Greedily Decomposing Proof Terms for String Rewriting into Multistep Derivations by Topological Multisorting

Vincent van Oostrom[1]

[1]Supported by EPSRC Project EP/R029121/1 Typed lambda-calculi with sharing and unsharing.

Causal equivalence in string rewriting

Example (Running)

string rewrite system (SRS) $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \rightarrow & A \\ \beta & : & AAB & \rightarrow & BAAB \end{array}$$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$ABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$ABAAB \to ABBAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \rightarrow & A \\ \beta & : & AAB & \rightarrow & BAAB \end{array}$$

$ABAAB \rightarrow ABBAAB \rightarrow AAAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$ABAAB \to ABBAAB \to AAAAB \to AABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\rightarrow A \\ \beta &: & AAB &\rightarrow BAAB \end{aligned}$$

$ABAAB \rightarrow ABBAAB \rightarrow AAAAB \rightarrow AABAAB \rightarrow BAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$ABAAB \to ABBAAB \to AAAAB \to AABAAB \to BAABAAB \to BBAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$ABAAB \to ABBAAB \to AAAAB \to AABAAB \to BAABAAB \to BBAABAAB \to AAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \rightarrow & A \\ \beta & : & AAB & \rightarrow & BAAB \end{array}$$

$ABAAB \rightarrow ABBAAB \rightarrow AAAAB \rightarrow AABAAB \rightarrow BAABAAB \rightarrow BBAABAAB \rightarrow AAABAAB \rightarrow ABAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$
\begin{aligned}
\alpha &: & BB &\rightarrow A \\
\beta &: & AAB &\rightarrow BAAB
\end{aligned}
$$

$$AB\underline{AAB} \rightarrow A\underline{BB}AAB \rightarrow AA\underline{AAB} \rightarrow \underline{AAB}AAB \rightarrow B\underline{AAB}AAB \rightarrow \underline{BB}AABAAB \rightarrow A\underline{AAB}AAB \rightarrow ABAABAAB$$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$$A B \underline{AAB} \to A \underline{BB} AAB \to AA \underline{AAB} \to \underline{AAB} AAB \to B \underline{AAB} AAB \to \underline{BB} AABAAB \to A \underline{AAB} AAB \to ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

observe 2nd–3rd steps causally independent, and 6th–7th steps too

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$ABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$$A B \underline{AAB} \to A \underline{BB} AAB \to AA \underline{AAB} \to \underline{AAB} AAB \to B \underline{AAB} AAB \to \underline{BB} AABAAB \to A \underline{AAB} AAB \to ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$$ABAAB \to ABBAAB$$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \rightarrow & A \\ \beta & : & AAB & \rightarrow & BAAB \end{array}$$

$$AB\underline{AAB} \rightarrow A\underline{BB}AAB \rightarrow AA\underline{AAB} \rightarrow \underline{AAB}AAB \rightarrow B\underline{AAB}AAB \rightarrow \underline{BB}AABAAB \rightarrow A\underline{AAB}AAB \rightarrow ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$$ABAAB \rightarrow ABBAAB \mathrel{-\!\!\circ\!\!\rightarrow} AABAAB$$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$ABAAB \to ABBAAB \multimap AABAAB \to BAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$ABAAB \to ABBAAB \multimap AABAAB \to BAABAAB \to BBAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{lcrcl} \alpha & : & BB & \rightarrow & A \\ \beta & : & AAB & \rightarrow & BAAB \end{array}$$

$AB\underline{AAB} \rightarrow A\underline{BB}AAB \rightarrow AA\underline{AAB} \rightarrow \underline{AAB}AAB \rightarrow B\underline{AAB}AAB \rightarrow \underline{BB}AABAAB \rightarrow A\underline{AAB}AAB \rightarrow ABAABAAB$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$ABAAB \rightarrow ABBAAB \multimap\!\!\rightarrow AABAAB \rightarrow BAABAAB \rightarrow BBAABAAB \multimap\!\!\rightarrow ABAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$
\begin{array}{lcrcl}
\alpha & : & BB & \rightarrow & A \\
\beta & : & AAB & \rightarrow & BAAB
\end{array}
$$

$$AB\underline{AAB} \rightarrow A\underline{BB}AAB \rightarrow AA\underline{AAB} \rightarrow \underline{AAB}AAB \rightarrow B\underline{AAB}AAB \rightarrow \underline{BB}AABAAB \rightarrow A\underline{AAB}AAB \rightarrow ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$$AB\underline{AAB} \rightarrow A\underline{BB}\,\underline{AAB} \xrightarrow{\circ} AAB\underline{AAB} \rightarrow B\underline{AAB}AAB \rightarrow \underline{BB}\,\underline{AAB}AAB \xrightarrow{\circ} ABAABAAB$$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P\rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$$AB\underline{AAB} \multimap\!\!\to A\underline{BB}\,\underline{AAB} \multimap\!\!\to \underline{AAB}AAB \multimap\!\!\to B\underline{AAB}AAB \multimap\!\!\to \underline{BB}\,\underline{AAB}AAB \multimap\!\!\to ABAABAAB$$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$$AB\underline{AAB} \multimap A\underline{BB}\,\underline{AAB} \multimap \underline{AAB}AAB \multimap B\underline{AAB}AAB \multimap \underline{BB}\,\underline{AAB}AAB \multimap ABAABAAB$$

multistep reduction $ABAAB \multimap\!\!\twoheadrightarrow ABAABAAB$

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$AB\underline{AAB} \multimap A\underline{BB}\underline{AAB} \multimap \underline{AAB}AAB \multimap B\underline{AAB}AAB \multimap \underline{BB}\underline{AAB}AAB \multimap ABAABAAB$

multistep reduction $ABAAB \multimap\!\!\twoheadrightarrow ABAABAAB$

observe both reductions do same amount of work: causally equivalent

Causal equivalence in string rewriting

Example (Running)

