# Approximating Relative Match-Bounds

Alfons Geser[1], *Dieter Hofbauer*[2], Johannes Waldmann[1]

[1]HTWK Leipzig (Germany), [2]ASW Saarland (Germany)

18th Workshop on Termination
Haifa, Israel, August 11–12, 2022



## Motivation

**The 595 problems from `TPDB/SRS_STANDARD/ICFP_2010` are**

- large: avg. 70 rules of size 2340 (non-`ICFP`: 3.3 of size 21.5)
- time consuming: VBS CPU time at termCOMP'21
  avg. 90", median 28" (non-`ICFP`: avg. 51", median 6")
- hard: VBS at termCOMP'21 solves 86 % (non-`ICFP`: 96 %)



## Motivation

**The 595 problems from `TPDB/SRS_STANDARD/ICFP_2010` are**

- large: avg. 70 rules of size 2340 (non-`ICFP`: 3.3 of size 21.5)
- time consuming: VBS CPU time at termCOMP'21
  avg. 90", median 28" (non-`ICFP`: avg. 51", median 6")
- hard: VBS at termCOMP'21 solves 86 % (non-`ICFP`: 96 %)

**termCOMP'21 versus '22**

| | Matchbox | MnM | VBS |
|---|---|---|---|
| termCOMP'21 | 510 | 417 | 514 |
| termCOMP'22 | 595 | 594 | 595 |



## Overview

**Methods from this talk** (timeout 10")

| | **rb** | **rel. rb** | **mb** | **rel. mb** |
|---|---|---|---|---|
| solved | 370 | 568 | 588 | 590 |
| % | 62.2 | 95.5 | 98.8 | 99.2 |
| avg. CPU time | 0.29" | 0.88" | 1.37" | 0.93" |

**rb**: right barren / **mb**: approx. RFC-match-bounded
combined with weights + reversal; iterated for rel.



## Overview

**Methods from this talk** (timeout 10")

| | **rb** | **rel. rb** | **mb** | **rel. mb** |
|---|---|---|---|---|
| solved | 370 | 568 | 588 | 590 |
| % | 62.2 | 95.5 | 98.8 | 99.2 |
| avg. CPU time | 0.29" | 0.88" | 1.37" | 0.93" |

**rb**: right barren / **mb**: approx. RFC-match-bounded
combined with weights + reversal; iterated for rel.

Example: `ICFP/180915` (180 rules on 6 letters)

$$180 \xrightarrow{\text{rev}} 180 \xrightarrow{\text{rel. mb (2)}} 45 \xrightarrow{\text{rev}} 45 \xrightarrow{\text{rel. mb (1)}} 0$$



## Overview

**Methods from this talk** (timeout 10")

| | **rb** | **rel. rb** | **mb** | **rel. mb** |
|---|---|---|---|---|
| solved | 370 | 568 | 588 | 590 |
| % | 62.2 | 95.5 | 98.8 | 99.2 |
| avg. CPU time | 0.29" | 0.88" | 1.37" | 0.93" |

**rb**: right barren / **mb**: approx. RFC-match-bounded
combined with weights + reversal; iterated for rel.

Example: `ICFP/180915` (180 rules on 6 letters)

$$180 \xrightarrow{\text{rev}} 180 \xrightarrow{\text{rel. mb (2)}} 45 \xrightarrow{\text{rev}} 45 \xrightarrow{\text{rel. mb (1)}} 0$$

- Idea: remove relatively (on RFC) match-bounded rules (H/W'10)
- New: approximate this property fast
- Ingredients: (Dershowitz'81); (Büchi'64); (McNaughton'94, Geser'01); automata completion (various authors)
- Independent implementations in Matchbox and MnM



## Termination of (string) rewriting

**Modular termination proofs by removing rules**

- SN($R$): $R$ is *terminating* (or: *strongly normalizing*) if every $R$-derivation contains only finitely many $R$-steps.
- SN($R/S$): $R$ is *terminating relative to* $S$ if every $(R \cup S)$-derivation contains only finitely many $R$-steps.
- Theorem: If SN($R/S$) and SN($S$) then SN($R \cup S$).



