

Combinatorics on Words formalized  
Binary codes that do not preserve primitivity

Štěpán Holub  
Martin Raška

June 7, 2026

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

# Contents

0.1	Lemmas for covered x square . . . . .	2
0.1.1	Two particular cases . . . . .	2
0.1.2	Main cases . . . . .	4
0.2	Considering the primitive root instead . . . . .	9
0.3	Square interpretation . . . . .	9
0.3.1	Locale: interpretation . . . . .	10
0.3.2	Locale with additional parameters . . . . .	17
0.3.3	Back to the main locale . . . . .	20
0.3.4	Locale: Extendable interpretation . . . . .	23
0.4	Global claims . . . . .	26
0.4.1	Examples . . . . .	28
0.5	General primitivity not preserving codes . . . . .	31
0.6	Covered uniform square . . . . .	34
0.6.1	Primitivity (non)preserving uniform binary codes . . . . .	38
0.7	The main theorem . . . . .	40
0.7.1	Imprimitive words with single y . . . . .	40
0.7.2	Conjugate words . . . . .	41
0.7.3	Square factor of the longer word and both words primitive (was all_assms) . . . . .	42
0.7.4	Obtaining primitivity with two squares (refining) . . . . .	47
0.7.5	Obtaining the square of the longer word (gluing) . . . . .	49
0.8	Examples . . . . .	52
0.9	Primitivity non-preserving binary code . . . . .	52
0.9.1	The target theorem . . . . .	54
0.10	Upper bound of the power exponent in the canonical imprimitivity witness . . . . .	56
0.10.1	Optimality of the exponent upper bound . . . . .	59
0.11	Characterization of binary primitivity preserving morphisms given by a pair of words . . . . .	60
0.11.1	Code equation for <i>bin-prim</i> predicate . . . . .	63
0.12	Characterization of binary imprimitivity codes . . . . .	64

**theory** *Binary-Square-Interpretation*

**imports**

*Combinatorics-Words.Submonoids*

*Combinatorics-Words.Equations-Basic*

**begin**

## 0.1 Lemmas for covered x square

This section explores various variants of the situation when  $x \cdot x$  is covered with  $x \cdot y^{\textcircled{a}} k \cdot u \cdot v \cdot y^{\textcircled{a}} l \cdot x$ , with  $y = u \cdot v$ , and the displayed dots being synchronized.

### 0.1.1 Two particular cases

**lemma** *pref-suf-pers-short*: **assumes**  $x \leq_p v \cdot x$  **and**  $|v \cdot u| < |x|$  **and**  $x \leq_s r \cdot u \cdot v \cdot u$  **and**  $r \in \langle \{u, v\} \rangle$

—  $x \cdot x$  is covered by  $(p \cdot u \cdot v \cdot u) \cdot v \cdot x$ , the displayed dots being synchronized

— That is, the condition on the first  $x$  in  $x \cdot y^{\textcircled{a}} k \cdot u \cdot v \cdot y^{\textcircled{a}} l \cdot x$  is relaxed

**shows**  $u \cdot v = v \cdot u$

**proof** (*rule nemp-comm*)

**have**  $v \cdot u <_s x$

**using** *suf-prod-long-less*[*OF* -  $\langle |v \cdot u| < |x| \rangle$ , *of*  $r \cdot u$ , *unfolded rassoc*, *OF*  $\langle x \leq_s r \cdot u \cdot v \cdot u \rangle$ ].

**assume**  $u \neq \varepsilon$  **and**  $v \neq \varepsilon$

**obtain**  $q$  **where**  $x = q \cdot v \cdot u$  **and**  $q \neq \varepsilon$

**using**  $\langle v \cdot u <_s x \rangle$  **by** (*auto simp add: suffix-def*)

**hence**  $q \leq_s r \cdot u$

**using**  $\langle x \leq_s r \cdot u \cdot v \cdot u \rangle$  **by** (*auto simp add: suffix-def*)

**from** *suf-trans*[*OF primroot-suf this*]

**have**  $\varrho q \leq_s r \cdot u$ .

**have**  $q \cdot v = v \cdot q$

**using** *pref-marker*[*OF*  $\langle x \leq_p v \cdot x \rangle$ , *of*  $q$ ]  $\langle x = q \cdot v \cdot u \rangle$  **by** *simp*

**from** *suf-marker-per-root*[*OF*  $\langle x \leq_p v \cdot x \rangle$ , *of*  $q u$ , *unfolded rassoc*  $\langle x = q \cdot v \cdot u \rangle$ ]

**have**  $u <_p v \cdot u$

**using**  $\langle v \neq \varepsilon \rangle$  **by** *blast*

**from** *per-root-primroot*[*OF this*]

*comm-primroots'*[*OF*  $\langle q \neq \varepsilon \rangle \langle v \neq \varepsilon \rangle \langle q \cdot v = v \cdot q \rangle$ ]

**have**  $u \leq_p \varrho q \cdot u$

**by** *force*

**from** *gen-prim*[*OF*  $\langle r \in \langle \{u, v\} \rangle \rangle$ ]

**have**  $r \in \langle \{u, \varrho q\} \rangle$

**unfolding**  $\langle \varrho q = \varrho v \rangle$ .  
**from** *two-elim-root-suf-comm*[*OF*  $\langle u \leq p \varrho q \cdot u \rangle \langle \varrho q \leq s r \cdot u \rangle$  *this*]  
**show**  $u \cdot v = v \cdot u$   
**using** *comm-primroot-conv*[*of - v, folded*  $\langle \varrho q = \varrho v \rangle$ ] **by** *blast*  
**qed**

**lemma** *pref-suf-pers-large-overlap*:  
**assumes**  
 $p \leq p x$  **and**  $s \leq s x$  **and**  $p \leq p r \cdot p$  **and**  $s \leq s s \cdot r$  **and**  $|x| + |r| \leq |p| + |s|$   
**shows**  $x \cdot r = r \cdot x$   
**using** *assms*  
**proof** (*cases*  $r = \varepsilon$ )  
**assume**  $r \neq \varepsilon$  **hence**  $r \neq \varepsilon$  **by** *blast*  
**have**  $|s| \leq |x|$   
**using**  $\langle s \leq s x \rangle$  **unfolding** *suffix-def* **by** *force*  
**have**  $|p| \leq |x|$   
**using**  $\langle p \leq p x \rangle$  **by** (*force simp add: prefix-def*)  
**have**  $|r| \leq |p|$   
**using**  $\langle |x| + |r| \leq |p| + |s| \rangle \langle |s| \leq |x| \rangle$  **unfolding** *lenmorph* **by** *linarith*  
**have**  $|r| \leq |s|$   
**using**  $\langle |x| + |r| \leq |p| + |s| \rangle \langle |p| \leq |x| \rangle$  **unfolding** *lenmorph* **by** *linarith*  
**obtain**  $p1 \text{ } ov \text{ } s1$  **where**  $p1 \cdot ov \cdot s1 = x$  **and**  $p1 \cdot ov = p$  **and**  $ov \cdot s1 = s$   
**using** *pref-suf-overlapE*[*OF*  $\langle p \leq p x \rangle \langle s \leq s x \rangle$ ] **using**  $\langle |x| + |r| \leq |p| + |s| \rangle$   
**by** *auto*  
**have**  $|r| \leq |ov|$   
**using**  $\langle |x| + |r| \leq |p| + |s| \rangle$  [*folded*  $\langle p1 \cdot ov \cdot s1 = x \rangle \langle p1 \cdot ov = p \rangle \langle ov \cdot s1 = s \rangle$ ]  
**unfolding** *lenmorph* **by** *force*  
**have**  $r \leq p p$   
**using**  $\langle |r| \leq |p| \rangle$  [*unfolded swap-len*] *pref-prod-long*[*OF*  $\langle p \leq p r \cdot p \rangle$ ] **by** *blast*  
**hence**  $r \leq p x$   
**using**  $\langle p \leq p x \rangle$  **by** *auto*  
**have**  $r \leq s s$   
**using**  $\langle |r| \leq |s| \rangle$  [*unfolded swap-len*] *pref-prod-long*[*reversed, OF*  $\langle s \leq s s \cdot r \rangle$ ]  
**by** *blast*  
**hence**  $r \leq s x$   
**using**  $\langle s \leq s x \rangle$  **by** *auto*  
**obtain**  $k$  **where**  $p \leq p r^{\textcircled{a}} k$   $0 < k$   
**using** *per-root-powE*[*OF*  $\langle p \leq p r \cdot p \rangle \langle r \neq \varepsilon \rangle$ ] *sprefD1* **by** *metis*  
**hence**  $p1 \cdot ov \leq f r^{\textcircled{a}} k$   
**unfolding**  $\langle p1 \cdot ov = p \rangle$  **by** *blast*  
**obtain**  $l$  **where**  $s \leq s r^{\textcircled{a}} l$   $0 < l$   
**using** *per-root-powE*[*reversed, OF*  $\langle s \leq s s \cdot r \rangle \langle r \neq \varepsilon \rangle$ ] *ssufD1* **by** *metis*  
**hence**  $ov \cdot s1 \leq f r^{\textcircled{a}} l$   
**unfolding**  $\langle ov \cdot s1 = s \rangle$  **by** *blast*  
**from** *per-glue-facs*[*OF*  $\langle p1 \cdot ov \leq f r^{\textcircled{a}} k \rangle \langle ov \cdot s1 \leq f r^{\textcircled{a}} l \rangle \langle |r| \leq |ov| \rangle$ , *unfolded*  
 $\langle p1 \cdot ov \cdot s1 = x \rangle$ ]  
**obtain**  $m$  **where**  $x \leq f r^{\textcircled{a}} m$ .  
**show**  $x \cdot r = r \cdot x$

**using** *root-suf-comm*[*OF*  
*pref-pow-root*[*OF marker-fac-pref*[*OF*  $\langle x \leq_f r^{\textcircled{m}} \rangle \langle r \leq_p x \rangle$ ]]  
*suffix-appendI*[*OF*  $\langle r \leq_s x \rangle$ ]]..  
**qed** *simp*

### 0.1.2 Main cases

**locale** *pref-suf-pers* =  
**fixes**  $x\ u\ v\ k\ m$   
**assumes**  
*x-pref*:  $x \leq_p (v \cdot (u \cdot v)^{\textcircled{k}}) \cdot x \text{ --- } x \leq_p p \cdot x \text{ and } p \leq_p q \cdot p \text{ where } q = v \cdot u$   
**and**  
*x-suf*:  $x \leq_s x \cdot (u \cdot v)^{\textcircled{m}} \cdot u \text{ --- } \leq_s x (s \cdot x) \text{ and } \leq_s s (q' \cdot s) \text{ where } q' = u \cdot v$   
**and** *k-pos*:  $0 < k$  **and** *m-pos*:  $0 < m$   
**begin**

**lemma** *pref-suf-commute-all-commutes*:

**assumes**  $|u \cdot v| \leq |x|$  **and**  $u \cdot v = v \cdot u$   
**shows** *commutes*  $\{u, v, x\}$   
**using** *assms*

**proof** (*cases*  $u \cdot v = \varepsilon$ )  
**let**  $?p = (v \cdot (u \cdot v)^{\textcircled{k}})$   
**let**  $?s = (u \cdot v)^{\textcircled{m}} \cdot u$   
**note** *x-pref* *x-suf*

**assume**  $u \cdot v \neq \varepsilon$

**have**  $?p \neq \varepsilon$  **and**  $?s \neq \varepsilon$  **and**  $v \cdot u \neq \varepsilon$

**using**  $\langle u \cdot v \neq \varepsilon \rangle$  *pow-list-Nil-iff-Nil*[*of* -  $u \cdot v$ ] *k-pos* *m-pos* **by** *blast+*

**obtain**  $r$  **where**  $u \in \langle \{r\} \rangle$  **and**  $v \in \langle \{r\} \rangle$  **and** *primitive*  $r$

**using** *comm-primrootE'*[*OF*  $\langle u \cdot v = v \cdot u \rangle$ ] *pow-sing-gen* **by** *metis*

**hence**  $r \neq \varepsilon$

**by** *force*

**have**  $?p \in \langle \{r\} \rangle$  **and**  $?s \in \langle \{r\} \rangle$  **and**  $v \cdot u \in \langle \{r\} \rangle$  **and**  $v \cdot v \in \langle \{r\} \rangle$

**using**  $\langle u \in \langle \{r\} \rangle \rangle \langle v \in \langle \{r\} \rangle \rangle$

**by** (*simp-all add: hull-closed power-in*)

**have**  $x \leq_p r \cdot x$

**using**  $\langle ?p \in \langle \{r\} \rangle \rangle \langle x \leq_p ?p \cdot x \rangle \langle ?p \neq \varepsilon \rangle$  **by** *blast*

**have**  $v \cdot u \leq_s x$

**using** *ruler-le*[*reversed*, *OF* - -  $\langle |u \cdot v| \leq |x| \rangle$ ][*unfolded swap-len*[*of*  $u$ ]],

*of*  $(x \cdot (u \cdot v)^{\textcircled{m}} (m-1) \cdot u) \cdot v \cdot u$ , *OF* *triv-suf*, *unfolded rassoc*, *OF*  $\langle x \leq_s$   
 $x \cdot ?s \rangle$ ][*unfolded pow-pos2*[*OF* *m-pos*] *rassoc*]].

**have**  $r \leq_s v \cdot u$

**using** *per-root-suf*[*OF*  $\langle v \cdot u \neq \varepsilon \rangle \langle v \cdot u \in \langle \{r\} \rangle \rangle$ ].

