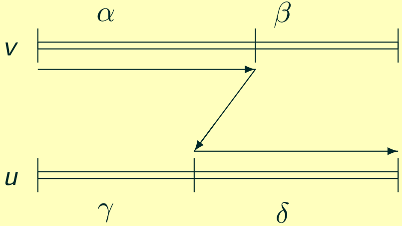
Reducing Repetitions

Peter Leupold[1]

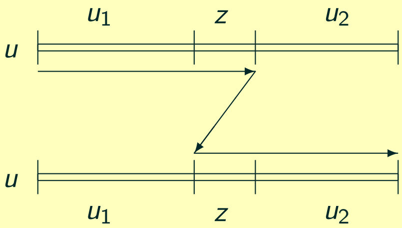
[1]Department of Mathematics, Faculty of Science
Kyoto Sangyo University, Japan

Prague Stringology Conference 2009

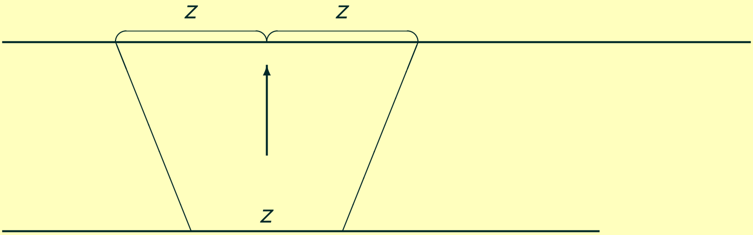
Crossing Over



Duplication



Graphical Display of Duplication



The Duplication Operation

We formalize this with the duplication relation $\heartsuit$ defined as

$$u \heartsuit v :\Leftrightarrow \exists z [z \in \Sigma^+ \wedge u = u_1 z u_2 \wedge v = u_1 z z u_2].$$

The Duplication Operation

We formalize this with the duplication relation $\heartsuit$ defined as

$$u \heartsuit v :\Leftrightarrow \exists z [z \in \Sigma^+ \wedge u = u_1 z u_2 \wedge v = u_1 z z u_2].$$

Note that the contexts u_1 and u_2 form part of the relation.

The Duplication Operation

We formalize this with the duplication relation $\heartsuit$ defined as

$$u \heartsuit v :\Leftrightarrow \exists z [z \in \Sigma^+ \wedge u = u_1 z u_2 \wedge v = u_1 z z u_2].$$

Note that the contexts u_1 and u_2 form part of the relation.

We also consider variants with length bounds $|z| \leq k$ or $|z| = k$ and we write $\heartsuit^{\leq k}$ or $\heartsuit^k$ respectively.

Duplication – Results and Open Problems

Theorem

$\heartsuit^*$ *preserves regularity over two-letter alphabet, but not over three-letter alphabet.*

Duplication – Results and Open Problems

Theorem

$\heartsuit^*$ *preserves regularity over two-letter alphabet, but not over three-letter alphabet.*

Theorem

$\heartsuit^{\leq k^*}$ *preserves context-freeness.*

Duplication – Results and Open Problems

Theorem

$\heartsuit^*$ *preserves regularity over two-letter alphabet, but not over three-letter alphabet.*

Theorem

$\heartsuit^{\leq k^*}$ *preserves context-freeness.*

Open Problem

Does $\heartsuit^*$ *preserve context-freeness?*

Duplication Roots

For reducing repetitions we will use the inverse of $\heartsuit$ denoted by $\rightarrowtail$.

Duplication Roots

For reducing repetitions we will use the inverse of $\heartsuit$ denoted by $\rightarrowtail$.

Definition

The <u>duplication root</u> of a non-empty word w is

$$\sqrt[\heartsuit]{w} := IRR(\rightarrowtail) \cap \{u : w \rightarrowtail^* u\}.$$

As usual, this notion is extended in the canonical way from words to languages such that

$$\sqrt[\heartsuit]{L} := \bigcup_{w \in L} \sqrt[\heartsuit]{w}.$$

The roots $\sqrt[\heartsuit \leq k]{w}$ and $\sqrt[\heartsuit k]{w}$ are defined in completely analogous ways

Duplication Roots – Examples

- All words in a duplication root are square-free, and over an alphabet of two letters only the seven square-free words $\{\lambda, a, b, ab, ba, aba, bab\}$ exist. They are uniquely determined by their first letter, the last letter, and the set of letters occurring in them.

