X(c) = Y(c') \)
- \( S(X) = f(\bullet, c'), S(Y) = f(c, \bullet) \)
The solution may have elements not present in the equation:

- \( \Sigma = \{ f/2, c/0, c'/0 \} \)
- \( X(c) = Y(c') \)
- \( S(X) = f(\bullet, c'), S(Y) = f(c, \bullet) \)

We introduce new symbols to the signature—problem?
The solution may have elements not present in the equation:

\[ \Sigma = \{f/2, c/0, c'/0\} \]

\[ X(c) = Y(c') \]

\[ S(X) = f(\bullet, c'), \quad S(Y) = f(c, \bullet) \]

We introduce new symbols to the signature—problem?

**Not a problem: Trimming**

If the context equation has a solution then it has it over the signature of letters in the equation (plus arbitrary constant and binary symbol).
The solution may have elements not present in the equation:

- \( \Sigma = \{f/2, c/0, c'/0\} \)
- \( X(c) = Y(c') \)
- \( S(X) = f(\bullet, c'), S(Y) = f(c, \bullet) \)

We introduce new symbols to the signature—problem?

Not a problem: Trimming

If the context equation has a solution then it has it over the signature of letters in the equation (plus arbitrary constant and binary symbol).

- the solution uses other letter \( f/k \): replace it with a fixed \( f_k/k \)
The solution may have elements not present in the equation:

- $\Sigma = \{ f/2, c/0, c'/0 \}$
- $X(c) = Y(c')$
- $S(X) = f(\bullet, c'), S(Y) = f(c, \bullet)$

We introduce new symbols to the signature—problem?

Not a problem: Trimming

If the context equation has a solution then it has it over the signature of letters in the equation (plus arbitrary constant and binary symbol).

- the solution uses other letter $f/k$: replace it with a fixed $f_k/k$
- when $f_k/k$ is not in the equation:
  - replace $f_k(t_1, \ldots, t_k)$ with $g(t_1, g(\ldots g(t_{k-1}, t_k) \ldots))$
  - $f_1(t_1) \rightarrow t_1$
  - $c_0 \rightarrow c$
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Local compressions
Idea

Use this approach for solving context equations.

▶ apply simple compression operations to the solutions
Idea

Use this approach for solving context equations.

- apply simple compression operations to the solutions
- perform them directly on the equation changes of the equation
Idea

Use this approach for solving context equations.

- apply simple compression operations to the solutions
- perform them directly on the equation changes of the equation
- keep the size small:
  choose a compression that preserves quadratic size of the equation
Idea

Use this approach for solving context equations.

- apply simple compression operations to the solutions
- perform them directly on the equation changes of the equation
- keep the size small: choose a compression that preserves quadratic size of the equation
- this yields PSPACE
‘Absorb’ $i$-leaf $c$ by its father $f$ (and change father’s label to $f'$). Replace $f(t_1, t_2, t_3, \ldots, t_{i-1}, c, t_{i+1}, \ldots, t_k)$ with $f'(t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$
*(f, i, c)*-leaf compression

- ‘Absorb’ *i*-leaf *c* by its father *f* (and change father’s label to *f’*).
  Replace \( f(t_1, t_2, t_3, \ldots, t_{i-1}, c, t_{i+1}, \ldots, t_k) \) with \( f'(t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k) \)
‘Absorb’ $i$-leaf $c$ by its father $f$ (and change father’s label to $f'$).

