Combinatorics on Words formalized
Lyndon Words

Štěpán Holub
Štěpán Starosta

March 17, 2025

Funded by the Czech Science Foundation grant GAČR 20-20621S.

Contents

theory *Lyndon*
imports *Combinatorics-Words.CoWBasic*
begin

Chapter 1

Lyndon words

A Lyndon word is a non-empty word that is lexicographically strictly smaller than any other word in its conjugacy class, i.e., than any its rotations. They are named after R. Lyndon who introduced them in [4] as "standard" sequences.

We present elementary results on Lyndon words, mostly covered by results in [3, Chapter 5] and [1, 2].

This definition assumes a linear order on letters given by the context.

1.1 Definition and elementary properties

1.1.1 Underlying order

lemma (**in** *linorder*) *lexordp-mid-pref*: *ord-class.lexordp* $u\ v \Longrightarrow$ *ord-class.lexordp* $v\ (u \cdot s) \Longrightarrow$
  $u \leq p\ v$
  ⟨*proof*⟩

lemma (**in** *linorder*) *lexordp-ext*: *ord-class.lexordp* $u\ v \Longrightarrow\ \neg\ u \leq p\ v \Longrightarrow$
  ord-class.lexordp $(u \cdot w)\ (v \cdot z)$
  ⟨*proof*⟩

context *linorder*
begin

abbreviation *Lyndon-less* :: *'a list* ⇒ *'a list* ⇒ *bool* (**infixl** ‹<*lex*› *50*)
  where *Lyndon-less xs ys* ≡ *ord-class.lexordp xs ys*

abbreviation *Lyndon-le* :: *'a list* ⇒ *'a list* ⇒ *bool* (**infixl** ‹≤*lex*› *50*)
  where *Lyndon-le xs ys* ≡ *ord-class.lexordp-eq xs ys*

interpretation *rlex*: *linorder* (≤*lex*) (<*lex*)
  ⟨*proof*⟩

interpretation *dual-rlex*: *linorder* λ *x y. y* $\leq$*lex x* λ *x y. y* <*lex x*
⟨*proof*⟩

lemma *sorted-dual-rev-iff*: *dual-rlex.sorted ws* $\longleftrightarrow$ *rlex.sorted* (*rev ws*)
⟨*proof*⟩

Several useful lemmas that are formulated for relations, interpreted for the default linear order.

lemmas *lexord-suf-linorder* = *lexord-sufE*[*of* - - - - {(*x*, *y*). *x* < *y*}, *folded lexordp-conv-lexord*]
and *lexord-append-leftI-linorder* = *lexord-append-leftI*[*of* - - {(*x*, *y*). *x* < *y*} -, *folded lexordp-conv-lexord*]
and *lexord-app-right-linorder* = *lexord-sufI*[*of* - - {(*x*, *y*). *x* < *y*} -, *folded lexordp-conv-lexord*]
and *lexord-take-index-conv-linorder* = *lexord-take-index-conv*[*of* - - {(*x*, *y*). *x* < *y*}, *folded lexordp-conv-lexord*]
and *mismatch-lexord-linorder* = *mismatch-lexord*[*of* - - {(*x*, *y*). *x* < *y*}, *folded lexordp-conv-lexord*]
and *lexord-cancel-right-linorder* = *lexord-cancel-right*[*of* - - - - {(*a*,*b*). *a* < *b*}, *folded lexordp-conv-lexord*]

1.1.2 Lyndon word definition

fun *Lyndon* :: *'a list* $\Rightarrow$ *bool*
where *Lyndon w* = (*w* $\neq \varepsilon \wedge$ ($\forall$ *n*. *0* < *n* $\wedge$ *n* < |*w*| $\longrightarrow$ *w* <*lex rotate n w*))

lemma *LyndonD*: *Lyndon w* $\Longrightarrow$ *0* < *n* $\Longrightarrow$ *n* < |*w*| $\Longrightarrow$ *w* <*lex rotate n w*
⟨*proof*⟩

lemma *LyndonD-nemp*: *Lyndon w* $\Longrightarrow$ *w* $\neq \varepsilon$
⟨*proof*⟩

lemma *LyndonI*: *w* $\neq \varepsilon \Longrightarrow \forall$ *n*. *0* < *n* $\wedge$ *n* < |*w*| $\longrightarrow$ *w* <*lex rotate n w* $\Longrightarrow$ *Lyndon w*
⟨*proof*⟩

lemma *Lyndon-sing*: *Lyndon* [*a*]
⟨*proof*⟩

lemma *Lyndon-prim*: **assumes** *Lyndon w*
shows *primitive w*
⟨*proof*⟩

lemma *Lyndon-conj-greater*: *Lyndon* (*u*·*v*) $\Longrightarrow$ *u* $\neq \varepsilon \Longrightarrow$ *v* $\neq \varepsilon \Longrightarrow$ *u*·*v* <*lex v*·*u*
⟨*proof*⟩