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$

reduction $ABAAB \twoheadrightarrow ABAABAAB$

$AB\underline{AAB} \multimap A\underline{BB}\,\underline{AAB} \multimap \underline{AAB}AAB \multimap B\underline{AAB}AAB \multimap \underline{BB}\,\underline{AAB}AAB \multimap ABAABAAB$

multistep reduction $ABAAB \multimap\!\!\!\twoheadrightarrow ABAABAAB$

this talk: 2nd is unique greedy multistep reduction causally equivalent to 1st

Methodology for defining equivalence of reductions

reduction in string rewrite system (Thue 1914)

Methodology for defining equivalence of reductions

reduction

↓ oudenadic / monadic embedding

proof term over signature, rule symbols, composition, and src / tgt (Meseguer 1990,Terese 2003)

Methodology for defining equivalence of reductions

reduction

↓ oudenadic / monadic embedding

proof term

↦ proof term algebra

causal graph (Wolfram 2002); trace relation / graph (Terese 2003 / here)

Methodology for defining equivalence of reductions

reduction

oudenadic / monadic embedding ↣

proof term

proof term algebra ↦

causal graph = tragr = proof term graph

composition of embedding and algebra maps induces equivalence on reductions

Methodology for defining equivalence of reductions

reduction

oudenadic / monadic embedding ↓

proof term

proof term algebra ↓

causal graph = tragr = proof term graph

composition of maps induces equivalence on reductions (via graph isomorphism)

Methodology for defining equivalence of deductions

morphism, deduction

inference as rewrite rules (Khasidashvili, 1995) ↓

reduction

oudenadic / monadic embedding ↓

proof term

proof term algebra ↦

causal graph = tragr = proof term graph

composition induces equivalence on morphisms, deductions (Guglielmi; paper)

Methodology for defining equivalence of deductions

morphism, deduction

↓ inference as rewrite rules

reduction

↓ oudenadic / monadic embedding

proof term

↦ proof term algebra

causal graph = tragr = proof term graph

composition induces equivalence on morphisms, deductions

Methodology for defining equivalence of reductions

reduction

↓ oudenadic / monadic embedding

proof term

↓ proof term algebra

causal graph = tragr = proof term graph

this talk: composition of maps induces equivalence on reductions

Embedding reductions into proof terms ($\rightarrowtail$)

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\alpha \ : \ BB \to A$$
$$\beta \ : \ AAB \to BAAB$$

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

$$\updownarrow$$

$$AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$$

replace redex-patterns by rule symbols α, β and arrows by composition symbol $\cdot$

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

$$\updownarrow$$

$$AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$$

$$AB\underline{AAB} \multimap A\underline{BB}\,\underline{AAB} \multimap \underline{AAB}AAB \multimap B\underline{AAB}AAB \multimap \underline{BB}\,\underline{AAB}AAB \multimap ABAABAAB$$

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{lcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

$$\downarrow$$

$$AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$$

$$AB\underline{AAB} \multimap A\underline{BB}\,\underline{AAB} \multimap \underline{AAB}AAB \multimap B\underline{AAB}AAB \multimap \underline{BB}\,\underline{AAB}AAB \multimap ABAABAAB$$

$$\downarrow$$

$$AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$$

multisteps may have multiple rule symbols; concurrent / parallel contraction

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \rightarrow & A \\ \beta & : & AAB & \rightarrow & BAAB \end{array}$$

- $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$
- $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

- $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$
- $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$

Definition (multistep and proof term)

multistep term over signature extended with rule symbols
proof term idem but also extended with composition $\cdot$ respecting src and tgt
for rule $\rho : \ell \to r$, $\mathrm{src}(\rho) := \ell$ and $\mathrm{tgt}(\rho) := r$; homomorphically extended

Embedding reductions into proof terms

Example

string rewrite system $\langle\Sigma, P\rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

- $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$
- $\mathrm{src}(\gamma) := \mathrm{src}(AB\beta) := AB\mathrm{src}(\beta) := ABAAB$

Definition (multistep and proof term)

multistep term over signature extended with rule symbols
proof term idem but also extended with composition $\cdot$ respecting src and tgt
for rule $\rho : \ell \to r$, $\mathrm{src}(\rho) := \ell$ and $\mathrm{tgt}(\rho) := r$; homomorphically extended

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{aligned} \alpha &: & BB &\to A \\ \beta &: & AAB &\to BAAB \end{aligned}$$

- $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$
- $\mathrm{src}(\gamma) := \mathrm{src}(AB\beta) := ABAAB$ and $\mathrm{tgt}(\gamma) := \mathrm{tgt}(A\beta AAB) := ABAABAAB$