Duplication Roots – Examples

- All words in a duplication root are square-free, and over an alphabet of two letters only the seven square-free words $\{\lambda, a, b, ab, ba, aba, bab\}$ exist. They are uniquely determined by their first letter, the last letter, and the set of letters occurring in them.
- By undoing duplications, i.e., by applying rules from $\rightarrowtail$, we obtain from the word $w = abcbabcbc$ the words in the set $\{abc, abcbc, abcbabc\}$.
  Thus we have the root $\sqrt[\heartsuit]{abcbabcbc} = \{abc, abcbabc\}$

Duplication Roots – Examples

- All words in a duplication root are square-free, and over an alphabet of two letters only the seven square-free words $\{\lambda, a, b, ab, ba, aba, bab\}$ exist. They are uniquely determined by their first letter, the last letter, and the set of letters occurring in them.
- By undoing duplications, i.e., by applying rules from $\succ\!\!\!\!\to$, we obtain from the word $w = abcbabcbc$ the words in the set $\{abc, abcbc, abcbabc\}$.
  Thus we have the root $\sqrt[\heartsuit]{abcbabcbc} = \{abc, abcbabc\}$
- $$\sqrt[\heartsuit]{babacabacbcabacb} = \{bacabacb, bacbcabacb, bacb\},$$
  and
  $$\sqrt[\heartsuit]{ababcbabcacbabcabacbabcab} =$$
  $$\{abcbabcabacbabcab, abcbabcab, abcacbabcab, abcabacbabcab, abcab\},$$

Duplication Roots and the Chomsky Hierarchy

Theorem

The closure properties of the classes of regular and context-free languages under the three duplication roots are as follows:

	$\sqrt[\heartsuit k]{L}$	$\sqrt[\heartsuit \leq k]{L}$	$\sqrt[\heartsuit]{L}$
REG	Y	Y	N
CF	?	?	N

The symbol Y stands for closure, N stands for non-closure, and ? means that the problem is open.

Repetition Complexity

In an effort to define a new measure for the complexity of words, Ilie et al. defined a reduction relation very similar to undoing duplications, which however remembers the steps it takes.
For the definition let $D = \{0, 1, \ldots 9\}$ be the set of decimal digits, and Σ be an alphabet disjoint from D. The alphabet for the reduction relation is $T := \Sigma \cup D \cup \{\langle, \rangle, EXP\}$.
Then the reduction relation $\Rightarrow$ is defined by $u \Rightarrow v$ iff $u = u_1 x^n u_2$, $v = u_1 \langle x \rangle EXP \langle \text{dec } n \rangle u_2$ for some $u_1, u_2 \in T^*$, $x = \Sigma^+$, $n > 2$. Finally, let h be the morphism erasing all symbols except the letters from Σ.

Unduplication versus Repetition Complexity

Example

For the word $ababcbc$ there are two irreducible forms under $\Rightarrow$, namely $\langle ab \rangle\, EXP\langle 2 \rangle cbc$ and $aba\, \langle bc \rangle\, EXP\langle 2 \rangle$.

Unduplication versus Repetition Complexity

Example

For the word $ababcbc$ there are two irreducible forms under $\Rightarrow$, namely $\langle ab \rangle\, EXP\langle 2 \rangle cbc$ and $aba\, \langle bc \rangle\, EXP\langle 2 \rangle$.

Under $\rightarrowtail$, however, the images of both words under h are further reducible to a common normal form: both $ababcbc \rightarrowtail abcbc \rightarrowtail abc$ and $ababcbc \rightarrowtail ababc \rightarrowtail abc$ are possible reductions leading to abc.

Unduplication versus Repetition Complexity

Example

For the word *ababcbc* there are two irreducible forms under $\Rightarrow$, namely $\langle ab\rangle\, EXP\langle 2\rangle cbc$ and $aba\,\langle bc\rangle\, EXP\langle 2\rangle$.

Under $\rightarrowtail$, however, the images of both words under h are further reducible to a common normal form: both $ababcbc \rightarrowtail abcbc \rightarrowtail abc$ and $ababcbc \rightarrowtail ababc \rightarrowtail abc$ are possible reductions leading to *abc*.

