Maximally Parallel Contextual String Rewriting

Traian Florin Șerbănuță[1]

University of Bucharest

Diaspora științifică 2016

[1] joint work with Liviu P. Dinu

Contents

Hyphenation rules!

(for Romanian)

- Hyphenation for Romanian seems simple
  - 8 insertion rules for *regular* words [Dinu, 2003]
  - Regular: no consecutive vowels (to exclude diphthongs, triphthongs).

In the rules below *v*s stand for vowels, *c*s for consonants

1. $(v_1, \text{-}, cv_2)$
2. $(v_1, \text{-}, c_1c_2v_2)$ if $c_1c_2 \in \{\text{ch, gh}\}$ or $(c_1, c_2) \in \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
3. $(v_1c_1, \text{-}, c_2v_2)$ if $c_1c_2 \notin \{\text{ch, gh}\}$ and $(c_1, c_2) \notin \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
4. $(v_1c_1c_2, \text{-}, c_3v_2)$ if $c_1c_2c_3 \in \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
5. $(v_1c_1, \text{-}, c_2c_3v_2)$ if $c_1c_2c_3 \notin \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
6. $(v_1c_1c_2, \text{-}, c_3c_4v_2)$ if $c_2c_3c_4 \in \{\text{gst, nbl}\}$
7. $(v_1c_1, \text{-}, c_2c_3c_4v_2)$ if $c_2c_3c_4 \notin \{\text{gst, nbl}\}$
8. $(v_1c_1c_2, \text{-}, c_3c_4c_5v_2)$ if $c_1c_2c_3c_4c_5 \in \{\text{ptspr, stscr}\}$

Hyphenation example

- lingvistică

1. $(v_1, \text{-}, cv_2)$
2. $(v_1, \text{-}, c_1c_2v_2)$ if $c_1c_2 \in \{\text{ch, gh}\}$ or $(c_1, c_2) \in \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
3. $(v_1c_1, \text{-}, c_2v_2)$ if $c_1c_2 \notin \{\text{ch, gh}\}$ and $(c_1, c_2) \notin \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
4. $(v_1c_1c_2, \text{-}, c_3v_2)$ if $c_1c_2c_3 \in \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
5. $(v_1c_1, \text{-}, c_2c_3v_2)$ if $c_1c_2c_3 \notin \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
6. $(v_1c_1c_2, \text{-}, c_3c_4v_2)$ if $c_2c_3c_4 \in \{\text{gst, nbl}\}$
7. $(v_1c_1, \text{-}, c_2c_3c_4v_2)$ if $c_2c_3c_4 \notin \{\text{gst, nbl}\}$
8. $(v_1c_1c_2, \text{-}, c_3c_4c_5v_2)$ if $c_1c_2c_3c_4c_5 \in \{\text{ptspr, stscr}\}$

Hyphenation example

- lingvistică
- lin-gvistică by rule (5)

1. $(v_1, \text{-}, cv_2)$
2. $(v_1, \text{-}, c_1c_2v_2)$ if $c_1c_2 \in \{\text{ch, gh}\}$ or $(c_1, c_2) \in \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
3. $(v_1c_1, \text{-}, c_2v_2)$ if $c_1c_2 \notin \{\text{ch, gh}\}$ and $(c_1, c_2) \notin \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
4. $(v_1c_1c_2, \text{-}, c_3v_2)$ if $c_1c_2c_3 \in \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
5. $(v_1c_1, \text{-}, c_2c_3v_2)$ if $c_1c_2c_3 \notin \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
6. $(v_1c_1c_2, \text{-}, c_3c_4v_2)$ if $c_2c_3c_4 \in \{\text{gst, nbl}\}$
7. $(v_1c_1, \text{-}, c_2c_3c_4v_2)$ if $c_2c_3c_4 \notin \{\text{gst, nbl}\}$
8. $(v_1c_1c_2, \text{-}, c_3c_4c_5v_2)$ if $c_1c_2c_3c_4c_5 \in \{\text{ptspr, stscr}\}$