**have**  $r \leq_s r \cdot x$

**using** *suf-trans*[*OF*  $\langle r \leq_s v \cdot u \rangle \langle v \cdot u \leq_s x \rangle$ , *THEN* *suffix-appendI*] **by** *blast*

**have**  $x \cdot r = r \cdot x$

**using** *root-suf-comm*[*OF*  $\langle x \leq_p r \cdot x \rangle \langle r \leq_s r \cdot x \rangle$ , *symmetric*].

**hence**  $x \in \langle \{r\} \rangle$   
**by** (*simp add: primitive r prim-comm-root*)  
**thus commutes**  $\{u, v, x\}$   
**using**  $\langle u \in \langle \{r\} \rangle \rangle \langle v \in \langle \{r\} \rangle \rangle$  *commutesI-root[of {u,v,x}]* **by blast**  
**qed simp**

**lemma no-overlap:**

**assumes**  
*len:  $|v \cdot (u \cdot v)^{\textcircled{k}}| + |(u \cdot v)^{\textcircled{m}} \cdot u| \leq |x|$  (is  $|?p| + |?s| \leq |x|$ ) and  $0 < k \ 0 < m$*   
**shows commutes**  $\{u, v, x\}$   
**using** *assms*  
**proof** (*cases  $u \cdot v = \varepsilon$* )  
**note** *x-pref x-suf*  
**assume**  $u \cdot v \neq \varepsilon$   
**have**  $?p \neq \varepsilon$  **and**  $?s \neq \varepsilon$   
**using**  $\langle u \cdot v \neq \varepsilon \rangle$  *m-pos k-pos by auto*  
**from** *per-lemma-pref-suf[OF per-rootI[OF  $\langle x \leq p \ ?p \cdot x \ \langle ?p \neq \varepsilon \rangle \rangle$  per-rootI[reversed,  $\langle x \leq s \ x \cdot ?s \ \langle ?s \neq \varepsilon \rangle \ \langle |?p| + |?s| \leq |x| \rangle \rangle$ ]*  
**obtain**  $r \ s \ kp \ ks \ mw$  **where**  $?p = (r \cdot s)^{\textcircled{kp}}$  **and**  $?s = (s \cdot r)^{\textcircled{ks}}$  **and**  $x = (r \cdot s)^{\textcircled{mw}} \cdot r$  **and primitive**  $(r \cdot s)$ .  
**hence**  $?p = r \cdot s$   
**using**  $\langle v \cdot (u \cdot v)^{\textcircled{k}} \neq \varepsilon \rangle$  *comm-primroots nemp-pow-nemp pow-comm prim-primroot by metis*  
**moreover have**  $?s = s \cdot r$   
**using** *primroot-unique[OF  $\langle ?s \neq \varepsilon \rangle - \langle ?s = (s \cdot r)^{\textcircled{ks}} \rangle$  prim-conjug[OF primitive (r \cdot s)]]* **by blast**  
**ultimately have**  $?p \sim ?s$   
**by force**  
**from** *conj-pers-conj-comm[OF this k-pos m-pos]*  
**have**  $u \cdot v = v \cdot u$ .  
  
**from** *pref-suf-commute-all-commutes[OF - this]*  
**show commutes**  $\{u, v, x\}$   
**using** *len by auto*  
**qed simp**

**lemma no-overlap':**

**assumes**  
*len:  $|v \cdot (u \cdot v)^{\textcircled{k}}| + |(u \cdot v)^{\textcircled{m}} \cdot u| \leq |x|$  (is  $|?p| + |?s| \leq |x|$ ) and  $0 < k \ 0 < m$*   
**shows**  $u \cdot v = v \cdot u$   
**by** (*rule commutesE[of {u,v,x}], simp-all add: no-overlap[OF assms]*)

**lemma short-overlap:**

**assumes**  
*len1:  $|x| < |v \cdot (u \cdot v)^{\textcircled{k}}| + |(u \cdot v)^{\textcircled{m}} \cdot u|$  (is  $|x| < |?p| + |?s|$ ) and*  
*len2:  $|v \cdot (u \cdot v)^{\textcircled{k}}| + |(u \cdot v)^{\textcircled{m}} \cdot u| \leq |x| + |u|$  (is  $|?p| + |?s| \leq |x| + |u|$ )*  
**shows commutes**  $\{u, v, x\}$

**proof** (*rule pref-suf-commute-all-commutes*)  
**show**  $|u \cdot v| \leq |x|$   
**using** *len2 unfolding pow-pos[OF k-pos] lenmorph by simp*  
**next**  
**note** *x-pref x-suf*  
— obtain the overlap  
  
**have**  $|?p| \leq |x|$   
**using** *len2 unfolding lenmorph by linarith*  
**hence**  $?p \leq_p x$   
**using**  $\langle x \leq_p ?p \cdot x \rangle$  *pref-prod-long by blast*  
  
**have**  $|?s| \leq |x|$   
**using** *len2 unfolding pow-pos[OF k-pos] pow-len lenmorph by auto*  
**hence**  $?s \leq_s x$   
**using** *suf-prod-long[OF  $\langle x \leq_s x \cdot ?s \rangle$ ] by blast*  
  
**from** *pref-suf-overlapE[OF  $\langle ?p \leq_p x \rangle \langle ?s \leq_s x \rangle$  less-imp-le[OF len1]]*  
**obtain**  $p1\ ov\ s1$  **where**  $p1 \cdot ov \cdot s1 = x$  **and**  $p1 \cdot ov = ?p$  **and**  $ov \cdot s1 = ?s$ .  
  
**from** *len1[folded this]*  
**have**  $ov \neq \varepsilon$   
**by** *fastforce*  
  
**have**  $|ov| \leq |u|$   
**using** *len2[folded  $\langle p1 \cdot ov \cdot s1 = x \rangle \langle p1 \cdot ov = ?p \rangle \langle ov \cdot s1 = ?s \rangle$ ] unfolding lenmorph by auto*  
  
**then obtain**  $s'$  **where**  $ov \cdot s' = u$  **and**  $s' \cdot v \cdot (u \cdot v)^{\otimes (m-1)} \cdot u = s1$   
**using** *eqdE[OF  $\langle ov \cdot s1 = ?s \rangle$ ] unfolded pow-pos[OF m-pos] rassoc] by auto*  
  
— obtain the left complement  
**from** *eqdE[reversed, of  $p1\ ov\ v \cdot (u \cdot v)^{\otimes (k-1)}\ u \cdot v$ , unfolded rassoc, OF  $\langle p1 \cdot ov = ?p \rangle$ ] unfolded pow-pos2[OF k-pos]]*  
**have**  $v \cdot (u \cdot v)^{\otimes (k-1)} \leq_p p1$   
**unfolding** *lenmorph prefix-def eq-commute[of p1] rassoc*  
**using** *trans-le-add1[OF  $\langle |ov| \leq |u| \rangle$ ] by metis*  
  
**then obtain**  $q$  **where**  $v \cdot (u \cdot v)^{\otimes (k-1)} \cdot q = p1$   
**by** (*force simp add: prefix-def*)  
  
— main proof using the lemma  $\llbracket ?u \cdot ?v \cdot ?v \cdot ?u \cdot ?p = ?q \cdot ?u \cdot ?v \cdot ?u; \leq_s ?p\ ?u; \leq_s ?q\ ?w; ?w \in \langle \{ ?u, ?v \} \rangle \rrbracket \implies ?v \cdot ?u = ?u \cdot ?v$   
  
**show**  $u \cdot v = v \cdot u$   
**proof** (*rule sym, rule uvu-suf-uvvu*)  
**show**  $s' \leq_s u$   
**using**  $\langle ov \cdot s' = u \rangle \langle ov \neq \varepsilon \rangle$  **by** *blast*  
**show**  $u \cdot v \cdot v \cdot u \cdot s' = q \cdot u \cdot v \cdot u$  — the main fact: the overlap situation

**proof-**  
**have**  $u \cdot v \cdot u \leq_p ?s$   
**unfolding**  $\text{pow-pos}[OF\ m\text{-pos}]$  *rassoc pref-cancel-conv shift-pow* **by** *blast*  
**hence**  $p1 \cdot u \cdot v \cdot u \leq_p x$   
**unfolding**  $\langle p1 \cdot ov \cdot s1 = x \rangle[\text{symmetric}] \langle ov \cdot s1 = ?s \rangle$  *pref-cancel-conv.*  
**hence**  $v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot q \cdot u \cdot v \cdot ov \leq_p x$   
**using**  $\langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot q = p1 \rangle \langle ov \cdot s' = u \rangle$  **by** (*force simp add:*  
*prefix-def*)

**have**  $v \cdot u \leq_p x$   
**using**  $\langle ?p \leq_p x \rangle[\text{unfolded pow-pos}[OF\ k\text{-pos}]]$  **by** (*auto simp add: prefix-def*)  
**have**  $|?p \cdot v \cdot u| \leq |x|$   
**using** *len2* **unfolding**  $\text{pow-pos}[OF\ m\text{-pos}]$  *lenmorph* **by** *force*  
**hence**  $?p \cdot v \cdot u \leq_p x$   
**using**  $\langle x \leq_p ?p \cdot x \rangle \langle v \cdot u \leq_p x \rangle$  *pref-prod-longer* **by** *blast*  
**hence**  $v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot u \cdot v \cdot v \cdot u \leq_p x$   
**unfolding**  $\text{pow-pos2}[OF\ k\text{-pos}]$  *rassoc.*

**have**  $|v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot u \cdot v \cdot v \cdot u| = |v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot q \cdot u \cdot v \cdot ov|$   
**using** *lenarg* $[OF\ \langle p1 \cdot ov = ?p \rangle[\text{folded } \langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot q = p1 \rangle, \text{unfolded } \text{pow-pos}[OF\ k\text{-pos}] \text{ rassoc cancel}]]$   
**by** *force*

**from** *ruler-eq-len* $[OF\ \langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot u \cdot v \cdot v \cdot u \leq_p x \rangle \langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot q \cdot u \cdot v \cdot ov \leq_p x \rangle]$  *this, unfolded cancel*  
**have**  $u \cdot v \cdot v \cdot u = q \cdot u \cdot v \cdot ov.$

**thus**  $u \cdot v \cdot v \cdot u \cdot s' = q \cdot u \cdot v \cdot u$   
**using**  $\langle ov \cdot s' = u \rangle$  **by** *auto*

**qed**  
**show**  $q \leq_s v \cdot u$   
**proof** (*rule ruler-le[reversed]*)  
**show**  $q \leq_s x$   
**proof** (*rule suf-trans*)  
**show**  $p1 \leq_s x$   
**using**  $\langle p1 \cdot ov \cdot s1 = x \rangle[\text{unfolded } \langle ov \cdot s1 = ?s \rangle] \langle x \leq_s x \cdot ?s \rangle$  *same-suffix-suffix*  
**by** *blast*  
**show**  $q \leq_s p1$   
**using**  $\langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot q = p1 \rangle$  **by** *auto*

**qed**  
**show**  $v \cdot u \leq_s x$   
**using**  $\langle ?s \leq_s x \rangle[\text{unfolded pow-pos2}[OF\ m\text{-pos}] \text{ rassoc}]$  *suf-extD* **by** *metis*  
**show**  $|q| \leq |v \cdot u|$   
**using** *lenarg* $[OF\ \langle u \cdot v \cdot v \cdot u \cdot s' = q \cdot u \cdot v \cdot u \rangle]$  *lenarg* $[OF\ \langle ov \cdot s' = u \rangle]$   
**by** *force*

**qed**  
**qed** *auto*  
**qed**

**lemma** *medium-overlap*:

**assumes**

*len1*:  $|x| + |u| < |v \cdot (u \cdot v)^{\textcircled{k}}| + |(u \cdot v)^{\textcircled{m}} \cdot u|$  (**is**  $|x| + |u| < |?p| + |?s|$ )

**and**

*len2*:  $|v \cdot (u \cdot v)^{\textcircled{k}}| + |(u \cdot v)^{\textcircled{m}} \cdot u| < |x| + |u \cdot v|$  (**is**  $|?p| + |?s| < |x| + |u \cdot v|$ )

**shows** *commutes*  $\{u, v, x\}$

**proof** (*rule pref-suf-commute-all-commutes*)

**show**  $|u \cdot v| \leq |x|$

**using** *len2* **unfolding** *pow-pos*[*OF k-pos*] **by force**

**next**

**note** *x-pref x-suf*

**have**  $|?p| \leq |x|$

**using** *len2* **unfolding** *pow-pos*[*OF m-pos*] **by auto**

**hence**  $?p \leq p \ x$

**using**  $\langle x \leq p \ ?p \cdot x \rangle$  *pref-prod-long* **by blast**

**hence**  $v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot u \cdot v \cdot v \leq p \ ?p \cdot x$

**using**  $\langle x \leq p \ ?p \cdot x \rangle$  **unfolding** *pow-pos2*[*OF k-pos*] *rassoc* **by** (*auto simp add: prefix-def*)

**have**  $|?s| \leq |x|$

**using** *len2* **unfolding** *pow-pos*[*OF k-pos*] *pow-len lenmorph* **by auto**

**hence**  $?s \leq s \ x$

**using** *suf-prod-long*[*OF*  $\langle x \leq s \ x \cdot ?s \rangle$ ] **by blast**

**then obtain**  $p'$  **where**  $p' \cdot u \cdot v \leq p \ x$  **and**  $p' \cdot ?s = x$

**unfolding** *pow-pos*[*OF m-pos*] **by** (*auto simp add: suffix-def*)

**have**  $|p' \cdot u \cdot v| \leq |?p \cdot v|$

**using** *len1*[*folded*  $\langle p' \cdot ?s = x \rangle$ ] **by force**

**have**  $|v \cdot (u \cdot v)^{\textcircled{k-1}}| < |p'|$

**using** *len2*[*folded*  $\langle p' \cdot ?s = x \rangle$ ] **unfolding** *pow-pos2*[*OF k-pos*] **by force**

**from** *less-imp-le*[*OF this*]

**obtain**  $p$  **where**  $v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot p = p'$

**using** *ruler-le*[*OF*  $\langle ?p \leq p \ x \rangle$   $\langle p' \cdot u \cdot v \leq p \ x \rangle$ ,

*unfolded pow-pos2*[*OF k-pos*] *lassoc*, *THEN pref-cancel-right*, *THEN pref-cancel-right*]