Replace $f(t_1, t_2, t_3, \ldots, t_{i-1}, c, t_{i+1}, \ldots, t_k)$ with $f'(t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$
‘Absorb’ $i$-leaf $c$ by its father $f$ (and change father’s label to $f’$). Replace $f(t_1, t_2, t_3, \ldots, t_{i-1}, c, t_{i+1}, \ldots, t_k)$ with $f’(t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$.
Replace each occurrence of $ab$ with $d$. 
Replace each occurrence of $ab$ with $d$. 
Replace each occurrence of $ab$ with $d$. 
Replace each occurrence of $ab$ with $d$. 
Replace each maximal chain $a^\ell$ with $a_\ell$, for all $\ell$. 
Replace each maximal chain $a^\ell$ with $a_\ell$, for all $\ell$. 
Replace each maximal chain $a^\ell$ with $a_\ell$, for all $\ell$. 
Replace each maximal chain $a^\ell$ with $a_\ell$, for all $\ell$. 
Moving to the equation.
Definition (pair types)

For a solution $S$ the occurrence of $ab$ is

- **explicit** it comes from the equation;
- **implicit** comes solely from $S(X)$ (or $S(x)$);
- **crossing** in other case.

$ab$ is **crossing** (for $S$) if it has a crossing occurrence (for $S$), non-crossing (for $S$) otherwise.
**PairNCrComp**

1. Let \( c \in \Sigma \) be an unused letter.
2. Replace each explicit \( ab \) in the equation by \( c \).
**PairNCrComp**

1: let $c \in \Sigma$ be an unused letter
2: replace each explicit $ab$ in the equation by $c$

**Lemma**

$\text{PairNCrComp}(a, b)$ *properly compresses noncrossing pairs*.
Non-crossing Pair Compression

PairNCrComp

1: let $c \in \Sigma$ be an unused letter
2: replace each explicit $ab$ in the equation by $c$

Lemma

PairNCrComp$(a, b)$ properly compresses noncrossing pairs.

Proof.

Every $ab$ is replaced:

- explicit pairs replaced explicitly
- implicit pairs replaced implicitly (in the solution)
- crossing there are none
Definition ($a$-chain types)

For a solution $S$ the occurrence of a maximal $a$-chain is

- **explicit** it comes from the equation;
- **implicit** comes solely from $S(X)$ (or $S(x)$);
- **crossing** in other case.

$a$ has **crossing chains** (for $S$) if it has a crossing $a$-chain occurrence (for $S$), non-crossing (for $S$) otherwise.
ChainNCr

1: for $\ell > 1$ do
2: replace each explicit maximal $a^\ell$ in the equation by $a_\ell$
ChainNCr

1: \textbf{for } \ell > 1 \textbf{ do}

2: replace each explicit maximal \( a^\ell \) in the equation by \( a_\ell \)

Lemma

ChainNCr(\(a\)) \textit{compresses noncrossing} \(a\)-\textit{chains}. 
Definition (father-i-leaf pair)

For a solution $S$ the occurrence of father-i-leaf $(f, i, c)$ is

- **explicit** if it comes from the equation;
- **implicit** if it comes solely from $S(X)$ (or $S(x)$);
- **crossing** in other case.

$(f, i, c)$ is crossing (for $S$) if it has a crossing occurrence (for $S$), non-crossing (for $S$) otherwise.
\begin{align*}
&\begin{tikzpicture}
\node (a) at (0,0) {$a$};
\node (b) at (1,0) {$b$};
\node (c) at (0.5,1) {$c$};
\node (f) at (0.5,2) {$f$};
\draw (a) -- (f);
\draw (b) -- (f);
\draw (c) -- (f);
\draw (a) -- (b);
\draw (a) -- (c);
\draw (b) -- (c);
\end{tikzpicture}
\quad = \quad
\begin{tikzpicture}
\node (a) at (0,0) {$a$};
\node (b) at (1,0) {$b$};
\node (c) at (0.5,1) {$c$};
\node (f) at (0.5,2) {$f$};
\draw (a) -- (f);
\draw (b) -- (f);
\draw (c) -- (f);
\draw (a) -- (b);
\draw (a) -- (c);
\draw (b) -- (c);
\end{tikzpicture}
\end{align*}
Leaf compression ctd.

LeafNCr$(f, i, c)$

1: replace $f$ with $c$ at positions $i$ by $f'$
Leaf compression ctd.