1.1.3 Code equations for Lyndon words

primrec *Lyndon-rec* :: *'a list* $\Rightarrow$ *nat* $\Rightarrow$ *bool* **where**

Lyndon-rec w 0 = True |
Lyndon-rec w (Suc n) = (if w <lex rotate (Suc n) w then Lyndon-rec w n else False)

lemma *Lyndon-rec-all*: **assumes** *Lyndon-rec (a # w) (|w|)*
shows $n < |a\#w| \Longrightarrow 0 < n \Longrightarrow$ *Lyndon-rec (a#w) n*
⟨*proof*⟩

lemma *Lyndon-Lyndon-rec*: **assumes** *Lyndon w*
shows $0 < n \Longrightarrow n < |w| \Longrightarrow$ *Lyndon-rec w n*
⟨*proof*⟩

lemma *Lyndon-code* [*code*]:
Lyndon Nil = False
Lyndon (a # w) = Lyndon-rec (a # w) (|w|)
⟨*proof*⟩

1.1.4 Properties of Lyndon words

Lyndon words are unbordered

theorem *Lyndon-unbordered*: **assumes** *Lyndon w* **shows** ¬ *bordered w*
⟨*proof*⟩

Each conjugacy class contains a Lyndon word

lemma *conjug-Lyndon-ex*: **assumes** *primitive w*
obtains *n* **where** *Lyndon (rotate n w)*
⟨*proof*⟩

lemma *conjug-Lyndon-ex'*: **assumes** *primitive w*
obtains *v* **where** $w \sim v$ **and** *Lyndon v*
⟨*proof*⟩

1.2 Characterization by suffixes

lemma *Lyndon-suf-less*: **assumes** *Lyndon w s ≤ns w s ≠ w*
shows *w <lex s*
⟨*proof*⟩

lemma *Lyndon-pref-suf-less*: **assumes** *Lyndon w p ≤p w s ≤ns w s ≠ w*
shows *p <lex s*
⟨*proof*⟩

lemma *suf-less-Lyndon*: **assumes** $w \neq \varepsilon$ **and** ∀ *s*. (*s ≤ns w ⟶ s ≠ w ⟶ w <lex s*)
shows *Lyndon w*
⟨*proof*⟩

corollary *Lyndon-suf-char*: $w \neq \varepsilon \Longrightarrow Lyndon\ w \longleftrightarrow (\forall s.\ s \leq ns\ w \longrightarrow s \neq w \longrightarrow w <lex\ s)$
⟨*proof*⟩

lemma *Lyndon-suf-le*: $Lyndon\ w \Longrightarrow s \leq ns\ w \Longrightarrow w \leq lex\ s$
⟨*proof*⟩

1.3 Unbordered prefix of a Lyndon word is Lyndon

lemma *unbordered-pref-Lyndon*: $Lyndon\ (u \cdot v) \Longrightarrow u \neq \varepsilon \Longrightarrow \neg\ bordered\ u \Longrightarrow Lyndon\ u$
⟨*proof*⟩

1.4 Concatenation of Lyndon words

theorem *Lyndon-concat*: **assumes** *Lyndon u* **and** *Lyndon v* **and** *u <lex v*
shows *Lyndon* $(u \cdot v)$
⟨*proof*⟩

1.5 Longest Lyndon suffix

fun *longest-Lyndon-suffix*:: *'a list* ⇒ *'a list* (‹*LynSuf*›) **where**
longest-Lyndon-suffix $\varepsilon = \varepsilon$ |
longest-Lyndon-suffix $(a\#w)$ = (*if Lyndon* $(a\#w)$ *then* $a\#w$ *else longest-Lyndon-suffix* w)

lemma *longest-Lyndon-suf-ext*: $\neg\ Lyndon\ (a\ \#\ w) \Longrightarrow LynSuf\ w = LynSuf\ (a\ \#\ w)$
⟨*proof*⟩

lemma *longest-Lyndon-suf-suf*: $w \neq \varepsilon \Longrightarrow LynSuf\ w \leq s\ w$
⟨*proof*⟩

lemma *longest-Lyndon-suf-max*:
$v \leq s\ w \Longrightarrow Lyndon\ v \Longrightarrow v \leq s\ (LynSuf\ w)$
⟨*proof*⟩

lemma *longest-Lyndon-suf-Lyndon-id*: **assumes** *Lyndon w*
shows *LynSuf* $w = w$
⟨*proof*⟩

lemma *longest-Lyndon-suf-longest*: $w \neq \varepsilon \Longrightarrow v' \leq s\ w \Longrightarrow Lyndon\ v' \Longrightarrow |v'| \leq |(LynSuf\ w)|$
⟨*proof*⟩