Hyphenation example

- lingvistică
- lin-gvistică by rule (5)
- lin-gvis-tică by rule (3)
- lin-gvis-ti-că by rule (1)

1. $(v_1, \text{-}, cv_2)$
2. $(v_1, \text{-}, c_1c_2v_2)$ if $c_1c_2 \in \{\text{ch, gh}\}$ or $(c_1, c_2) \in \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
3. $(v_1c_1, \text{-}, c_2v_2)$ if $c_1c_2 \notin \{\text{ch, gh}\}$ and $(c_1, c_2) \notin \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
4. $(v_1c_1c_2, \text{-}, c_3v_2)$ if $c_1c_2c_3 \in \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
5. $(v_1c_1, \text{-}, c_2c_3v_2)$ if $c_1c_2c_3 \notin \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$
6. $(v_1c_1c_2, \text{-}, c_3c_4v_2)$ if $c_2c_3c_4 \in \{\text{gst, nbl}\}$
7. $(v_1c_1, \text{-}, c_2c_3c_4v_2)$ if $c_2c_3c_4 \notin \{\text{gst, nbl}\}$
8. $(v_1c_1c_2, \text{-}, c_3c_4c_5v_2)$ if $c_1c_2c_3c_4c_5 \in \{\text{ptspr, stscr}\}$

Parallel hyphenation

Approach [Dinu, 2009]

- Insertion systems can model syllabification
- Insertion can be done in parallel by sharing the context
- Maximally parallel application of rules yields one-step hyphenation

- Why? Suggested as possible at cognitive level by some linguists
  lingvistică $\Longrightarrow$ lingvi-sti-că $\Longrightarrow$ lin-gvis-ti-că

$(v_1, \text{-}, cv_2)$
$(v_1c_1, \text{-}, c_2v_2)$ if $c_1c_2 \notin \{\text{ch, gh}\}$ and $(c_1, c_2) \notin \{\text{b, c, d, f, g, h, p, t}\} \times \{\text{l, r}\}$
$(v_1c_1, \text{-}, c_2c_3v_2)$ if $c_1c_2c_3 \notin \{\text{lpt, mpt, mpț, ncș, nct, ncț, ndv, rct, rtf, stm}\}$

Connections with other formalisms

Contextual description of rules

- Insertion/Deletion Systems [Kari, 1991]
  - String Rewriting + Insertion Rules + Deletion Rules + Parallel application
- $\mathbb{K}$ concurrent rewriting [Roșu, Șerbănuță, 2010]
  - Term rewriting + contextual $\mathbb{K}$ rules + Parallel application

Maximally parallel application of rules

- L-Systems [Lindenmayer, 1968]
  - Context free grammars with maximally parallel derivation
  - Context Sensitive L-Systems have some notion of sharing
  - However, the context is usually altered during the rewrite
- P Systems [Păun, 1998]
  - Multiset rewriting with local rules
  - Contextual sharing in the promoter/inhibitor variant

Contributions

- Contextual string rewriting rules
  - as a (slight) variation of rules from Insertion/Deletion Systems
  - as a specialization of $\mathbb{K}$ rules to string rewriting
- Formal definition for (maximal) parallel matching and derivation
- Transformations to TRSs
  - for simulating one-step maximal parallel derivation

Contents

Formal languages

- Alphabets: Σ, ...
  - are sets of letters: a, b, c, ...
- Words: u, v, w, x, y,...
  are (finite) sequences of letters
- The set of all words over alphabet Σ is denoted Σ^*
- The empty word is denoted by λ
- Concatenation of two words is obtained by juxtaposition $\_\_$ and is associative, with unit λ
- $(\Sigma^*, \_\_, \lambda)$ is the free monoid generated by Σ
- A formal language over Σ is a subset of Σ^*.

