Three termination problems

Patrick Dehornoy

Laboratoire de Mathématiques
Nicolas Oresme, Université de Caen

- Three unrelated termination problems : partial specific answers known, but no global understanding: can some general tools be useful?

- Plan :

    1. The Polish Algorithm for Left-Selfdistributivity
    2. Handle reduction of braids
    3. Subword reversing for positively presented groups

1. The Polish Algorithm for Left-Selfdistributivity
2. Handle reduction of braids
3. Subword reversing for positively presented groups

The baby problem I

- A "bi-term rewrite system" (????)

- The associativity law $(\boldsymbol{A})$: $x * (y * z) = (x * y) * z$,
  ... and the corresponding Word Problem:

  Given two terms $\boldsymbol{t}, \boldsymbol{t}'$, decide whether $\boldsymbol{t}$ and $\boldsymbol{t}'$ are $\boldsymbol{A}$-equivalent.

- A trivial problem: $\boldsymbol{t}, \boldsymbol{t}'$ are A-equivalent iff become equal when brackets are removed.

- (Right-) Polish expression of a term: "$\boldsymbol{t}_1 \boldsymbol{t}_2 *$" for $\boldsymbol{t}_1 * \boldsymbol{t}_2$ (no bracket needed)
  Example: In Polish, associativity is $x\, y\, z * * = x\, y * z *$.

- Definition.— The Polish Algorithm for A: starting with two terms $\boldsymbol{t}, \boldsymbol{t}'$ (in Polish):

```
- while $\boldsymbol{t} \neq \boldsymbol{t}'$ do
    - $\boldsymbol{p} :=$ first clash between $\boldsymbol{t}$ and $\boldsymbol{t}'$ ($p$th letter of $\boldsymbol{t} \neq p$th letter of $\boldsymbol{t}'$)
    - case type of $\boldsymbol{p}$ of
        - "variable vs. blank" :     return NO;
        - "blank vs. variable" :     return NO;
        - "variable vs. variable" :  return NO;
        - "variable vs. $*$" :       apply $A^+$ to $\boldsymbol{t}$;   $(t_1 t_2 \boldsymbol{t_3} * * \rightarrow t_1 t_2 * t_3 *)$
        - "$*$ vs. variable" :       apply $A^+$ to $\boldsymbol{t}'$;  $(t_1 t_2 \boldsymbol{t_3} * * \rightarrow t_1 t_2 * t_3 *)$
- return YES.
```

The baby problem II

- Remember : in Polish, associativity is $\begin{cases} x\,y\,z*\,* \\ x\,y*z\,* \end{cases}$.

- Example: $\boldsymbol{t} = x*(x*(x*x))$, $\boldsymbol{t}' = ((x*x)*x)*x$, i.e., in Polish,

$$\begin{aligned} \boldsymbol{t}_0 &= x\,x\,x\,x***\\ \boldsymbol{t}_0' &= x\,x*x*x* \end{aligned}$$

$$\begin{aligned} \boldsymbol{t}_1 &= x\,x*x\,x**\\ \boldsymbol{t}_1' &= x\,x*x*x* \end{aligned}$$

$$\begin{aligned} \boldsymbol{t}_2 &= x\,x*x*x*\\ \boldsymbol{t}_2' &= x\,x*x*x* \end{aligned}$$

So $\boldsymbol{t}_2 = \boldsymbol{t}_2'$, hence $\boldsymbol{t}_0$ and $\boldsymbol{t}_0'$ are A-equivalent.

- "Theorem".— The Polish Algorithm works for associativity.
(In particular, it terminates.)

The real problem I

- Left-selfdistributivity (LD) : $x*(y*z) = (x*y)*(x*z)$,

  i.e., in Polish, $\begin{cases} x\,y\,z\,*\,* \\ x\,y\,*\,x\,z\,*\,* \end{cases}$ compare with associativity $\begin{cases} x\,y\,z\,*\,* \\ x\,y\,*\,x\,* \end{cases}$

- Polish Algorithm: the same as for associativity.