Definition (multistep and proof term)

multistep term over signature extended with rule symbols
proof term idem but also extended with composition $\cdot$ respecting src and tgt
for rule $\rho : \ell \to r$, $\mathrm{src}(\rho) := \ell$ and $\mathrm{tgt}(\rho) := r$; homomorphically extended

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{lcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

- $\gamma : ABAAB \geqslant ABAABAAB$, target string P-reachable from source string
- $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$

Definition (multistep and proof term)

multistep term over signature extended with rule symbols
proof term idem but also extended with composition $\cdot$ respecting src and tgt
for rule $\rho : \ell \to r$, $\mathrm{src}(\rho) := \ell$ and $\mathrm{tgt}(\rho) := r$; homomorphically extended

Embedding reductions into proof terms

Example

string rewrite system $\langle \Sigma, P \rangle$; alphabet $\Sigma = \{A, B\}$ with letters A, B; rules P:

$$\begin{array}{rcrcl} \alpha & : & BB & \to & A \\ \beta & : & AAB & \to & BAAB \end{array}$$

- γ : $ABAAB \geqslant ABAABAAB$
- γ' : $ABAAB \geqslant ABAABAAB$

Definition (multistep and proof term)

multistep term over signature extended with rule symbols
proof term idem but also extended with composition $\cdot$ respecting src and tgt
for rule $\rho : \ell \to r$, $\mathrm{src}(\rho) := \ell$ and $\mathrm{tgt}(\rho) := r$; homomorphically extended

Properties of embedding $\rightarrowtail$

Lemma (multistep reductions as proof terms)

- *is injective (obvious);*

Properties of embedding $\rightarrowtail$

Lemma (multistep reductions as proof terms)

- *is injective;*
- *maps reductions to compositions of steps*

Properties of embedding $\rightarrowtail$

Lemma (multistep reductions as proof terms)

- *is injective;*
- *maps reductions to compositions of steps*
- *maps multistep reductions to compositions of multisteps*

Properties of embedding $\rightarrowtail$

Lemma (multistep reductions as proof terms)

- *is injective;*
- *maps reductions to compositions of steps*
- *maps multistep reductions to compositions of multisteps*
- *unique modulo associativity of composition* $\cdot$

Properties of embedding $\rightarrowtail$

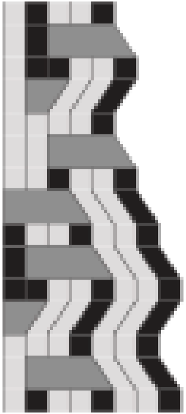
Lemma (multistep reductions as proof terms)

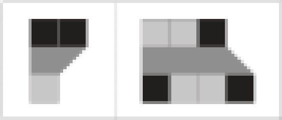
- *is injective;*
- *maps reductions to compositions of steps*
- *maps multistep reductions to compositions of multisteps*
- *unique modulo associativity of composition* $\cdot$

Upshot

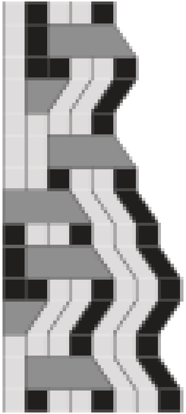
harmless to speak of (multistep) reductions to refer to the corresponding proof term modulo associativity

Evolution: visualisation of reduction γ (Wolfram 2002)





Evolution: visualisation of proof term γ

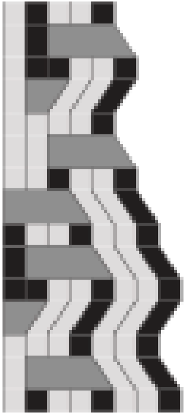


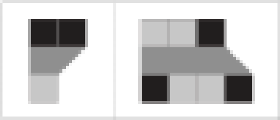
$ABAAB$
$\quad\Downarrow AB\beta$
$ABBAAB$
$\quad\Downarrow A\alpha AAB$
$AAAAB$
$\quad\Downarrow AA\beta$
$AABAAB$
$\quad\Downarrow \beta AAB$
$BAABAAB$
$\quad\Downarrow B\beta AAB$
$BBAABAAB$
$\quad\Downarrow \alpha AABAAB$
$AAABAAB$
$\quad\Downarrow A\beta AAB$
$ABAABAAB$

BB $\Downarrow \alpha$ A	AAB $\Downarrow \beta$ $BAAB$

$A \mapsto$ , $B \mapsto$ , $\alpha \mapsto$ , and $\beta \mapsto$ ; traces show causality (Terese 2003)

Evolution: visualisation of proof terms



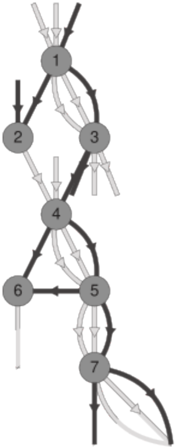


$$
\begin{array}{l}
ABAAB \\
\quad \downarrow AB\beta \\
ABBAAB \\
\quad \downarrow A\alpha AAB \\
AAAAB \\
\quad \downarrow AA\beta \\
AABAAB \\
\quad \downarrow \beta AAB \\
BAABAAB \\
\quad \downarrow B\beta AAB \\
BBAABAAB \\
\quad \downarrow \alpha AABAAB \\
AAABAAB \\
\quad \downarrow A\beta AAB \\
ABAABAAB
\end{array}
$$