## Termination of (string) rewriting

**Modular termination proofs by removing rules**

- SN($R$): $R$ is *terminating* (or: *strongly normalizing*) if every $R$-derivation contains only finitely many $R$-steps.
- SN($R/S$): $R$ is *terminating relative to* $S$ if every $(R \cup S)$-derivation contains only finitely many $R$-steps.
- Theorem: If SN($R/S$) and SN($S$) then SN($R \cup S$).

**How to prove SN($R$), or prove SN($R/S$)?**

- Ad hoc approach: $0 \in$ finitely many.
  Show that $R$-steps do not occur in any $R$-derivation, or
  show that $R$-steps do not occur in any $(R \cup S)$-derivation.

## Termination of (string) rewriting

### Modular termination proofs by removing rules

- SN($R$): $R$ is *terminating* (or: *strongly normalizing*) if every $R$-derivation contains only finitely many $R$-steps.
- SN($R/S$): $R$ is *terminating relative to* $S$ if every $(R \cup S)$-derivation contains only finitely many $R$-steps.
- Theorem: If SN($R/S$) and SN($S$) then SN($R \cup S$).

### How to prove SN($R$), or prove SN($R/S$)?

- Ad hoc approach: $0 \in$ finitely many. Show that $R$-steps do not occur in any $R$-derivation, or show that $R$-steps do not occur in any $(R \cup S)$-derivation.
- Nonsensical, this is never the case . . . . . . but could work for a restricted set of derivations.



## Restricting the set of derivations

### Definition: Right-hand sides of forward closures

- $\mathrm{RFC}(R) = (\to_R \cup \rightharpoonup_{\mathrm{right}(R)})^*(\mathrm{rhs}(R))$, where $\rightharpoonup$ is suffix rewriting, and $\mathrm{right}(R) = \{\ell_1 \to r \mid (\ell_1\ell_2 \to r) \in R, \ell_1 \neq \epsilon \neq \ell_2\}$.
- $\to_R$ are *inner steps*, $\rightharpoonup_{\mathrm{right}(R)}$ are *suffix extension steps*.



## Restricting the set of derivations

### Definition: Right-hand sides of forward closures

- $\mathrm{RFC}(R) = (\to_R \cup \rightharpoonup_{\mathrm{right}(R)})^*(\mathrm{rhs}(R))$, where $\rightharpoonup$ is suffix rewriting, and $\mathrm{right}(R) = \{\ell_1 \to r \mid (\ell_1\ell_2 \to r) \in R, \ell_1 \neq \epsilon \neq \ell_2\}$.
- $\to_R$ are *inner steps*, $\rightharpoonup_{\mathrm{right}(R)}$ are *suffix extension steps*.

### Theorem (Dershowitz'81)

$R$ is terminating iff $R$ is terminating on RFC($R$).



## Restricting the set of derivations

### Definition: Right-hand sides of forward closures

- $\mathrm{RFC}(R) = (\to_R \cup \rightharpoonup_{\mathrm{right}(R)})^*(\mathrm{rhs}(R))$, where $\rightharpoonup$ is suffix rewriting, and $\mathrm{right}(R) = \{\ell_1 \to r \mid (\ell_1\ell_2 \to r) \in R, \ell_1 \neq \epsilon \neq \ell_2\}$.
- $\to_R$ are *inner steps*, $\rightharpoonup_{\mathrm{right}(R)}$ are *suffix extension steps*.

### Theorem (Dershowitz'81)

$R$ is terminating iff $R$ is terminating on RFC($R$).

### Example: $R = \{ab \to ba\}$

Here, $\mathrm{right}(R) = \{a \to ba\}$, so $\mathrm{RFC}(R) = (\to_R \cup \rightharpoonup_{\mathrm{right}(R)})^*(ba) = b^+a$. RFC($R$) contains no $R$-redex, so $R$ is terminating.



## Right barren string rewriting

Generalizing McNaughton'94, Geser'01 from 1-rule to arbitrary finite systems:

### Definition: $R$ is *right barren*

if no $\ell \in \mathrm{lhs}(R)$ is factor of a string in RFC($R$).



## Right barren string rewriting

Generalizing McNaughton'94, Geser'01 from 1-rule to arbitrary finite systems:

### Definition: $R$ is *right barren*

if no $\ell \in \mathrm{lhs}(R)$ is factor of a string in RFC($R$).

### Theorem

This property is decidable, and it implies termination.