Insertion/Deletion Systems [Kari, 1991]

Syntax

An insertion-deletion system is a tuple $\Gamma = (\Sigma, T, A, I, D)$, where

- Σ is an alphabet
- $T \subseteq \Sigma$ is the terminal alphabet
- $A \subseteq \Sigma^*$ is the set of axioms (starting symbols)
- I (the insertion rules) and D (the deletion rules) are
  - finite sets of triples of the form (u, x, v), where $u, x, v \in \Sigma^*$ with $x \neq \lambda$

(Sequential) Semantics

- Insertion rule (u, x, v) specifies insertion of x in context (u, v)

  u v $\Rightarrow$ u x v

- Deletion rule (u, x, v) specifies deletion of x in context (u, v)

  u x v $\Rightarrow$ u v

- The language generated by Γ is $L(\Gamma) = \{w \in T^* \mid \exists u \in A . u \Rightarrow^* w\}$

Insertion grammars with (maximum) parallel derivation

[Dinu, 2009]

Syntax

An insertion grammar $\Gamma = (\Sigma, A, I)$ is a special case of insertion/deletion systems, where

- There are no non-terminals ($T = \Sigma$)
- There are no deletion rules ($D = \emptyset$)

Parallel Semantics

$w \Rrightarrow_\Gamma z$ iff

- $w =$ $\ldots$, where $r \geq 1$ (r maximal for maximum derivation)
- for all $1 \leq i \leq r$, there exist $(u_i, x_i, v_i) \in I$ such that
  - ends with u_i
  - starts with v_i
- $z =$ x_1 $x_2 \ldots$ x_r

Insertion grammars derivation examples

Example

Let $\Sigma = \{a, b, c, d\}$, $A = \{abcd\}$, and $I = \{(a, a, b), (b, b, c), (c, cd, d)\}$

Language for non-maximally parallel derivation: $a^+b^+c^nd^n$

Language for maximum parallel derivation: $a^nb^nc^nd^n$

Example

Let $\Sigma = \{a, b\}$, $A = \{a, b\}$, and $I = \{(a, b, \lambda), (\lambda, ba, b)\}$.

Language for non-maximally parallel derivation:

$$\{a\} \cup \{(ab^+)^+\} \cup \{b(ab^+)^*\}$$

Language for maximum parallel derivation: "Fibonacci words":

$$\{a, b, ab, bab, abbab, bababbab, abbabbababbab, \ldots\}$$

$\mathbb{K}$ rewriting [Roșu, Șerbănuță, 2010]

Contextual rules for concurrent term rewriting with sharing

Syntax

A $\mathbb{K}$ rule is of the form $k[l_1 \to r_1, \ldots, l_n \to r_n]$ where

- k is an n-ary context (shared by concurrent rule applications)
- $l_1, \ldots, l_n$ are the parts to be rewritten
- $r_1, \ldots, r_n$ are their corresponding replacements
- 2D notation: $k[\underbrace{l_1}_{r_1}, \ldots, \underbrace{l_n}_{r_n}]$

Semantics

Non-Concurrent through flattening to term rewrite rules

$$k[l\_1, \ \ldots \ , l\_n] \Rightarrow k[r\_1, \ \ldots \ , r\_n]$$

Concurrent through graph rewriting [Șerbănuță, Roșu, 2011]

Insertion/Deletion rules as $\mathbb{K}$ rules

Insertion rule (u,x,v): $\dfrac{u \quad \bullet \quad v}{x}$

Deletion rule (u,x,v): $\dfrac{u \quad x \quad v}{\bullet}$

- $\bullet$ is the generic unit of $\mathbb{K}$ for the given (associative) operation
  It will be instantiated by λ in this particular case.

Natural Question

Why not having general rules of the form $\dfrac{u \quad x \quad v}{y}$?

