Uniwersytet Wrocławski

Deciding Context Unification with regular constraints

Artur Jeż
University of Wrocław
Developments in Language Theory
Warsaw 08.08.2019

Equations over terms

- Equations over terms (unification)

Equations over terms

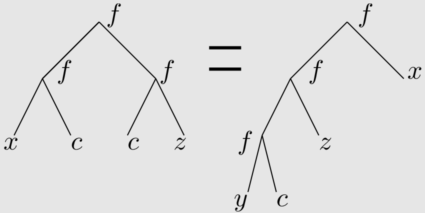
- Equations over terms (unification)
- What can the variables represent?

(First order) term unification

Variables represent closed terms

(First order) term unification

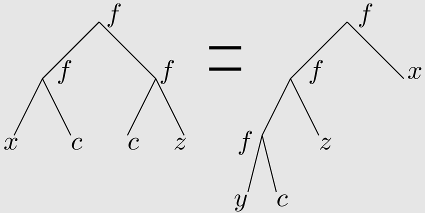
Variables represent closed terms



Uniwersytet Wrocławski

(First order) term unification

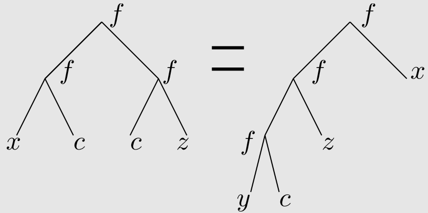
Variables represent closed terms: polynomial



- iterative decomposition

(First order) term unification

Variables represent closed terms: polynomial



- iterative decomposition

Not covered: “functions”

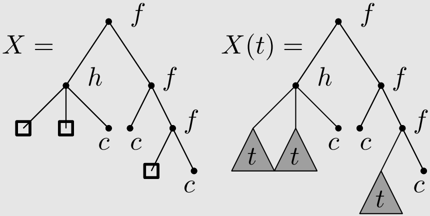
- open terms
- λ-terms
- ...

Second order unification

- Variables have arguments that can be used. Terms with ‘holes’.

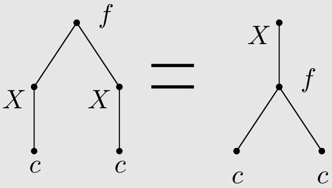
Second order unification

- Variables have arguments that can be used. Terms with 'holes'.



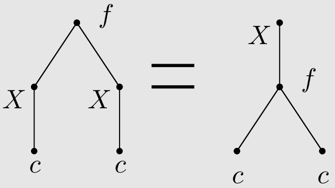
Example

Equation

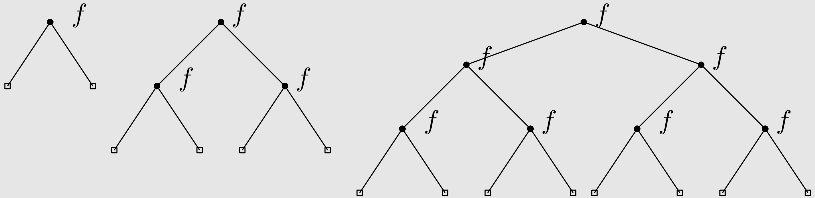


Example

Equation



Solutions: full binary trees



Problems

Undecidable.

Problems

Undecidable.

In very restricted cases:

- one argument
- one binary symbol
- ...

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 $\in \Sigma^*$.
- $aXbXYbbb = XabaabYbY$ $(X = aa, Y = bb)$

Word equations

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

Restrictions of second order unification

Word equations

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

Decidable. [Makanin $\to$ Plandowski (PSPACE)]

Context unification

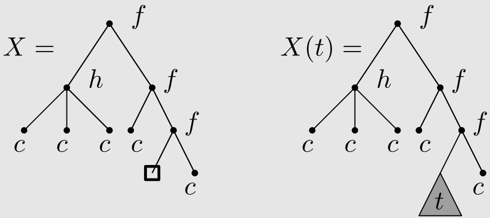
In words equation arguments are used once.

In words equation arguments are used once.

Context unification

Second order unification

Substitution for X uses argument exactly once.



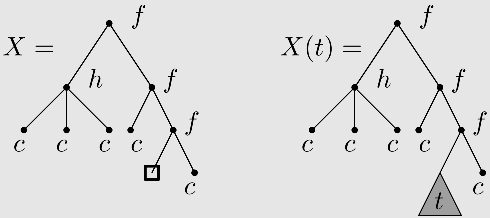
Context unification

In words equation arguments are used once.

Context unification

Second order unification

Substitution for X uses argument exactly once.



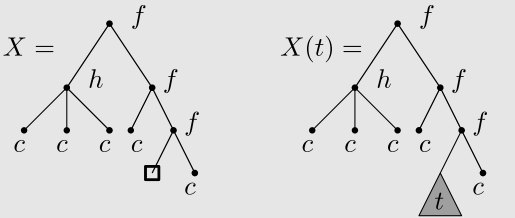
Easy reduction: k-arguments $\to$ one argument (NP-reduction)

In words equation arguments are used once.

Context unification

Second order unification

Substitution for X uses argument exactly once.



Easy reduction: k-arguments $\to$ 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

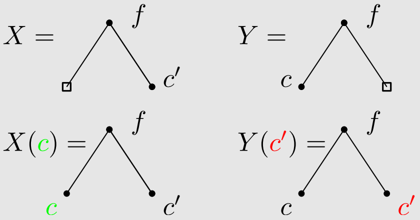
- to a term variable: a closed term over a signature
- to a context variable: a term over a signature plus a unique occurrence of 'hole' $(\bullet)$
- extend in a natural way
  $S(X(t))$: in $S(X)$ we replace $\bullet$ with $S(t)$.

Example

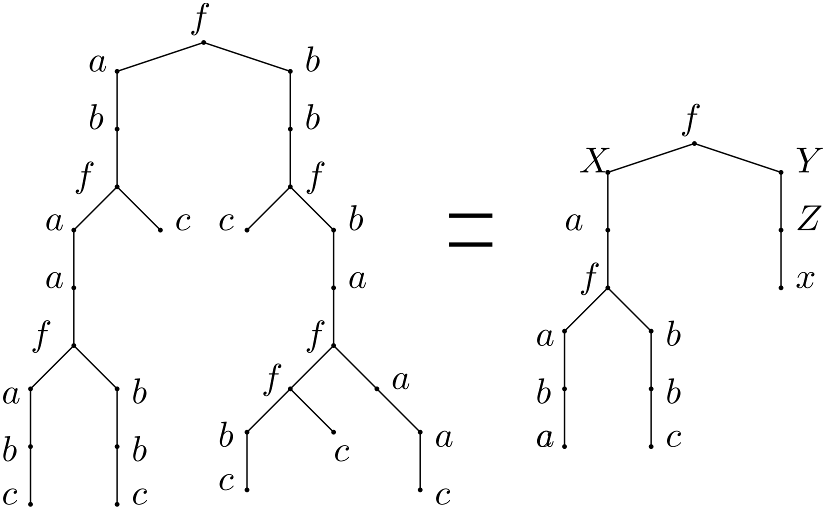
- f: arity 2, c, c': arity 0
- $X(c) = Y(c')$

Example

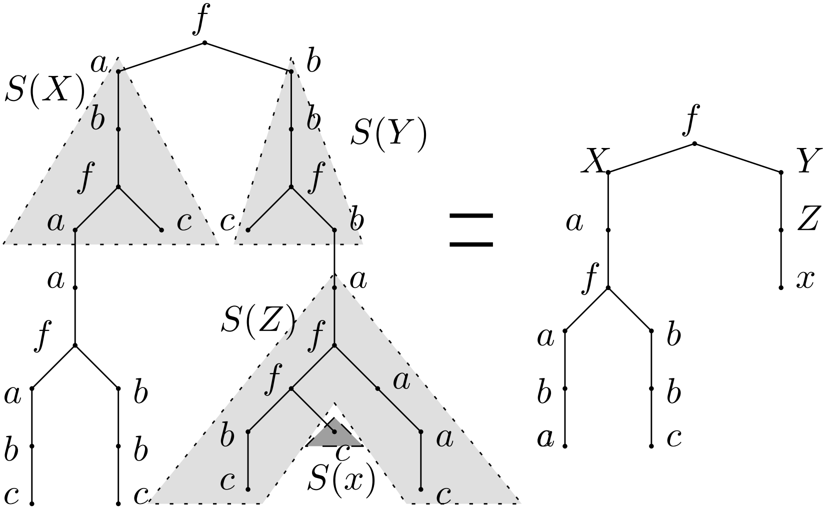
- f: arity 2, c, c': arity 0
- $X(c) = Y(c')$



Larger example



Larger example



Regular constraints

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)

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 $\implies$ 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 $\implies$ undecidability
- equivalent to linear second order unification
- regular constraints are somehow expressive
- very successful for word equations

- representation of Ls matters for computational complexity.
- some representations are easier to handle

Why

In between important problems:

- word equations , first order-unification
- second-order unification

Why

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)

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)

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

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

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

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.