Notice how the brackets block the further reduction of *abab* in $aba\,\langle bc\rangle\, EXP\langle 2\rangle$ and of *bcbc* in $\langle ab\rangle\, EXP\langle 2\rangle cbc$.

Unduplication versus Repetition Complexity

There are two main differences between the two relations.

Unduplication versus Repetition Complexity

There are two main differences between the two relations.

1. A reduction $u^n \Rightarrow \langle u \rangle\, EXP\langle n \rangle$ is done in a single step while the reduction $u^n \succ\!\!\!\rightarrow^* u$ will always take $n-1$ steps.

Unduplication versus Repetition Complexity

There are two main differences between the two relations.

1. A reduction $u^n \Rightarrow \langle u \rangle\, EXP\langle n \rangle$ is done in a single step while the reduction $u^n \succ\!\!\rightarrow^* u$ will always take $n-1$ steps.
2. If $w \Rightarrow^* u$ then $w \succ\!\!\rightarrow^* h(u)$, but the reverse does not hold, see the Example above.

Unduplication versus Repetition Complexity

There are two main differences between the two relations.

1. A reduction $u^n \Rightarrow \langle u \rangle\, EXP \langle n \rangle$ is done in a single step while the reduction $u^n \succ\!\!\to^* u$ will always take $n-1$ steps.
2. If $w \Rightarrow^* u$ then $w \succ\!\!\to^* h(u)$, but the reverse does not hold, see the Example above.

Despite these differences, the similarities are evident, and $\Rightarrow^*$ can be embedded in $\succ\!\!\to^*$. We state a further relation.

Theorem

For a word w, if $\sqrt[\heartsuit]{w} \subseteq \{h(u) : w \Rightarrow^* u\}$ then $|\sqrt[\heartsuit]{w}| = 1$.

The Number of Duplication Roots

We have seen from the examples above that the number of possible duplication roots seems to increase with increasing word length.
Our main interest here is to investigate the behaviour of the function:

$$\mathrm{duproots}(n) := \max\{|\sqrt[\heartsuit]{w}| : |w| = n\}.$$

The Number of Duplication Roots

We have seen from the examples above that the number of possible duplication roots seems to increase with increasing word length. Our main interest here is to investigate the behaviour of the function:

$$\mathsf{duproots}(n) := \max\{|\sqrt[\heartsuit]{w}| : |w| = n\}.$$

Because it has often turned out to be very useful to consider problems about duplications with a length restriction, we also define the function

$$\mathsf{bduproots}{\leq}\, k(n) := \max\{|\sqrt[\heartsuit \leq k]{w}| : |w| = n\}.$$

The Number of Duplication Roots

We have seen from the examples above that the number of possible duplication roots seems to increase with increasing word length.
Our main interest here is to investigate the behaviour of the function:

$$\mathrm{duproots}(n) := \max\{|\sqrt[\heartsuit]{w}| : |w| = n\}.$$

Because it has often turned out to be very useful to consider problems about duplications with a length restriction, we also define the function

$$\mathrm{bduproots}_{\leq k}(n) := \max\{|\sqrt[\heartsuit \leq k]{w}| : |w| = n\}.$$

Notice that we do not bound the alphabet size.

Bounding from Above

Obviously, rules from $\succ\!\!\rightarrow$ can only be applied on square factors. Thus the number of squares is the number of possible distinct rule applications in a string.

Bounding from Above

Obviously, rules from $\succ$ can only be applied on square factors. Thus the number of squares is the number of possible distinct rule applications in a string.

However, when we are interested in rule applications with distinct results and thus with potentially distinct roots, the number of runs captures this more exactly.

Bounding from Above

Obviously, rules from $\rightarrowtail$ can only be applied on square factors. Thus the number of squares is the number of possible distinct rule applications in a string.

However, when we are interested in rule applications with distinct results and thus with potentially distinct roots, the number of runs captures this more exactly.

Fact

Let w be a word with period k. Then all applications of rules from $\rightarrowtail_k$ will result in the same word, i.e. $\{u : w \rightarrowtail_k u\}$ is a singleton set.

Bounding from Above

Obviously, rules from $\rightarrowtail$ can only be applied on square factors. Thus the number of squares is the number of possible distinct rule applications in a string.