- Example: $\boldsymbol{t} = x*((x*x)*(x*x))$, $\boldsymbol{t}' = (x*x)*(x*(x*x))$, i.e., in Polish,

$$\begin{aligned}
\boldsymbol{t}_0 &= x\,x\,x\,*\,x\,x\,*\,*\,* \\
\boldsymbol{t}'_0 &= x\,x\,*\,x\,x\,x\,*\,*\,* \\[1ex]
\boldsymbol{t}_1 &= x\,x\,*\,x\,x\,*\,*\,x\,x\,x\,*\,*\,* \\
\boldsymbol{t}'_1 &= x\,x\,*\,x\,x\,x\,*\,*\,* && (= \boldsymbol{t}'_0) \\[1ex]
\boldsymbol{t}_2 &= x\,x\,*\,x\,x\,*\,*\,x\,x\,x\,*\,*\,* && (= \boldsymbol{t}_1) \\
\boldsymbol{t}'_2 &= x\,x\,*\,x\,x\,*\,x\,x\,*\,* \\[1ex]
\boldsymbol{t}_3 &= x\,x\,*\,x\,x\,*\,*\,x\,x\,x\,*\,*\,* && (= \boldsymbol{t}_2) \\
\boldsymbol{t}'_3 &= x\,x\,*\,x\,x\,*\,*\,x\,x\,*\,x\,x\,*\,*\,* \\[1ex]
\boldsymbol{t}_4 &= x\,x\,*\,x\,x\,*\,*\,x\,x\,*\,x\,x\,*\,*\,* \\
\boldsymbol{t}'_4 &= x\,x\,*\,x\,x\,*\,*\,x\,x\,*\,x\,x\,*\,*\,* && (= \boldsymbol{t}'_3)
\end{aligned}$$

So $\boldsymbol{t}_4 = \boldsymbol{t}'_4$, hence $\boldsymbol{t}_0$ and $\boldsymbol{t}'_0$ are LD-equivalent.

The real problem II

- Conjecture.— The Polish Algorithm works for left-selfdistributivity.

- Known.— (i) If it terminates, the Polish Algorithm works for left-selfdistributivity.
  (ii) The smallest counter-example to termination (if any) is huge.

1. The Polish Algorithm for Left-Selfdistributivity
2. Handle reduction of braids
3. Subword reversing for positively presented groups

The baby problem I

- A true (but infinite) rewrite system.

- Alphabet: $\mathtt{a}, \mathtt{b}, \mathtt{A}, \mathtt{B}$ (think of $\mathtt{A}$ as an inverse of $\mathtt{a}$, etc.)

- Rewrite rules:
  - $\mathtt{aA} \longrightarrow \varepsilon$, $\mathtt{Aa} \longrightarrow \varepsilon$, $\mathtt{bB} \longrightarrow \varepsilon$, $\mathtt{Bb} \longrightarrow \varepsilon$ (so far trivial: "free group reduction")
  - $\mathtt{abA} \longrightarrow \mathtt{Bab}$, $\mathtt{aBA} \longrightarrow \mathtt{BAb}$, $\mathtt{Aba} \longrightarrow \mathtt{baB}$, $\mathtt{ABa} \longrightarrow \mathtt{bAB}$,

and, more generally,
  - $\mathtt{ab}^i\mathtt{A} \longrightarrow \mathtt{Ba}^i\mathtt{b}$, $\mathtt{aB}^i\mathtt{A} \longrightarrow \mathtt{BA}^i\mathtt{b}$, $\mathtt{Ab}^i\mathtt{a} \longrightarrow \mathtt{ba}^i\mathtt{B}$, $\mathtt{AB}^i\mathtt{a} \longrightarrow \mathtt{bA}^i\mathtt{B}$ for $i \geqslant 1$.

- Aim: obtain a word that does not contain both $\mathtt{a}$ and $\mathtt{A}$.