This talk

Context unification is in PSPACE.

This talk

Context unification is in PSPACE.

Trivially NP-hard

This talk

Context unification is in PSPACE.

Trivially NP-hard

- generalises earlier (simpler) solution for word equations
- applies compression to the equation

This talk

Context unification is in PSPACE.

Trivially NP-hard

- generalises earlier (simpler) solution for word equations
- applies compression to the equation

- quite natural
- proof reasonable

This talk

Context unification is in PSPACE.

Trivially NP-hard

- generalises earlier (simpler) solution for word equations
- applies compression to the equation

- quite natural
- proof reasonable

With regular constraints: in EXPTIME
Easily EXPTIME-hard.

Preliminary notes: signature

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

Preliminary notes: signature

- 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?

Preliminary notes: signature

- 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).

Preliminary notes: signature

- 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

Preliminary notes: signature

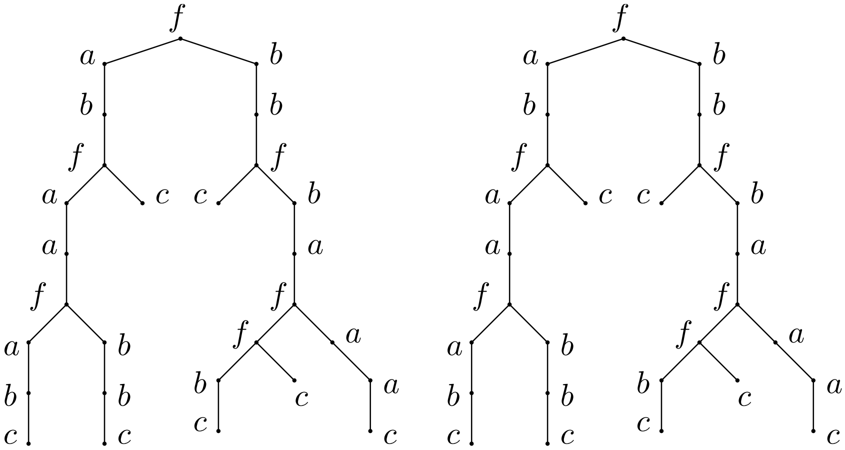
- The solution may have elements not present in the equation:
  - $\Sigma = \{f/2, c/0, c'/0\}$
  - $X(c) = Y(c')$
  - $S(X) = f(\bullet, c')$, $S(Y) = f(c, \bullet)$
- We introduce new symbols to the signature—problem?

Not a problem: Trimming

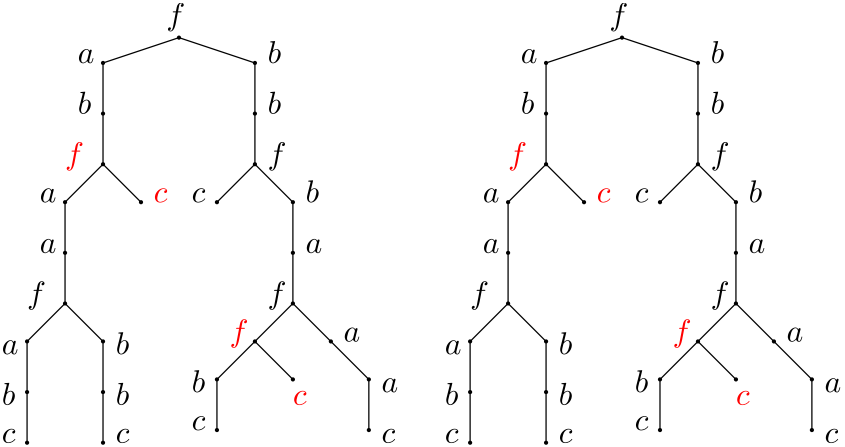
If the context equation has a solution then it has it over the signature of letters in the equation (plus arbitrary constant and binary symbol).

- 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 $f_k(t_1, \dots, t_k)$ with $g(t_1, g(\dots g(t_{k-1}, t_k)\dots))$
  - $f_1(t_1) \to t_1$
  - $c_0 \to c$