However, when we are interested in rule applications with distinct results and thus with potentially distinct roots, the number of runs captures this more exactly.

Fact

Let w be a word with period k. Then all applications of rules from $\rightarrowtail_k$ will result in the same word, i.e. $\{u : w \rightarrowtail_k u\}$ is a singleton set.

As a consequence of this, the number of distinct descendants of w with respect to $\rightarrowtail$ is equal to the number of runs in w.

Bounding from Above

Every reduction via $\rightarrowtail$ removes at least one letter, thus there can be at most $n-1$ steps in the reduction of a word of length n. So there are at most duproots$(n) \leq runs(n)^{n-3}$ different reductions.

Bounding from Above

Every reduction via $\rightarrowtail$ removes at least one letter, thus there can be at most $n-1$ steps in the reduction of a word of length n. So there are at most duproots$(n) \leq runs(n)^{n-3}$ different reductions.

Lemma

If for two words $u, v \in \Sigma^*$ we have $seq(u) = seq(v)$, then also $\sqrt[\heartsuit]{u} = \sqrt[\heartsuit]{v} = \sqrt[\heartsuit]{seq(u)}$.

Bounding from Above

Every reduction via $\succ\!\!\!\rightarrow$ removes at least one letter, thus there can be at most $n-1$ steps in the reduction of a word of length n. So there are at most $\mathrm{duproots}(n) \leq runs(n)^{n-3}$ different reductions.

Lemma

If for two words $u, v \in \Sigma^*$ we have $seq(u) = seq(v)$, then also $\sqrt[\heartsuit]{u} = \sqrt[\heartsuit]{v} = \sqrt[\heartsuit]{seq(u)}$.

This means that we can first do all the possible reductions of the form $x^2 \to x$ for single letters x. For possible splits to different duplication roots we can assume that at least two letters are deleted in every step.

Bounding from Above

Every reduction via $\succ\!\!\!\rightarrow$ removes at least one letter, thus there can be at most $n-1$ steps in the reduction of a word of length n. So there are at most duproots$(n) \leq runs(n)^{n-3}$ different reductions.

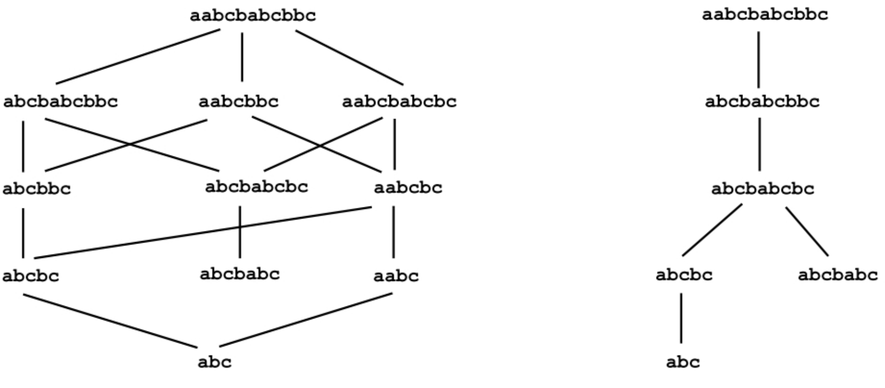
Lemma

If for two words $u, v \in \Sigma^*$ we have $seq(u) = seq(v)$, then also $\sqrt[\heartsuit]{u} = \sqrt[\heartsuit]{v} = \sqrt[\heartsuit]{seq(u)}$.

This means that we can first do all the possible reductions of the form $x^2 \rightarrow x$ for single letters x. For possible splits to different duplication roots we can assume that at least two letters are deleted in every step.

This improves our upper bound to $runs(n)^{\frac{n-3}{2}}$.

One-Letter-Squares First



10 versus 2 paths for the word *aabcbabcbbc*, by first reducing one-letter squares from left to right. The direction of reductions is top to bottom.

Bounding from Below

- We construct an example of a sequence of words w_n, which are simply powers of a word w, namely $w_n := w^n$. The number of roots increases exponentially in n.