- Example:

$$\begin{aligned} w_0 &= \mathtt{a}\underline{\mathtt{abA}}\mathtt{bbAA} \\ w_1 &= \mathtt{aB}\underline{\mathtt{abbbA}}\mathtt{A} \\ w_2 &= \mathtt{aBBaa}\underline{\mathtt{abA}} \\ w_3 &= \mathtt{aBBaaBab}, \end{aligned}$$

⇝ a word without $\mathtt{A}$

The baby problem II

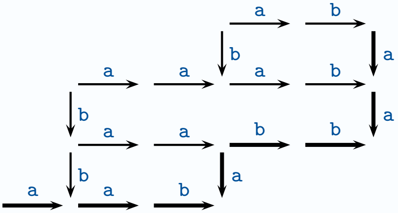
- **Theorem.—** The process terminates in quadratic time.

- Proof: (Length does not increase, but could cycle.)
Associate with the sequence of reductions a rectangular grid (quadratic area).

For the example:

$$\begin{aligned} \boldsymbol{w}_0 &= \texttt{aabAbbAA} \\ \boldsymbol{w}_1 &= \texttt{aBabbbAA} \\ \boldsymbol{w}_2 &= \texttt{aBBaaabA} \\ \boldsymbol{w}_3 &= \texttt{aBBaaBab} \end{aligned}$$

draw the grid:



□

- This is the braid handle reduction procedure;
  so far: case of "3-strand" braids; now: case of "4-strand" braids
  (case of "n strand" braids entirely similar for every n).

- Alphabet: $\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{A}, \mathtt{B}, \mathtt{C}$.
- Rewrite rules:
  - $\mathtt{aA} \longrightarrow \varepsilon$, $\mathtt{Aa} \longrightarrow \varepsilon$, $\mathtt{bB} \longrightarrow \varepsilon$, $\mathtt{Bb} \longrightarrow \varepsilon$, $\mathtt{cC} \longrightarrow \varepsilon$, $\mathtt{Cc} \longrightarrow \varepsilon$, (as above)
  - for w in $\{\mathtt{b}, \mathtt{c}, \mathtt{C}\}^*$ or $\{\mathtt{B}, \mathtt{c}, \mathtt{C}\}^*$: $\mathtt{a}w\mathtt{A} \longrightarrow \phi_{\mathtt{a}}(w)$, $\mathtt{A}w\mathtt{a} \longrightarrow \phi_{\mathtt{A}}(w)$,
    with $\phi_{\mathtt{a}}(w)$ obtained from w by $\mathtt{b} \longrightarrow \mathtt{Bab}$ and $\mathtt{B} \longrightarrow \mathtt{BAb}$,
    and $\phi_{\mathtt{A}}(w)$ obtained from w by $\mathtt{b} \longrightarrow \mathtt{baB}$ and $\mathtt{B} \longrightarrow \mathtt{bAB}$,
  - for w in $\{\mathtt{c}\}^*$ or $\{\mathtt{C}\}^*$: $\mathtt{b}w\mathtt{B} \longrightarrow \phi_{\mathtt{b}}(w)$, $\mathtt{B}w\mathtt{b} \longrightarrow \phi_{\mathtt{B}}(w)$,
    with $\phi_{\mathtt{b}}(w)$ obtained from w by $\mathtt{c} \longrightarrow \mathtt{Cbc}$ and $\mathtt{C} \longrightarrow \mathtt{CBc}$,
    and $\phi_{\mathtt{B}}(w)$ obtained from w by $\mathtt{c} \longrightarrow \mathtt{cbC}$ and $\mathtt{C} \longrightarrow \mathtt{cBC}$.

- Remark.— $\mathtt{ab}^i\mathtt{A} \longrightarrow (\mathtt{Bab})^i \longrightarrow \mathtt{Ba}^i\mathtt{b}$: extends the 3-strand case.