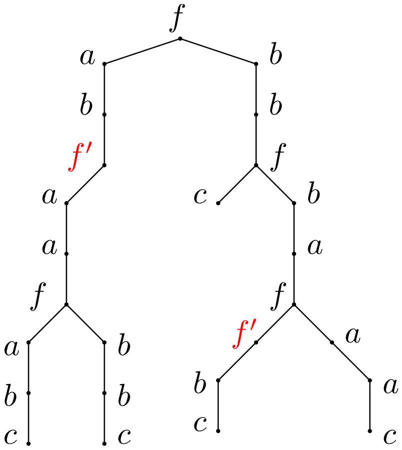
Local compressions



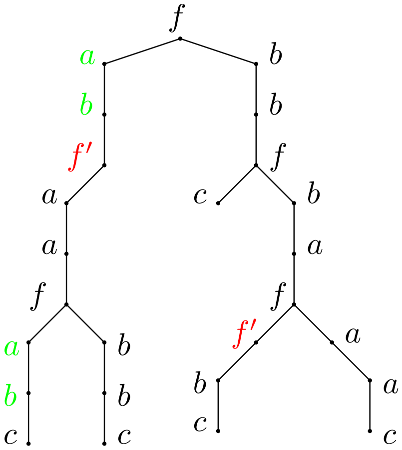
Local compressions



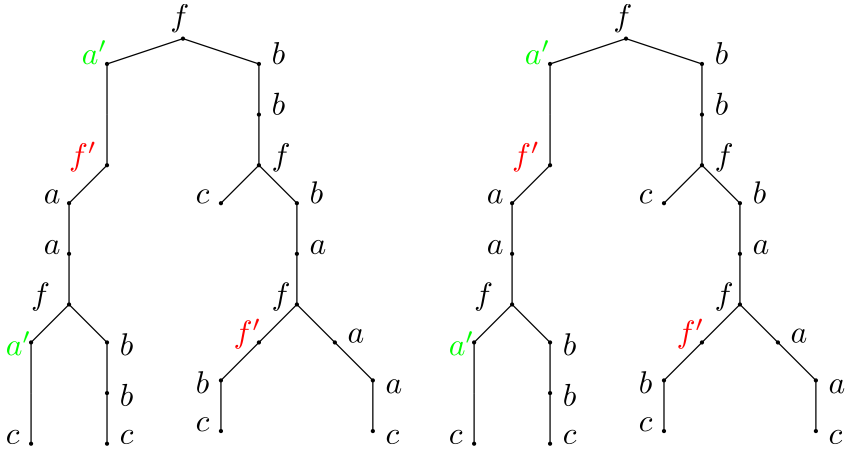
Local compressions



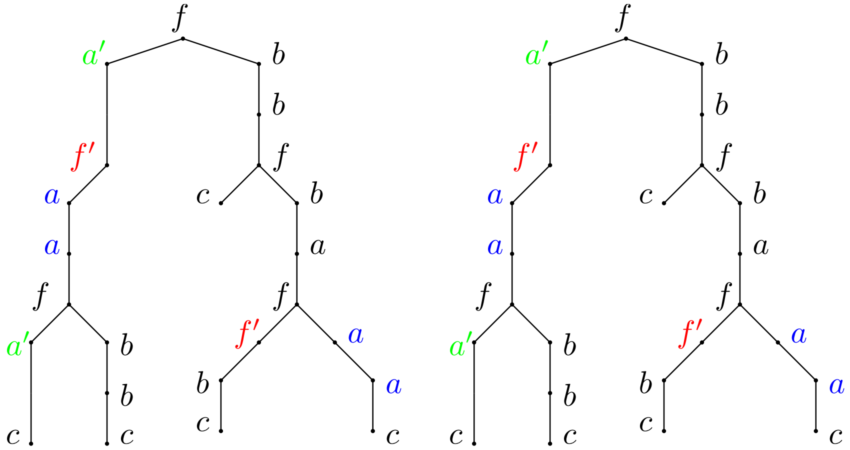
Local compressions



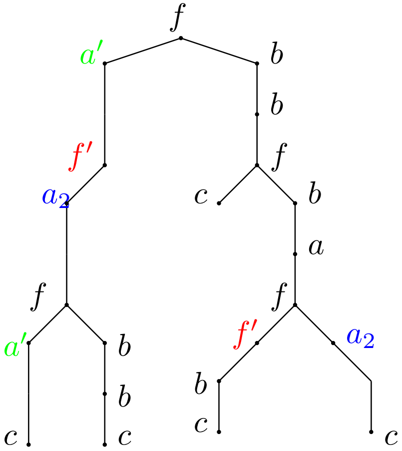
Local compressions



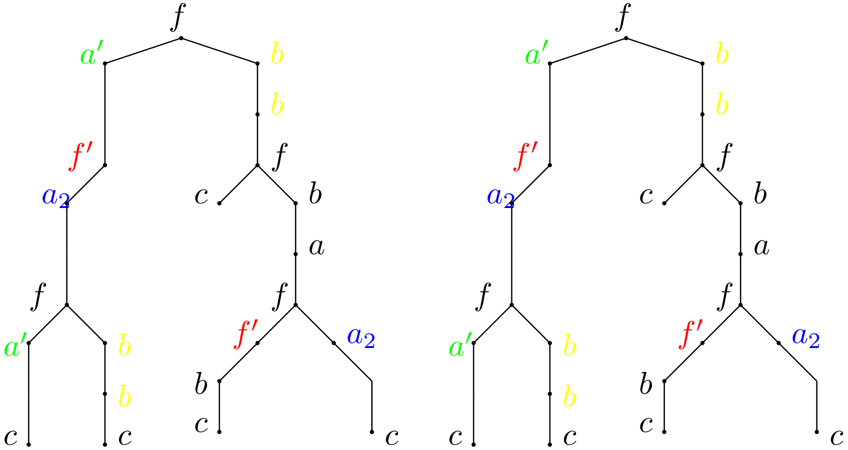
Local compressions



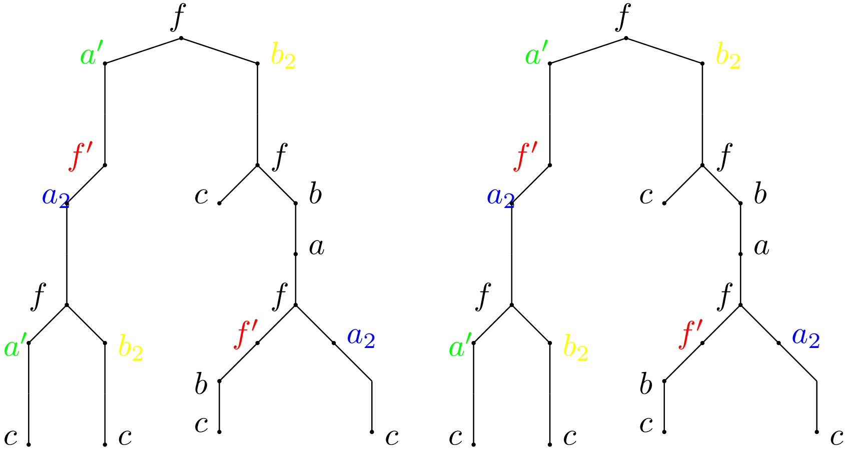
Local compressions



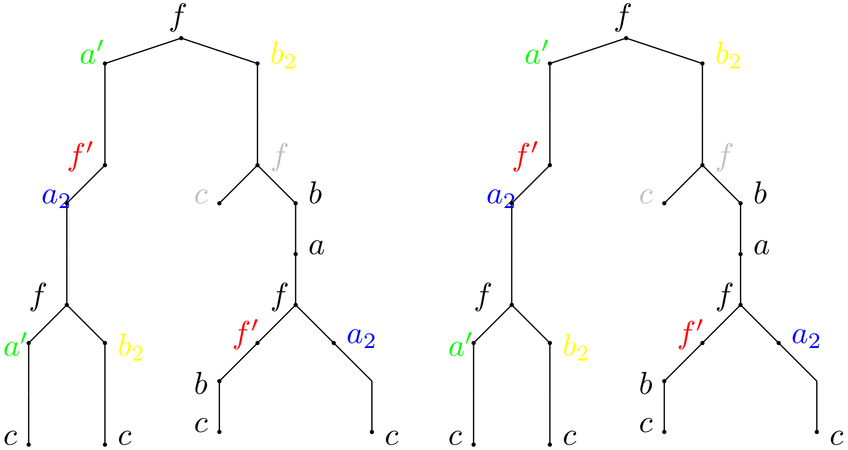
Local compressions



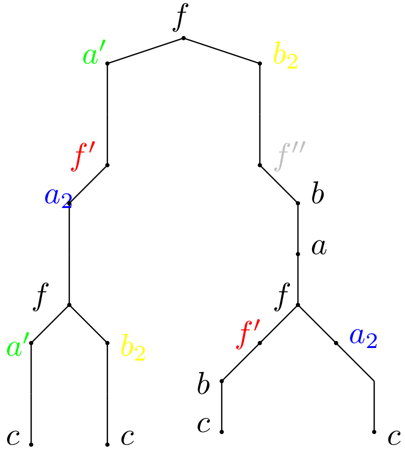
Local compressions



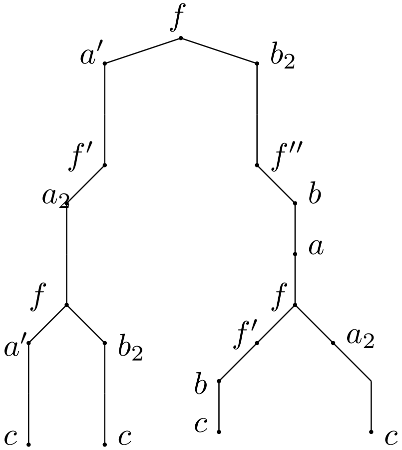
Local compressions

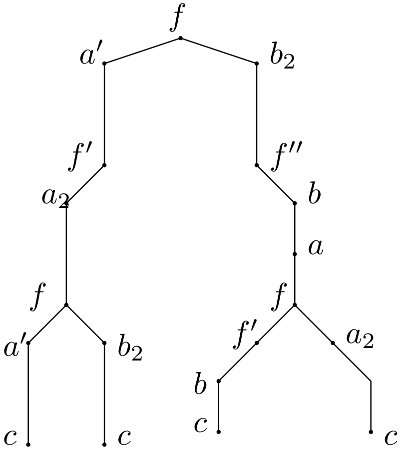


Local compressions



Local compressions





Idea

Idea

Use this approach for solving context equations.

- apply simple compression operations to the solutions

Idea

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

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.

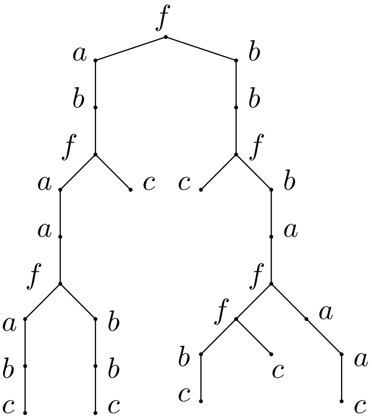
- apply simple compression operations to the solutions
- perform them directly on the equation
  changes of the equation
- keep the size small:
  choose a compression that preserves quadratic size of the equation
- this yields PSPACE

(f, i, c)-leaf compression

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

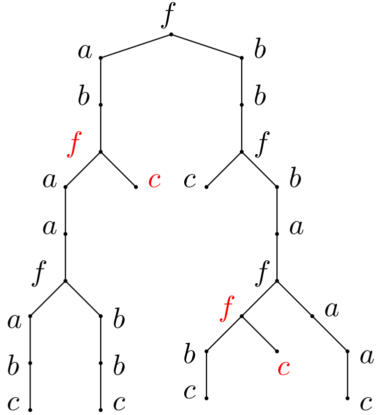
(f, i, c)-leaf compression

- '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 $f'(t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$



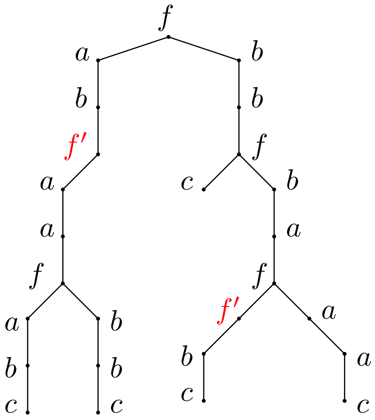
(f, i, c)-leaf compression

- '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 $f'(t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$



(f, i, c)-leaf compression

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



a, b-pair compression

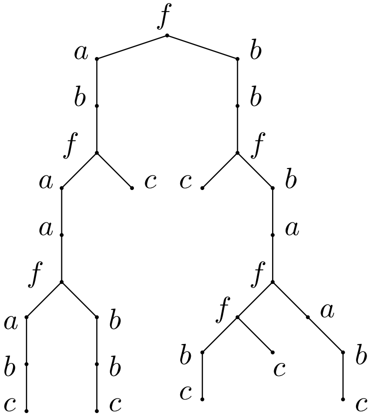
ab-compression

Replace each occurrence of ab with d.

a, b-pair compression

ab-compression

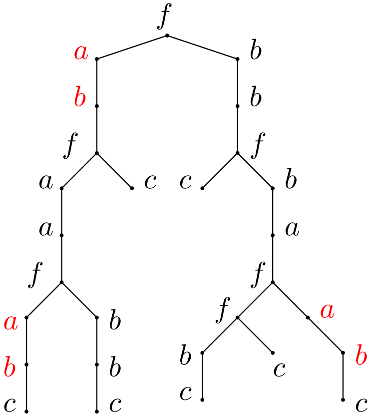
Replace each occurrence of ab with d.



a, b-pair compression

ab-compression

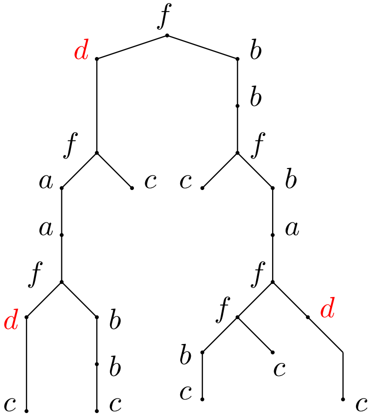
Replace each occurrence of ab with d.



a, b-pair compression

ab-compression

Replace each occurrence of ab with d.