**unfolding** *lenmorph* **by** (*auto simp add: prefix-def*)

**have**  $|p| \leq |v|$

**using**  $\langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot p = p' \rangle$   $\langle |p' \cdot u \cdot v| \leq |?p \cdot v| \rangle$  **unfolding** *pow-pos2*[*OF k-pos*] **by force**

**show**  $u \cdot v = v \cdot u$

**proof** (*rule uv-fac-uvv*)

**show**  $p \cdot u \cdot v \leq p \ u \cdot v \cdot v$

**proof** (*rule pref-cancel*[*of*  $v \cdot (u \cdot v)^{\textcircled{k-1}}$ ], *rule ruler-le*)

**show**  $(v \cdot (u \cdot v)^{\textcircled{k-1}}) \cdot p \cdot u \cdot v \leq p \ ?p \cdot x$

**unfolding** *lassoc*  $\langle v \cdot (u \cdot v)^{\textcircled{k-1}} \cdot p = p' \rangle$ [*unfolded lassoc*]

**using**  $\langle p' \cdot u \cdot v \leq p \ x \ \langle x \leq p \ ?p \cdot x \rangle$  **unfolding** *pow-pos2[OF k-pos]* **by**  
*force*  
**show**  $(v \cdot (u \cdot v)^{\textcircled{a}} (k-1)) \cdot u \cdot v \cdot v \leq p \ (v \cdot (u \cdot v)^{\textcircled{a}} k) \cdot x$   
**unfolding** *pow-pos2[OF k-pos]* *rassoc*  
**using**  $\langle v \cdot (u \cdot v)^{\textcircled{a}} k \leq p \ x \rangle$  **by** (*auto simp add: prefix-def*)  
**show**  $|(v \cdot (u \cdot v)^{\textcircled{a}} (k-1)) \cdot p \cdot u \cdot v| \leq |(v \cdot (u \cdot v)^{\textcircled{a}} (k-1)) \cdot u \cdot v \cdot v|$   
**using**  $\langle v \cdot (u \cdot v)^{\textcircled{a}} (k-1) \cdot p = p' \rangle \ \langle |p' \cdot u \cdot v| \leq |?p \cdot v| \rangle$  **unfolding**  
*pow-pos2[OF k-pos]* **by** *force*  
**qed**

**have**  $p \leq s \ x$   
**using**  $\langle p' \cdot ?s = x \rangle$  [*folded*  $\langle v \cdot (u \cdot v)^{\textcircled{a}} (k-1) \cdot p = p' \rangle$ ]  $\langle x \leq s \ x \cdot ?s \rangle$  *suf-cancel*  
*suf-extD* **by** *metis*

**from** *ruler-le[reversed, OF this*  $\langle ?s \leq s \ x \rangle$ , *unfolded* *pow-pos2[OF m-pos]* *rassoc]*  
**show**  $p \leq s \ (u \cdot v)^{\textcircled{a}} (m-1) \cdot u \cdot v \cdot u$   
**using**  $\langle |p| \leq |v| \rangle$  **unfolding** *lenmorph* **by** *auto*

**show**  $(u \cdot v)^{\textcircled{a}} (m-1) \cdot u \cdot v \cdot u \in \langle \{u, v\} \rangle$   
**by** (*simp add: gen-in hull-closed power-in*)

**show**  $p \neq \varepsilon$   
**using**  $\langle |v \cdot (u \cdot v)^{\textcircled{a}} (k-1)| < |p'| \rangle \ \langle v \cdot (u \cdot v)^{\textcircled{a}} (k-1) \cdot p = p' \rangle$  **by** *force*  
**qed**  
**qed**

**thm**  
*no-overlap*  
*short-overlap*  
*medium-overlap*

**end**

**thm**  
*pref-suf-pers.no-overlap*  
*pref-suf-pers.short-overlap*  
*pref-suf-pers.medium-overlap*  
*pref-suf-pers.large-overlap*

## 0.2 Considering the primitive root instead

### 0.3 Square interpretation

In this section fundamental description is given of (the only) possible  $\{x, y\}$ -interpretation of the square  $x \cdot x$ , where  $|y| \leq |x|$ . The proof is divided into several locales.

**lemma** *cover-not-disjoint:*

**shows** *primitive* (a·b·a·b·a·b·a) (is *primitive* ?x) **and**  
*primitive* (a·b) (is *primitive* ?y) **and**  
(a·b·a·b·a·b·a) · (a·b) ≠ (a·b) · (a·b·a·b·a·b·a)  
(is ?x · ?y ≠ ?y · ?x) **and**  
 $\varepsilon$  (a·b·a·b·a·b·a) · (a·b·a·b·a·b·a) (b·a·b·a)  $\sim_{\mathcal{I}}$  [(a·b·a·b·a·b·a), (a·b), (a·b), (a·b·a·b·a·b·a)]  
(is  $\varepsilon$  ?x · ?x ?s  $\sim_{\mathcal{I}}$  [?x, ?y, ?y, ?x])  
**unfolding** *factor-interpretation-def*  
**by** *primitivity-inspection+* *force*

### 0.3.1 Locale: interpretation

**term** *refine*

**locale** *square-interp* =

— The basic set of assumptions  
— The goal is to arrive at  $ws = [x] \cdot [y]^{\otimes k} \cdot [x]$  including the description of the interpretation in terms of the first and the second occurrence of x in the interpreted square.

**fixes** *x y p s ws*

**assumes**

*non-comm*:  $x \cdot y \neq y \cdot x$  **and**

*y-le-x*:  $|y| \leq |x|$  **and**

*ws-lists*:  $ws \in \text{lists } \{x, y\}$  **and**

*nconjug*:  $\neg \varrho x \sim \varrho y$  **and**

*disj-root*:  $p (\text{Ref } \{\varrho x, y\}[x, x]) s \sim_{\mathcal{D}} ws$

**begin**

**interpretation** *xy-code*: *binary-code* *x y*

**using** *non-comm* **by** *unfold-locales*

**interpretation** *yx-code*: *binary-code* *y x*

**using** *non-comm*[*symmetric*] **by** *unfold-locales*

**lemma** *interp*:  $p (x \cdot x) s \sim_{\mathcal{I}} ws$

**using** *disj-interpD0*[*OF disj-root*] **unfolding** *refine-def list.simps*

*xy-code.bin-roots-decompose(1)* **by** *simp*

**lemma** *nconjug'*:  $\neg x \sim y$

**using** *conjug-primroot nconjug* **by** *auto*

**lemma** *pref-xx-root-expE*: **assumes**  $us \leq_p [x, x]$

**obtains** *e* **where**  $\text{concat } us = (\varrho x)^{\otimes e}$  **and**  $e \leq e_{\varrho} x * 2$

**proof**–

**consider**  $\text{concat } us = \varrho x^{\otimes (e_{\varrho} x * 0)} \mid \text{concat } us = \varrho x^{\otimes (e_{\varrho} x * 1)} \mid \text{concat } us = \varrho x^{\otimes (e_{\varrho} x * 2)}$

**using**  $\langle us \leq_p [x, x] \rangle$  **unfolding** *pow-mult prefix-Cons* **by** *fastforce*

**thus** *thesis*

**by cases** (*use that*[*OF - mult-le-mono2*] *one-le-numeral in blast*)+

qed

**lemma** *pref-xx-exp-le*: **assumes**  $(\varrho x)^{\textcircled{e}} \leq_p x \cdot x$   
**shows**  $e \leq_{e_\varrho} x * 2$

**proof**–

**have**  $x \cdot x = (\varrho x)^{\textcircled{e}}(e_\varrho x * 2)$

**unfolding** *pow-mult pow-list-2* **by** *simp*

**from** *pref-len*[*OF assms, unfolded this pow-len*]

**show**  $e \leq_{e_\varrho} x * 2$

**using** *nemp-len*[*OF primroot-nemp* [*OF xy-code.bin-fst-nemp*]] **by** *force*

qed

**lemma** *prim-disj-interp*: **assumes** *primitive x* **shows**  $p [x,x] s \sim_{\mathcal{D}} ws$

**using** *disj-root*[*unfolded refine-def*]

**unfolding** *prim-primroot*[*OF assms*] *list.simps xy-code.bin-roots-decompose*(5)

**by** *simp*

**lemma** *disjoint*: **assumes**  $p1 \leq_p [x,x]$   $p2 \leq_p ws$  **shows**  $p \cdot \text{concat } p1 \neq \text{concat } p2$

**proof** (*rule pref-xx-root-expE* [*OF*  $\langle p1 \leq_p [x,x] \rangle$ , *of* *?thesis*])

**fix**  $e$

**assume**  $\text{concat } p1 = \varrho x^{\textcircled{e}} e$   $e \leq_{e_\varrho} x * 2$

**show**  $p \cdot \text{concat } p1 \neq \text{concat } p2$

**using** *disj-interpD1*[*OF disj-root -*  $\langle p2 \leq_p ws \rangle$ , *of*  $[\varrho x]^{\textcircled{e}} e$ ]

**unfolding**  $\langle \text{concat } p1 = \varrho x^{\textcircled{e}} e \rangle$  *xy-code.ref-fst-sq concat-pow-list concat-sing'*

**using** *le-exps-pref*[*OF*  $\langle e \leq_{e_\varrho} x * 2 \rangle$ ].

qed

**lemmas** *interpret-concat = fac-interpD*(3)[*OF interp*]

**lemma** *p-nemp*:  $p \neq \varepsilon$

**using** *disj-root disj-interp-nemp*(1) **by** *metis*

**lemma** *s-nemp*:  $s \neq \varepsilon$

**using** *disj-root disj-interp-nemp*(2) **by** *metis*

**lemma** *ws-nemp*:  $ws \neq \varepsilon$

**using** *xy-code.bin-fst-nemp fac-interp-nemp*[*OF - interp*] **by** *blast*

**lemma** *hd-ws-lists*:  $\text{hd } ws \in \{x, y\}$

**using** *lists-hd-in-set ws-lists ws-nemp* **by** *auto*

**lemma** *last-ws-lists*:  $\text{last } ws \in \{x, y\}$

**using** *lists-hd-in-set*[*reversed, OF ws-nemp ws-lists*].

**lemma** *kE*: **obtains**  $k$  **where**  $[\text{hd } ws] \cdot [y]^{\textcircled{k}} \cdot [\text{last } ws] = ws$

**proof**–

**from** *list.collapse*[*OF ws-nemp*] *hd-word*

**obtain**  $ws'$  **where**  $ws = [\text{hd } ws] \cdot ws'$

by *metis*  
 hence  $|hd\ ws| \leq |x|$   
 using *two-elem-cases*[*OF lists-hd-in-set*[*OF ws-nemp ws-lists*]] *y-le-x* by *blast*  
 hence  $|x| \leq |concat\ ws'|$   
 using *lenarg*[*OF interpret-concat, unfolded lenmorph*]  
 unfolding *concat.simps emp-simps arg-cong*[*OF*  $\langle ws = [hd\ ws] \cdot ws' \rangle$ , of  $\lambda x.$   
 $|concat\ x|$ , *unfolded concat-morph lenmorph*]  
 by *linarith*  
 hence  $ws' \neq \varepsilon$   
 using *nemp-len*[*OF xy-code.bin-fst-nemp*] by *fastforce*  
 then obtain *mid-ws* where  $ws' = mid-ws \cdot [last\ ws]$   
 using  $\langle ws = [hd\ ws] \cdot ws' \rangle$  *append-butlast-last-id last-appendR* by *metis*  
 note  $\langle ws = [hd\ ws] \cdot ws' \rangle$ [*unfolded this*]  
*fac-interpD*[*OF interp*]  
 obtain  $p'$  where [*symmetric*]:  $p \cdot p' = hd\ ws$  and  $p' \neq \varepsilon$   
 using *spref-exE*[*OF*  $\langle p <_p\ hd\ ws \rangle$ ].  
 obtain  $s'$  where [*symmetric*]:  $s' \cdot s = last\ ws$  and  $s' \neq \varepsilon$   
 using *spref-exE*[*reversed, OF*  $\langle s <_s\ last\ ws \rangle$ ].  
 have  $p' \cdot concat\ mid-ws \cdot s' = x \cdot x$   
 using  $\langle ws = [hd\ ws] \cdot mid-ws \cdot [last\ ws] \rangle$ [*unfolded*  $\langle hd\ ws = p \cdot p' \rangle$   $\langle last\ ws =$   
 $s' \cdot s \rangle$ ]  
 $\langle p \cdot (x \cdot x) \cdot s = concat\ ws \rangle$  by *simp*  
 note *over = prim-overlap-sqE*[*folded this*]  
 have  $mid-ws \in lists\ \{x, y\}$   
 using  $\langle ws = [hd\ ws] \cdot ws' \rangle$   $\langle ws' = mid-ws \cdot [last\ ws] \rangle$  *append-in-lists-conv ws-lists*  
 by *metis*  
 have  $x \notin set\ mid-ws$   
 proof  
 assume  $x \in set\ mid-ws$   
 then obtain  $rs\ qs$  where  $mid-ws = rs \cdot [x] \cdot qs$   
 unfolding *in-set-conv-decomp-first*[*of x mid-ws*] by *auto*  
 have  $[hd\ ws] \cdot rs \leq_p\ ws$   
 using  $\langle ws = [hd\ ws] \cdot ws' \rangle$ [*unfolded*  $\langle ws' = mid-ws \cdot [last\ ws] \rangle$   $\langle mid-ws = rs \cdot$   
 $[x] \cdot qs \rangle$ ] by *force*  
 have  $p' \cdot concat\ rs \cdot x \leq_p\ x \cdot x$   
 unfolding  $\langle p' \cdot concat\ mid-ws \cdot s' = x \cdot x \rangle$ [*symmetric*]  
 $\langle mid-ws = rs \cdot [x] \cdot qs \rangle$  by *simp*  
 from *comm-primroot-exp*[*OF xy-code.bin-fst-nemp pref-marker-sq*[*OF this*[*unfolded*  
*lassoc*]]]  
 obtain  $e$  where  $e: concat\ ([\varrho\ x]^{\otimes} e) = p' \cdot concat\ rs$   
 unfolding *concat-sing' concat-pow-list*.  
 hence  $concat\ ([hd\ ws] \cdot rs) = p \cdot ([\varrho\ x]^{\otimes} e)$   
 unfolding  $\langle hd\ ws = p \cdot p' \rangle$  by *fastforce*  
 from  $\langle p' \cdot concat\ rs \cdot x \leq_p\ x \cdot x \rangle$ [*folded rassoc e, unfolded concat-sing' con-*  
*cat-pow-list*]  
 have  $e \leq_{e_\varrho}\ x * 2$   
 using *pref-xx-exp-le append-prefixD* by *blast*  
 from *le-exps-pref*[*OF this*] *disj-interpD1*[*OF disj-root -*  $\langle [hd\ ws] \cdot rs \leq_p\ ws \rangle$ ,  
*unfolded xy-code.ref-fst-sq, of*  $[\varrho\ x]^{\otimes} e$ ]