The real problem

- Example:

$\mathtt{\underline{abcbA}BABCBA}$
$\mathtt{BabcBa\underline{bB}ABCBA}$
$\mathtt{BabcB\underline{aA}BCBA}$
$\mathtt{Ba\underline{bcB}BCBA}$
$\mathtt{BaC\underline{bcB}CBA}$
$\mathtt{BaCCb\underline{cC}BA}$
$\mathtt{BaCC\underline{bB}A}$
$\mathtt{B\underline{aCCA}}$
$\mathtt{BCC}$

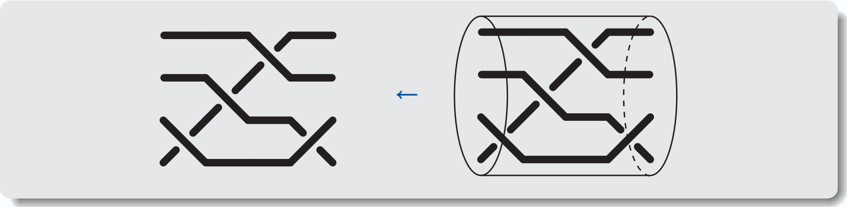
⇝ Terminates: the final word does not contain both `a` and `A`
(by the way: contains neither `a` nor `A`, and not both `b` and `B`.)

- Theorem.— Handle reduction always terminates in exponential time
(and *id.* for n-strand version).

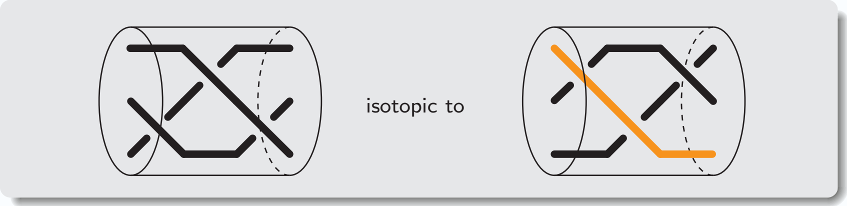
- Experimental evidence.— It terminates in quadratic time (for every n).

Braids

- A 4-strand braid diagram = 2D-projection of a 3D-figure:



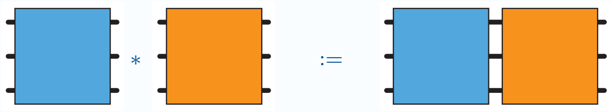
- isotopy = move the strands but keep the ends fixed:



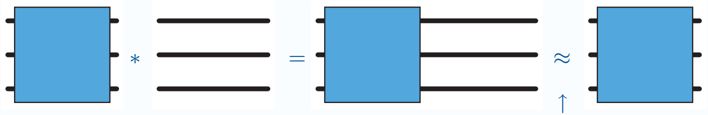
isotopic to

- a braid := an isotopy class $\rightsquigarrow$ represented by 2D-diagram,
  but different 2D-diagrams may give rise to the same braid.

- Product of two braids:

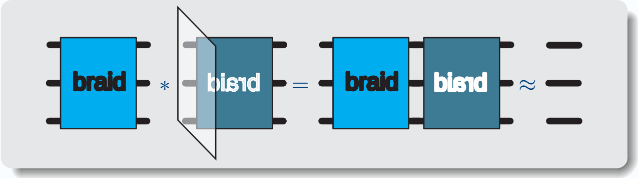


- Then well-defined with respect to isotopy), associative, admits a unit:



and inverses:

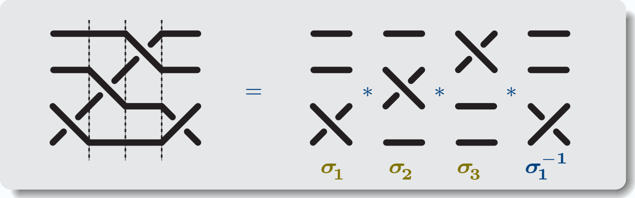
isotopic to



⇝ For each n, the group B_n of n-strand braids (E.Artin, 1925).