Contents

Contextual String Rewriting

Definition

A Contextual string rewrite rule is of the form $u\langle x \rightarrow y \rangle v$, where

- u, x, y, v are words over Σ
- $uxv, xy \in \Sigma^+$

Reading: x rewrites to y in the context (u, v)

A Contextual SRS is a set $\mathcal{R}$ of contextual string rewrite rules.

Non concurrent semantics

Through rewrite rules: u x v => u y v

Formalizing a parallel matching instance

Definition

$$\pi : \quad \langle y_1 \rangle^{\rho_1} \quad \langle y_2 \rangle^{\rho_2} \quad \dots \quad \langle y_r \rangle^{\rho_r}$$

is a parallel instance of $\mathcal{R}$ on w if for all $1 \leq i \leq r$,

- ρ_i is the rule $u_i \langle x_i \rightarrow y_i \rangle v_i \in \mathcal{R}$, and
  - ends with u_i
  - starts with v_i

The parallel rewriting step induced by π is

$$x_1 \quad x_2 \quad \dots x_r \quad \Rrightarrow_{\mathcal{R},\pi} \quad y_1 \quad y_2 \quad \dots y_r$$

Parallel matching instance

Example

Given the rules:

$$\rho_1 : \mathit{licen} \langle \, \mathrm{c} \to \mathrm{s} \, \rangle \, \mathrm{e}$$

and

$$\rho_2 : (\langle \, \textvisiblespace \to \lambda \, \rangle$$

a parallel instance for

$$\mathrm{my}\textvisiblespace\mathrm{licence}(\textvisiblespace\mathrm{driving})$$

could be

$$\pi : \mathrm{my}\textvisiblespace\mathrm{licen} \langle \, \mathrm{s} \, \rangle^{\rho_1} \, \mathrm{e}(\langle \, \lambda \, \rangle^{\rho_2} \, \mathrm{driving})$$

and we have that

$$\mathrm{my}\textvisiblespace\mathrm{licence}(\textvisiblespace\mathrm{driving}) \Rrightarrow_{\mathcal{R},\pi} \mathrm{my}\textvisiblespace\mathrm{license}(\mathrm{driving})$$

Parallel instance refinement

Let $\pi : \quad \langle y_1 \rangle^{\rho_1} \quad \langle y_2 \rangle^{\rho_2} \quad \ldots \quad \langle y_r \rangle^{\rho_r}$ be an instance on a word w and let ρ_i be $u_i \langle x_i \rightarrow y_i \rangle v_i$ for each $1 \leq i \leq r$. Then:

- Each starts with v_i and ends with u_{i+1}

Definition ($\pi \succ \pi'$)

π is refinable if any w_i, $0 \leq i \leq r$, admits some parallel instance

$$\pi_i : \quad \langle y_{i,1} \rangle^{\rho_{i,1}}$$

such that by replacing w_i by π_i in π we still get a parallel instance of w.

- i.e., still starts with v_i and still ends with u_{i+1}
- iff the "split" of was not inside either v_i or u_{i+1}

π is maximal if cannot be refined anymore.

Note: To refine π, v_i and u_{i+1} must not overlap on w_i for some $0 \leq i \leq r$

Parallel instance refinement

Example

Given the rules:

ρ_1 : *licen* $\langle$ c $\rightarrow$ s $\rangle$ e $\quad$ ρ_2 : ($\langle$ ␣ $\rightarrow$ λ $\rangle$ $\quad$ ρ_3 : $\langle$ λ $\rightarrow$ ␣ $\rangle$ (

Then, the previous parallel instance of my␣licence(␣driving):

$$\pi : \text{my␣licen} \langle \text{ s } \rangle^{\rho_1} \text{ e(} \langle \lambda \rangle^{\rho_2} \text{ driving)}$$

can be further refined using ρ_3 to

$$\pi' : \text{my␣licen} \langle \text{ s } \rangle^{\rho_1} \text{ e} \langle \text{␣} \rangle^{\rho_3} \quad \langle \lambda \rangle^{\rho_2} \text{ driving)}$$

which is maximal w.r.t the rules above, and we have that

$$\text{my␣licence(␣driving)} \Rrightarrow_{\mathcal{R},\pi'} \text{my␣license (driving)}$$