$BB \downarrow \alpha\ A$	$AAB \downarrow \beta\ BAAB$

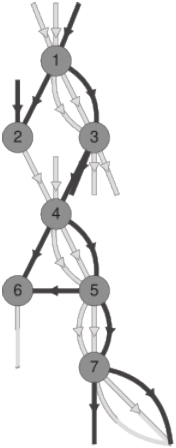
$A \mapsto$ □, $B \mapsto$ ■, $\alpha \mapsto$ , and $\beta \mapsto$ ; traces show causality (Terese 2003)

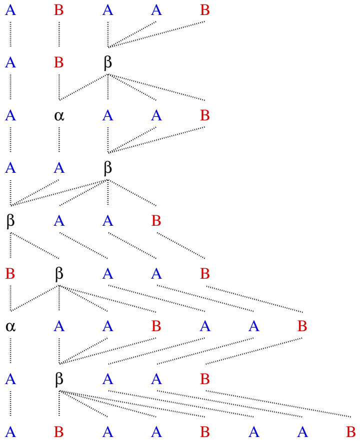
Causal graph of reduction γ (Wolfram 2002)



causal graph: rules as nodes with src and tgt symbols as edges

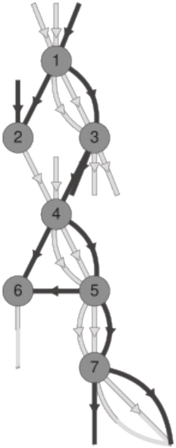
Trace relation of proof term γ (Terese 2003)

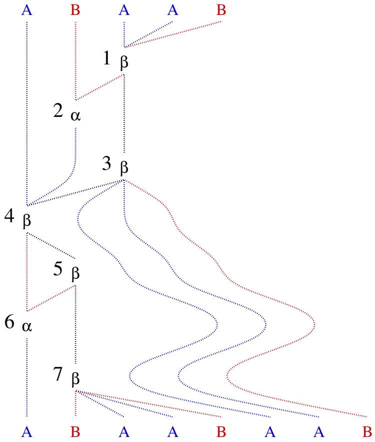




trace relation: rule and symbol positions with tracing as relation

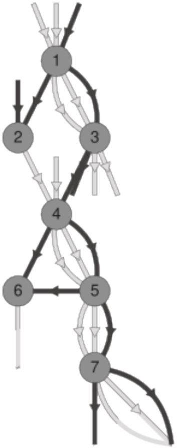
Trace relation of proof term γ

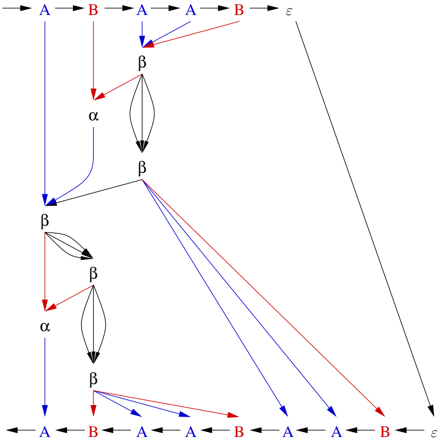




trace relation: rule positions with tracing as relation

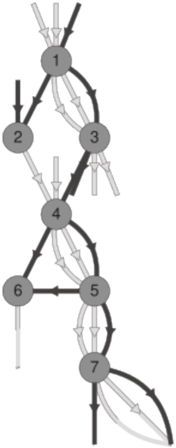
Trace graph of proof term γ

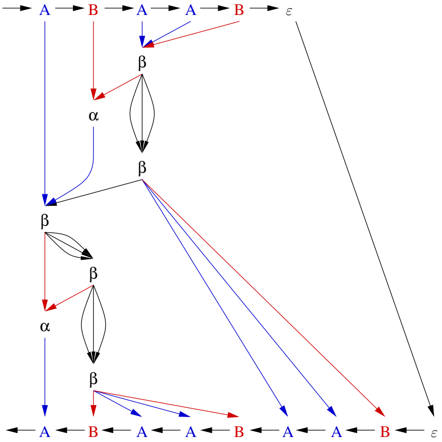




trace graph: rule positions with tracing as graph

Tragr of proof terms γ and γ'



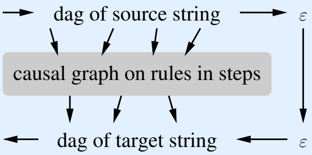


tragr: rule positions with tracing as graph

Tragrs by proof term algebra

Definition (tragr : symbol- and rule-labelled planar dag)

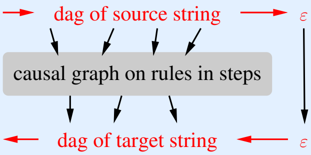
directed acyclic multigraph



Tragrs by proof term algebra

Definition (tragr : symbol- and rule-labelled planar dag)

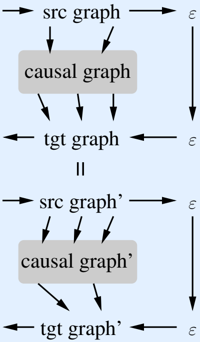
having source and target dags as interface