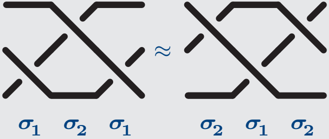
Artin presentation of B_n

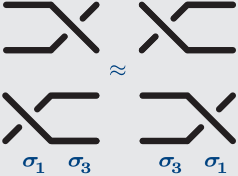
- Artin generators of B_n:



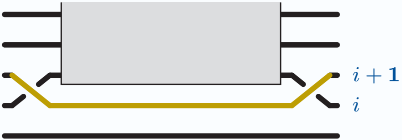
- Theorem (Artin): The group B_n is generated by $\sigma_1, \ldots, \sigma_{n-1}$,

subject to $\begin{cases} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{for } |i-j| = 1, \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{for } |i-j| \geqslant 2. \end{cases}$

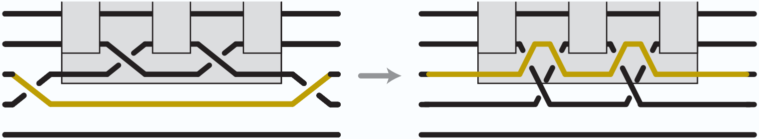

≈
σ_1 σ_2 σ_1
σ_2 σ_1 σ_2



- A $\boldsymbol{\sigma}_i$-handle:



- Reducing a handle:



- Handle reduction is an isotopy; It extends free group reduction;
  Terminal words cannot contain both $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_1^{-1}$.

- Theorem.— Every sequence of handle reductions terminates.

1. The Polish Algorithm for Left-Selfdistributivity
2. Handle reduction of braids
3. Subword reversing for positively presented groups

The baby problem I

- This time: a truly true rewrite system...

- Alphabet: $\mathtt{a}, \mathtt{b}, \mathtt{A}, \mathtt{B}$ (think of $\mathtt{A}$ as an inverse of a, etc.)

- Rewrite rules:
  - $\mathtt{Aa} \longrightarrow \varepsilon$, $\mathtt{Bb} \longrightarrow \varepsilon$ ("free group reduction" as usual, but only one direction)
  - $\mathtt{Ab} \longrightarrow \mathtt{bA}$, $\mathtt{Ba} \longrightarrow \mathtt{aB}$. ("reverse $-+$ patterns into $+-$ patterns")

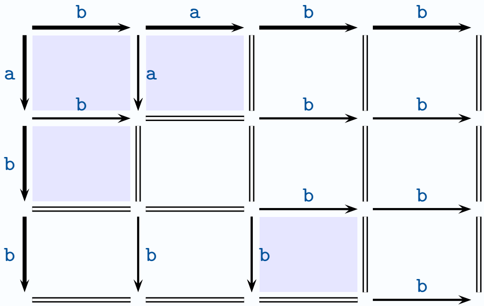
- Aim: transforming an arbitrary signed word into a positive–negative word.

- Example: $\mathtt{BB\underline{Ab}abb} \longrightarrow \mathtt{B\underline{Bb}Aabb} \longrightarrow \mathtt{B\underline{Aa}bb} \longrightarrow \mathtt{\underline{Bb}b} \longrightarrow \mathtt{b}$.

The baby problem II

- "Theorem".— It terminates in quadratic time.

- Proof: (obvious). Construct a reversing grid:



⇝ Clear that reversing terminates with quadratic time upper bound (and linear space upper bound).

- Obviously: *id.* for any number of letters.

The real problem I

- Example 2:
- Same alphabet: $\texttt{a}, \texttt{b}, \texttt{A}, \texttt{B}$

- Rewrite rules:
  - $\texttt{Aa} \longrightarrow \varepsilon$, $\texttt{Bb} \longrightarrow \varepsilon$ (free group reduction in one direction)
  - $\texttt{Ab} \longrightarrow \texttt{baBA}$, $\texttt{Ba} \longrightarrow \texttt{abAB}$. ("reverse $-+$ into $+-$", but different rule)

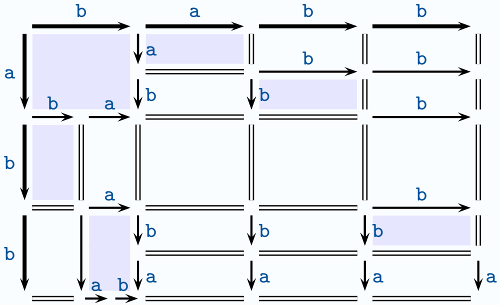
    ⇝ Again: transforms an arbitrary signed word into a positive–negative word.