Bounding from Below

- We construct an example of a sequence of words w_n, which are simply powers of a word w, namely $w_n := w^n$. The number of roots increases exponentially in n.
- We start from $u = abcbabcbc$ with the two roots $u_1 = abc$ and $u_2 = abcbabc$.

Bounding from Below

- We construct an example of a sequence of words w_n, which are simply powers of a word w, namely $w_n := w^n$. The number of roots increases exponentially in n.
- We start from $u = abcbabcbc$ with the two roots $u_1 = abc$ and $u_2 = abcbabc$.
- Let ρ be the morphism, which simply renames letters according to the scheme $a \rightarrow b \rightarrow c \rightarrow a$. Then $\rho(u)$ has the two roots $\rho(u_1)$ and $\rho(u_2)$; similarly, $\rho(\rho(u))$ has the two roots $\rho(\rho(u_1))$ and $\rho(\rho(u_2))$.

Bounding from Below

- We construct an example of a sequence of words w_n, which are simply powers of a word w, namely $w_n := w^n$. The number of roots increases exponentially in n.
- We start from $u = abcbabcbc$ with the two roots $u_1 = abc$ and $u_2 = abcbabc$.
- Let ρ be the morphism, which simply renames letters according to the scheme $a \to b \to c \to a$. Then $\rho(u)$ has the two roots $\rho(u_1)$ and $\rho(u_2)$; similarly, $\rho(\rho(u))$ has the two roots $\rho(\rho(u_1))$ and $\rho(\rho(u_2))$.
- $w = ud\rho(u)d\rho(\rho(u))d = abcbabcbc \cdot d \cdot bcacbcaca \cdot d \cdot cabacabab \cdot d$.

Bounding from Below

Thus the duplication root of w contains among others the three words

$$\begin{aligned} w_a = &\quad abc \cdot d \cdot bca \cdot d \cdot cabacab \cdot d \\ w_b = &\quad abc \cdot d \cdot bcacbca \cdot d \cdot cab \cdot d \\ w_c = &\quad abcbabc \cdot d \cdot bca \cdot d \cdot cab \cdot d, \end{aligned}$$

which are square-free.

Bounding from Below

Thus the duplication root of w contains among others the three words

$$\begin{aligned} w_a = & \quad abc \cdot d \cdot bca \cdot d \cdot cabacab \cdot d \\ w_b = & \quad abc \cdot d \cdot bcacbca \cdot d \cdot cab \cdot d \\ w_c = & \quad abcbabc \cdot d \cdot bca \cdot d \cdot cab \cdot d, \end{aligned}$$

which are square-free.

We now need to recall that a morphism h is called square-free, iff $h(v)$ is square-free for all square-free words v. Crochemore has shown that a uniform morphism h is square-free iff it is square-free for all square-free words of length 3.

Bounding from Below

Thus the duplication root of w contains among others the three words

$$\begin{aligned} w_a = &\quad abc \cdot d \cdot bca \cdot d \cdot cabacab \cdot d \\ w_b = &\quad abc \cdot d \cdot bcacbca \cdot d \cdot cab \cdot d \\ w_c = &\quad abcbabc \cdot d \cdot bca \cdot d \cdot cab \cdot d, \end{aligned}$$

which are square-free.

We now need to recall that a morphism h is called square-free, iff $h(v)$ is square-free for all square-free words v. Crochemore has shown that a uniform morphism h is square-free iff it is square-free for all square-free words of length 3.

The morphism we define now is $\varphi(x) := w_x$ for all $x \in \{a, b, c\}$.

Checking Square-freeness

$$
\begin{aligned}
\varphi(aba) &= abcdbcadcabacabdabcdbcacbcadcabdabcdbcadcabacabd\\
\varphi(abc) &= abcdbcadcabacabdabcdbcacbcadcabdabcbabcdbcadcabd\\
\varphi(aca) &= abcdbcadcabacabdabcbabcdbcadcabdabcdbcadcabacabd\\
\varphi(acb) &= abcdbcadcabacabdabcbabcdbcadcabdabcdbcacbcadcabd\\
\varphi(bab) &= abcdbcacbcadcabdabcdbcadcabacabdabcdbcacbcadcabd\\
\varphi(bac) &= abcdbcacbcadcabdabcdbcadcabacabdabcbabcdbcadcabd\\
\varphi(bca) &= abcdbcacbcadcabdabcbabcdbcadcabdabcdbcadcabacabd\\
\varphi(bcb) &= abcdbcacbcadcabdabcbabcdbcadcabdabcdbcacbcadcabd\\
\varphi(cac) &= abcbabcdbcadcabdabcdbcadcabacabdabcbabcdbcadcabd\\
\varphi(cab) &= abcbabcdbcadcabdabcdbcadcabacabdabcdbcacbcadcabd\\
\varphi(cba) &= abcbabcdbcadcabdabcdbcacbcadcabdabcdbcadcabacabd\\
\varphi(cbc) &= abcbabcdbcadcabdabcdbcacbcadcabdabcbabcdbcadcabd,
\end{aligned}
$$