Results Summary

Rewriting using parallel instances has the following properties:

- Is sound and complete for regular string rewriting
- The refinement relation terminates
  - Each refinement matches or consumes at least one letter ($uxv \neq \lambda$)
  - Each letter can be matched by at most two rules
  - Therefore, the length of any refinement chain for $|w|$ is at most $2|w|$

Contents

Rasing the flag

To each letter a in a matching instance we can associate two flags to track when it is matched by rules

- A $\lceil$ flag raised to the left when a was matched by some u
- A $\rceil$ flag to the right when a was matched by some v
- The y parts of an instance are marked in red (and treated as a single letter)

Example

Given $\rho_1 : licen\langle c \to s \rangle e$ and $\rho_2 : (\langle \sqcup \to \lambda \rangle$, the flag encoding of

$$my\sqcup licen\langle s \rangle^{\rho_1} e(\langle \lambda \rangle^{\rho_2} driving)$$

would be

$$m\, y\, \sqcup \lceil l \lceil i \lceil c \lceil e \lceil n\, s\, e \rceil \rceil (\lceil \lambda\, d\, r\, i\, v\, i\, n\, g \rceil)$$

Encoding rules

Idea: make sure the split of a w_i does not occur inside the v_i or u_{i+1} part

- $u\langle x \to y \rangle v$ and $x \neq \lambda$ or $uv \neq \lambda$ $\quad u\ x\ v \to u\ y\ v$
- $\langle \lambda \to y \rangle v$ $\quad Z\ v \to Z\ y\ v$, and $\quad ?\ v \to ?\ y\ v$,
- $u\langle \lambda \to y \rangle$ $\quad u\ Z \to u\ y\ Z$, and $\quad u\ ? \to u\ y\ ?$,[2]

Example

We can use the encoding $?\ (\to ?\ \sqcup\ ($ of rule $\langle \lambda \to \sqcup \rangle ($ to refine

$$m\ y\ \sqcup\ l\ i\ c\ e\ n\ s\ e\ (\lambda\ d\ r\ i\ v\ i\ n\ g)$$

to

$$m\ y\ \sqcup\ l\ i\ c\ e\ n\ s\ e\ \sqcup\ (\lambda\ d\ r\ i\ v\ i\ n\ g)$$

[2] indicates the flags are not modified by the rule (i.e., stands for flag variables); ? is a variable of sort letter; Z is a variable of sort word

Results Summary

The proposed transformations have the following properties:

- Properly encoding the refinement of parallel instances
- Erasing the extra flags and marks one obtains the rewritten word (through the current instance)
- Solve the problem of computing a maximal instance
  - Every normal form of the transformed systems encodes a maximal parallel instance
  - The transformed systems terminate for any input
  - Any maximal parallel instance of w is computed in at most $2|w|$ steps

Bibliography

Dinu, 2003 An approach to syllables via some extensions of Marcus contextual grammars LP Dinu - Grammars

Dinu, 2009 On Insertion Grammars with Maximum Parallel Derivation - Fundamenta Informaticae 93 (4), 357-369

Kari, 1991 On Insertion and Deletion in Formal Languages. Ph.D. Thesis, University of Turku, 1991

Lindenmayer, 1968 Mathematical models for cellular interaction in development. J. Theoret. Biology, 18:280—315

Păun, 1998 Computing with Membranes. TUCS Report 208. Turku Centre for Computer Science

Roșu, Șerbănuță, 2010 An overview of the K semantic framework. Journal of Logic and Algebraic Programming

Șerbănuță, Roșu, 2011 A Truly Concurrent Semantics for the $\mathbb{K}$ Framework Based on Graph Transformations. ICGT'11