## Right barren string rewriting

Generalizing McNaughton'94, Geser'01 from 1-rule to arbitrary finite systems:

### Definition: $R$ is *right barren*

if no $\ell \in \mathrm{lhs}(R)$ is factor of a string in RFC($R$).

### Theorem

This property is decidable, and it implies termination.

### Proof of decidability

If $R$ is right barren, $\mathrm{RFC}(R) = \rightharpoonup_{\mathrm{right}(R)}{}^*(\mathrm{rhs}(R))$. This set is regular, since regularity is preserved under suffix rewriting (Büchi'64).



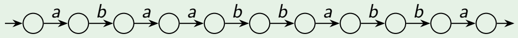
## Right barren string rewriting (cont'd)

### Example: $R = \{babbaba \to abaabbabba\}$

## Right barren string rewriting (cont'd)

**Example: $R = \{babbaba \to abaabbabba\}$**

Automaton accepting rhs($R$):

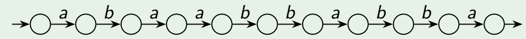




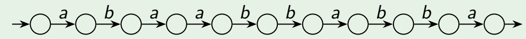

## Right barren string rewriting (cont'd)

**Example: $R = \{babbaba \to abaabbabba\}$**

Automaton accepting rhs($R$):




Closure under $\to_{\text{right}(R)}$ by adding epsilon transitions:

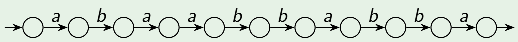




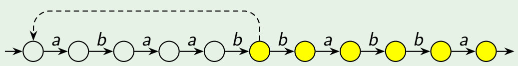

## Right barren string rewriting (cont'd)

**Example: $R = \{babbaba \to abaabbabba\}$**

Automaton accepting rhs($R$):




Closure under $\to_{\text{right}(R)}$ by adding epsilon transitions:

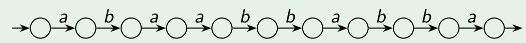




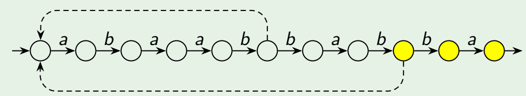

## Right barren string rewriting (cont'd)

**Example: $R = \{babbaba \to abaabbabba\}$**

Automaton accepting rhs($R$):




Closure under $\to_{\text{right}(R)}$ by adding epsilon transitions:

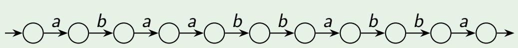




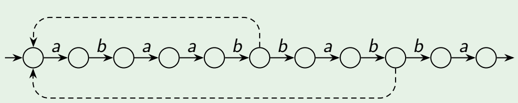

## Right barren string rewriting (cont'd)

**Example: $R = \{babbaba \to abaabbabba\}$**

Automaton accepting rhs($R$):




Closure under $\to_{\text{right}(R)}$ by adding epsilon transitions:



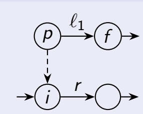

The left-hand side of $R$ is not a factor of any accepted string, so $R$ is right barren, thus terminating.



## Right barren string rewriting (cont'd)

**Closure algorithm: suffix matches**

For state $p$, final state $f$, $(\ell_1 \to r) \in \text{right}(R)$:
If there is a path $p \xrightarrow{\ell_1} f$, add $p \xrightarrow{\epsilon} i$,
where $i$ is the initial state of the path for $r$.

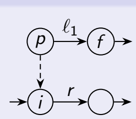





## Right barren string rewriting (cont'd)

**Closure algorithm: suffix matches**

For state $p$, final state $f$, $(\ell_1 \to r) \in \text{right}(R)$:
If there is a path $p \xrightarrow{\ell_1} f$, add $p \xrightarrow{\epsilon} i$,
where $i$ is the initial state of the path for $r$.




- Termination of this algorithm: No new nodes, so there are only finitely many possible epsilon transitions.
- Decide whether $\ell \in lhs(R)$ is a factor of some accepted string: check for path $p \xrightarrow{\ell} q$ (states are accessible and co-accessible).



## Removing relatively right barren rules

**Definition: $S \subseteq R$ is *relatively right barren* w. r. t. $R \setminus S$**

if no $\ell \in \text{lhs}(S)$ is factor of a string in RFC($R$).

