Grammar-based Compression of Unranked Trees

Carl Philipp Reh, University of Siegen, joint work with Adrià Gascón,
Markus Lohrey, Sebastian Maneth and Kurt Sieber

Compression on Text

- Compression is used to save space. This is especially important when a lot of data is transferred over slow connections.
- Algorithms for string compression: LZ77, LZ78, RePair, etc.

A *Straight-Line Program* (SLP) compresses text by sharing common sub-strings.

Example

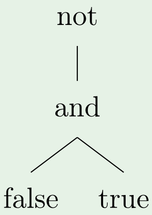
$$A \to \text{a very long text}$$
$$B \to AA$$
$$C \to BB$$

An SLP is similar to a context-free grammar but produces exactly one string instead of a language.

Ranked trees

Ranked trees are trees where the label of a node determines the number of its children.

Example



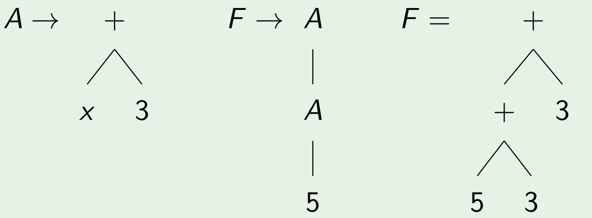
The unary not has one child, the binary and has two, and the nullary true and false have none.

Compression of Ranked trees

A *Tree Straight-Line Program* (TSLP) compresses trees not only by sharing common sub-trees but also by sharing common *sub-tree contexts*.

Example

$A \to +(x, 3) \qquad F \to A(A(5)) \qquad F = +(+(5,3),3)$



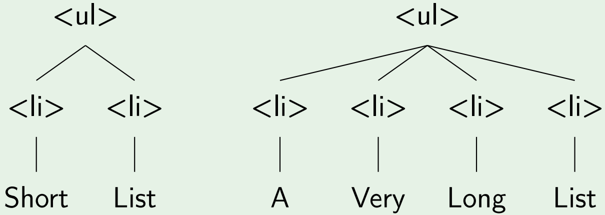
A context must contain exactly one x!

Unranked Trees

Unranked Trees are trees where the label of a node **does not** determine the number of its children. A *Forest* is a list of unranked trees.

- Forests are very common, for example in XML. In HTML, an -node does not determine the number of its children.

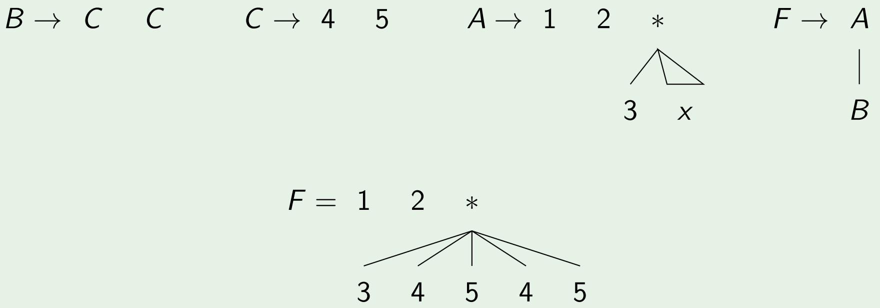
Example



Compression of Forests

A *Forest Straight-Line Program* (FSLP) compresses forests similar to a TSLP. It can also concatenate forests horizontally.

Example



Algorithms on compressed data

- FSLPs can be exponentially smaller, e.g. a^{2^n} can be represented by n variables.
- Instead of working on the uncompressed forest, we would like to work on the FSLPs directly.

Example

- We can compute the size of the uncompressed data.
- We can navigate (go left, right, up, down, print the current symbol) the uncompressed forest, where each step takes constant time.