**show** *False*  
**unfolding** *concat-sing' concat-pow-list <concat ([hd ws] · rs) = p · ρ x<sup>@</sup> e>*  
**by** *blast*  
**qed**  
**hence** *mid-ws ∈ lists {y}*  
**using** *<mid-ws ∈ lists {x,y}>* **by** *force*  
**from** *that sing-lists-exp[OF this]*  
**show** *thesis*  
**using** *<ws = [hd ws] · mid-ws · [last ws]>* **by** *metis*  
**qed**

**lemma** *l-mE*: **obtains** *m u v l* **where** *(hd ws) · y<sup>@m</sup> · u = p · x* **and** *v · y<sup>@l</sup> · (last ws) = x · s* **and**  
*u · v = y u ≠ ε v ≠ ε* **and** *x · (v · u) ≠ (v · u) · x*  
**proof**–  
**note** *fac-interpD[OF interp]*  
**obtain** *k* **where** *[hd ws] · [y]<sup>@k</sup> · [last ws] = ws*  
**using** *kE*.  
**from** *arg-cong[OF this, of concat, folded interpret-concat, unfolded concat-morph rassoc concat-sing' concat-pow-list-single]*  
**have** *hd ws · y<sup>@k</sup> · last ws = p · x · x · s*.  
**have** *|hd ws| ≤ |p · x|*  
**unfolding** *lenmorph* **by** *(rule two-elem-cases[OF hd-ws-lists])*  
*(use dual-order.trans[OF le-add2 y-le-x] le-add2[of |x|] in fast)+*  
**from** *eqd[OF - this]*  
**obtain** *ya* **where** *hd ws · ya = p · x*  
**using** *<hd ws · y<sup>@k</sup> · last ws = p · x · x · s>* **by** *auto*  
**have** *|last ws| ≤ |x|*  
**unfolding** *lenmorph* **using** *dual-order.trans last-ws-lists y-le-x* **by** *auto*  
**hence** *|last ws| < |x · s|*  
**unfolding** *lenmorph* **using** *nemp-len[OF s-nemp]* **by** *linarith*  
**from** *eqd[reversed, OF - less-imp-le[OF this]]*  
**obtain** *yb* **where** *yb · (last ws) = x · s*  
**using** *<(hd ws) · y<sup>@k</sup> · (last ws) = p · x · x · s>* *rassoc* **by** *metis*  
**hence** *yb ≠ ε*  
**using** *s-nemp <|last ws| < |x · s|>* **by** *force*  
**have** *ya · yb = y<sup>@k</sup>*  
**using** *<(hd ws) · y<sup>@k</sup> · (last ws) = p · x · x · s>* *[folded <yb · (last ws) = x · s>, unfolded lassoc cancel-right, folded <(hd ws) · ya = p · x>, unfolded rassoc cancel, symmetric]*.  
**from** *pref-mod-pow'[OF sprefI[OF prefI[OF this]], folded this]*  
**obtain** *m u* **where** *m < k* **and** *u <p y* **and** *y<sup>@m</sup> · u = ya*  
**using** *<yb ≠ ε>* **by** *blast*  
**have** *y<sup>@m</sup> · u · (u<sup>-1</sup> > y) · y<sup>@(k-m-1)</sup> = y<sup>@m</sup> · y · y<sup>@(k-m-1)</sup>*  
**using** *<u <p y>* **by** *(auto simp add: prefix-def)*  
**also have** *... = y<sup>@(m+1+(k-m-1))</sup>*  
**using** *rassoc pow-add pow-list-1* **by** *comparison*  
**also have** *... = y<sup>@k</sup>*  
**using** *<m < k>* **by** *auto*

**finally obtain**  $l v$  **where**  $u \cdot v = y$  **and**  $y^{\textcircled{m}} \cdot u \cdot v \cdot y^{\textcircled{l}} = y^{\textcircled{k}}$   
**using**  $\langle u \leq_p y \rangle$  *lq-pref* **by** *blast*  
**have**  $\text{concat}([hd\ ws] \cdot [y]^{\textcircled{m}}) = hd\ ws \cdot y^{\textcircled{m}}$   
**by** *simp*  
**have**  $v \neq \varepsilon$   
**using**  $\langle u \leq_p y \rangle$   $\langle u \cdot v = y \rangle$  **by** *force*  
**have**  $[hd\ ws] \cdot [y]^{\textcircled{m}} \leq_p ws$   
**using**  $\langle [hd\ ws] \cdot [y]^{\textcircled{k}} \cdot [last\ ws] = ws \rangle$  *folded pop-pow[OF less-imp-le[OF  $\langle m < k \rangle$ ]]* **by** *force*  
**from** *disjoint[OF - this, of [x], unfolded  $\langle \text{concat}([hd\ ws] \cdot [y]^{\textcircled{m}}) = hd\ ws \cdot y^{\textcircled{m}} \rangle$ ]*  
**have**  $u \neq \varepsilon$   
**using**  $\langle (hd\ ws) \cdot ya = p \cdot x \rangle$  *folded  $\langle y^{\textcircled{m}} \cdot u = ya \rangle$*  *s-nemp* **by** *force*  
**have**  $x \cdot (v \cdot u) \neq (v \cdot u) \cdot x$   
**proof**  
**assume**  $x \cdot v \cdot u = (v \cdot u) \cdot x$   
**from** *comm-primroots'[OF xy-code.bin-fst-nemp suf-nemp[OF  $\langle u \neq \varepsilon \rangle$ ] this]*  
**have**  $\varrho\ x = \varrho\ (v \cdot u)$ .  
**thus** *False*  
**using**  $\langle u \cdot v = y \rangle$  *nconjug conjug-primroot[OF conjugI', of v u, unfolded  $\langle u \cdot v = y \rangle$ ]* **by** *simp*  
**qed**  
**with** *that[of m u u<sup>-1</sup> > y l, OF  $\langle hd\ ws \cdot ya = p \cdot x \rangle$  folded  $\langle y^{\textcircled{m}} \cdot u = ya \rangle$ , folded  $\langle yb \cdot last\ ws = x \cdot s \rangle$   $\langle u \cdot v = y \rangle$ , unfolded *lq-triv lassoc cancel-right, OF - -  $\langle u \neq \varepsilon \rangle \langle v \neq \varepsilon \rangle$  this[unfolded lassoc]]*  
**show** *thesis*  
**using**  $\langle y^{\textcircled{m}} \cdot u \cdot v \cdot y^{\textcircled{l}} = y^{\textcircled{k}} \rangle$  *folded  $\langle ya \cdot yb = y^{\textcircled{k}} \rangle$   $\langle y^{\textcircled{m}} \cdot u = ya \rangle$ , unfolded *rassoc cancel, folded  $\langle u \cdot v = y \rangle$ ]* **by** *blast*  
**qed****

**lemma** *last-ws: last ws = x*  
**proof**(*rule ccontr*)  
**assume**  $last\ ws \neq x$   
**hence**  $last\ ws = y$   
**using** *last-ws-lists* **by** *blast*  
**obtain**  $l\ m\ u\ v$  **where**  $(hd\ ws) \cdot y^{\textcircled{m}} \cdot u = p \cdot x$  **and**  $v \cdot y^{\textcircled{l}} \cdot (last\ ws) = x \cdot s$  **and**  
 $u \cdot v = y$  **and**  $u \neq \varepsilon$  **and**  $v \neq \varepsilon$  **and**  $x \cdot v \cdot u \neq (v \cdot u) \cdot x$   
**using** *l-mE* **by** *metis*  
**note**  $y\text{-le-}x$  *folded  $\langle u \cdot v = y \rangle$ , unfolded swap-len[of u]]*  
**from**  $\langle v \cdot y^{\textcircled{l}} \cdot (last\ ws) = x \cdot s \rangle$  *unfolded  $\langle last\ ws = y \rangle$ , folded  $\langle u \cdot v = y \rangle$ ]*  
**have**  $x \leq_p (v \cdot u)^{\textcircled{Suc\ l}} \cdot v$   
**unfolding** *pow-Suc2 rassoc* **using** *append-eq-appendI prefix-def shift-pow* **by** *metis*  
**moreover** **have**  $(v \cdot u)^{\textcircled{Suc\ l}} \cdot v \leq_p (v \cdot u) \cdot (v \cdot u)^{\textcircled{Suc\ l}} \cdot v$   
**unfolding** *lassoc pow-comm[symmetric]* **using** *rassoc* **by** *blast*  
**ultimately** **have**  $x \leq_p (v \cdot u) \cdot x$   
**using** *pref-keeps-per-root* **by** *blast*

```

thus False
proof (cases  $m = 0$ )
  assume  $m \neq 0$ 
  have  $v \cdot u \leq_s x$ 
  using  $\langle (hd\ ws) \cdot y^{\textcircled{m}} \cdot u = p \cdot x \rangle$  [folded  $\langle u \cdot v = y \rangle$ , unfolded pow-pos2 [OF  $\langle m \neq 0 \rangle$ ] [unfolded neg0-conv]] rassoc
    suf-extD [THEN suf-prod-long [OF -  $\langle |v \cdot u| \leq |x| \rangle$ ], of  $p\ hd\ ws \cdot (u \cdot v)^{\textcircled{m}}$ 
  ( $m-1$ )  $\cdot u$ , unfolded rassoc] by simp
  have [symmetric]:  $(v \cdot u) \cdot x = x \cdot (v \cdot u)$ 
  using root-suf-comm' [OF -  $\langle x \leq_p (v \cdot u) \cdot x \rangle$   $\langle (v \cdot u) \leq_s x \rangle$ ].
  thus False
  using  $\langle x \cdot v \cdot u \neq (v \cdot u) \cdot x \rangle$  by blast
next
assume  $m = 0$ 
thus False
proof (cases  $hd\ ws = y$ )
  assume  $hd\ ws = y$ 
  have  $p \cdot (x \cdot x) \cdot s = y^{\textcircled{m}}\ Suc\ (Suc\ (Suc\ (m+l)))$ 
  unfolding rassoc  $\langle v \cdot y^{\textcircled{l}} \cdot (last\ ws) = x \cdot s \rangle$  [unfolded  $\langle last\ ws = y \rangle$ ,
symmetric] power-Suc2
  unfolding lassoc  $\langle (hd\ ws) \cdot y^{\textcircled{m}} \cdot u = p \cdot x \rangle$  [unfolded  $\langle hd\ ws = y \rangle$ , symmetric]
   $\langle u \cdot v = y \rangle$  [symmetric]
  by comparison
  have  $\varrho\ x \sim \varrho\ y$ 
  proof (rule fac-two-conjug-primroot)
  show  $x \cdot x \leq_f y^{\textcircled{m}}\ Suc\ (Suc\ (Suc\ (m+l)))$ 
  using facI [of  $x \cdot x\ p\ s$ , unfolded  $\langle p \cdot (x \cdot x) \cdot s = y^{\textcircled{m}}\ Suc\ (Suc\ (Suc\ (m+l))) \rangle$ ].
  show  $x \cdot x \leq_f x^{\textcircled{2}}$ 
  unfolding pow-list-2 by blast
  show  $|x| + |y| \leq |x \cdot x|$ 
  using y-le-x unfolding lenmorph by auto
qed
thus False
  using nconjug by blast
next
assume  $hd\ ws \neq y$ 
hence  $hd\ ws = x$ 
  using hd-ws-lists by auto

  have  $x \leq_s x \cdot u$ 
  using  $\langle (hd\ ws) \cdot y^{\textcircled{m}} \cdot u = p \cdot x \rangle$  [unfolded  $\langle m = 0 \rangle$   $\langle hd\ ws = x \rangle$  pow-zero
emp-simps]
  by (simp add: suffix-def)
  have  $v \cdot u \leq_p x$ 
  using  $\langle x \leq_p (v \cdot u) \cdot x \rangle$  y-le-x [folded  $\langle u \cdot v = y \rangle$ , unfolded swap-len [of  $u$ ]]
  pref-prod-long by blast
  hence  $|v \cdot u| < |x|$ 
  using nconjug conjugI [OF -  $\langle u \cdot v = y \rangle$ , of  $x$ , THEN conjug-primroot]
  spref-len sprefI by meson

```

```

have  $u \cdot v = v \cdot u$ 
proof (rule pref-suf-pers-short[reversed])
  from  $\langle x \leq_p (v \cdot u)^{\textcircled{a}} \text{Suc } l \cdot v \rangle$ 
  show  $x \leq_p ((v \cdot u) \cdot v) \cdot (u \cdot v)^{\textcircled{a}} l$ 
    by comparison
  show  $(u \cdot v)^{\textcircled{a}} l \in \langle \{v, u\} \rangle$ 
    by blast
qed fact+
from pref-extD[OF  $\langle v \cdot u \leq_p x \rangle$ [folded  $\langle u \cdot v = v \cdot u \rangle$ ]]
have  $x \cdot u = u \cdot x$ 
  using  $\langle x \leq_s x \cdot u \rangle$  suf-root-pref-comm by blast
with comm-trans[OF this  $\langle u \cdot v = v \cdot u \rangle$ [symmetric]  $\langle u \neq \varepsilon \rangle$ ]
have  $x \cdot (v \cdot u) = (v \cdot u) \cdot x$ 
  using comm-prod by blast
thus False
  using  $\langle x \cdot v \cdot u \neq (v \cdot u) \cdot x \rangle$  by blast
qed
qed
qed

```

**lemma** *rev-primroot-exp* [*simp*, *cow-simps*]:  $e_{\varrho} (\text{rev } x) = e_{\varrho} x$   
**unfolding** *primitive-root-exp-def* *primroot-rev* *rev-is-Nil-conv* *rev-pow*[*symmetric*]  
*rev-is-rev-conv*..