## Removing relatively right barren rules

**Definition:** $S \subseteq R$ is *relatively right barren* w. r. t. $R \setminus S$
if no $\ell \in \mathrm{lhs}(S)$ is factor of a string in $\mathrm{RFC}(R)$.

**Theorem:** Let $S \subseteq R$ be relatively right barren w. r. t. $R \setminus S$.
Then $\mathrm{SN}(R \setminus S)$ implies $\mathrm{SN}(R)$.



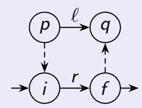
## Removing relatively right barren rules

**Definition:** $S \subseteq R$ is *relatively right barren* w. r. t. $R \setminus S$
if no $\ell \in \mathrm{lhs}(S)$ is factor of a string in $\mathrm{RFC}(R)$.

**Theorem:** Let $S \subseteq R$ be relatively right barren w. r. t. $R \setminus S$.
Then $\mathrm{SN}(R \setminus S)$ implies $\mathrm{SN}(R)$.

**Closure algorithm: suffix and redex matches**

Closure steps for suffix matches as before.
Closure steps for redex matches:
For states $p$, $q$, and $(\ell \to r) \in R$:
If there is a path $p \xrightarrow{\ell} q$, add $p \xrightarrow{\epsilon} i$ and $f \xrightarrow{\epsilon} q$,
where $i$ and $f$ are the initial resp. final state of the path for $r$.





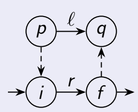

## Removing relatively right barren rules

**Definition:** $S \subseteq R$ is *relatively right barren* w. r. t. $R \setminus S$
if no $\ell \in \mathrm{lhs}(S)$ is factor of a string in $\mathrm{RFC}(R)$.

**Theorem:** Let $S \subseteq R$ be relatively right barren w. r. t. $R \setminus S$.
Then $\mathrm{SN}(R \setminus S)$ implies $\mathrm{SN}(R)$.

**Closure algorithm: suffix and redex matches**

Closure steps for suffix matches as before.
Closure steps for redex matches:
For states $p$, $q$, and $(\ell \to r) \in R$:
If there is a path $p \xrightarrow{\ell} q$, add $p \xrightarrow{\epsilon} i$ and $f \xrightarrow{\epsilon} q$,
where $i$ and $f$ are the initial resp. final state of the path for $r$.



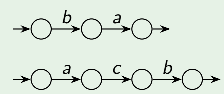

The resulting automaton over-approximates $\mathrm{RFC}(R)$.



## Removing relatively right barren rules (cont'd)

**Example:** $R = \{ab \to ba, ba \to acb\}$ (`Zantema_04/z006`)

Automaton for $\mathrm{rhs}(R)$:

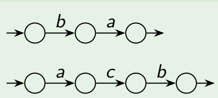




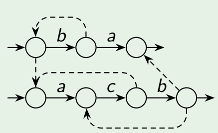

## Removing relatively right barren rules (cont'd)

**Example:** $R = \{ab \to ba, ba \to acb\}$ (`Zantema_04/z006`)

Automaton for $\mathrm{rhs}(R)$:




Closure under $\to_R \cup \to_{\mathrm{right}(R)}$:

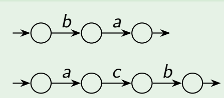




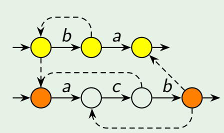

## Removing relatively right barren rules (cont'd)

**Example:** $R = \{ab \to ba, ba \to acb\}$ (`Zantema_04/z006`)

Automaton for $\mathrm{rhs}(R)$:




Closure under $\to_R \cup \to_{\mathrm{right}(R)}$:

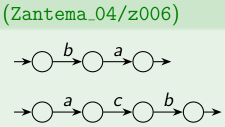




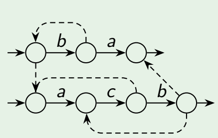

## Removing relatively right barren rules (cont'd)

**Example:** $R = \{ab \to ba, ba \to acb\}$ (`Zantema_04/z006`)

Automaton for $\mathrm{rhs}(R)$:




Closure under $\to_R \cup \to_{\mathrm{right}(R)}$:




There is no path labelled by the left-hand side of $S = \{ab \to ba\}$:
$S$ is relatively right barren w. r. t. $R \setminus S$. As $R \setminus S = \{ba \to acb\}$ is terminating (it is right barren), $R$ is terminating.