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\ ]\!]$)

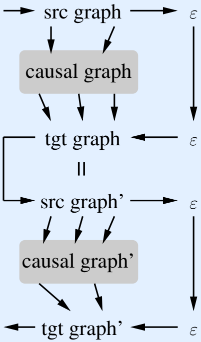
- composition $\gamma \cdot \gamma' \mapsto$ vertical (serial) composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

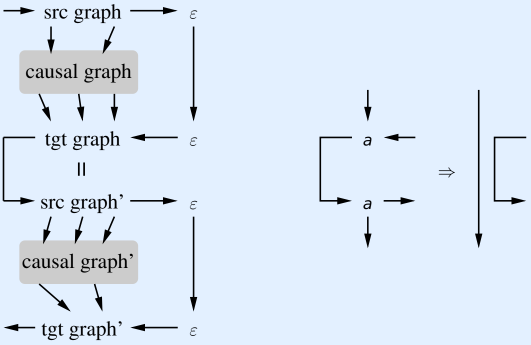
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

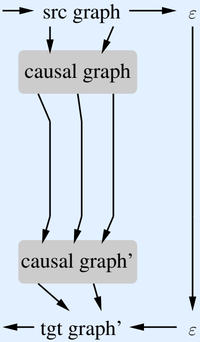
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$ + elision



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\ ]\!]$)

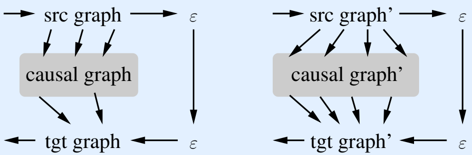
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

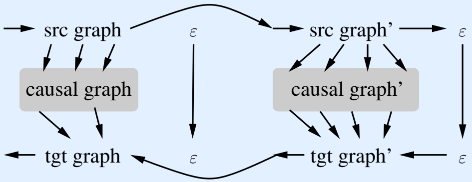
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal (parallel) composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

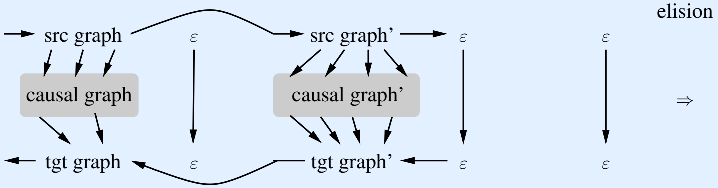
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

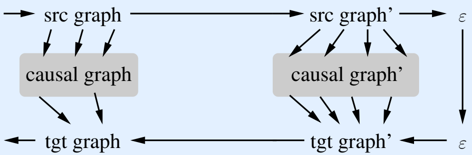
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$ + elision



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\;]\!]$)

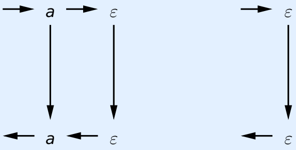
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

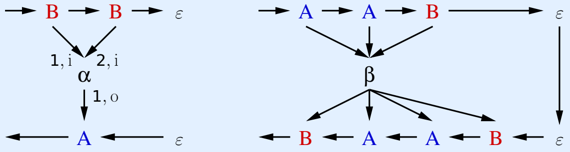
- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol *a* and empty string $\mapsto$ identity graph with 'itself' as source, target



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\ ]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph from dag of source string to dag of target string



Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph

this tragr algebra $[\![\,]\!]$ induces causal equivalence on proof terms

Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph

this tragr algebra $[\![\,]\!]$ induces causal equivalence on proof terms, $[\![\gamma]\!] = [\![\gamma']\!]$

Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\;]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph

Definition (permutation equivalence $\equiv$ (Lévy, Stark,...))

(left unit)	$s \cdot \gamma \equiv \gamma$	(associativity)	$(\gamma \cdot \delta) \cdot \zeta \equiv \gamma \cdot (\delta \cdot \zeta)$
(right unit)	$\gamma \cdot t \equiv \gamma$	(exchange)	$\gamma\delta \cdot \zeta\eta \equiv (\gamma \cdot \zeta)(\delta \cdot \eta)$

strings of (non-rule) symbols as vertical unit

Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph

Definition (permutation equivalence $\equiv$)

(left unit)	$\varepsilon\gamma \equiv \gamma$	(associativity)	$(\gamma\delta)\zeta \equiv \gamma(\delta\zeta)$
(right unit)	$\gamma\varepsilon \equiv \gamma$	(exchange)	$\gamma\delta \cdot \zeta\eta \equiv (\gamma \cdot \zeta)(\delta \cdot \eta)$

empty string ε as horizontal unit

Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\,]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph

Definition (permutation equivalence $\equiv$)

(left unit)	$\varepsilon\gamma \equiv \gamma$	(associativity)	$(\gamma\delta)\zeta \equiv \gamma(\delta\zeta)$
(right unit)	$\gamma\varepsilon \equiv \gamma$	(exchange)	$\gamma\delta \cdot \zeta\eta \equiv (\gamma \cdot \zeta)(\delta \cdot \eta)$