**lemma** *rev-square-interp*:  
*square-interp* (rev  $x$ ) (rev  $y$ ) (rev  $s$ ) (rev  $p$ ) (rev (map rev  $ws$ ))  
**proof** (*unfold-locales*)  
**show**  $\text{rev } (\text{map rev } ws) \in \text{lists } \{\text{rev } x, \text{rev } y\}$   
**using** *ws-lists* **by** *force*  
**show**  $|\text{rev } y| \leq |\text{rev } x|$   
**using** *y-le-x* **by** *simp*  
**show**  $\neg \varrho (\text{rev } x) \sim \varrho (\text{rev } y)$   
**unfolding** *primroot-rev* *conjug-rev-conv* **using** *nconjug*.  
**show**  $\text{rev } x \cdot \text{rev } y \neq \text{rev } y \cdot \text{rev } x$   
**using** *non-comm* **unfolding** *comm-rev-iff*.  
**interpret** *xy-rev*: *binary-code* rev  $x$  rev  $y$   
**by** (*unfold-locales*, *unfold* *rev-append*[*symmetric*]) (*use* *non-comm*[*symmetric*])  
**in** *blast*)  
**have** *aux*:  $\text{Ref } \{\varrho (\text{rev } x), \text{rev } y\} [\text{rev } x, \text{rev } x] = \text{rev } (\text{map rev } (\text{Ref } \{\varrho x, y\} [x, x]))$   
**unfolding** *xy-code.ref-fst-sq* *xy-rev.ref-fst-sq* *rev-map* *rev-pow* *sing-pow-palindrom'*  
**unfolding** *primroot-rev* *cow-simps* *list.simps* *map-pow-list*..  
**show** (rev  $s$ ) (Ref  $\{\varrho (\text{rev } x), \text{rev } y\} [\text{rev } x, \text{rev } x]$ ) (rev  $p$ )  $\sim_{\mathcal{D}}$  rev (map rev  $ws$ )  
**using** *disj-root* **unfolding** *aux* *rev-disj-interp-iff*.  
**qed**

**lemma** *hd-ws*:  $\text{hd } ws = x$   
**using** *square-interp.last-ws*[*reversed*, *OF* *rev-square-interp*]  
**unfolding** *hd-map*[*OF* *ws-nemp*]

by *simp*

**lemma** *p-pref*:  $p <_p x$   
 using *fac-interpD(1) hd-ws interp* by *auto*

**lemma** *s-suf*:  $s <_s x$   
 using *fac-interpD(2) last-ws interp* by *auto*

**end**

### 0.3.2 Locale with additional parameters

**locale** *square-interp-plus* = *square-interp* +  
 fixes  $l\ m\ u\ v$   
 assumes *fst-x*:  $x \cdot y^{\textcircled{m}} \cdot u = p \cdot x$  and  
*snd-x*:  $v \cdot y^{\textcircled{l}} \cdot x = x \cdot s$  and  
*uv-y*:  $u \cdot v = y$  and  
*u-nemp*:  $u \neq \varepsilon$  and *v-nemp*:  $v \neq \varepsilon$  and  
*vu-x-non-comm*:  $x \cdot (v \cdot u) \neq (v \cdot u) \cdot x$   
**begin**

**interpretation** *binary-code*  $x\ y$   
 using *non-comm* by *unfold-locales*

**lemma** *rev-square-interp-plus*: *square-interp-plus* (*rev*  $x$ ) (*rev*  $y$ ) (*rev*  $s$ ) (*rev*  $p$ )  
 (*rev* (*map* *rev*  $ws$ ))  $m\ l$  (*rev*  $v$ ) (*rev*  $u$ )

**proof**–  
**note** *fac-interpD*[*OF* *interp*, *unfolded hd-ws last-ws*]  
**note**  $\langle s <_s x \rangle$ [*unfolded strict-suffix-to-prefix*]  
**note**  $\langle p <_p x \rangle$ [*unfolded spref-rev-suf-iff*]

**interpret** *i*: *square-interp* (*rev*  $x$ ) (*rev*  $y$ ) (*rev*  $s$ ) (*rev*  $p$ ) (*rev* (*map* *rev*  $ws$ ))  
 using *rev-square-interp*.  
**show** *?thesis*  
 by *standard*  
 (*simp-all* *del*: *rev-append add*: *rev-pow*[*symmetric*] *rev-append*[*symmetric*],  
*simp-all* *add*: *fst-x snd-x uv-y v-nemp u-nemp vu-x-non-comm*[*symmetric*,  
*unfolded rassoc*])  
**qed**

### Exactly one of the exponents is zero: impossible.

Uses lemma  $\llbracket ?x \leq_p ?v \cdot ?x; |?v \cdot ?u| < |?x|; \leq_s ?x (?r \cdot ?u \cdot ?v \cdot ?u); ?r \in \langle \{ ?u, ?v \} \rangle \rrbracket \implies ?u \cdot ?v = ?v \cdot ?u$  and exploits the symmetric interpretation.

**lemma** *fst-exp-zero*: **assumes**  $m = 0$  **and**  $0 < l$  **shows** *False*  
**proof** (*rule* *notE*[*OF* *vu-x-non-comm*])  
**note** *y-le-x*[*folded uv-y*, *unfolded swap-len*[*of*  $u$ ]]  
**have**  $x \leq_p (v \cdot (u \cdot v)^{\textcircled{l}}) \cdot x$

**unfolding** *rassoc* **using** *snd-x[folded uv-y]* **by** *blast*  
**have**  $v \cdot (u \cdot v)^{\textcircled{l}} \neq \varepsilon$   
**using** *v-nemp* **by** *force*  
**obtain** *exp* **where**  $x \leq_p (v \cdot (u \cdot v)^{\textcircled{l}})^{\textcircled{exp}}$   $0 < \text{exp}$   
**using** *per-root-powE[OF <x ≤p (v · (u · v) <sup>ⓐ</sup> l) · x> <v · (u · v) <sup>ⓐ</sup> l ≠ ε>, of thesis]* **by** *blast*

**have**  $x \leq_s x \cdot u$   
**using** *fst-x[unfolded <m = 0> pow-zero emp-simps]* **by** (*simp add: suffix-def*)  
**have**  $((v \cdot u) \cdot v) \cdot ((u \cdot v)^{\textcircled{l-1}}) \cdot (v \cdot (u \cdot v)^{\textcircled{l}})^{\textcircled{exp-1}} = (v \cdot (u \cdot v)^{\textcircled{l}})^{\textcircled{exp}}$   
**(is**  $((v \cdot u) \cdot v) \cdot ?\text{suf} = (v \cdot (u \cdot v)^{\textcircled{l}})^{\textcircled{exp}}$ **)**  
**using**  $\langle 0 < l \rangle \langle 0 < \text{exp} \rangle$  **by** *comparison*  
**have**  $v \cdot u \leq_p x$   
**using** *pref-prod-longer[OF <x ≤p (v · (u · v) <sup>ⓐ</sup> l) · x>[unfolded rassoc] - <|v · u| ≤ |x>]*  
**unfolding** *pow-pos[OF <0 < l>]* *rassoc* **by** *blast*  
**hence**  $|v \cdot u| < |x|$   
**using** *nconjug' conjugI[OF - uv-y, of x] <|v · u| ≤ |x>*  
*le-neq-implies-less pref-same-len* **by** *blast*  
**have**  $u \cdot v = v \cdot u$   
**proof** (*rule pref-suf-pers-short[reversed]*)  
**show**  $x \leq_p ((v \cdot u) \cdot v) \cdot ?\text{suf}$   
**unfolding**  $\langle ((v \cdot u) \cdot v) \cdot ?\text{suf} = (v \cdot (u \cdot v)^{\textcircled{l}})^{\textcircled{exp}} \rangle$  **by** *fact*  
**show**  $((u \cdot v)^{\textcircled{l-1}}) \cdot (v \cdot (u \cdot v)^{\textcircled{l}})^{\textcircled{exp-1}} \in \langle \{v, u\} \rangle$   
**by** (*simp add: gen-in hull-closed power-in*)  
**qed** *fact+*  
**show**  $x \cdot v \cdot u = (v \cdot u) \cdot x$   
**using** *root-suf-comm[OF - <x ≤s x · u>] pref-keeps-per-root comm-trans[OF <u · v = v · u>[symmetric] - u-nemp, symmetric] <v · u ≤p x> comm-prod prefI*  
**by** *metis*  
**qed**

**lemma** *snd-exp-zero: assumes*  $0 < m$  **and**  $l = 0$  **shows** *False*  
**using** *square-interp-plus.fst-exp-zero[OF rev-square-interp-plus, reversed, rotated, OF assms]*.

## Both exponents positive: impossible

**lemma** *both-exps-pos: assumes*  $0 < m$  **and**  $0 < l$  **shows** *False*

**proof**–

**note** *fac-interpD[OF interp, unfolded hd-ws last-ws]*  
**have**  $|p| \leq |x|$  **and**  $|s| \leq |x|$   
**using** *pref-len[OF sprefD1[OF <p <p x>]] suf-len[OF ssufD1[OF <s <s x>]]*.

**have**  $x \leq_p (v \cdot (u \cdot v)^{\textcircled{l}}) \cdot x$   
**(is**  $x \leq_p ?\text{pref} \cdot x$ **)**  
**using** *snd-x[folded uv-y]* **by** *force*  
**moreover** **have**  $x \leq_s x \cdot ((u \cdot v)^{\textcircled{m}} \cdot u)$

(is  $x \leq_s x \cdot ?suf$ )  
**using**  $fst-x[folded\ uv-y]$  **by** *force*

**ultimately interpret**  $pref-suf-pers\ x\ u\ v\ l\ m$   
**using**  $\langle 0 < l \rangle \langle 0 < m \rangle$  **by** *unfold-locales*

**have**  $?pref \leq_p x$   
**using**  $snd-x[folded\ uv-y\ rassoc,\ symmetric]$   $eqd[reversed,\ OF - \langle |s| \leq |x| \rangle]$  **by**  
*blast*

**have**  $?suf \leq_s x$   
**using**  $fst-x[folded\ uv-y,\ symmetric]$   $eqd[OF - \langle |p| \leq |x| \rangle]$  **by** *blast*

**have** *in-particular: commutes*  $\{u, v, x\} \implies x \cdot (v \cdot u) = (v \cdot u) \cdot x$   
**unfolding** *commutes-def* **by** (*rule comm-prod*) *blast+*

— Case analysis based on (slightly modified) lemmas for covered x square.

**note** *no-overlap-comm* = *no-overlap*[*THEN in-particular*] **and**  
*short-overlap-comm* = *short-overlap*[*THEN in-particular*] **and**  
*medium-overlap-comm* = *medium-overlap*[*THEN in-particular*] **and**  
*large-overlap-conjug* = *pref-suf-pers-large-overlap*[*OF*  $\langle ?pref \leq_p x \rangle \langle ?suf \leq_s x \rangle$ ,  
*of v · u*]

**consider**

(*no-overlap*)  $|?pref| + |?suf| \leq |x|$  |

(*short-overlap*)  $|x| < |?pref| + |?suf| \wedge |?pref| + |?suf| \leq |x| + |u|$  |

(*medium-overlap*)  $|x| + |u| < |?pref| + |?suf| \wedge |?pref| + |?suf| < |x| + |u \cdot v|$  |

(*large-overlap*)  $|x| + |v \cdot u| \leq |?pref| + |?suf|$

**unfolding** *swap-len*[*of v*] **by** *linarith*

**thus** *False*

**proof** (*cases*)

**case** *no-overlap*

**then show** *False*

**using** *no-overlap-comm vu-x-non-comm*  $\langle 0 < l \rangle \langle 0 < m \rangle$  **by** *blast*

**next**

**case** *short-overlap*

**then show** *False*

**using** *short-overlap-comm vu-x-non-comm* **by** *blast*

**next**

**case** *medium-overlap*

**then show** *False*

**using** *medium-overlap-comm vu-x-non-comm* **by** *blast*

**next**

**case** *large-overlap*

**show** *False*

**thm** *large-overlap-conjug nconjug*

**proof** (*rule notE*[*OF vu-x-non-comm*], *rule large-overlap-conjug*[*OF - - large-overlap*])

**have**  $(u \cdot v) \stackrel{\textcircled{a}}{(l-1)} \leq_p (u \cdot v) \stackrel{\textcircled{a}}{Suc\ (l-1)}$

```

    using pref-pow-ext by blast
  thus  $v \cdot (u \cdot v)^{\textcircled{a} l} \leq_p (v \cdot u) \cdot v \cdot (u \cdot v)^{\textcircled{a} l}$ 
    unfolding pow-pos[OF <0 < l>] pow-Suc rassoc pref-cancel-conv.
  show  $(u \cdot v)^{\textcircled{a} m} \cdot u \leq_s ((u \cdot v)^{\textcircled{a} m} \cdot u) \cdot v \cdot u$ 
    by comparison
qed
qed
qed

```

**thm** *suf-cancel-conv*

**end**

### 0.3.3 Back to the main locale

**context** *square-interp*

**begin**

**definition** *u where*  $u = x^{-1} \triangleright (p \cdot x)$

**definition** *v where*  $v = (x \cdot s)^{\triangleleft -1} x$

**lemma** *cover-xyx: ws = [x,y,x] and vu-x-non-comm:  $x \cdot (v \cdot u) \neq (v \cdot u) \cdot x$  and uv-y:  $u \cdot v = y$  and*