- Termination? Not clear: length may increase...

- Example: $\texttt{BB}\underline{\texttt{Ab}}\texttt{abb} \longrightarrow \texttt{B}\underline{\texttt{Bb}}\texttt{aBAabb} \longrightarrow \underline{\texttt{Ba}}\texttt{BAabb}$
  $\longrightarrow \texttt{abABB}\underline{\texttt{Aa}}\texttt{bb} \longrightarrow \texttt{abAB}\underline{\texttt{Bb}}\texttt{b} \longrightarrow \texttt{abA}\underline{\texttt{Bb}} \longrightarrow \texttt{abA}.$

The real problem III

- Reversing grid: same, but possibly smaller and smaller arrows.



- <span>Theorem.— Reversing terminates in quadratic time (in this specific case).</span>

- Proof: Return to the baby case = find a (finite) set of words $\boldsymbol{S}$ that includes the alphabet and closed under reversing.

↑ for all $\boldsymbol{u}, \boldsymbol{v}$ in $\boldsymbol{S}$, exist $\boldsymbol{u}', \boldsymbol{v}'$ in $\boldsymbol{S}$ s.t. ∃ reversing grid

$$\begin{tikzcd}
{} \arrow[r, "\boldsymbol{v}"] \arrow[d, "\boldsymbol{u}"'] & {} \arrow[d, "\boldsymbol{u}'"] \\
{} \arrow[r, "\boldsymbol{v}'"'] & {}
\end{tikzcd}$$

Here: works with $\boldsymbol{S} = \{\mathtt{a}, \mathtt{b}, \mathtt{ab}, \mathtt{ba}\}$. □

Many more real problems

- Always like that? Not really...

- Example 3:

Alphabet $\mathtt{a}, \mathtt{b}, \mathtt{A}, \mathtt{B}$, rules $\mathtt{Aa} \longrightarrow \varepsilon$, $\mathtt{Bb} \longrightarrow \varepsilon$, plus $\mathtt{Ab} \longrightarrow \underbrace{\mathtt{baba}\ldots}_{m\,\text{letters}}\underbrace{\ldots\mathtt{BABA}}_{m\,\text{letters}}$, $\mathtt{Ba} \longrightarrow \underbrace{\mathtt{abab}\ldots}_{m\,\text{letters}}\underbrace{\ldots\mathtt{ABAB}}_{m\,\text{letters}}$.

⇝ Here : terminating in quadratic time and linear space

- Example 4:

Alphabet $\mathtt{a}, \mathtt{b}, \mathtt{A}, \mathtt{B}$, rules $\mathtt{Aa} \longrightarrow \varepsilon$, $\mathtt{Bb} \longrightarrow \varepsilon$, plus $\mathtt{Ab} \longrightarrow \mathtt{abA}$, $\mathtt{Ba} \longrightarrow \mathtt{aBA}$

Start with $\mathtt{Bab}$: $\underset{\substack{\uparrow\\ w}}{\underline{\mathtt{Ba}}\mathtt{b}} \longrightarrow \mathtt{aB}\underline{\mathtt{Ab}} \longrightarrow \underset{\substack{\uparrow\\ \mathtt{a}w\mathtt{A}}}{\mathtt{a}\underline{\mathtt{Ba}}\mathtt{bA}} \longrightarrow \mathtt{aaB}\underline{\mathtt{Ab}}\mathtt{A} \longrightarrow \underset{\substack{\uparrow\\ \mathtt{a}^2w\mathtt{A}^2}}{\mathtt{aa}\underline{\mathtt{Ba}}\mathtt{bAA}} \longrightarrow \mathtt{aaaB}\underline{\mathtt{Ab}}\mathtt{AA} \longrightarrow \mathtt{aaa}\underline{\mathtt{Ba}}\mathtt{bAAA} \longrightarrow \mathtt{aaaaB}\underline{\mathtt{Ab}}\mathtt{AAA}$