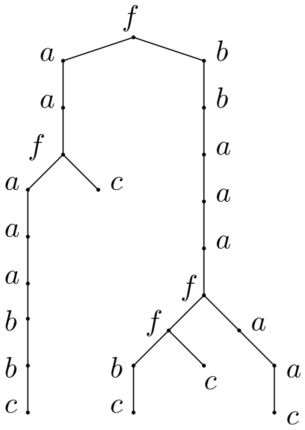
a-chain compression

a-compression

Replace each maximal chain a^ℓ with a_ℓ, for all ℓ.

a-compression

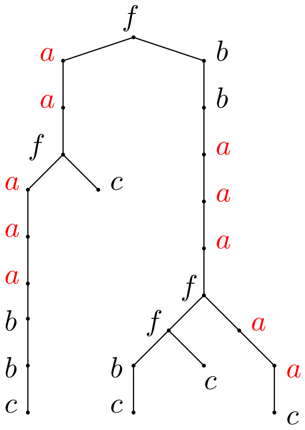
Replace each maximal chain a^ℓ with a_ℓ, for all ℓ.



a-chain compression

a-compression

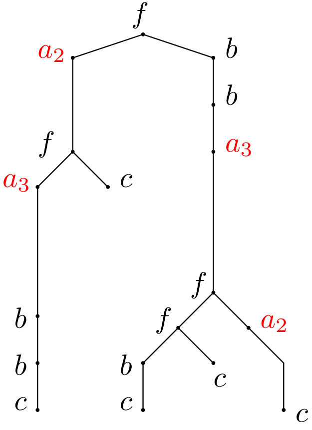
Replace each maximal chain a^ℓ with a_ℓ, for all ℓ.



a-chain compression

a-compression

Replace each maximal chain a^ℓ with a_ℓ, for all ℓ.



Moving to the equation.

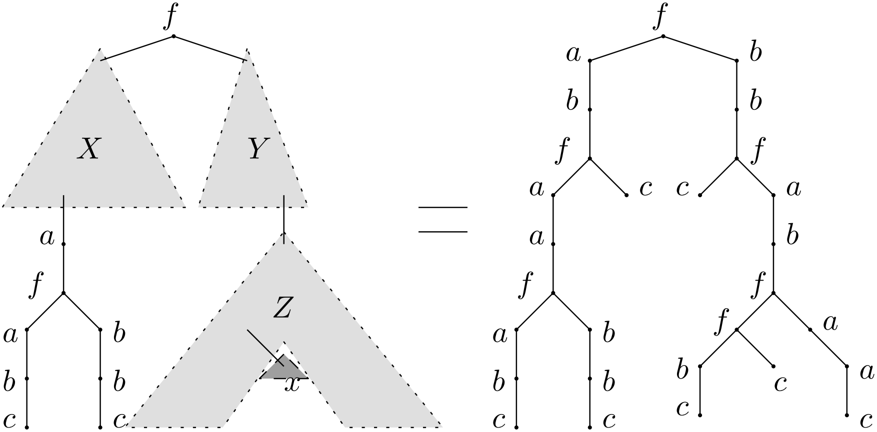
Easy case (pair compression)

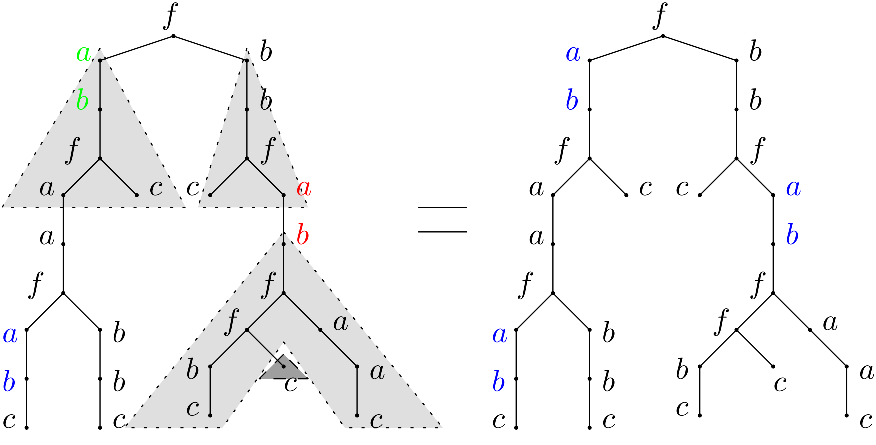
Definition (pair types)

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.


f
X
Y
a
f
a
b
b
b
c
c
Z
x
=
f
a
b
b
b
f
f
a
c
c
a
a
b
f
f
f
a
a
b
b
c
b
b
c
a
c
c
c



Non-crossing Pair Compression

PairNCrComp

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

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.*

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.

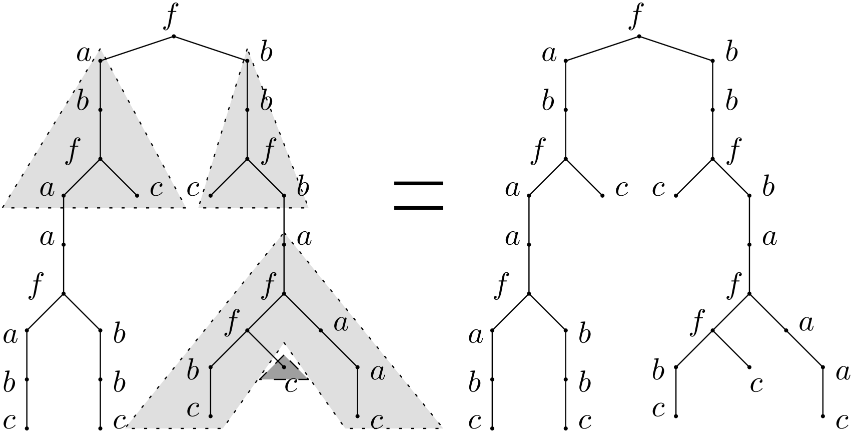
Every ab is replaced:

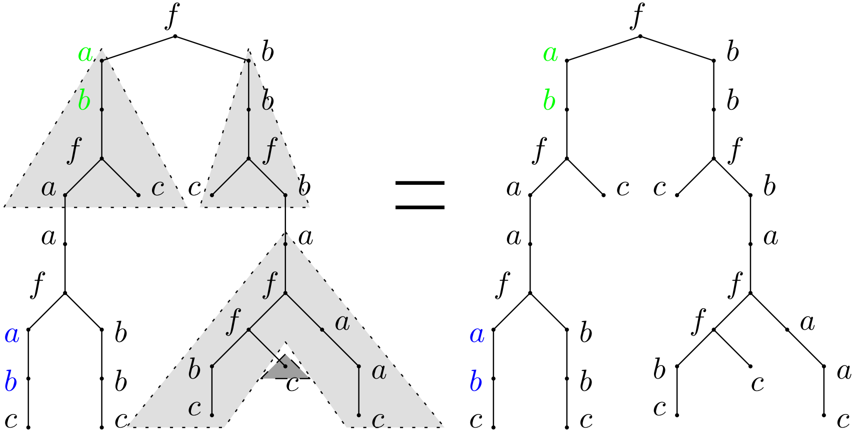
explicit pairs replaced explicitly

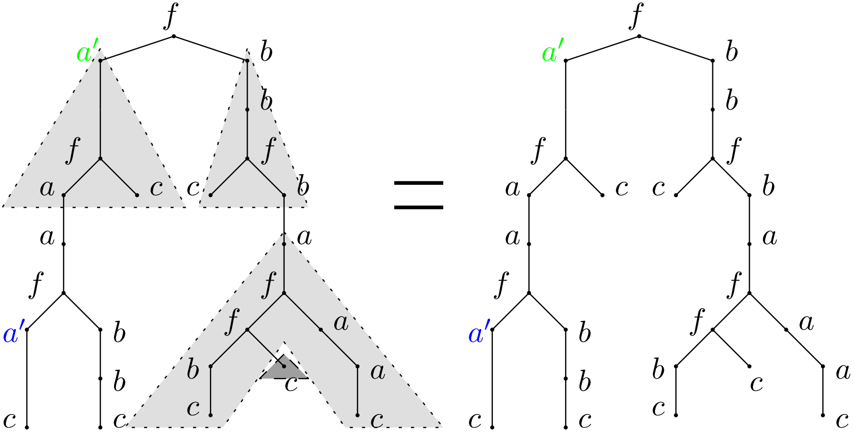
implicit pairs replaced implicitly (in the solution)