Leaf$\text{NCr}(f, i, c)$

1: replace $f$ with $c$ at positions $i$ by $f'$

Lemma

Leaf$\text{NCr}$ performs leaf compression when $(f, i, c)$ is non-crossing.
Uncrossing
Uncrossing pairs

\[ a \quad b \]
\[ y \]

\[ X \]
\[ X \]
\[ a \quad b \quad y \]

If \( S(Y) \) is empty then remove \( Y \).

Lemma

After performing this for all variables, \( ab \) is no longer crossing.

Compact it!
Uncrossing pairs

- replace $Y$ with $bY$ and replace $X$ with $Xa$
- implicitly change $S(Y) = bt$, $S(X) = t'a$ to $S(Y) = t$, $S(X) = t'$

Lemma
After performing this for all variables, $ab$ is no longer crossing.

Compress it!
Uncrossing pairs

replace $Y$ with $bY$ and replace $X$ with $Xa$
implicitly change $S(Y) = bt, S(X) = t'a$ to $S(Y) = t,$
$S'(X) = t'$

If $S(Y)$ is empty then remove $Y$.
Uncrossing pairs

replace $Y$ with $bY$ and replace $X$ with $Xa$

implicitly change $S(Y) = bt$, $S(X) = t'a$ to $S(Y) = t$, $S(X) = t'$

If $S(Y)$ is empty then remove $Y$.

Lemma

After performing this for all variables, $ab$ is no longer crossing.
Uncrossing pairs

- replace $Y$ with $bY$ and replace $X$ with $Xa$
  - implicitly change $S(Y) = bt$, $S(X) = t'a$ to $S(Y) = t$, $S(X) = t'$
- If $S(Y)$ is empty then remove $Y$.

Lemma

After performing this for all variables, $ab$ is no longer crossing.

Compress it!
Crossing $a$-chains?

- Crossing $a$-chain: similar to crossing $ab$ (equiv. to crossing $aa$).
Crossing $a$-chains?

- Crossing $a$-chain: similar to crossing $ab$ (equiv. to crossing $aa$).
- pop whole $a$-prefix and $a$-suffix
  - $S(X) = a^\ell X t a^r X$: change it to $S(X) = t$
  - replace $X$ in equation by $a^\ell X a^r X$
  - they cannot be too long: exponential
Crossing $a$-chains?

- Crossing $a$-chain: similar to crossing $ab$ (equiv. to crossing $aa$).
- Pop whole $a$-prefix and $a$-suffix
  - $S(X) = \alpha^X t \alpha^{-X}$: change it to $S(X) = t$
  - replace $X$ in equation by $\alpha^X X \alpha^{-X}$
  - they cannot be too long: exponential
- do for all variables
Crossing $a$-chains?

- Crossing $a$-chain: similar to crossing $ab$ (equiv. to crossing $aa$).
- Pop whole $a$-prefix and $a$-suffix
  - $S(X) = a^\ell x t a^\ell x$: change it to $S(X) = t$
  - replace $X$ in equation by $a^\ell x X a^\ell x$
  - they cannot be too long: exponential
- Do for all variables
- $a$ has no crossing chains!
Crossing $a$-chains?

- Crossing $a$-chain: similar to crossing $ab$ (equiv. to crossing $aa$).
- Pop whole $a$-prefix and $a$-suffix
  - $S(X) = a^\ell X t a^r X$: change it to $S(X) = t$
  - replace $X$ in equation by $a^\ell X a^r X$
    - they cannot be too long: exponential
- Do for all variables
- $a$ has no crossing chains!
- Compress $a$-chains
Uncrossing father-leaf pair

\[ f \]
\[ c \]
\[ y \]

\[ X \]
\[ f \]
\[ c \]
\[ y \]

\[ X \]
\[ f \]
\[ c \]
\[ y \]

Replace \( y \) with \( c \)

\[ f(\ldots, x_{i-1}, \ldots, x_i+1, \ldots, x_\ell) \]