Lemma

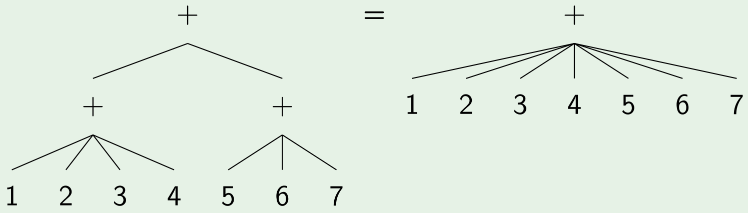
Given two variables, we can check in quadratic time if they produce the same forest.

This has been proven for SLPs by Artur Jeż and is easy to adapt to FSLPs.

Associative symbols

Associative symbols allow to "delete" nested occurrences of them:

Example



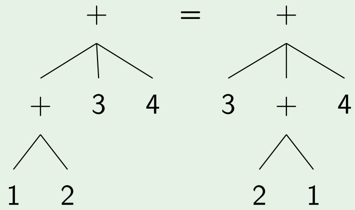
Lemma

For each pair of variables of an FSLP we can check in PTIME if they produce forests that are equal w.r.t. associative symbols.

Commutative symbols

Commutative symbols allow for reordering of children:

Example



Theorem

For each pair of variables of an FSLP we can check in PTIME if they produce forests that are equal w.r.t. commutative symbols.

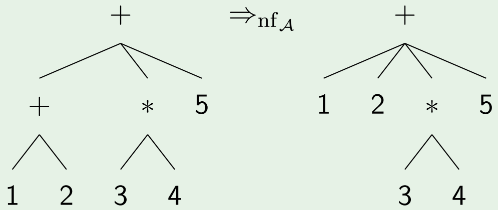
Proof ideas

- General technique: Instead of checking whether $P(f, g)$ for some forests f, g and property P, we instead transform them into a *normal form* nf_P and check if $\mathrm{nf}_P(f) = \mathrm{nf}_P(g)$.
- Implement nf_P on FSLPs directly without uncompressing them. The size of the FSLP must not increase too much!
- Then check if two variables produce the same forest.

Associative normal form

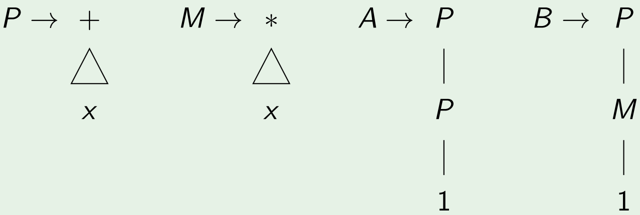
$\mathrm{nf}_{\mathcal{A}}$: Delete all nested associative symbols.

Example



FSLPs to $\mathrm{nf}_{\mathcal{A}}$

Example



FSLPs to $\mathrm{nf}_{\mathcal{A}}$

Example

$$P \to \begin{array}{c} + \\ \triangle \\ x \end{array} \qquad M \to \begin{array}{c} * \\ \triangle \\ x \end{array} \qquad A \to \begin{array}{c} P \\ | \\ P \\ | \\ 1 \end{array} \qquad B \to \begin{array}{c} P \\ | \\ M \\ | \\ 1 \end{array}$$

$$A = \begin{array}{c} + \\ | \\ + \\ | \\ 1 \end{array} \qquad \mathrm{nf}_{\mathcal{A}}(A) = \begin{array}{c} + \\ | \\ 1 \end{array} \qquad B = \begin{array}{c} + \\ | \\ * \\ | \\ 1 \end{array} \qquad \mathrm{nf}_{\mathcal{A}}(B) = \begin{array}{c} + \\ | \\ * \\ | \\ 1 \end{array}$$

FSLPs to $\mathrm{nf}_{\mathcal{A}}$

Create multiple versions of each variable (A_c means "c is above A").

Example

$$P_+ \to x \qquad P_* \to \begin{array}{c} + \\ \triangle \\ x \end{array} \qquad M_+ \to \begin{array}{c} * \\ \triangle \\ x \end{array} \qquad M_* \to x$$

FSLPs to $\mathrm{nf}_{\mathcal{A}}$

Create multiple versions of each variable (A_c means "c is above A").