crossing there are none

$\square$







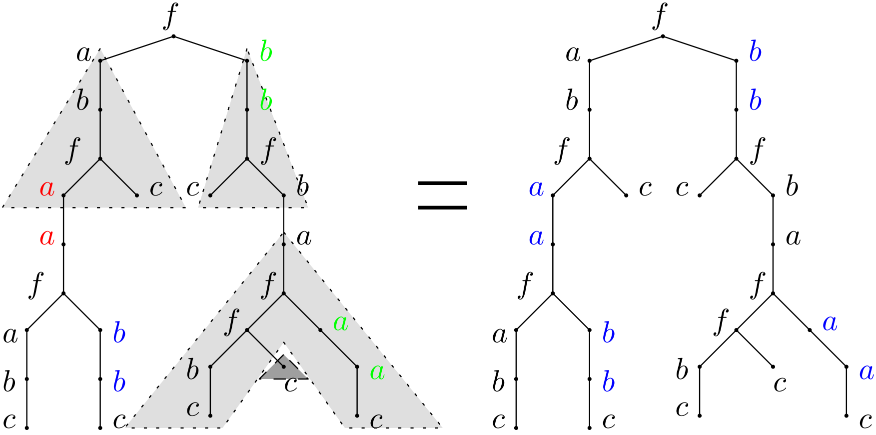
f
a′ b
b
f f
a c c b
a a
f f
f a
a′ b
b a
b c
c c c c
=
f
a′ b
b
f f
a c c b
a a
f f
f a
a′ b
b a
b c
c c
c c

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

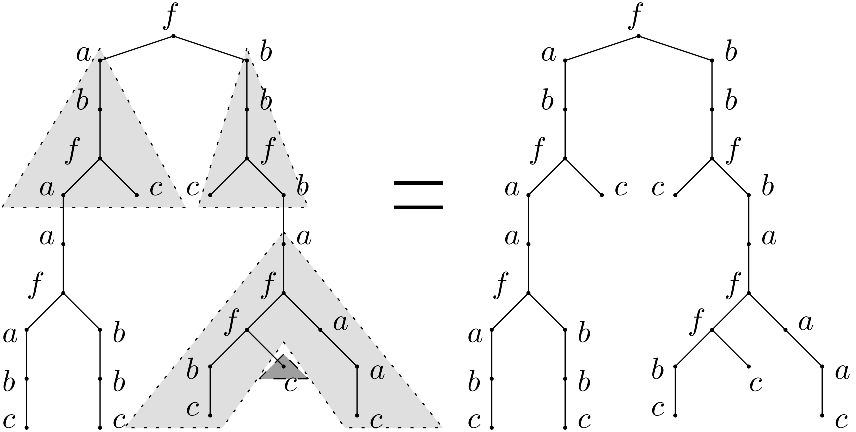
1: **for** $\ell > 1$ **do**
2: replace each explicit maximal a^ℓ in the equation by a_ℓ

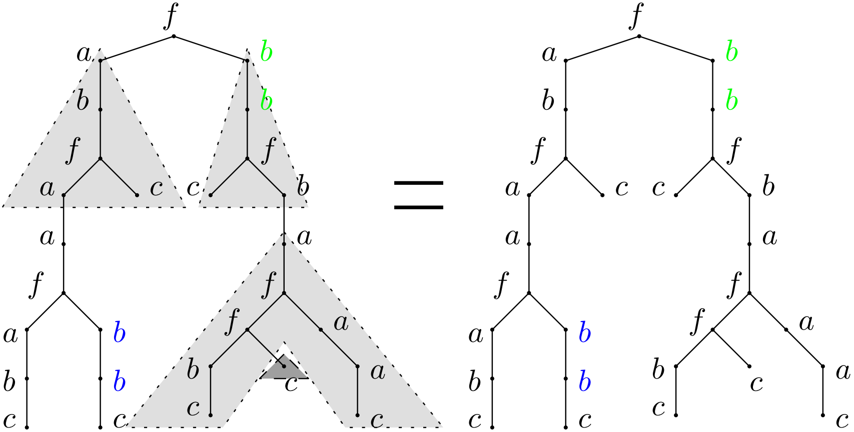
ChainNCr

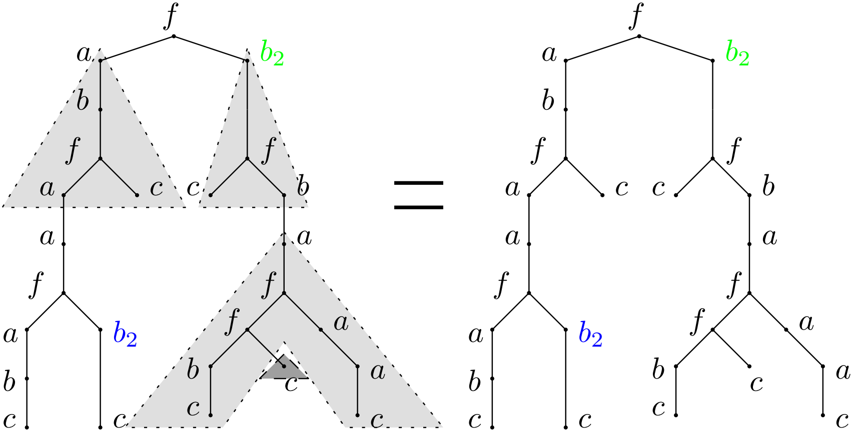
for $\ell > 1$ **do**
    replace each explicit maximal a^ℓ in the equation by a_ℓ

Lemma

ChainNCr(a) *compresses noncrossing a-chains.*





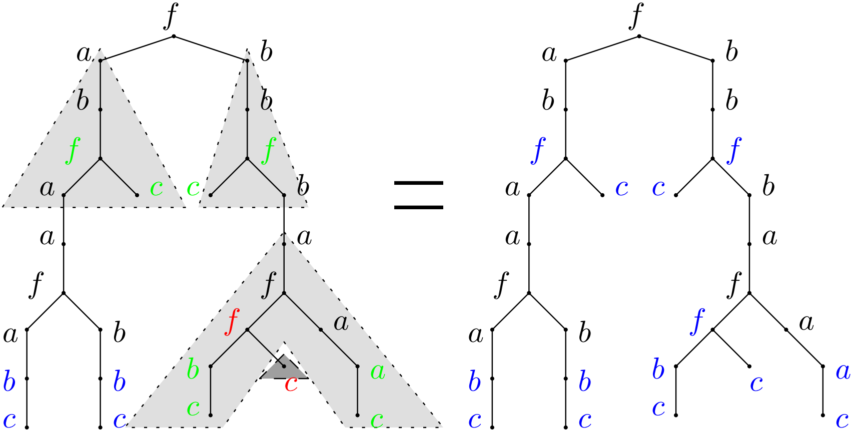

f
a
b_2
b
f
f
a
c
c
b
a
a
f
f
a
b_2
f
a
b
b
a
c
c
c
c
c
=
f
a
b_2
b
f
f
a
c
c
b
a
a
f
f
a
b_2
f
a
b
b
a
c
c
c
c
c

Definition (father-i-leaf pair)

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

- **explicit** it comes from the equation;
- **implicit** 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.



Leaf compression ctd.

LeafNCr(f, i, c)

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

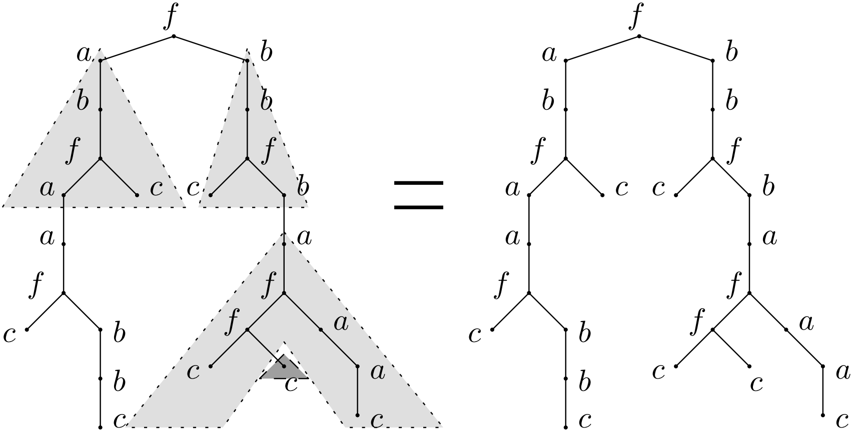
Leaf compression ctd.