Remove \( X \) when \( S(X) \) is empty

\[ \text{new variables } x_{i-1}, \ldots, x_i+1, \ldots, x_\ell \]
- replace $y$ with $c$
- replace $X$ with $X(f(x_1, \ldots, x_{i-1}, \bullet, x_{i+1}, \ldots, x_\ell))$
Uncrossing father-leaf pair

- replace $y$ with $c$
- replace $X$ with $X(f(x_1, \ldots x_{i-1}, \bullet, x_{i+1}, \ldots, x_\ell))$

**new variables** $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_\ell$
Uncrossing father-leaf pair

- replace $y$ with $c$
- replace $X$ with $X(f(x_1, \ldots, x_{i-1}, \bullet, x_{i+1}, \ldots, x_\ell))$
  new variables $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_\ell$
- remove $X$ when $S(X)$ is empty
Uncrossing father-leaf pair

- replace $y$ with $c$
- replace $X$ with $X(f(x_1, \ldots x_{i-1}, \bullet, x_{i+1}, \ldots, x_\ell))$
  new variables $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_\ell$
- remove $X$ when $S(X)$ is empty
- there are no crossing $(f, i, c)$ father-$i$-leaf pairs
- compress them
**while** equation is nontrivial **do**
The algorithm

while equation is nontrivial do
    choose some $ab$, $a$ or $(f, i, c)$ to compress
    if it is crossing then
        uncross it
    compress it
Lemma

The maximal arity of letters in $\Sigma$ does not increase.
Lemma

The maximal arity of letters in $\Sigma$ does not increase.

Does not depend on the nondeterministic choices.
Lemma

The maximal arity of letters in $\Sigma$ does not increase.

Does not depend on the nondeterministic choices.

Proof.

Compression operations do not increase arity.
Lemma

The maximal arity of letters in $\Sigma$ does not increase.

Does not depend on the nondeterministic choices.

Proof.

Compression operations do not increase arity.

Call it $k$. 
Lemma

There are at most

- $n$ context variables
- $kn$ variables

($n$: size of the input equation; $k$—maximal arity of functions)
Lemma

There are at most

- \( n \) context variables
- \( kn \) variables

\((n: \text{size of the input equation}; k—\text{maximal arity of functions})\)

Does not depend on the nondeterministic choices.
Lemma

There are at most
- $n$ context variables
- $kn$ variables

($n$: size of the input equation; $k$—maximal arity of functions)

Does not depend on the nondeterministic choices.

Proof.
- we do not introduce new context variables
- we can associate each ‘new’ variable with context variable at most $(k - 1)$ are associated with one context variable
Controlling new variables: details

When to pop-down

- we pop only when needed: \( X(c) \) and last letter of \( S(X) \) is \( f \)
- at most \( k - 1 \) new variables per \( X \)
- they are all below \( X \).
Controlling new variables: details

When to pop-down

- we pop only when needed: $X(c)$ and last letter of $S(X)$ is $f$
- at most $k - 1$ new variables per $X$
- they are all below $X$.

When $X$ pops again

- $X(c')$ is in equation
- no variable is below $X$: all were removed
When to pop-down

- we pop only when needed: $X(c)$ and last letter of $S(X)$ is $f$
- at most $k - 1$ new variables per $X$
- they are all below $X$.

When $X$ pops again

- $X(c')$ is in equation
- no variable is below $X$: all were removed

There are at most $kn$ variables
Soundness
If the new equation has a solution, then also the original one had.
Correctness

Soundness
If the new equation has a solution, then also the original one had.

Just roll back the changes.
Correctness

Soundness
If the new equation has a solution, then also the original one had.

Just roll back the changes.