Lemma (permutation)

permutation equivalence induces causal equivalence: if $\gamma \equiv \delta$ *then* $[\![\gamma]\!] = [\![\delta]\!]$

Tragrs by proof term algebra

Definition (tragr proof term algebra $[\![\;]\!]$)

- composition $\gamma \cdot \gamma' \mapsto$ vertical composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- juxtaposition $\gamma\gamma' \mapsto$ horizontal composition of graphs $[\![\gamma]\!]$ and $[\![\gamma']\!]$
- symbol $\mapsto$ identity graph
- rule $\mapsto$ trace graph

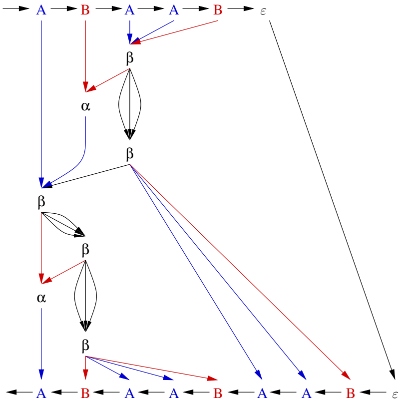
Definition (permutation equivalence $\equiv$)

(left unit)	$\varepsilon\gamma \equiv \gamma$	(associativity)	$(\gamma\delta)\zeta \equiv \gamma(\delta\zeta)$
(right unit)	$\gamma\varepsilon \equiv \gamma$	(exchange)	$\gamma\delta \cdot \zeta\eta \equiv (\gamma \cdot \zeta)(\delta \cdot \eta)$

Lemma (permutation)

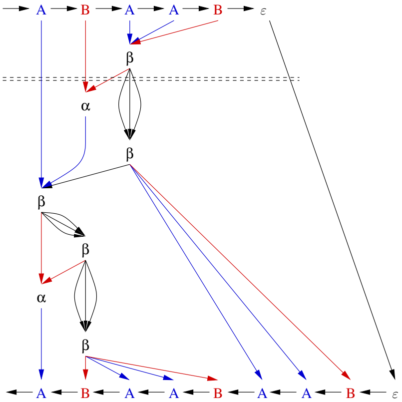
permutation equivalence induces causal equivalence; conversely?

Reading back multistep reductions from tragrs by TS



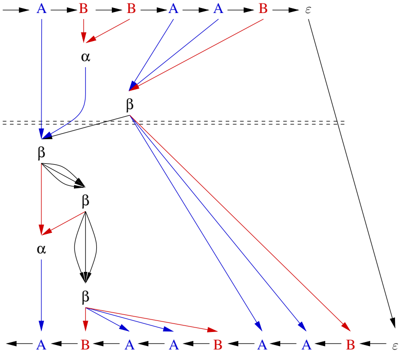
idea : by topological multisorting; maximal rule-parallelism

Reading back multistep reductions from tragrs by TS



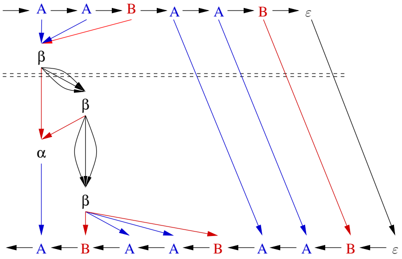
$\color{red}{A}\color{blue}{B}\beta \cdot \ldots$; later steps caused by this β

Reading back multistep reductions from tragrs by TS



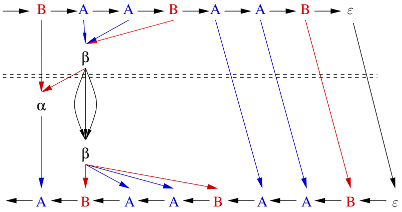
$AB\beta \cdot A\alpha\beta \cdot \ldots$; α and β independent; later steps caused by (one of) them

Reading back multistep reductions from tragrs by TS



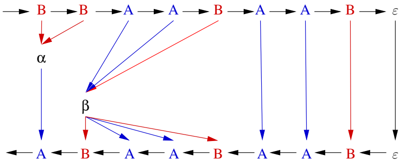
$\color{red}{A}\color{blue}{B}\beta \cdot \color{red}{A}\alpha\beta \cdot \beta\color{red}{AA}\color{blue}{B} \cdot \ldots$; later steps caused by this β

Reading back multistep reductions from tragrs by TS



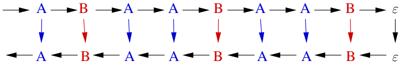
$\color{red}{A}\color{blue}{B}\beta \cdot \color{red}{A}\alpha\beta \cdot \beta\color{red}{A}\color{red}{A}\color{blue}{B} \cdot \color{blue}{B}\beta\color{red}{A}\color{red}{A}\color{blue}{B} \cdot \ldots$; later steps caused by this β

Reading back multistep reductions from tragrs by TS



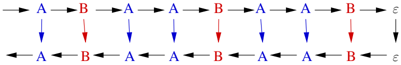
$\color{red}{A}\color{blue}{B}\beta \cdot \color{red}{A}\alpha\beta \cdot \beta\color{red}{AA}\color{blue}{B} \cdot \color{blue}{B}\beta\color{red}{AA}\color{blue}{B} \cdot \alpha\beta\color{red}{AA}\color{blue}{B} \cdot \ldots$; α and β independent; no later steps

Reading back multistep reductions from tragrs by TS



$AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$

Reading back multistep reductions from tragrs by TS



$AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB = \gamma'$!