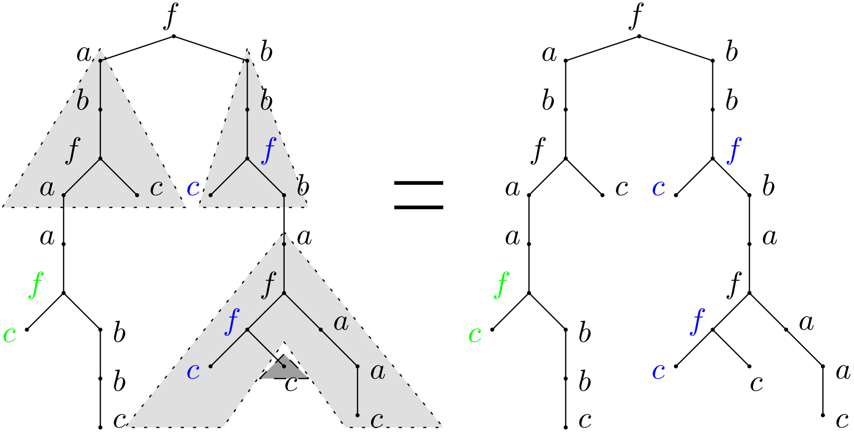
LeafNCr(f, i, c)

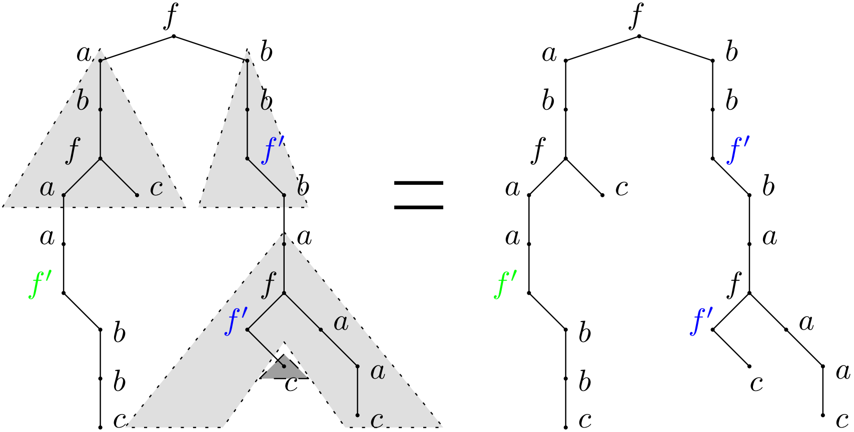
1: replace f with c at positions i by f'

Lemma

LeafNCr *performs leaf compression when* (f, i, c) *is non-crossing.*


f f
a b a b
b b b b
f f f f
a c c b = a c c b
a a a a
f f f f
c b f a c b f a
b c c a c c a
c c c c

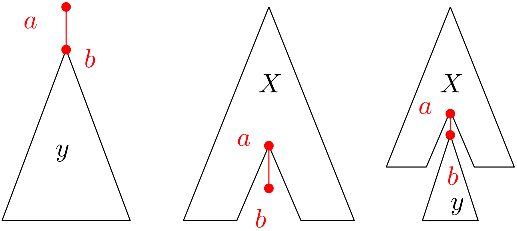

f
a b
b b
f f
a c c b
a a
f f
f a
c b
c a
b c
c c
=
f
a b
b b
f f
a c c b
a a
f f
f a
c b
c a
b c
c c



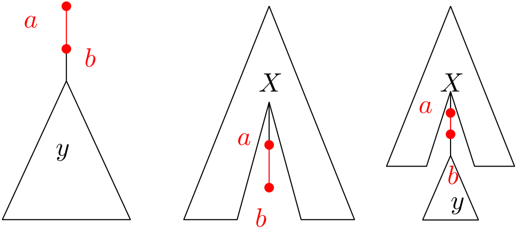
f
a
b
b
b
f
f′
a
c
b
a
a
f′
f
f′
a
b
a
b
c
c
c
=
f
a
b
b
b
f
f′
a
c
b
a
a
f′
f
f′
a
b
a
b
c
c
c

Uncrossing

Uncrossing pairs

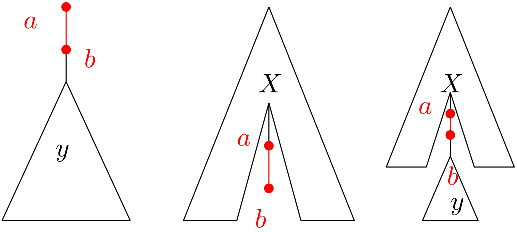


Uncrossing pairs



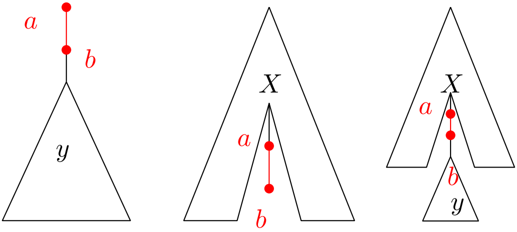
- replace Y with bY and replace X with Xa
  implicitly change $S(Y) = bt, S(X) = t'a$ to $S(Y) = t$, $S(X) = t'$

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

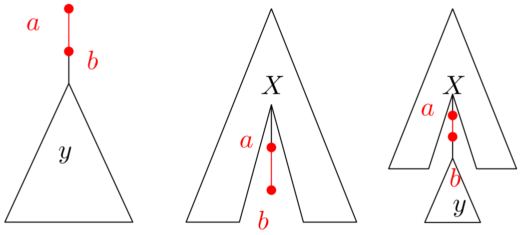


- replace Y with bY and replace X with Xa
  implicitly change $S(Y) = bt, S(X) = t'a$ to $S(Y) = t$, $S(X) = t'$
- If $S(Y)$ is empty then remove Y.

Lemma

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

Uncrossing pairs

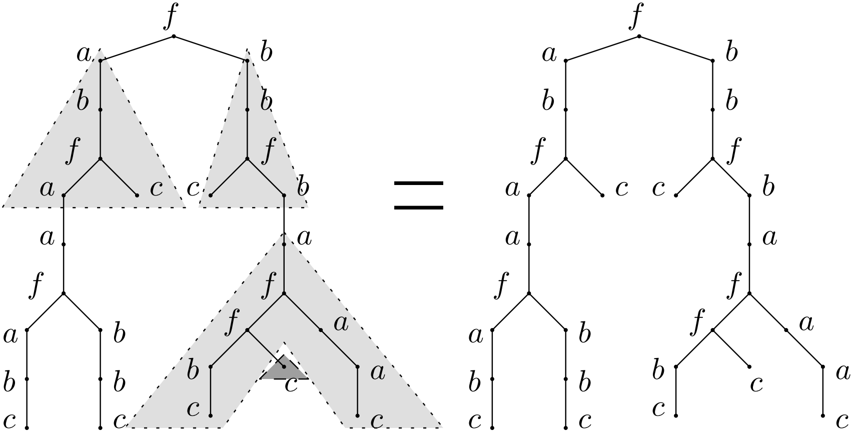


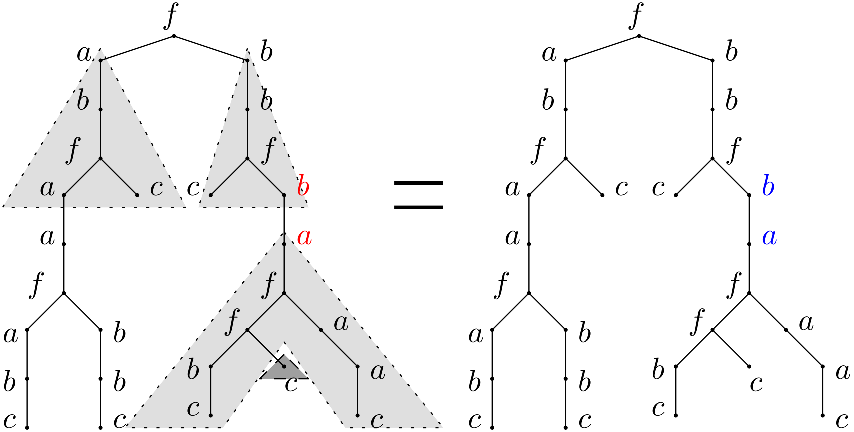
- replace Y with bY and replace X with Xa
  implicitly change $S(Y) = bt, S(X) = t'a$ to $S(Y) = t$, $S(X) = t'$
- If $S(Y)$ is empty then remove Y.