Completeness
If the equation has the solution, then for some nondeterministic choices the new equation has a corresponding one.
### Correctness

<table>
<thead>
<tr>
<th><strong>Soundness</strong></th>
<th>If the new equation has a solution, then also the original one had.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Just roll back the changes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Completeness</strong></th>
<th>If the equation has the solution, then for some nondeterministic choices the new equation has a corresponding one.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Make the choices according to the solution.</td>
</tr>
</tbody>
</table>
Noncrossing

- decreases the size of the solution
## Size

<table>
<thead>
<tr>
<th>Noncrossing</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ decreases the size of the solution</td>
</tr>
<tr>
<td>▶ decreases the size of the equation</td>
</tr>
<tr>
<td>Noncrossing</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>▶ decreases the size of the solution</td>
</tr>
<tr>
<td>▶ decreases the size of the equation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crossing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Noncrossing</td>
<td>Crossing</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>▶ decreases the size of the solution</td>
<td>▶ decreases the size of the solution</td>
</tr>
<tr>
<td>▶ decreases the size of the equation</td>
<td></td>
</tr>
</tbody>
</table>
### Size

<table>
<thead>
<tr>
<th>Noncrossing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ decreases the size of the solution</td>
<td></td>
</tr>
<tr>
<td>▶ decreases the size of the equation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crossing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ decreases the size of the solution</td>
<td></td>
</tr>
<tr>
<td>▶ increases the size of the equation (popped letters)</td>
<td></td>
</tr>
</tbody>
</table>
### Noncrossing
- Decreases the size of the solution
- Decreases the size of the equation

### Crossing
- Decreases the size of the solution
- Increases the size of the equation (popped letters)
- Decreases the size of the equation (compressed letters)
<table>
<thead>
<tr>
<th><strong>Noncrossing</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decreases the size of the solution</td>
</tr>
<tr>
<td></td>
<td>decreases the size of the equation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Crossing</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decreases the size of the solution</td>
</tr>
<tr>
<td></td>
<td>increases the size of the equation (popped letters)</td>
</tr>
<tr>
<td></td>
<td>decreases the size of the equation (compressed letters)</td>
</tr>
</tbody>
</table>

We will terminate, but what about the size?
Lemma (Fixed solution)

There are at most $kn + 2n$ different crossing letters, pairs, father-$i$-leaf pairs
Lemma (Fixed solution)

There are at most $kn + 2n$ different crossing letters, pairs, father-i-leaf pairs

Proof.

Each is associated with an occurrence of a (context) variable.
Lemma (Fixed solution)

There are at most $kn + 2n$ different crossing letters, pairs, father-i-leaf pairs

Proof.

Each is associated with an occurrence of a (context) variable.

Lemma (Fixed solution)

Uncrossing introduces at most $kn + 2n$ letters to the equation.
Lemma (Fixed solution)

There are at most \( kn + 2n \) different crossing letters, pairs, father-\( i \)-leaf pairs.

Proof.

Each is associated with an occurrence of a (context) variable.

Lemma (Fixed solution)

Uncrossing introduces at most \( kn + 2n \) letters to the equation.

Proof.

Each variable pops one up and one down for \( a \)-chains: it is compressed immediately afterwards.
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size $m \rightarrow m'$:

\[
\begin{align*}
\text{Size} & \quad m \rightarrow m' \\
\text{Compressed} & \quad m' \\
\text{With} & \quad m' \\
\text{Better analysis} & \quad O(n^2 k^2) \text{ steps}
\end{align*}
\]
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size $m \rightarrow m'$:

- popped letters: $+kn + 2n$
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size $m \rightarrow m'$:
- popped letters: $+kn + 2n$
- only crossing: only $kn + 2n$ different pairs, chains, $(f, i, c)$
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size $m \rightarrow m'$:
- popped letters: $+ kn + 2n$
- only crossing: only $kn + 2n$ different pairs, chains, $(f, i, c)$
- some covers $\frac{1}{2} \frac{m}{kn+2n}$ letters (requires some argument)
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size $m \rightarrow m'$:

- popped letters: $+kn + 2n$
- only crossing: only $kn + 2n$ different pairs, chains, $(f, i, e)$
- some covers $\frac{1}{2} \frac{m}{kn + 2n}$ letters (requires some argument)

$$m' \leq m - \frac{m}{2(kn + 2n)} + kn + 2n \implies O(n^2k^2)$$
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size $m \rightarrow m'$:

- popped letters: $+kn + 2n$
- only crossing: only $kn + 2n$ different pairs, chains, $(f, i, c)$
- some covers $\frac{1}{2} \frac{m}{kn+2n}$ letters (requires some argument)

$$m' \leq m - \frac{m}{2(kn + 2n)} + kn + 2n \implies \mathcal{O}(n^2k^2)$$

We terminate for positive, never terminate for negative.
Strategy

- If there is something non-crossing: compress it.
- Only crossing: choose one that minimises the equation.