## Approximating match-bounds

- Refine the approximation of $\mathrm{RFC}(R)$ by match-heights (G/H/W'03).

## Approximating match-bounds

- Refine the approximation of RFC($R$) by match-heights (G/H/W'03).
- Fix $B \in \mathbb{N}$ and start with $B+1$ disjoints paths for each $r \in rhs(R)$. Layer $h \leq B$ corresponds to height $h$.
- Initial and final states are at height 0.



## Approximating match-bounds

- Refine the approximation of RFC($R$) by match-heights (G/H/W'03).
- Fix $B \in \mathbb{N}$ and start with $B+1$ disjoints paths for each $r \in rhs(R)$. Layer $h \leq B$ corresponds to height $h$.
- Initial and final states are at height 0.
- Suffix matches always link to height 0.
- Redex matches have height $h = \min$(layer of letter transition); epsilon transitions have no height. Reject if $h = B$, otherwise link to reduct path at height $h+1$.



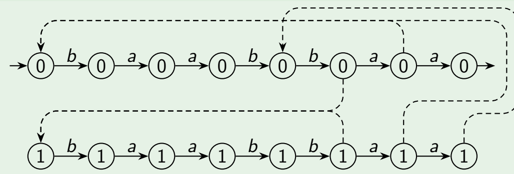
## Approximating match-bounds

- Refine the approximation of RFC($R$) by match-heights (G/H/W'03).
- Fix $B \in \mathbb{N}$ and start with $B+1$ disjoints paths for each $r \in rhs(R)$. Layer $h \leq B$ corresponds to height $h$.
- Initial and final states are at height 0.
- Suffix matches always link to height 0.
- Redex matches have height $h = \min$(layer of letter transition); epsilon transitions have no height. Reject if $h = B$, otherwise link to reduct path at height $h+1$.
- In case of success: complete automaton is a certificate for match-bound $B$ on RFC($R$).



## Approximating match-bounds (cont'd)

**Example:** $R = \{abaab \to baabbaa\}$ (`Zantema_04/z034` reversed)

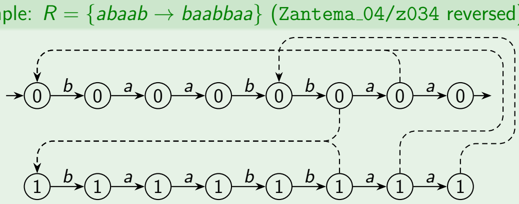


Complete automaton is a certificate for match-bound 1 on RFC($R$).



## Approximating match-bounds (cont'd)

**Example:** $R = \{abaab \to baabbaa\}$ (`Zantema_04/z034` reversed)



Complete automaton is a certificate for match-bound 1 on RFC($R$).

**Removing relatively match-bounded rules (sketch)**

- Now layer $B$ represents all heights $\geq B$; we never reject.
- After completion, remove those rules where all redex heights are $< B$: they are match-bounded relative to the remaining rules by $B$ on RFC, so they terminating relative to the remaining rules.



## Summary and discussion

- This method solves `SRS_STANDARD/ICFP_2010`. Weaker on non-`ICFP`: Solves 164 of 1056.
- Cannot solve `Zantema_04/z001`.
- But, by iteration, solves problems that are not (RFC-)match-bounded.
- Two independent implementations: Confidence, no certification.
- Combined with *drop common prefix/suffix*, nearly solves `Wenzel_16`: MnM solves 222 of 226.
- Implementation: keep the set of epsilon transitions transitively closed.
- Strategy: fix B = 2 or choose B = 0, 1, ...?



## Summary and discussion

- This method solves `SRS_STANDARD/ICFP_2010`. Weaker on non-`ICFP`: Solves 164 of 1056.
- Cannot solve `Zantema_04/z001`.
- But, by iteration, solves problems that are not (RFC-)match-bounded.
- Two independent implementations: Confidence, no certification.
- Combined with *drop common prefix/suffix*, nearly solves `Wenzel_16`: MnM solves 222 of 226.
- Implementation: keep the set of epsilon transitions transitively closed.
- Strategy: fix B = 2 or choose B = 0, 1, ...?

- Challenge: merge this method with the exact RFC-method (Endrullis/H/W'06).
- Challenge: termCOMP needs more SRS benchmarks — that are independent of any specific method. Continue systematic or random enumeration.