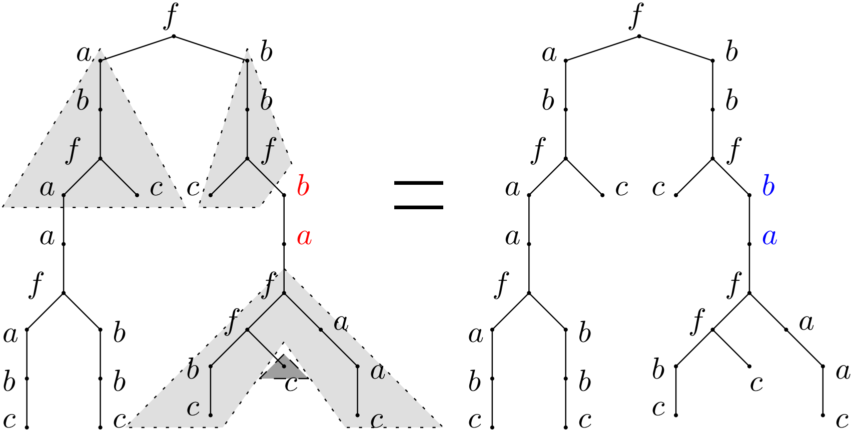
Lemma

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

Compress it!


f f
a b a b
b b b b
f f f f
a c c b a c c b
a a a a
f f f f
a b f a a b f a
b b b a b b b a
c c c c c c c c c c
=





Uniwersytet Wrocławski

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 $a^{\ell_X} X a^{r_X}$
  - they cannot be too long: exponential

Crossing a-chains?

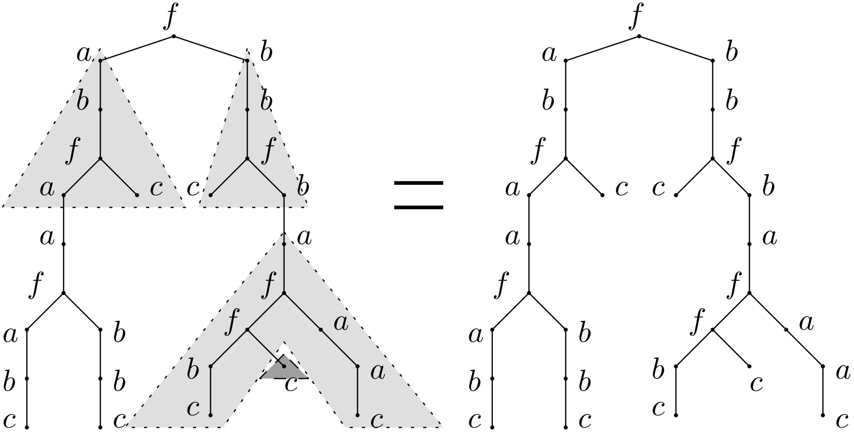
- Crossing a-chain: similar to crossing ab (equiv. to crossing aa).
- pop whole a-prefix and a-suffix
  - $S(X) = a^{\ell_X} t a^{r_X}$: change it to $S(X) = t$
  - replace X in equation by $a^{\ell_X} X a^{r_X}$
  - they cannot be too long: exponential
- do for all variables

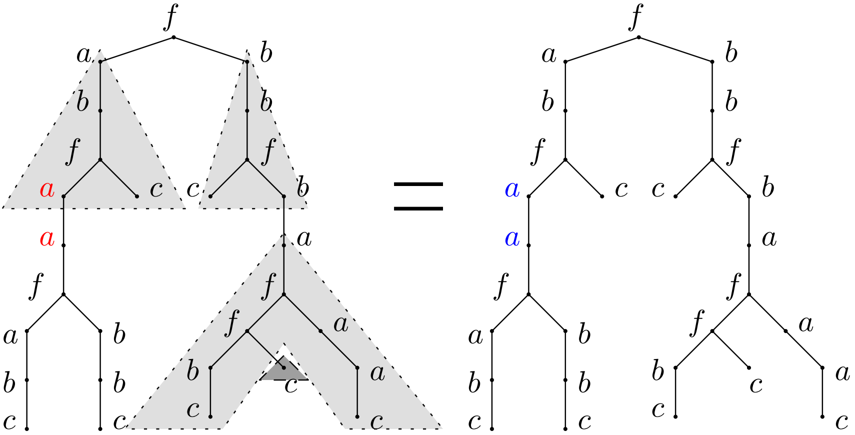
Crossing a-chains?

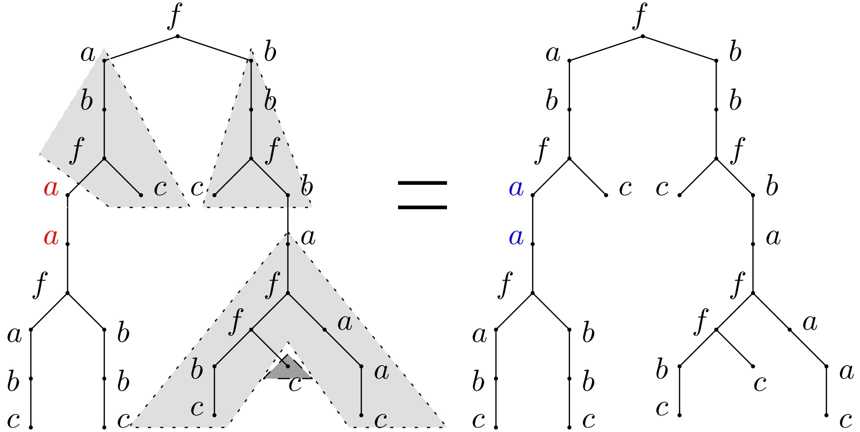
- Crossing a-chain: similar to crossing ab (equiv. to crossing aa).
- pop whole a-prefix and a-suffix
  - $S(X) = a^{\ell_X} t a^{r_X}$: change it to $S(X) = t$
  - replace X in equation by $a^{\ell_X} X a^{r_X}$
  - they cannot be too long: exponential
- do for all variables
- a has no crossing chains!

Crossing a-chains?

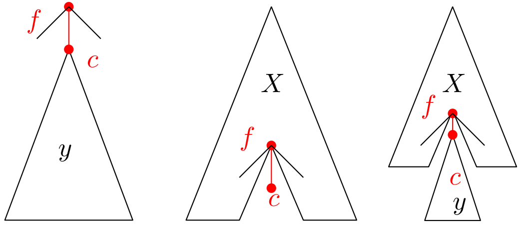
- Crossing a-chain: similar to crossing ab (equiv. to crossing aa).
- pop whole a-prefix and a-suffix
  - $S(X) = a^{\ell_X} t a^{r_X}$: change it to $S(X) = t$
  - replace X in equation by $a^{\ell_X} X a^{r_X}$
  - they cannot be too long: exponential
- do for all variables
- a has no crossing chains!
- compress a-chains



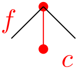


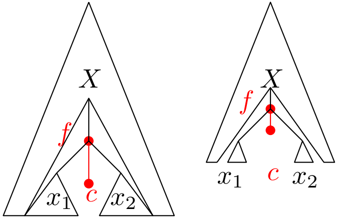


Uncrossing father-leaf pair



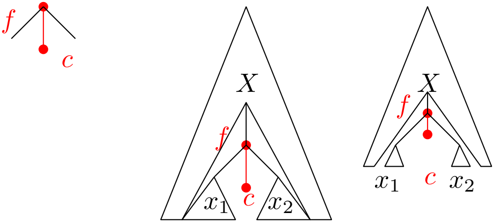
Uncrossing father-leaf pair





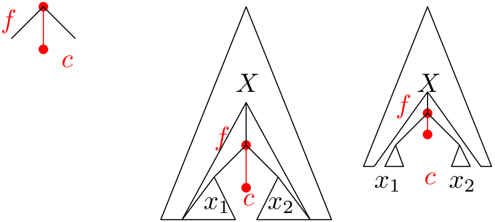
- replace y with c
- replace X with $X(f(x_1, \dots x_{i-1}, \bullet, x_{i+1}, \dots, x_\ell))$

Uncrossing father-leaf pair



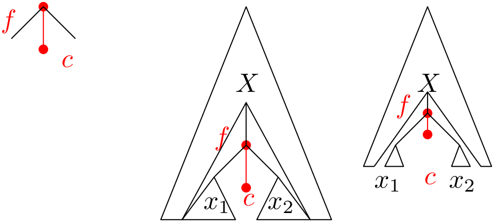
- replace y with c
- replace X with $X(f(x_1, \dots x_{i-1}, \bullet, x_{i+1}, \dots, x_\ell))$
  new variables $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell$