Size \( m \rightarrow m' \):

- popped letters: +\( kn + 2n \)
- only crossing: only \( kn + 2n \) different pairs, chains, \((f, i, c)\)
- some covers \( \frac{1}{2} \frac{m}{kn+2n} \) letters (requires some argument)

\[
m' \leq m - \frac{m}{2(kn + 2n)} + kn + 2n \implies O(n^2k^2)
\]

We terminate for positive, never terminate for negative.

With better analysis

- termination in \( O(poly(n) \log N) \) steps (\( N \): size of the solution)
- space \( O(nk^2) \)
Regular constraints
(Nondeterministic) tree automaton

<table>
<thead>
<tr>
<th>States $Q \supseteq F$: accepting states</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each $f \in \Sigma$ there is $\delta_f \subseteq Q^{\text{rank}(f)+1}$ (transition function/relation)</td>
</tr>
<tr>
<td>$Q$-labelling consistent with transition functions</td>
</tr>
<tr>
<td>Accepting: root label in $F$</td>
</tr>
</tbody>
</table>
Regular languages

(Nondeterministic) tree automaton

States $Q \supseteq F$: accepting states
For each $f \in \Sigma$ there is $\delta_f \subseteq Q^{\text{rank}(f)+1}$ (transition function/relation)
$Q$-labelling consistent with transition functions
Accepting: root label in $F$

Generalize transition to arbitrary trees with holes:
$t$ with $k$ arguments has transition $\delta_t \subseteq Q^{k+1}$:
$\{(q_1, \ldots, q_k, q_{k+1}) : (q_1, \ldots, q_k, q_{k+1}) \in \delta_t \iff \text{there is a } Q\text{-labeling with holes labelled } q_1, \ldots, q_k \text{ and root } q_{k+1}\}$. 
**Regular languages**

(Nondeterministic) tree automaton

- **States** $Q \supseteq F$: accepting states
- For each $f \in \Sigma$ there is $\delta_f \subseteq Q^{\text{rank}(f)+1}$ (transition function/relation)
- $Q$-labelling consistent with transition functions
- Accepting: root label in $F$

Generalize transition to arbitrary trees with holes:
- $t$ with $k$ arguments has transition $\delta_t \subseteq Q^{k+1}$:
- $(q_1, \ldots, q_k, q_{k+1}) \in \delta_t \iff$ there is a $Q$-labeling with holes labelled $q_1, \ldots, q_k$ and root $q_{k+1}$.

$$
\begin{align*}
\delta_f(\cdot, \cdot) &= \{(q, q', p) : p \in \{q + 1, q'\}\} \\
\delta_f(f(\cdot, \cdot), \cdot) &= \{(q, q', q'', p) : p \in \{q + 2, q' + 1, q''\}\}
\end{align*}
$$
Regular constraints

Constraints: given automaton $N$ declare

- $\delta^N_{S(x)} \subseteq Q$
- $\delta^N_{S(X)} \subseteq Q^2$
- $\delta^N_{S(u)} \subseteq Q$
Regular constraints

Constraints: given automaton $N$ declare

- $\delta^N_S(x) \subseteq Q$
- $\delta^N_S(X) \subseteq Q^2$
- $\delta^N_S(u) \subseteq Q$

Do this separately for several automata.
Vector of transition functions.
Regular constraints

Constraints: given automaton $N$ declare

- $\delta^N_{S(x)} \subseteq Q$
- $\delta^N_{S(X)} \subseteq Q^2$
- $\delta^N_{S(u)} \subseteq Q$

Do this separately for several automata.