*px-xu:  $p \cdot x = x \cdot u$  and vx-xs:  $v \cdot x = x \cdot s$  and u-nemp:  $u \neq \varepsilon$  and v-nemp:  $v \neq \varepsilon$*

**proof**–

**obtain** *k where*  $ws: [x] \cdot [y]^{\textcircled{a} k} \cdot [x] = ws$

**using** *kE[unfolded hd-ws last-ws].*

**obtain** *m u' v' l where*  $x \cdot y^{\textcircled{a} m} \cdot u' = p \cdot x$  **and**  $v' \cdot y^{\textcircled{a} l} \cdot x = x \cdot s$  **and**  $u' \cdot v' = y$

**and**  $u' \neq \varepsilon$  **and**  $v' \neq \varepsilon$  **and**  $x \cdot v' \cdot u' \neq (v' \cdot u') \cdot x$

**using** *l-mE[unfolded hd-ws last-ws].*

**then interpret** *square-interp-plus x y p s ws l m u' v'*

**by** *(unfold-locale)*

**have**  $m = 0$  **and**  $l = 0$  **and**  $y \neq \varepsilon$

**using** *both-exps-pos snd-exp-zero fst-exp-zero <u' \cdot v' = y> <u' \neq \varepsilon>* **by** *blast+*

**have**  $u' = u$

**unfolding** *u-def*

**using** *conjug-lq[OF fst-x[unfolded <m = 0> pow-zero emp-simps, symmetric]].*

**have**  $v' = v$

**unfolding** *v-def*

**using** *conjug-lq[reversed, OF snd-x[unfolded <l = 0> pow-zero emp-simps, symmetric]].*

**have**  $x \cdot y^{\textcircled{a} m} \cdot (u' \cdot v') \cdot y^{\textcircled{a} l} \cdot x = \text{concat } ws$

**unfolding** *interpret-concat[symmetric] using fst-x snd-x by force*

**from** *this[folded ws, unfolded <u' \cdot v' = y> <m = 0> <l = 0> pow-zero emp-simps]*

**have**  $k = 1$

**unfolding** *eq-pow-exp[OF <y \neq \varepsilon>, of k 1, symmetric] pow-list-1 concat-morph*

*concat-pow-list*  
**by simp**  
**from** *ws*[*unfolded this pow-list-1*]  
**show**  $ws = [x, y, x]$  **by simp**  
**show**  $u \cdot v = y$   
**unfolding**  $\langle u' = u \rangle$ [*symmetric*]  $\langle v' = v \rangle$ [*symmetric*] **by fact+**  
**show**  $p \cdot x = x \cdot u$   
**using**  $\langle x \cdot y \rangle^{\textcircled{m}} \cdot u' = p \cdot x$ [*unfolded*  $\langle m = 0 \rangle$   $\langle u' = u \rangle$  *pow-zero emp-simps*,  
*symmetric*].  
**show**  $v \cdot x = x \cdot s$   
**using**  $\langle v' \cdot y \rangle^{\textcircled{l}} \cdot x = x \cdot s$ [*unfolded*  $\langle l = 0 \rangle$   $\langle v' = v \rangle$  *pow-zero emp-simps*].  
**show**  $x \cdot (v \cdot u) \neq (v \cdot u) \cdot x$   
**using**  $\langle x \cdot v' \cdot u' \rangle \neq \langle v' \cdot u' \rangle \cdot x$ [*unfolded*  $\langle u' = u \rangle$   $\langle v' = v \rangle$ ].  
**show**  $u \neq \varepsilon$  **and**  $v \neq \varepsilon$   
**using**  $\langle u' \neq \varepsilon \rangle$   $\langle v' \neq \varepsilon \rangle$  **unfolding**  $\langle u' = u \rangle$   $\langle v' = v \rangle$ .  
**qed**

**lemma** *cover*:  $x \cdot y \cdot x = p \cdot x \cdot x \cdot s$   
**using** *interpret-concat cover-xyx* **by auto**

**lemma** *conjug-facs*:  $\varrho u \sim \varrho v$

**proof**–

**note** *sufI*[*OF px-xu*]  
**have**  $u \neq \varepsilon$   
**using** *p-nemp px-xu* **by force**  
**obtain** *expu* **where**  $x <_s u^{\textcircled{}}$  *expu*  $0 < \text{expu}$   
**using** *per-root-powE*[*reversed*, *OF*  $\langle x \leq_s x \cdot u \rangle$   $\langle u \neq \varepsilon \rangle$ ].  
**hence**  $x \leq_f u^{\textcircled{}}$  *expu*  
**using** *ssufD1* **by blast**

**note** *prefI*[*OF vx-xs*[*symmetric*]]  
**have**  $v \neq \varepsilon$   
**using** *s-nemp vx-xs* **by force**  
**obtain** *expv* **where**  $x <_p v^{\textcircled{}}$  *expv*  $0 < \text{expv}$   
**using** *per-root-powE*[*OF*  $\langle x \leq_p v \cdot x \rangle$   $\langle v \neq \varepsilon \rangle$ ].  
**hence**  $x \leq_f v^{\textcircled{}}$  *expv* **by blast**

**show**  $\varrho u \sim \varrho v$   
**proof**(*rule fac-two-conjug-primroot*[*OF*  $\langle x \leq_f u^{\textcircled{}}$  *expu*  $\langle x \leq_f v^{\textcircled{}}$  *expv*])  
**show**  $|u| + |v| \leq |x|$   
**using** *y-le-x*[*folded uv-y*, *unfolded lenmorph*] **by fastforce**

**qed**  
**qed**

**term** *square-interp.v*

— We have a detailed information about all words

**lemma** *bin-sq-interpE*: **obtains**  $r t m k l$   
**where**  $(t \cdot r)^{\textcircled{k}} = u$  **and**  $(r \cdot t)^{\textcircled{l}} = v$  **and**

$(r \cdot t)^{\textcircled{m}} \cdot r = x$  and  $(t \cdot r)^{\textcircled{k}} \cdot (r \cdot t)^{\textcircled{l}} = y$   
**and**  $(r \cdot t)^{\textcircled{k}} = p$  and  $(t \cdot r)^{\textcircled{l}} = s$  and  $r \cdot t \neq t \cdot r$  and  
 $0 < k$  and  $0 < m$  and  $0 < l$  and  $k + l \leq m$

**proof-**

**obtain**  $r \ t \ k \ m$  **where**  $(r \cdot t)^{\textcircled{k}} = p$  and  $(t \cdot r)^{\textcircled{k}} = u$  and  $(r \cdot t)^{\textcircled{m}} \cdot r = x$

**and**

$t \neq \varepsilon$  and  $0 < k$  and *primitive*  $(r \cdot t)$

**using** *conjug-eq-primrootE*[*OF px-xu p-nemp*].

**have**  $t \cdot r = \varrho \ u$

**using** *prim-conjug*[*OF*  $\langle$ *primitive*  $(r \cdot t)$  $\rangle$ , *THEN primroot-unique*[*OF u-nemp*],  
*OF conjugI'*  $\langle$  $(t \cdot r)^{\textcircled{k}} = u$  $\rangle$ [*symmetric*]].

**have**  $0 < m$

**proof** (*rule ccontr*)

**assume**  $\neg 0 < m$

**hence**  $x = r$  **using**  $\langle$  $(r \cdot t)^{\textcircled{m}} \cdot r = x$  $\rangle$  **by** *simp*

**show** *False*

**using**  $\langle 0 < k \rangle$   $\langle$  $(r \cdot t)^{\textcircled{k}} = p$  $\rangle$   $\langle x = r \rangle$  *comp-pows-pref-zero p-pref* **by** *blast*

**qed**

**from**  $\langle$  $(r \cdot t)^{\textcircled{m}} \cdot r = x$  $\rangle$ [*unfolded pow-pos*[*OF*  $\langle 0 < m \rangle$ ]]

**have**  $r \cdot t \leq_p x$

**by** *blast*

**have**  $r \cdot t = \varrho \ v$

**proof** (*rule ruler-eq-len*[*of*  $\varrho \ v \ x \ r \cdot t$ , *symmetric*])

**have**  $|\varrho \ v| \leq |x|$

**unfolding** *conjug-len*[*OF conjug-facs, symmetric*]  $\langle t \cdot r = \varrho \ u \rangle$ [*symmetric*]

**unfolding**  $\langle$  $(r \cdot t)^{\textcircled{m}} \cdot r = x$  $\rangle$ [*symmetric*] *pow-pos*[*OF*  $\langle 0 < m \rangle$ ]

*lenmorph pow-len* **by** *auto*

**from** *ruler-le*[*OF* - - *this, of v · x*]

**show**  $\varrho \ v \leq_p x$

**using** *vx-xs prefI prefix-prefix primroot-pref v-nemp* **by** *metis*

**show**  $r \cdot t \leq_p x$  **by** *fact*

**show**  $|\varrho \ v| = |r \cdot t|$

**unfolding** *conjug-len*[*OF conjug-facs, symmetric, folded*  $\langle t \cdot r = \varrho \ u \rangle$ ] *lenmorph*

**by** *simp*

**qed**

**then obtain**  $l$  **where**  $(r \cdot t)^{\textcircled{l}} = v$  and  $0 < l$

**using** *primroot-expE v-nemp* **by** *metis*

**have**  $(t \cdot r)^{\textcircled{l}} = s$

**using** *vx-xs*[*folded*  $\langle$  $(r \cdot t)^{\textcircled{m}} \cdot r = x$  $\rangle$   $\langle$  $(r \cdot t)^{\textcircled{l}} = v$  $\rangle$ , *unfolded lassoc*  
*pows-comm*[*of* - - *m*],  
*unfolded rassoc cancel, unfolded shift-pow cancel*].

**have**  $r \cdot t \neq t \cdot r$   
**proof**  
**assume**  $r \cdot t = t \cdot r$   
**hence**  $aux: r \cdot (t \cdot r)^{\textcircled{a}} e = (t \cdot r)^{\textcircled{a}} e \cdot r$  **for**  $e$   
**by** *comparison*  
**have**  $x \cdot (v \cdot u) = (v \cdot u) \cdot x$   
**unfolding**  $\langle (t \cdot r)^{\textcircled{a}} k = u \rangle$  *[symmetric]*  $\langle (r \cdot t)^{\textcircled{a}} l = v \rangle$  *[symmetric]*  
**unfolding**  $\langle (r \cdot t)^{\textcircled{a}} m \cdot r = x \rangle$  *[symmetric]* *pow-add[symmetric]*  $\langle r \cdot t = t \cdot$   
 $r \rangle$  *aux rassoc*  
**unfolding** *lassoc cancel-right pow-add[symmetric]*  
**by** *(simp add: add.commute)*  
**thus** *False*  
**using** *vu-x-non-comm* **by** *blast*  
**qed**

**have**  $r \neq \varepsilon \ t \neq \varepsilon \ r \cdot t \neq \varepsilon$   
**using**  $\langle r \cdot t \neq t \cdot r \rangle$  **by** *blast+*

**find-theorems**  $?k < ?m \ ?k + 1$

**note**  $y = uv \cdot y$  *[folded]*  $\langle (t \cdot r)^{\textcircled{a}} k = u \rangle \langle (r \cdot t)^{\textcircled{a}} l = v \rangle$   
**have**  $k + l \leq m$   
**proof** *(rule ccontr, unfold not-le, unfold less-eq-Suc-le)*  
**assume**  $Suc \ m \leq k + l$   
**show** *False*  
**using**  $y \cdot le \cdot x$  *[folded]*  $\langle (r \cdot t)^{\textcircled{a}} m \cdot r = x \rangle$   $y$   
**unfolding** *lenmorph*  
**unfolding** *pow-len swap-len nemp-len[OF <t ≠ ε>]* *add-mult-distrib[symmetric]*  
**using** *mult-le-mono1[OF <Suc m ≤ k + l>, of |r · t|]* *nemp-len[OF <t ≠ ε>]*  
**unfolding** *Suc-eq-plus1 add-mult-distrib* **by** *auto*  
**qed**

**show** *thesis*  
**using** *that[OF <(t · r)<sup>ⓐ</sup> k = u> <(r · t)<sup>ⓐ</sup> l = v> <(r · t)<sup>ⓐ</sup> m · r = x>*  
 $y \langle (r \cdot t)^{\textcircled{a}} k = p \rangle \langle (t \cdot r)^{\textcircled{a}} l = s \rangle \langle r \cdot t \neq t \cdot r \rangle$   
 $\langle 0 < k \rangle \langle 0 < m \rangle \langle 0 < l \rangle \langle k + l \leq m \rangle$ .  
**qed**

**end**

### 0.3.4 Locale: Extendable interpretation

Further specification follows from the assumption that the interpretation is extendable, that is, the covered  $x \cdot x$  is a factor of a word composed of  $\{x, y\}$ . Namely,  $u$  and  $v$  are then conjugate by  $x$ .