Uncrossing father-leaf pair



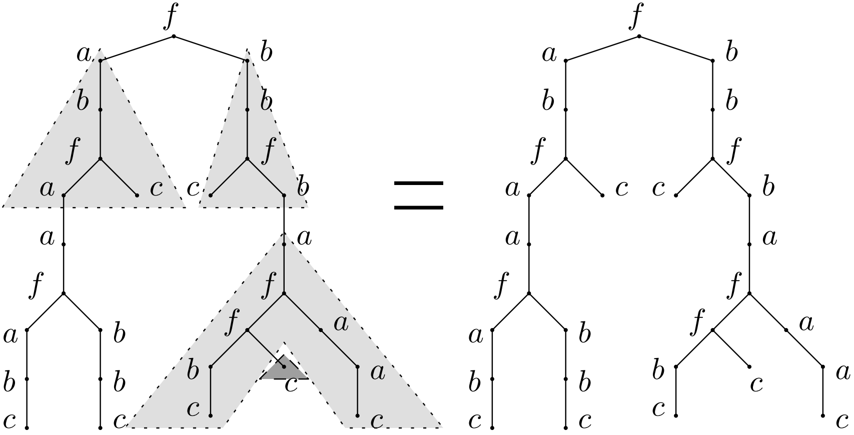
- replace y with c
- replace X with $X(f(x_1, \dots x_{i-1}, \bullet, x_{i+1}, \dots, x_\ell))$
  new variables $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell$
- remove X when $S(X)$ is empty

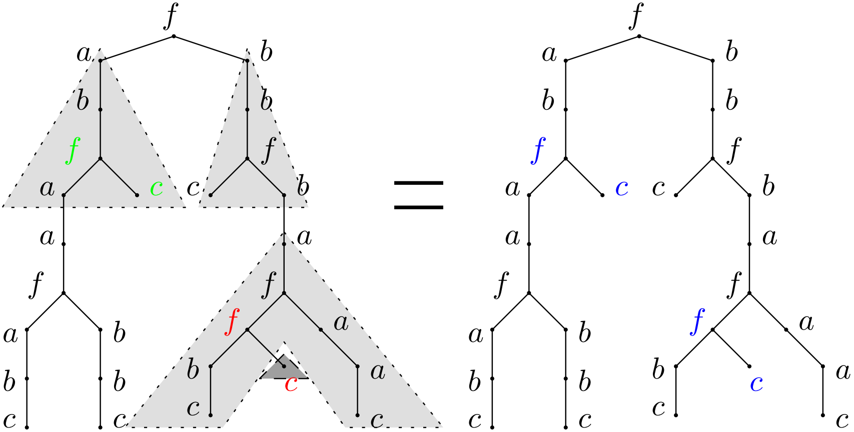
Uncrossing father-leaf pair

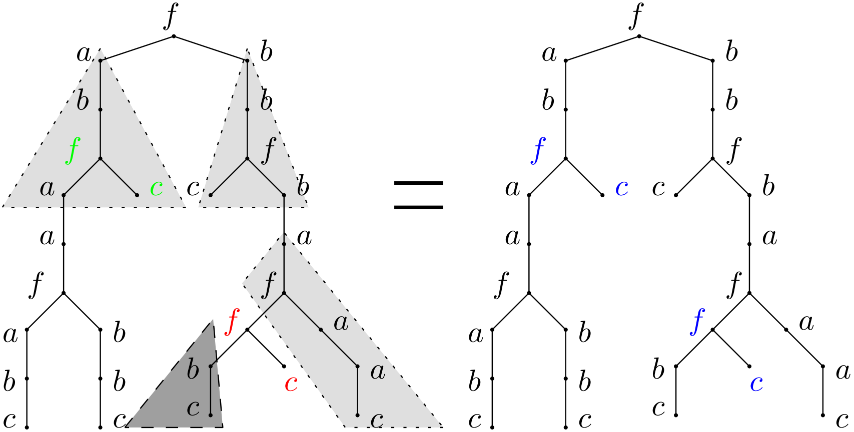


- replace y with c
- replace X with $X(f(x_1, \dots x_{i-1}, \bullet, x_{i+1}, \dots, x_\ell))$
  new variables $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell$
- remove X when $S(X)$ is empty

- there are no crossing (f, i, c) father-i-leaf pairs
- compress them






f
a b
b b
f f
a c c b
a a
f f
a b f a
b b b c a
c c c c
=
f
a b
b b
f f
a c c b
a a
f f
a b f a
b b b a
c c c c

The algorithm

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

Arity

Lemma

The maximal arity of letters in Σ does not increase.

Arity

Lemma

The maximal arity of letters in Σ does not increase.

Does not depend on the nondterministic choices.

Arity

Lemma

The maximal arity of letters in Σ does not increase.

Does not depend on the nondterministic choices.

Proof.

Compression operations do not increase arity. $\square$

Arity

Lemma

The maximal arity of letters in Σ does not increase.

Does not depend on the nondterministic choices.

Proof.

Compression operations do not increase arity. $\square$

Call it k.

Controlling the variables

Lemma

There are at most

- n *context variables*
- kn *variables*

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

Controlling the variables

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.

Controlling the variables

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.

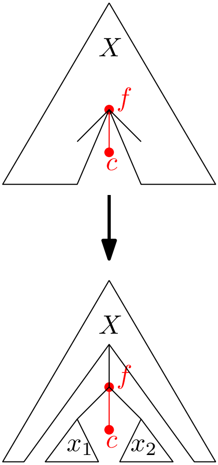
Proof.

- 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 $\square$

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.



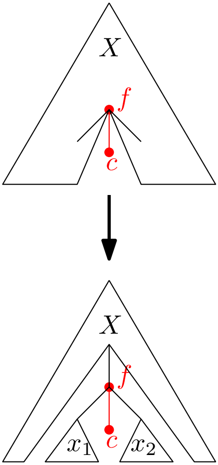
Controlling new variables: details

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



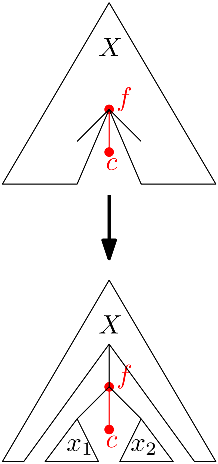
Controlling new variables: details

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



There are at most kn variables

Correctness

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

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.

Make the choices according to the solution.

Size

Noncrossing

- decreases the size of the solution

Size

Noncrossing

- decreases the size of the solution
- decreases the size of the equation

Size

Noncrossing

- decreases the size of the solution
- decreases the size of the equation

Crossing

Size

Noncrossing

- decreases the size of the solution
- decreases the size of the equation

Crossing

- decreases the size of the solution

Size

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)

Size

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)

Size

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)

We will terminate, but what about the size?

Crossing: analysis

Lemma (Fixed solution)

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

Crossing: analysis

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. □

Crossing: analysis

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.

Crossing: analysis

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 $m \to m'$:

Strategy

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

Size $m \to m'$:

- popped letters: $+kn + 2n$

Strategy

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

Size $m \to m'$:

- 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 $m \to m'$:

- 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 $m \to m'$:

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

Strategy

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

Size $m \to m'$:

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

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 $m \to m'$:

- 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.

With better analysis

- termination in $\mathcal{O}(poly(n) \log N)$ steps (N: size of the solution)
- space $\mathcal{O}(nk^2)$

Regular constraints

(Nondterministic) tree automaton

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

(Nondterministic) tree automaton

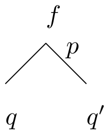
States $Q \supseteq F$: accepting states
For each $f \in \Sigma$ there is $\delta_f \subseteq Q^{\mathrm{rank}(f)+1}$ (transition function/relation)
Q-labelling consistent with transition functions
Accepting: root label in F

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}.

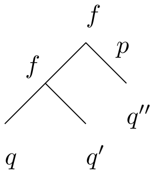
(Nondterministic) tree automaton

States $Q \supseteq F$: accepting states
For each $f \in \Sigma$ there is $\delta_f \subseteq Q^{\mathrm{rank}(f)+1}$ (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}.



$\delta_{f(\cdot,\cdot)} =$
$\{(q, q', p) \ : \ p \in \{q+1, q'\}\}$



$\delta_{f(f(\cdot,\cdot),\cdot)} =$
$\{(q, q', q'', p) \ : \ p \in \{q+2, q'+1, q''\}\}$

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.

Extensions

Easy

Compression: When a, b are compressed: $\delta_c \leftarrow \delta_a \circ \delta_b$
Popping: when x pops a: set δ'_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 δ'_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 δ 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

Transition functions

Each needed transition δ 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!

Transition functions

Each needed transition δ 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 Σ for each $\delta_1, \ldots, \delta_\ell$ as above and realised by a tree.

- equisatisfiable
- this allows trimming
- Σ is stored using an EXPTIME oracle (verified at popping)

Transition functions

Each needed transition δ 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 $a_{\delta_1,\dots,\delta_\ell}$ to Σ for each $\delta_1, \dots, \delta_\ell$ as above and realised by a tree.

- equisatisfiable
- this allows trimming
- Σ is stored using an EXPTIME oracle (verified at popping)
  - EXPTIME: exponentially many different transition vectors
  - sometimes the oracle is simpler (below EXPTIME)

Open problems

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?)