Multistep reductions read back by TS are greedy

Definition (cf. greedy decomposition of Dehornoy et al. 2015)

- proof term greedy if multistep reduction without loath pairs

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ
  $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$ is not greedy

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

  $\gamma := AB\beta \cdot \overline{A\alpha\underline{AAB} \cdot AA\underline{\beta}} \cdot \beta AAB \cdot B\beta AAB \cdot \overline{\alpha\underline{AAB}AAB \cdot A\underline{\beta}AAB}$ loath pairs

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ
  $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$ is greedy; no loath pairs

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

Theorem (bijection)

bijection between greedy proof terms and tragrs (tragr algebra, topological sort)

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

Theorem (bijection)

bijection between greedy proof terms and tragrs

Proof.

topological sort of tragr gives greedy multistep reduction: by induction using that for multistep constructed from first layer, all later steps are (transitively) caused by some rule in that layer / multistep by sorting topologically □

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

Theorem (bijection)

bijection between greedy proof terms and tragrs

Proof.

identity if tragr obtained from greedy proof term by tragr algebra: by induction showing that for a greedy proof term its multisteps induce the layers of the topological sort when read back, since consecutive multisteps are not loath □

Multistep reductions read back by TS are greedy

Definition (cf. being sorted / standard if no out-of-order pairs)

- proof term greedy if multistep reduction without loath pairs
- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

Theorem (bijection)

bijection between greedy proof terms and tragrs

Example

reading back from the tragr of γ' yields γ' again, since it is greedy; not for γ

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ not caused by rule in Φ

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$; intuition: increase parallelism in 1st multistep

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Example

- $A\alpha\underline{AAB} \cdot AA\underline{\beta}$ swaps into $A\alpha\underline{\beta} \cdot AA\underline{BAAB}$

inverse of 1st multistep and step in 2nd multistep orthogonal

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Example

- $A\alpha\underline{AAB} \cdot AA\underline{\beta}$ swaps into $A\alpha\underline{\beta} \cdot AA\underline{BAAB}$
- $\alpha\underline{AAB}AAB \cdot A\underline{\beta}AAB$ swaps into $\alpha\underline{\beta}AAB \cdot A\underline{BAAB}AAB$

inverse of 1st multistep and step in 2nd multistep orthogonal

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Example

- $A\alpha\underline{AAB} \cdot AA\underline{\beta}$ swaps into $A\alpha\underline{\beta} \cdot AA\underline{BAAB}$
- $\alpha\underline{AAB}AAB \cdot A\underline{\beta}AAB$ swaps into $\alpha\underline{\beta}AAB \cdot A\underline{BAAB}AAB$
- γ greedily decomposes into $\gamma' \cdot ABAABAAB \cdot ABAABAAB$

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping + removing empty multisteps

Example

- $A\alpha\underline{AAB} \cdot AA\underline{\beta}$ swaps into $A\alpha\underline{\beta} \cdot AA\underline{BAAB}$
- $\alpha\underline{AAB}AAB \cdot A\underline{\beta}AAB$ swaps into $\alpha\underline{\beta}AAB \cdot A\underline{BAAB}AAB$
- γ greedily decomposes into γ'

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Theorem (greedy decomposition)

greedy decomposition γ' of γ exists (swapping terminates) and $\gamma \equiv \gamma'$

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Theorem (greedy decomposition)

greedy decomposition γ' of γ exists and is permutation equivalent to γ: $\gamma \equiv \gamma'$

Greedy multistep reductions by swapping loath pairs

Definition (swapping loath pairs)

- consecutive multisteps $\Phi \cdot \Psi$ loath if some rule in Ψ can be swapped into Φ: $\exists X$ such that $\Phi \subseteq X$ having residual step $\psi := X/\Phi$ with $\psi \subseteq \Psi$
- result of swap is $X \cdot (\Psi/\psi)$

greedy decomposition by exhaustive swapping

Theorem (greedy decomposition)

greedy decomposition γ' of γ exists and is permutation equivalent to γ: $\gamma \equiv \gamma'$

Proof.

termination : inverse lexicographic size (Huet & Lévy) of multisteps decreases
equivalence : loath pair equivalent to result of swap ($\Phi \cdot \Psi \equiv X \cdot (\Psi/\psi)$) □

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Lemma (confluence-by-evaluation (Plaisted 1985 / Hardin 1989))

rewrite system $\to$ *is confluent, if* nf *function on the objects and*

1. $\to$ *is normalising (WN)*
2. *if* $a \to b$ *then* $\mathrm{nf}(a) = \mathrm{nf}(b)$
3. *if* a *is a normal form, then* $\mathrm{nf}(a) = a$

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Lemma (CbE)

rewrite system $\to$ *is confluent, if* nf *function on the objects and*

1. $\to$ *is normalising*
2. *if* $a \to b$ *then* $\mathsf{nf}(a) = \mathsf{nf}(b)$
3. *if* a *is a normal form, then* $\mathsf{nf}(a) = a$

Proof.