**locale** *square-interp-ext = square-interp +*  
**assumes** *p-extend:  $\exists pe. pe \in \{x, y\} \wedge p \leq s \ pe$*  **and**  
*s-extend:  $\exists se. se \in \{x, y\} \wedge s \leq p \ se$*

**begin**

**lemma** *s-pref-y*:  $s \leq_p y$

**proof**–

**obtain**  $sy\ ry\ eu\ ev\ ex$

**where**  $(ry \cdot sy)^{\textcircled{a}} eu = u$  **and**  $(sy \cdot ry)^{\textcircled{a}} ev = v$  **and**

$(sy \cdot ry)^{\textcircled{a}} eu = p$  **and**  $(ry \cdot sy)^{\textcircled{a}} ev = s$  **and**

$(sy \cdot ry)^{\textcircled{a}} ex \cdot sy = x$  **and**  $sy \cdot ry \neq ry \cdot sy$  **and**

$0 < eu$  **and**  $0 < ev$  **and**  $0 < ex$

**using** *bin-sq-interpE*.

**obtain**  $se$  **where**  $se \in \langle \{x, y\} \rangle$  **and**  $s \leq_p se$

**using** *s-extend* **by** *blast*

**hence**  $se \neq \varepsilon$  **using** *s-nemp* **by** *force*

**from**  $\langle (sy \cdot ry)^{\textcircled{a}} ex \cdot sy = x \rangle$

**have**  $sy \cdot ry \leq_p x$

**unfolding** *pow-pos*[*OF*  $\langle 0 < ex \rangle$ ] *rassoc* **by** *force*

**have**  $x \leq_p se \vee y \leq_p se$

**using**  $\langle se \neq \varepsilon \rangle$  *hull.cases*[*OF*  $\langle se \in \langle \{x, y\} \rangle \rangle$ , *of*  $x \leq_p se \vee y \leq_p se$ ]

*prefix-append* *triv-pref* *two-elem-cases* **by** *blast*

**moreover** **have**  $\neg x \leq_p se$

**proof**

**assume**  $x \leq_p se$

**from** *ruler-eq-len*[*of*  $sy \cdot ry\ se\ ry \cdot sy$ , *OF* *pref-trans*[*OF*  $\langle sy \cdot ry \leq_p x \rangle$  *this*]]

**show** *False*

**using**  $\langle s \leq_p se \rangle$ [*folded*  $\langle (ry \cdot sy)^{\textcircled{a}} ev = s \rangle$ [*unfolded* *pow-pos*[*OF*  $\langle 0 < ev \rangle$ ]]]

$\langle sy \cdot ry \neq ry \cdot sy \rangle$  **by** (*force* *simp* *add*: *prefix-def*)

**qed**

**ultimately** **have** *y-pref-se*:  $y \leq_p se$  **by** *blast*

**from** *ruler-le*[*OF*  $\langle s \leq_p se \rangle$  *this*]

**show**  $s \leq_p y$

**using** *lenarg*[*OF*  $vx\ xs$ ] **unfolding** *wv-y*[*symmetric*] *lenmorph* **by** *linarith*

**qed**

**lemma** *rev-square-interp-ext*: *square-interp-ext* (*rev*  $x$ ) (*rev*  $y$ ) (*rev*  $s$ ) (*rev*  $p$ ) (*rev* (*map* *rev*  $ws$ ))

**proof**–

**interpret**  $i$ : *square-interp* (*rev*  $x$ ) (*rev*  $y$ ) (*rev*  $s$ ) (*rev*  $p$ ) (*rev* (*map* *rev*  $ws$ ))

**using** *rev-square-interp*.

**show** *?thesis*

**proof**

**show**  $\exists pe. pe \in \langle \{rev\ x, rev\ y\} \rangle \wedge rev\ s \leq_s pe$

**using** *s-pref-y* **unfolding** *pref-rev-suf-iff* **by** *blast*

**obtain**  $pe$  **where**  $pe \in \langle \{x, y\} \rangle$  **and**  $p \leq_s pe$

**using** *p-extend* **by** *blast*

**hence**  $rev\ pe \in \langle \{rev\ x, rev\ y\} \rangle$

by (*simp add: rev-hull rev-in-conv*)  
 thus  $\exists se. se \in \{\text{rev } x, \text{rev } y\} \wedge \text{rev } p \leq p \text{ se}$   
 using  $\langle p \leq s \text{ pe} \rangle$ [*unfolded suf-rev-pref-iff prefix-def*] *rev-rev-ident* by *blast*  
 qed  
 qed

**lemma** *p-suf-y*:  $p \leq s \ y$

**proof**–

interpret *i*: *square-interp-ext* (*rev x*) (*rev y*) (*rev s*) (*rev p*) (*rev (map rev ws)*)  
 using *rev-square-interp-ext*.

from *i.s-pref-y*[*reversed*]

show  $p \leq s \ y$ .

qed

**theorem** *bin-sq-interp-extE*: obtains  $r \ t \ k \ m$  where  $(r \cdot t)^{\textcircled{m}} \cdot r = x$  and  $(t \cdot r)^{\textcircled{k}} \cdot (r \cdot t)^{\textcircled{k}} = y$   
 $(r \cdot t)^{\textcircled{k}} \cdot k = p$  and  $(t \cdot r)^{\textcircled{k}} \cdot k = s$  and  $r \cdot t \neq t \cdot r$  and  $u = s$  and  $v = p$  and  
 $|p| = |s|$  and  
 $0 < k$  and  $0 < m$  and  $k + k \leq m$

**proof**–

obtain  $r \ t \ k \ k' \ m$

where  $u: (t \cdot r)^{\textcircled{k}} \cdot k = u$  and  $v: (r \cdot t)^{\textcircled{k}} \cdot k' = v$  and

$p: (r \cdot t)^{\textcircled{k}} \cdot k = p$  and  $s: (t \cdot r)^{\textcircled{k}} \cdot k' = s$  and

$x: (r \cdot t)^{\textcircled{k}} \cdot m \cdot r = x$  and *code*:  $r \cdot t \neq t \cdot r$  and

$0 < k' \ 0 < m \ 0 < k \ k + k' \leq m$

using *bin-sq-interpE*.

have  $|u \cdot v| = |s \cdot p|$

using *lenarg[OF px-xu, unfolded lenmorph]* *lenarg[OF vx-xs, unfolded lenmorph]*

by *simp*

hence  $u \cdot v = s \cdot p$

unfolding *uv-y* using *s-pref-y p-suf-y* by (*auto simp add: prefix-def suffix-def*)

note  $eq = \langle u \cdot v = s \cdot p \rangle$ [*unfolded*  $\langle (t \cdot r)^{\textcircled{k}} \cdot k = u \rangle$ [*symmetric*]  $\langle (r \cdot t)^{\textcircled{k}} \cdot k' = v \rangle$ [*symmetric*],

*unfolded*  $\langle (t \cdot r)^{\textcircled{k}} \cdot k' = s \rangle$ [*symmetric*]  $\langle (r \cdot t)^{\textcircled{k}} \cdot k = p \rangle$ [*symmetric*]]

from *pows-comm-comm*[*OF this*]

have  $k = k'$

using  $\langle r \cdot t \neq t \cdot r \rangle$  *eqd-eq(1)*[*OF - swap-len, of t r*] by *fastforce*

have  $|p| = |s|$

using *lenarg[OF p]* *lenarg[OF s]* unfolding  $\langle k = k' \rangle$  *pow-len lenmorph add commute*[*of |r|*] by *fastforce*

thus *thesis*

using *that*[*OF x uv-y*][*folded u v*  $\langle k = k' \rangle$ ] *p s* [*folded*  $\langle k = k' \rangle$ ] *code - - -*  $\langle 0 < k \rangle$   
 $\langle 0 < m \rangle$  *u v p s*  $\langle k + k' \leq m \rangle$  unfolding  $\langle k = k' \rangle$  by *argo*

qed

**lemma** *ps-len*:  $|p| = |s|$  and *p-eq-v*:  $p = v$  and *s-eq-u*:  $s = u$

using *bin-sq-interp-extE* by *blast+*

```

lemma v-x-x-u:  $v \cdot x = x \cdot u$ 
  using vx-xs unfolding s-eq-u.

lemma sp-y:  $s \cdot p = y$ 
  using p-eq-v s-eq-u uv-y by auto

lemma p-x-x-s:  $p \cdot x = x \cdot s$ 
  by (simp add: px-xu s-eq-u)

lemma xy-root:  $x \cdot x \cdot y = (x \cdot p) \cdot (x \cdot p)$ 
  using p-x-x-s sp-y by force

theorem sq-ext-interp:  $ws = [x, y, x] \ s \cdot p = y \ p \cdot x = x \cdot s$ 
  using cover-xyx sp-y p-x-x-s.

end

lemma prim-sq-interp:
  assumes  $x \cdot y \neq y \cdot x$  and primitive x and  $|y| \leq |x|$  and  $ws \in \text{lists } \{x, y\}$  and
   $\neg x \sim y$  and
   $p [x, x] \ s \sim_{\mathcal{D}} \ ws$ 
  shows square-interp x y p s ws
proof
  interpret binary-code x y
  using  $\langle x \cdot y \neq y \cdot x \rangle$  by unfold-locales
  have  $y \neq \varepsilon$ 
  using  $\langle x \cdot y \neq y \cdot x \rangle$  by blast
  show  $x \cdot y \neq y \cdot x$ 
  using  $\langle x \cdot y \neq y \cdot x \rangle$ .
  show  $|y| \leq |x|$ 
  using  $\langle |y| \leq |x| \rangle$ .
  show  $ws \in \text{lists } \{x, y\}$ 
  using  $\langle ws \in \text{lists } \{x, y\} \rangle$ .
  have  $\varrho \ x = x$ 
  by (simp add: asms(2) prim-primroot)
  show  $\neg \varrho \ x \sim \varrho \ y$ 
  unfolding prim-primroot[OF  $\langle \text{primitive } x \rangle$ ]
  using  $\langle \neg x \sim y \rangle \ \langle |y| \leq |x| \rangle$  short-primroot[OF  $\langle y \neq \varepsilon \rangle$ ] prim-primroot-conv[OF
   $\langle y \neq \varepsilon \rangle$ ]
  by (cases primitive y) (simp, use conjug-len[of  $x \ \varrho \ y$ ] in linarith)
  show  $p \ \text{Ref } \{\varrho \ x, y\} \ [x, x] \ s \sim_{\mathcal{D}} \ ws$ 
  using  $\langle p [x, x] \ s \sim_{\mathcal{D}} \ ws \rangle$  disj-interpI'[OF disj-interpD0[OF  $\langle p [x, x] \ s \sim_{\mathcal{D}} \ ws \rangle$ ]]
  unfolding refine-def prim-primroot[OF  $\langle \text{primitive } x \rangle$ ] using bin-roots-decompose(5)
by auto
qed

```

## 0.4 Global claims

**theorem** *bin-sq-interpE*:

**assumes**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  $\neg \varrho x \sim \varrho y$  **and**  
 $p \text{ Ref } \{\varrho x, y\} [x, x] s \sim_{\mathcal{D}} ws$   
**obtains**  $r t m k l$  **where**  $(r \cdot t)^{\textcircled{a}} m \cdot r = x$  **and**  $(t \cdot r)^{\textcircled{a}} k \cdot (r \cdot t)^{\textcircled{a}} l = y$   
 $(r \cdot t)^{\textcircled{a}} k = p$  **and**  $(t \cdot r)^{\textcircled{a}} l = s$  **and**  $r \cdot t \neq t \cdot r$  **and**  $0 < k \ 0 < m \ 0 < l \ k$   
 $+ l \leq m$   
**using** *square-interp.bin-sq-interpE*[*OF square-interp.intro, OF assms*].

**theorem** *bin-sq-interp-primE*:

**assumes**  $x \cdot y \neq y \cdot x$  **and** *primitive*  $x$  **and**  $|y| \leq |x|$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  
 $\neg x \sim y$  **and**  
 $p [x, x] s \sim_{\mathcal{D}} ws$   
**obtains**  $r t m k l$  **where**  $(r \cdot t)^{\textcircled{a}} m \cdot r = x$  **and**  $(t \cdot r)^{\textcircled{a}} k \cdot (r \cdot t)^{\textcircled{a}} l = y$   
 $(r \cdot t)^{\textcircled{a}} k = p$  **and**  $(t \cdot r)^{\textcircled{a}} l = s$  **and**  $r \cdot t \neq t \cdot r$  **and**  $0 < k \ 0 < m \ 0 < l \ k$   
 $+ l \leq m$

**proof**–

**interpret** *square-interp*  $x y p s ws$   
**using** *prim-sq-interp*[*OF assms*].  
**from** *bin-sq-interpE*[*of thesis*]  
**show** *thesis*  
**using** *that by blast*

**qed**

**theorem** *bin-sq-interp*:

**assumes**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  $\neg \varrho x \sim \varrho y$  **and**  
 $p \text{ Ref } \{\varrho x, y\} [x, x] s \sim_{\mathcal{D}} ws$   
**shows**  $ws = [x, y, x]$   
**using** *square-interp.cover-xyx*[*OF square-interp.intro, OF assms*].

**theorem** *bin-sq-interp-prim*:

**assumes**  $x \cdot y \neq y \cdot x$  **and** *primitive*  $x$  **and**  $|y| \leq |x|$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  
 $\neg x \sim y$  **and**  
 $p [x, x] s \sim_{\mathcal{D}} ws$   
**shows**  $ws = [x, y, x]$   
**using** *square-interp.cover-xyx*[*OF prim-sq-interp*[*OF assms*]].

**theorem** *bin-sq-interp-extE*:

**assumes**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  $\neg \varrho x \sim \varrho y$   
**and**  
 $p \text{ Ref } \{\varrho x, y\} [x, x] s \sim_{\mathcal{D}} ws$  **and**  
 $p\text{-extend}: \exists pe. pe \in \langle \{x, y\} \rangle \wedge p \leq s \text{ pe}$  **and**  
 $s\text{-extend}: \exists se. se \in \langle \{x, y\} \rangle \wedge s \leq p \text{ se}$   
**obtains**  $r t m k$  **where**  $(r \cdot t)^{\textcircled{a}} m \cdot r = x$  **and**  $(t \cdot r)^{\textcircled{a}} k \cdot (r \cdot t)^{\textcircled{a}} k = y$   
 $(r \cdot t)^{\textcircled{a}} k = p$  **and**  $(t \cdot r)^{\textcircled{a}} k = s$  **and**  $r \cdot t \neq t \cdot r$  **and**  $0 < k$  **and**  $0 < m$   
**using** *square-interp-ext.bin-sq-interp-extE*[*OF square-interp-ext.intro, OF square-interp.intro*  
*square-interp-ext-axioms.intro, OF assms*].