Example

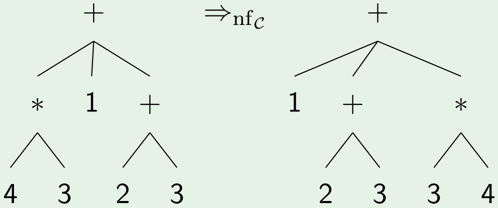
$$P_+ \to x \qquad P_* \to \begin{array}{c} + \\ \triangle \\ x \end{array} \qquad M_+ \to \begin{array}{c} * \\ \triangle \\ x \end{array} \qquad M_* \to x$$

$$A_+ \to \begin{array}{c} P_+ \\ | \\ P_+ \\ | \\ 1 \end{array} \qquad A_+ = 1 \qquad A_* \to \begin{array}{c} P_* \\ | \\ P_+ \\ | \\ 1 \end{array} \qquad A_* = \begin{array}{c} + \\ | \\ 1 \end{array}$$

Commutative normal form

$\mathrm{nf}_{\mathcal{C}}$: Sort all forests below commutative symbols using a length-lexicographical order (assume $+ < *$):

Example



How do we deal with this?

FSLPs to $\mathrm{nf}_{\mathcal{C}}$

Lemma

We can transform FSLPs in PTIME into the following form:

1. $A \to \varepsilon$,
2. $A \to BC$,
3. $A(x) \to z(LxR)$,
4. $A(x) \to B(C(x))$,
5. $A \to a(B)$,
6. $A \to B(C)$, *where* C *must not contain* x *and produce a tree.*

FSLPs to $\mathrm{nf}_{\mathcal{C}}$

Lemma

We can transform FSLPs in PTIME into the following form:

1. $A \to \varepsilon$,
2. $A \to BC$,
3. $A(x) \to z(LxR)$,
4. $A(x) \to B(C(x))$,
5. $A \to a(B)$,
6. $A \to B(C)$, *where C must not contain x and produce a tree.*

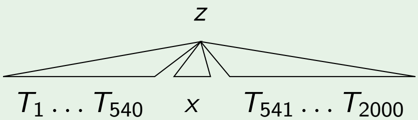
- L and R in $A(x) \to z(LxR)$ are produced by repeated use of $A \to BC$ until either $A \to \varepsilon$ or $A \to B(C)$ is reached.
- B in $A \to B(C)$ is produced by repeated use of $A(x) \to B(C(x))$ until $A(x) \to z(LxR)$ is reached again.

FSLPs to $\mathrm{nf}_{\mathcal{C}}$: $A(x) \to z(LxR)$

Key observation: Although L and R can yield exponentially wide forests, they always consist of linearly many *different* trees.

Example

$A(x) =$

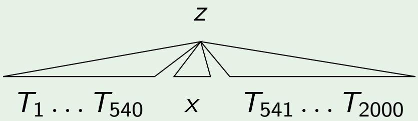


FSLPs to $\mathrm{nf}_{\mathcal{C}}$: $A(x) \to z(LxR)$

Key observation: Although L and R can yield exponentially wide forests, they always consist of linearly many *different* trees.

Example

$A(x) =$



Say $\{U_1, \ldots, U_{20}\}$ are the different trees of $\{T_1, \ldots, T_{2000}\}$ and $n_1, \ldots, n_{20}$ how often they occur. Assume $U_1 < \cdots < U_{20}$.

FSLPs to $\mathrm{nf}_{\mathcal{C}}$: $A(x) \to z(LxR)$

Key observation: Although L and R can yield exponentially wide forests, they always consist of linearly many *different* trees.

Example

$$A(x) = z$$

$$T_1 \dots T_{540} \qquad x \qquad T_{541} \dots T_{2000}$$

Say $\{U_1, \dots, U_{20}\}$ are the different trees of $\{T_1, \dots, T_{2000}\}$ and $n_1, \dots, n_{20}$ how often they occur. Assume $U_1 < \cdots < U_{20}$.
We create 21 translations, $A_0, \dots, A_{20}$, one for each x position, e.g.