if $b \twoheadleftarrow a \twoheadrightarrow c$ $\square$

semantical; local confluence / Newman's Lemma not used

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Lemma (CbE)

rewrite system $\to$ *is confluent, if* nf *function on the objects and*

1. $\to$ *is normalising*
2. *if* $a \to b$ *then* $\mathsf{nf}(a) = \mathsf{nf}(b)$
3. *if* a *is a normal form, then* $\mathsf{nf}(a) = a$

Proof.

then $b' \twoheadleftarrow b \twoheadleftarrow a \twoheadrightarrow c \twoheadrightarrow c'$ for normal forms b', c' by (1) $\square$

semantical; local confluence / Newman's Lemma not used

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Lemma (CbE)

rewrite system $\to$ *is confluent, if* nf *function on the objects and*

1. $\to$ *is normalising*
2. *if* $a \to b$ *then* $\mathrm{nf}(a) = \mathrm{nf}(b)$
3. *if* a *is a normal form, then* $\mathrm{nf}(a) = a$

Proof.

hence $\mathrm{nf}(b') = \mathrm{nf}(c')$ by convertibility of b' and c' and (2) $\square$

semantical; local confluence / Newman's Lemma not used

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ proof terms γ, $\exists!$ greedy multistep reduction γ' such that $\gamma \equiv \gamma'$

Lemma (CbE)

rewrite system $\to$ is confluent, if nf *function on the objects and*

1. *$\to$ is normalising*
2. *if $a \to b$ then $\mathrm{nf}(a) = \mathrm{nf}(b)$*
3. *if a is a normal form, then $\mathrm{nf}(a) = a$*

Proof.

so $b' = c'$ by (3), i.e. $b \twoheadrightarrow b' = c' \twoheadleftarrow c$ $\square$

semantical; local confluence / Newman's Lemma not used

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\,]\!]$ followed by read back TS:

$\square$

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\,]\!]$ followed by read back TS:

1. swapping is terminating (by greedy decomposition theorem), hence normalising

□

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\ ]\!]$ followed by read back TS:

1. swapping is terminating, hence normalising
2. nf is preserved by swapping since $[\![\ ]\!]$ is by permutation lemma using: proof term $\equiv$ multistep reduction (serialisation)

□

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\,]\!]$ followed by read back TS:

1. swapping is terminating, hence normalising
2. nf is preserved by swapping since $[\![\,]\!]$ is by permutation lemma using: proof term $\equiv$ greedy multistep reduction (greedy decomposition theorem)

$\square$

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\ ]\!]$ followed by read back TS:

1. swapping is terminating, hence normalising
2. nf is preserved by swapping since $[\![\ ]\!]$ is
3. nf is identity on greedy normal forms

□

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\,]\!]$ followed by read back TS:

1. swapping is terminating, hence normalising
2. nf is preserved by swapping since $[\![\,]\!]$ is
3. nf is identity on greedy normal forms

by CbE swapping is complete (confluent and terminating) $\square$

Greedy multistep reduction represents $\equiv$-class

Theorem (permutation equivalence via causal equivalence)

$\forall$ *proof terms* γ, $\exists!$ *greedy multistep reduction* γ' *such that* $\gamma \equiv \gamma'$

Proof.

for swap rewrite system and nf mapping to $[\![\,]\!]$ followed by read back TS:

1. swapping is terminating, hence normalising
2. nf is preserved by swapping since $[\![\,]\!]$ is
3. nf is identity on greedy normal forms

by CbE swapping is complete (confluent and terminating) □

Upshot

permutation $\simeq$ causal equivalence; greedy multistep reduction $\simeq$ causal graph

Conclusions / directions

1. physics (causal graph; Wolfram)

Conclusions / directions

1. physics, Garside theory (greedy decomposition; Dehornoy)

Conclusions / directions

1. physics, Garside theory and concurrency theory (CTS; Stark)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting ($\equiv$; Lévy)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic (myopic; intentional?)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same (sorted $\simeq$ decomposed $\simeq$ standard)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS (nullary, modulo AC)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding (unary)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation? ($abc \to ac \to d$? for rules $b \to \varepsilon$, $ac \to d$; see paper)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity? (area? width (parallel) vs. length (serial))

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting? cf. sharing graphs (Lamping 1990)
   TRS non-linear: replication vs. causation (Terese 2003)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE? (ground confluence of $0, S, A$; Futatsugi)

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction $\rightarrowtail$ proof term

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction $\rightarrowtail$ proof term modulo causality

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction $\rightarrowtail$ proof term modulo causality $\leftrightarrow$ causal graph

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction $\rightarrowtail$ proof term modulo causality $\leftrightarrow$ tragr

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction $\rightarrowtail$ proof term modulo causality $\leftrightarrow$ proof term graph

Conclusions / directions

1. physics, Garside theory and concurrency theory mirror rewriting: causality
2. cross-citing sporadic, methods same
3. oudenadic embedding of SRS in TRS; in paper monadic embedding
4. empty causation?
5. complexity?
6. extend to term rewriting?
7. application / automation of CbE?
8. morphism, deduction $\rightarrowtail$ proof term modulo causality $\leftrightarrow$ proof term graph

thank you

(return to NL tomorrow night; contact me after at `oostrom@javakade.nl`)