Vector of transition functions.

- EXPTIME-hard (nonemptiness of intersection of tree automata)
- NP-equivalent to other natural formulations
Regular constraints

Constraints: given automaton $N$ declare

- $\delta^N_{S(x)} \subseteq Q$
- $\delta^N_{S(X)} \subseteq Q^2$
- $\delta^N_{S(u)} \subseteq Q$

Do this separately for several automata.

Vector of transition functions.

- EXPTIME-hard (nonemptiness of intersection of tree automata)
- NP-equivalent to other natural formulations

Extend the algorithm.
### Easy

**Compression:** When $a, b$ are compressed: 
\[ \delta_c \leftarrow \delta_a \circ \delta_b \]

**Popping:** when $x$ pops $a$: set $\delta'_x$ so that 
\[ \delta_x = \delta_a \circ \delta'_x \]
Extensions

Easy

Compression: When $a, b$ are compressed: $\delta_c \leftarrow \delta_a \circ \delta_b$

Popping: when $x$ pops $a$: set $\delta'_x$ so that $\delta_x = \delta_a \circ \delta'_x$

Less easy

- Size of the transition function?
- Size of the signature?

We cannot trim—this affects transition function.
Transition functions
Each needed transition $\delta$ is in one of the forms:
- $\delta \in Q$, so with 0 parameters (not so many)
- $\delta \in Q^2$, so with 1 parameter (not so many)
- is a projection of a transition from the input
<table>
<thead>
<tr>
<th>Transition functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each needed transition $\delta$ is in one of the forms:</td>
</tr>
<tr>
<td>- $\delta \in Q$, so with 0 parameters (not so many)</td>
</tr>
<tr>
<td>- $\delta \in Q^2$, so with 1 parameter (not so many)</td>
</tr>
<tr>
<td>- is a projection of a transition from the input</td>
</tr>
</tbody>
</table>

Consequence of the compression operations. This is **not robust**!
Transition functions

Each needed transition $\delta$ is in one of the forms:

- $\delta \in Q$, so with 0 parameters (not so many)
- $\delta \in Q^2$, so with 1 parameter (not so many)
- is a projection of a transition from the input

Consequence of the compression operations. This is not robust!

Signature

Add $a_{\delta_1, \ldots, \delta_\ell}$ to $\Sigma$ for each $\delta_1, \ldots, \delta_\ell$ as above and realised by a tree.

- equisatisfiable
- this allows trimming
- $\Sigma$ is stored using an EXPTIME oracle (verified at popping)
Transition functions

Each needed transition \( \delta \) is in one of the forms:
- \( \delta \in Q \), so with 0 parameters (not so many)
- \( \delta \in Q^2 \), so with 1 parameter (not so many)
- is a projection of a transition from the input

Consequence of the compression operations. This is not robust!

Signature

Add \( a_{\delta_1, \ldots, \delta_\ell} \) to \( \Sigma \) for each \( \delta_1, \ldots, \delta_\ell \) as above and realised by a tree.
- equisatisfiable
- this allows trimming
- \( \Sigma \) is stored using an EXPTIME oracle (verified at popping)
  - EXPTIME: exponentially many different transition vectors
  - sometimes the oracle is simpler (below EXPTIME)
Open problems

- computational complexity (maybe in NP?)
  - lower bound already for very simple word equations (encoding of integer programming)
  - enough to show exponential size of smallest solution
Open problems

- computational complexity (maybe in NP?)
  - lower bound already for very simple word equations (encoding of integer programming)
  - enough to show exponential size of smallest solution
- One-step term rewriting
  - existential formulas: solvable by context unification
  - positive theory?
  - existential theory?
Open problems

- Computational complexity (maybe in NP?)
  - Lower bound already for very simple word equations (encoding of integer programming)
  - Enough to show exponential size of smallest solution

- One-step term rewriting
  - Existential formulas: solvable by context unification
  - Positive theory?
  - Existential theory?

- Fragment with one context variable (in P?)