⇝ Here : non-terminating

- Example 5:

Alphabet $\mathtt{a}, \mathtt{b}, \mathtt{A}, \mathtt{B}$, rules $\mathtt{Aa} \longrightarrow \varepsilon$, $\mathtt{Bb} \longrightarrow \varepsilon$, plus $\mathtt{Ba} \longrightarrow \varepsilon$, $\mathtt{Ab} \longrightarrow \mathtt{abab}^2\mathtt{ab}^2\mathtt{abab}$

⇝ Here : terminating in cubic time and quadratic space

- What are we doing? We are working with a semigroup presentation and trying to represent the elements of the presented group by fractions.

- A semigroup presentation: list of generators (alphabet), plus list of relations, e.g., $\{\mathtt{a}, \mathtt{b}\}$, plus $\{\mathtt{aba} = \mathtt{bab}\}$. $\rightsquigarrow$ monoid $\langle \mathtt{a}, \mathtt{b} \mid \mathtt{aba} = \mathtt{bab} \rangle^+$, group $\langle \mathtt{a}, \mathtt{b} \mid \mathtt{aba} = \mathtt{bab} \rangle$.

- Definition.— Assume (A, R) semigroup presentation and, for all $s \neq t$ in A, there is exactly one relation $s ... = t ...$ in R, say $sC(s,t) = tC(t,s)$.
Then reversing is the rewrite system on $A \cup \overline{A}$ (a copy of A, here : capitalized letters) with rules $\overline{s}s \to \varepsilon$ and $\overline{s}t \to C(s,t)\overline{C(t,s)}$ for $s \neq t$ in A.

- Reversing does not change the element of the group that is represented;
$\rightsquigarrow$ if it terminates, every element of the group is a fraction fg^{-1} with f, g positive.

- Example 1 = reversing for the free Abelian group: $\langle \mathtt{a}, \mathtt{b} \mid \mathtt{ab} = \mathtt{ba} \rangle$;
- Example 2 = reversing for the 3-strand braid group: $\langle \mathtt{a}, \mathtt{b} \mid \mathtt{aba} = \mathtt{bab} \rangle$;
- Example 3 = reversing for type $I_2(m+1)$ Artin group: $\langle \mathtt{a}, \mathtt{b} \mid \underbrace{\mathtt{abab}...}_{m+1} = \underbrace{\mathtt{baba}...}_{m+1} \rangle$;
- Example 4 = reversing for the Baumslag–Solitar group: $\langle \mathtt{a}, \mathtt{b} \mid \mathtt{ab}^2 = \mathtt{ba} \rangle$;
- Example 5 = reversing for the ordered group: $\langle \mathtt{a}, \mathtt{b} \mid \mathtt{a} = \mathtt{babab}^2\mathtt{ab}^2\mathtt{abab} \rangle$.

- The only known facts:
  - reduction to the baby case ⇒ termination;
  - self-reproducing pattern ⇒ non-termination;

  - if reversing is complete for $(\boldsymbol{A}, \boldsymbol{R})$, then it is terminating iff any two elements of the monoid $\langle \boldsymbol{A} \mid \boldsymbol{R} \rangle^{+}$ admit a common right-multiple.

- Question.— What are YOU say about reversing?

For the Polish Algorithm:

• P. Dehornoy, Braids and selfdistributivity, Progress in math. vol 192, Birkhaüser 2000 (Chapter VIII)

• O. Deiser, Notes on the Polish Algorithm, deiser@tum.de (Technishe Universität München)

For Handle Reduction of braids:

• P. Dehornoy, with I. Dynnikov, D. Rolfsen, B. Wiest, Braid ordering, Math. Surveys and Monographs vol. 148, Amer. Math. Soc. 2008 (Chapter V)

For reversing associated with a semigroup presentation:

• P. Dehornoy, with F. Digne, E. Godelle, D. Krammer, J. Michel, Foundations of Garside Theory, submitted www.math.unicaen.fr/∼dehornoy/ (Chapter II)