lemma *longest-Lyndon-suf-sing*: *LynSuf* $[a] = [a]$
⟨*proof*⟩

lemma *longest-Lyndon-suf-Lyndon*: $w \neq \varepsilon \Longrightarrow$ *Lyndon* (*LynSuf* w)
⟨*proof*⟩

lemma *longest-Lyndon-suf-nemp*: $w \neq \varepsilon \Longrightarrow$ *LynSuf* $w \neq \varepsilon$
⟨*proof*⟩

lemma *longest-Lyndon-sufI*:
assumes $q \leq s\ w$ **and** *Lyndon* q **and** *all-s*: $(\forall\ s.\ (s \leq s\ w \wedge \textit{Lyndon}\ s) \longrightarrow s \leq s\ q)$
shows *LynSuf* $w = q$
⟨*proof*⟩

corollary *longest-Lyndon-sufI'*:
assumes $q \leq s\ w$ **and** *Lyndon* q **and** *all-s*: $\forall\ s.\ (s \leq s\ w \wedge \textit{Lyndon}\ s) \longrightarrow |s| \leq |q|$
shows *LynSuf* $w = q$
⟨*proof*⟩

The next lemma is fabricated to suit the upcoming definition of longest Lyndon factorization.

lemma *longest-Lyndon-suf-shorter*: **assumes** $w \neq \varepsilon$
shows $|w^{<-1}(\textit{LynSuf}\ w)| < |w|$
⟨*proof*⟩

1.6 Lyndon factorizations

function *Lyndon-fac*::*'a list* ⇒ *'a list list* (‹*LynFac*›)
where *Lyndon-fac* w = (*if* $w \neq \varepsilon$ *then* ((*Lyndon-fac* ($w\ ^{<-1}$(*LynSuf* w))) · [*LynSuf* w]) *else* ε)
⟨*proof*⟩
termination
⟨*proof*⟩

The factorization *LynFac* w obtained by taking always the longest Lyndon suffix is well defined, and called "Lyndon factorization (of w)".

lemma *Lyndon-fac-simp*: $w \neq \varepsilon \Longrightarrow$ *Lyndon-fac* w = *Lyndon-fac* ($w^{<-1}$*LynSuf* w) · (*LynSuf* w # ε)
⟨*proof*⟩

lemma *Lyndon-fac-emp*: *Lyndon-fac* $\varepsilon = \varepsilon$
⟨*proof*⟩

Note that the Lyndon factorization of a Lyndon word is trivial.

lemma *Lyndon-fac-longest-Lyndon-id*: *Lyndon* $w \Longrightarrow$ *Lyndon-fac* $w = [w]$
⟨*proof*⟩

Lyndon factorization is composed of Lyndon words ...

lemma *Lyndon-fac-set*: $z \in$ *set* (*Lyndon-fac* w) $\Longrightarrow$ *Lyndon* z

⟨*proof*⟩

...and it indeed is a factorization of the argument.

lemma *Lyndon-fac-longest-dec*: *concat* (*Lyndon-fac w*) = *w*
⟨*proof*⟩

The following lemma makes explicit the inductive character of the definition of *LynFac*.

lemma *Lyndon-fac-longest-pref*: *us* ≤*p* *Lyndon-fac w* ⟹ *Lyndon-fac* (*concat us*) = *us*
⟨*proof*⟩

We give name to an important predicate: monotone (nonincreasing) list of Lyndon words.

definition *Lyndon-mono* :: *'a list list* ⇒ *bool* **where**
Lyndon-mono ws ⟷ (∀ *u* ∈ *set ws*. *Lyndon u*) ∧ (*rlex.sorted* (*rev ws*))

lemma *Lyndon-mono-set*: *Lyndon-mono ws* ⟹ *u* ∈ *set ws* ⟹ *Lyndon u*
⟨*proof*⟩

lemma *Lyndon-mono-sorted*: *Lyndon-mono ws* ⟹ *rlex.sorted* (*rev ws*)
⟨*proof*⟩

lemma *Lyndon-mono-nth*: *Lyndon-mono ws* ⟹ *i* ≤ *j* ⟹ *j* < |*ws*| ⟹ *ws*!*j* ≤*lex* *ws*!*i*
⟨*proof*⟩

lemma *Lyndon-mono-empty*[*simp*]: *Lyndon-mono* ε
⟨*proof*⟩

lemma *Lyndon-mono-sing*: *Lyndon u* ⟹ *Lyndon-mono* [*u*]
⟨*proof*⟩

lemma *Lyndon-mono-fac-Lyndon-mono*:
assumes *ps* ≤*f* *ws* **and** *Lyndon-mono ws* **shows** *Lyndon-mono ps*
⟨*proof*⟩

Lyndon factorization is monotone! Altogether, we have shown that the Lyndon factorization is a monotone factorization into Lyndon words.