Bounding from Below

Now let t be an infinite square-free word over the letters a, b and c. Such a word exists.

Bounding from Below

Now let t be an infinite square-free word over the letters a, b and c. Such a word exists.

Then all the words in $\varphi(\text{pref}(t))$ are square-free, too. From the construction of φ we know that for any word z of length i we can reach $\varphi(z)$ from w^i by undoing duplications.

Bounding from Below

Now let t be an infinite square-free word over the letters a, b and c. Such a word exists.

Then all the words in $\varphi(\mathrm{pref}(t))$ are square-free, too. From the construction of φ we know that for any word z of length i we can reach $\varphi(z)$ from w^i by undoing duplications.

Therefore $\varphi(\mathrm{pref}(t)) \subseteq \sqrt[\heartsuit]{w^+}$. For two distinct square-free words t_1 and t_2, also $\varphi(t_1) \neq \varphi(t_2)$.

Bounding from Below

Now let t be an infinite square-free word over the letters a, b and c. Such a word exists.

Then all the words in $\varphi(\text{pref}(t))$ are square-free, too. From the construction of φ we know that for any word z of length i we can reach $\varphi(z)$ from w^i by undoing duplications.

Therefore $\varphi(\text{pref}(t)) \subseteq \sqrt[\heartsuit]{w^+}$. For two distinct square-free words t_1 and t_2, also $\varphi(t_1) \neq \varphi(t_2)$.

Finally, notice that for all positive $i \leq n$ we have $w^n \rightarrowtail^* w^i$.

Counting the Roots

We conclude that $\text{bduproots}_{\leq 30} \leq s$, where $s(n)$ is the number of ternary square-free words of length up to n.

Counting the Roots

We conclude that $\mathrm{bduproots}_{\leq 30} \leq s$, where $s(n)$ is the number of ternary square-free words of length up to n.

This function's value is not known, however, it was first bounded to $6 \cdot 1.032^n \leq s(n) \leq 6 \cdot 1.379^n$ by Brandenburg. A better lower bound was found by Sun $s(n) \geq 110^{\frac{n}{42}}$.

Counting the Roots

We conclude that $\mathrm{bduproots}_{\leq 30} \leq s$, where $s(n)$ is the number of ternary square-free words of length up to n.

This function's value is not known, however, it was first bounded to $6 \cdot 1.032^n \leq s(n) \leq 6 \cdot 1.379^n$ by Brandenburg. A better lower bound was found by Sun $s(n) \geq 110^{\frac{n}{42}}$.

w itself is of length $3|u| + 3 = 30$. So we see that $\mathrm{bduproots}_{\leq 30}(n) \geq \frac{1}{30} 110^{\frac{n}{42}}$.

The Bounds

Theorem

$\frac{1}{30}110^{\frac{n}{42}} \leq duproots(n) \leq 2^n$ for all $n > 0$.

Theorem

$\frac{1}{30}110^{\frac{n}{42}} \leq bduproots_{\leq 30}(n) \leq \max\{812^{\frac{n-3}{2}}, 2^n\}$ for all $n > 0$.

For ternary alphabet, the upper bound $6 \cdot 1.379^n$ on the number of ternary words by Brandenburg can replace 2^n in both Propositions.

Open Problems

- In how far does duproots(n) depend on the alphabet size?

Open Problems

- In how far does duproots(n) depend on the alphabet size?
- An example for exponential growth with only three letters.

Open Problems

- In how far does duproots(n) depend on the alphabet size?
- An example for exponential growth with only three letters.
- How complicated is it to compute duproots(n)?