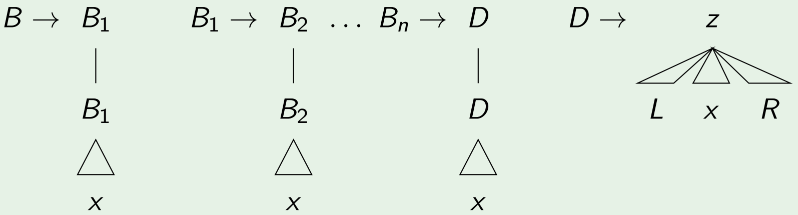
$$A_5(x) \to z$$

$$U_1^{n_1} \dots U_5^{n_5} \qquad x \qquad U_6^{n_6} \dots U_{20}^{n_{20}}$$

FSLPs to $\mathrm{nf}_{\mathcal{C}}$: $A \to B(C)$

When B is produced, this can lead to an exponentially long chain of productions of the form $A(x) \to B(C(x))$.

Example

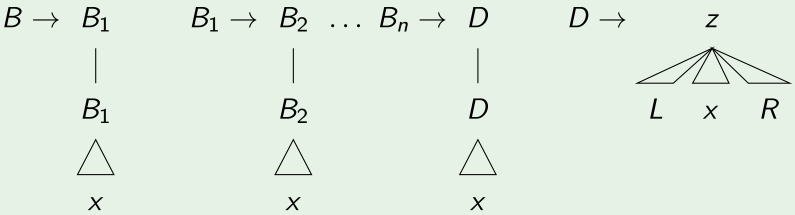


So $B(C) = D^{2^n}(C)$.

FSLPs to $\mathrm{nf}_{\mathcal{C}}$: $A \to B(C)$

When B is produced, this can lead to an exponentially long chain of productions of the form $A(x) \to B(C(x))$.

Example



So $B(C) = D^{2^n}(C)$.

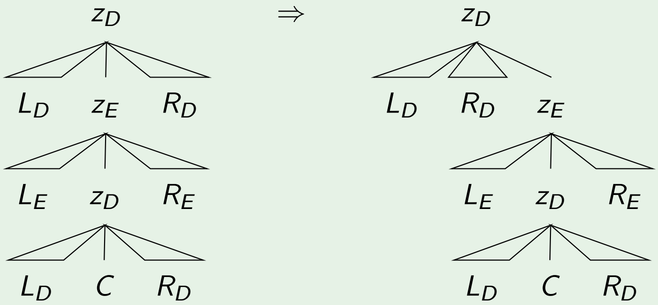
Problem: We cannot introduce 2^n many different translations for D.

FSLPs to $\mathrm{nf}_{\mathcal{C}}$: $A \to B(C)$

Key observation: When D occurs more than once, then in all occurrences of D except the *lowest*, x must go in the last position.

Example

$D(E(D(C)))$, $D \to z_D(L_D x R_D)$, $E \to z_E(L_E x R_E)$



Since $|E(D(C))| > |L_D R_D|$, $E(D(C))$ comes after $L_D R_D$.

Other formalisms

In FSLPs we allow arbitrary horizontal composition, e.g. fg.

FCNS: Canonical representation like Head/Tail for lists: $a(f)g$. TSLPs are used to compress FCNS.

Top Trees: Most basic forms are $a(b)$ and $a(b_x)$. When concatenating, the same symbols are merged, e.g. $a(b)a(c)$ becomes $a(bc)$ and $a(b_x(b(c)))$ becomes $a(b(c))$. Top Dags are used to compress Top Trees.

Lemma

- *We can translate between TSLPs for FCNS and FSLPs in each direction and maintain the size.*
- *We can translate from Top Dags to FSLPs and maintain the size.*
- *We can translate from FSLPs to Top Dags but the size increases by a factor of $|\Sigma|$.*

Summary

- FSLPs compress forests. They allow for sharing of forest contexts and arbitrary horizontal composition.
- We can check if two variables produce forests that are equal w.r.t. associative or commutative symbols without uncompressing the FSLP.

Thank you!