theorem *fac-Lyndon-mono*: *Lyndon-mono* (*Lyndon-fac w*)
⟨*proof*⟩

Now we want to show the converse: any monotone factorization into Lyndon words is the Lyndon factorization

The last element in the Lyndon factorization is the smallest suffix.

lemma *Lyndon-mono-last-smallest*: *Lyndon-mono ws* ⟹*s* ≤*ns* (*concat ws*) ⟹ *last ws* ≤*lex* *s*

⟨*proof*⟩

A monotone list, if seen as a factorization, must end with the longest suffix

lemma *Lyndon-mono-last-longest*: **assumes** $ws \neq \varepsilon$ **and** *Lyndon-mono ws*
shows *LynSuf* (*concat ws*) = *last ws*
⟨*proof*⟩

Therefore, by construction, any monotone list is the Lyndon factorization of its concatenation

lemma *Lyndon-mono-fac*:
Lyndon-mono ws $\Longrightarrow$ *ws* = *Lyndon-fac* (*concat ws*)
⟨*proof*⟩

This implies that the Lyndon factorization can be characterized in two equivalent ways: as the (unique) monotone factorization (into Lyndon words) or as the "suffix greedy" factorization (into Lyndon words).

corollary *Lyndon-mono-fac-iff*: *Lyndon-mono ws* $\longleftrightarrow$ *ws* = *LynFac* (*concat ws*)
⟨*proof*⟩

corollary *Lyndon-mono-unique*: **assumes** *Lyndon-mono ws* **and** *Lyndon-mono zs* **and** *concat ws* = *concat zs*
shows *ws* = *zs*
⟨*proof*⟩

1.6.1 Standard factorization

lemma *Lyndon-std*: **assumes** *Lyndon w* $1 < |w|$
obtains *l m* **where** $w = l \cdot m$ **and** *Lyndon l* **and** *Lyndon m* **and** *l* <*lex m*
⟨*proof*⟩

corollary *Lyndon-std-iff*:
Lyndon w $\longleftrightarrow$ ($|w| = 1 \vee (\exists\ l\ m.\ w = l \cdot m \wedge$ *Lyndon l* $\wedge$ *Lyndon m* $\wedge$ *l* <*lex m*))
(**is** *?L* $\longleftrightarrow$ *?R*)
⟨*proof*⟩

end — end context linorder

end

theory *Lyndon-Addition*
imports *Lyndon Szpilrajn.Szpilrajn*

begin

1.6.2 The minimal relation

We define the minimal relation which guarantees the lexicographic minimality of w compared to its nontrivial conjugates.

inductive-set *rotate-rel* :: *'a list* ⇒ *'a rel* **for** *w*
where $0 < n \Longrightarrow n < |w| \Longrightarrow$ (*mismatch-pair w* (*rotate n w*)) ∈ *rotate-rel w*

A word is Lyndon iff the corresponding order of letters is compatible with *rotate-rel w*.

lemma (**in** *linorder*) *rotate-rel-iff*: **assumes** $w \neq \varepsilon$
shows *Lyndon w* ⟷ *rotate-rel w* ⊆ {(*x*,*y*). $x < y$} (**is** *?L* ⟷ *?R*)
⟨*proof*⟩

It is well known that an acyclic order can be extended to a total strict linear order. This means that a word is Lyndon with respect to some order iff its *rotate-rel w* is acyclic.

lemma *Lyndon-rotate-rel-iff*:
acyclic (*rotate-rel w*) ⟷ (∃ *r*. *strict-linear-order r* ∧ *rotate-rel w* ⊆ *r*) (**is** *?L* ⟷ *?R*)
⟨*proof*⟩

lemma *slo-linorder*: *strict-linear-order r* ⟹ *class.linorder* (λ *a b*. (*a*,*b*) ∈ $r^=$) (λ *a b*. (*a*,*b*) ∈ *r*)
⟨*proof*⟩

Application examples

lemma assumes $w \neq \varepsilon$ **and** *acyclic* (*rotate-rel w*) **shows** *primitive w*
⟨*proof*⟩

lemma assumes $w \neq \varepsilon$ **and** *acyclic* (*rotate-rel w*) **shows** ¬ *bordered w*
⟨*proof*⟩

end

References

[1] J.-P. Duval. Mots de Lyndon et périodicité. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 14(2):181–191, 1980.

[2] J. P. Duval. Factorizing words over an ordered alphabet. *Journal of Algorithms*, 4(4):363–381, Dec. 1983.

[3] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopaedia of Mathematics and its Applications*. Addison-Wesley, Reading, Mass., 1983. Reprinted in the *Cambridge Mathematical Library*, Cambridge University Press, Cambridge UK, 1997.

[4] R. C. Lyndon. On Burnside's problem. *Transactions of the American Mathematical Society*, 77(2):202–202, Feb. 1954.