**theorem** *bin-sq-interp-ext-primE*:

**assumes**  $x \cdot y \neq y \cdot x$  **and** *primitive*  $x$  **and**  $|y| \leq |x|$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  
 $\neg x \sim y$  **and**

$p [x,x] s \sim_{\mathcal{D}} ws$  **and**  
*p-extend*:  $\exists pe. pe \in \langle \{x,y\} \rangle \wedge p \leq s pe$  **and**  
*s-extend*:  $\exists se. se \in \langle \{x,y\} \rangle \wedge s \leq p se$   
**obtains**  $r t m k$  **where**  $(r \cdot t)^{\otimes} m \cdot r = x$  **and**  $(t \cdot r)^{\otimes} k \cdot (r \cdot t)^{\otimes} k = y$   
 $(r \cdot t)^{\otimes} k = p$  **and**  $(t \cdot r)^{\otimes} k = s$  **and**  $r \cdot t \neq t \cdot r$  **and**  $0 < k$  **and**  $0 < m$   
**using** *square-interp-ext.bin-sq-interp-extE*[*OF square-interp-ext.intro*, *OF - square-interp-ext-axioms.intro*,  
*OF prim-sq-interp*, *OF assms*].

### 0.4.1 Examples

Basic example of an extendable cover

**lemma** *example-imprim-sq-cover*:

**fixes**  $x y p s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot b \cdot a \cdot b \cdot a$  **and**  $y \equiv b \cdot a \cdot a \cdot b$  **and**  
 $p \equiv a \cdot b$  **and**  $s \equiv b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$  **and** *primitive*  $x$   
 $p x \cdot x s \sim_{\mathcal{I}} [x,y,x]$   
 $\neg$  *primitive*  $(x \cdot x \cdot y)$

**proof**–

**show**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$   
**unfolding** *x-def y-def* **by** *simp-all*  
**show**  $\neg$  *primitive*  $(x \cdot x \cdot y)$  **and** *primitive*  $x$   
**unfolding** *x-def y-def* **by** *primitivity-inspection+*  
**show**  $p x \cdot x s \sim_{\mathcal{I}} [x,y,x]$   
**unfolding** *x-def y-def p-def s-def*  
**by** (*rule fac-interpI*) *force+*

**qed**

Example of a non-extendable cover

**lemma** *example-prim-sq-cover*:

**fixes**  $x y p s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot b \cdot a \cdot b \cdot a$  **and**  $y \equiv b \cdot a \cdot a \cdot b \cdot a \cdot b$  **and**  
 $p \equiv a \cdot b$  **and**  $s \equiv b \cdot a \cdot b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$  **and** *primitive*  $x$   
 $p x \cdot x s \sim_{\mathcal{I}} [x,y,x]$   
*primitive*  $(x \cdot x \cdot y)$

**proof**–

**show**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$   
**unfolding** *x-def y-def* **by** *simp-all*  
**show** *primitive*  $(x \cdot x \cdot y)$  **and** *primitive*  $x$   
**unfolding** *x-def y-def* **by** *primitivity-inspection+*  
**show**  $p x \cdot x s \sim_{\mathcal{I}} [x,y,x]$   
**unfolding** *x-def y-def p-def s-def*  
**by** (*rule fac-interpI*) *force+*

**qed**

Cube cover with a long  $y$

**lemma** *example-cube-cover*:

**fixes**  $x y p s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot b \cdot a$  and  $y \equiv b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot b$  and  
 $p \equiv a \cdot b$  and  $s \equiv b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$  and  $|x| + |y| < |x \cdot x \cdot x|$  and *primitive*  $x$   
 $p \cdot x \cdot x \cdot x \cdot s \sim_{\mathcal{I}} [x, y, x]$   
*primitive*  $(x \cdot x \cdot y)$   
**proof**–  
**show**  $x \cdot y \neq y \cdot x$  and  $|x| + |y| < |x \cdot x \cdot x|$   
**unfolding**  $x$ -def  $y$ -def **by** *simp-all*  
**show** *primitive*  $(x \cdot x \cdot y)$  and *primitive*  $x$   
**unfolding**  $x$ -def  $y$ -def **by** *primitivity-inspection*+  
**show**  $p \cdot x \cdot x \cdot x \cdot s \sim_{\mathcal{I}} [x, y, x]$   
**unfolding**  $x$ -def  $y$ -def  $p$ -def  $s$ -def  
**by** (*rule fac-interpI*) *force*+  
**qed**

**lemma** *example-pow-cover*:

**fixes**  $x y p s n$   
**assumes**  $2 \leq n$   
**defines**  $x \equiv a \cdot b \cdot a \cdot b \cdot a$  and  $y \equiv b \cdot a \cdot x^{\textcircled{n}} \cdot a \cdot b$  and  
 $p \equiv a \cdot b$  and  $s \equiv b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$   
and  $|x| + |y| \leq |x^{\textcircled{n}}|$   
and *primitive*  $x$   
 $p \cdot x^{\textcircled{n}} \cdot s \sim_{\mathcal{I}} [x, y, x]$

**proof**–  
**have**  $xn$ :  $x^{\textcircled{n}} = x \cdot x^{\textcircled{n-2}} \cdot x$   
**unfolding** *pow-Suc2[symmetric]* *pow-Suc[symmetric]* *Suc-minus2[OF 2 ≤ n]*..  
**show**  $x \cdot y \neq y \cdot x$  and  $|x| + |y| \leq |x^{\textcircled{n}}|$   
**unfolding**  $xn$  **unfolding**  $x$ -def  $y$ -def **by** *simp-all*  
**show** *primitive*  $x$   
**unfolding**  $x$ -def  $y$ -def **by** *primitivity-inspection*  
**show**  $p \cdot x^{\textcircled{n}} \cdot s \sim_{\mathcal{I}} [x, y, x]$   
**unfolding**  $xn$  **unfolding**  $x$ -def  $y$ -def  $p$ -def  $s$ -def *concat.simps*  
**by** (*rule fac-interpI*) *force*+  
**qed**

Not root-disjoint covers

**lemma** *example-root-joint-sq-cover*:

**fixes**  $x y p s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot a \cdot b \cdot a$  and  $y \equiv a \cdot b$  and  
 $p \equiv a \cdot b \cdot a$  and  $s \equiv b \cdot a \cdot a \cdot b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$  and  $|y| \leq |x|$  and  $\neg$  *primitive*  $x$   
 $p \cdot x \cdot x \cdot s \sim_{\mathcal{I}} [x, x, y, x]$   
*primitive*  $(x \cdot x \cdot y)$

**proof**–  
**show**  $x \cdot y \neq y \cdot x$  and  $|y| \leq |x|$   
**unfolding**  $x$ -def  $y$ -def **by** *simp-all*  
**show** *primitive*  $(x \cdot x \cdot y)$  and  $\neg$  *primitive*  $x$

**unfolding**  $x$ -def  $y$ -def by *primitivity-inspection*+  
**show**  $p \cdot x \cdot x \cdot s \sim_{\mathcal{I}} [x, x, y, x]$   
**unfolding**  $x$ -def  $y$ -def  $p$ -def  $s$ -def  
**by** (*rule fac-interpI*) *force*+  
**qed**

**lemma** *example-root-joint-xyy-cover*:  
**fixes**  $x \ y \ p \ s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot a \cdot b$  and  $y \equiv a \cdot b \cdot a$  and  
 $p \equiv a \cdot b \cdot a \cdot a \cdot b$  and  $s \equiv b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$   
**and**  $\neg$  *primitive*  $x$   
 $p \cdot x \cdot x \cdot y \cdot y \cdot s \sim_{\mathcal{I}} [x, x, x, y]$   
**proof**-  
**show**  $x \cdot y \neq y \cdot x$   
**unfolding**  $x$ -def  $y$ -def by *simp*  
**show**  $\neg$  *primitive*  $x$   
**unfolding**  $x$ -def by *primitivity-inspection*  
**show**  $p \cdot x \cdot x \cdot y \cdot y \cdot s \sim_{\mathcal{I}} [x, x, x, y]$   
**unfolding**  $x$ -def  $y$ -def  $p$ -def  $s$ -def *concat.simps*  
**by** (*rule fac-interpI*) *force*+  
**qed**

**lemma** *example-root-joint-xyy-cover*:  
**fixes**  $x \ y \ p \ s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot a \cdot b \cdot a$  and  $y \equiv a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b$  and  
 $p \equiv a \cdot b \cdot a$  and  $s \equiv a$   
**shows**  $x \cdot y \neq y \cdot x$   
**and**  $\neg$  *primitive*  $x$   
 $p \cdot x \cdot x \cdot y \cdot s \sim_{\mathcal{I}} [x, x, x, x, x]$   
**proof**-  
**show**  $x \cdot y \neq y \cdot x$   
**unfolding**  $x$ -def  $y$ -def by *simp*  
**show**  $\neg$  *primitive*  $x$   
**unfolding**  $x$ -def by *primitivity-inspection*  
**show**  $p \cdot x \cdot x \cdot y \cdot s \sim_{\mathcal{I}} [x, x, x, x, x]$   
**unfolding**  $x$ -def  $y$ -def  $p$ -def  $s$ -def *concat.simps*  
**by** (*rule fac-interpI*) *force*+  
**qed**

**lemma** *example-root-joint-no-overlap*:  
**fixes**  $x \ y \ p \ s$   
**defines**  $x \equiv a \cdot b \cdot a \cdot a \cdot b \cdot a$  and  $y \equiv a$  and  
 $p \equiv a \cdot b$  and  $s \equiv b \cdot a$   
**shows**  $x \cdot y \neq y \cdot x$   
**and**  $\neg$  *primitive*  $x$   
 $p \cdot y \cdot y \cdot s \sim_{\mathcal{I}} [x]$   
**proof**-

```

show  $x \cdot y \neq y \cdot x$ 
  unfolding  $x$ -def  $y$ -def by simp
show  $\neg$  primitive  $x$ 
  unfolding  $x$ -def by primitivity-inspection
show  $p \ y \cdot y \ s \sim_{\mathcal{I}} [x]$ 
  unfolding  $x$ -def  $y$ -def  $p$ -def  $s$ -def concat.simps
  by (rule fac-interpI) force+
qed

end

```

```

theory Binary-Code-Imprimitive
  imports
    Combinatorics-Words-Graph-Lemma.Glued-Codes
    Binary-Square-Interpretation
begin

```

This theory focuses on the characterization of imprimitive words which are concatenations of copies of two words (forming a binary code). We follow the article [1] (mainly Théorème 2.1 and Lemme 3.1), while substantially optimizing the proof. See also [3] for an earlier result on this question, and [2] for another proof.

## 0.5 General primitivity not preserving codes

```

context code

```

```

begin

```

Two nontrivially conjugate elements generated by a code induce a disjoint interpretation.

**lemma** *shift-disjoint*:

```

assumes  $ws \in \text{lists } \mathcal{C}$  and  $ws' \in \text{lists } \mathcal{C}$  and  $z \notin \langle \mathcal{C} \rangle$  and  $z \cdot \text{concat } ws = \text{concat } ws' \cdot z$ 

```

```

   $us \leq_p ws^{\textcircled{n}}$  and  $vs \leq_p ws'^{\textcircled{n}}$ 

```

```

shows  $z \cdot \text{concat } us \neq \text{concat } vs$ 

```

```

using  $\langle z \notin \langle \mathcal{C} \rangle \rangle$ 

```

```

proof (elim contrapos-nn)

```

```

  assume  $z \cdot \text{concat } us = \text{concat } vs$ 

```

```

  have  $z \neq \varepsilon$ 

```

```

    using  $\langle z \notin \langle \mathcal{C} \rangle \rangle$  by blast

```

```

  obtain  $us'$  where  $ws^{\textcircled{n}} = us \cdot us'$ 

```

```

    using prefixE[OF us ≤p wsⓈn].

```

```

  obtain  $vs'$  where  $ws'^{\textcircled{n}} = vs \cdot vs'$ 

```

```

    using prefixE[OF vs ≤p ws'Ⓢn].

```

```

  from conjug-pow[OF z · concat ws = concat ws' · z [symmetric], symmetric]

```

```

  have  $z \cdot \text{concat } (ws^{\textcircled{n}}) = \text{concat } (ws'^{\textcircled{n}}) \cdot z$ 

```

**unfolding** *concat-pow-list*.  
**from** *this*[ *unfolded*  $\langle ws^{\textcircled{n}} = us \cdot us' \rangle$   $\langle ws^{r\textcircled{n}} = vs \cdot vs' \rangle$  *concat-morph rassoc*  
 $\langle z \cdot \text{concat } us = \text{concat } vs \rangle$  [*symmetric*] *cancel*]  
**have** *concat*  $vs' \cdot z = \text{concat } us'$ .  
**show**  $z \in \langle \mathcal{C} \rangle$   
**proof** (*rule stability*)  
**have**  $us \in \text{lists } \mathcal{C}$  **and**  $us' \in \text{lists } \mathcal{C}$  **and**  $vs \in \text{lists } \mathcal{C}$  **and**  $vs' \in \text{lists } \mathcal{C}$   
**using**  $\langle ws \in \text{lists } \mathcal{C} \rangle$   $\langle ws' \in \text{lists } \mathcal{C} \rangle$   $\langle ws^{r\textcircled{n}} = vs \cdot vs' \rangle$   $\langle ws^{\textcircled{n}} = us \cdot us' \rangle$   
**by** *inlists*  
**thus**  $z \cdot \text{concat } us \in \langle \mathcal{C} \rangle$  **and**  $\text{concat } vs' \in \langle \mathcal{C} \rangle$  **and**  $\text{concat } us \in \langle \mathcal{C} \rangle$  **and** *concat*  
 $vs' \cdot z \in \langle \mathcal{C} \rangle$   
**unfolding**  $\langle \text{concat } vs' \cdot z = \text{concat } us' \rangle$   $\langle z \cdot \text{concat } us = \text{concat } vs \rangle$   
**by** (*simp-all add: concat-in-hull'*)  
**qed**  
**qed**

This in particular yields a disjoint extendable interpretation of any prefix

**lemma** *shift-interp*:

**assumes**  $ws \in \text{lists } \mathcal{C}$  **and**  $ws' \in \text{lists } \mathcal{C}$  **and**  $z \notin \langle \mathcal{C} \rangle$  **and**  
 $\text{conjug: } z \cdot \text{concat } ws = \text{concat } ws' \cdot z$  **and**  $|z| \leq |\text{concat } ws'|$   
**and**  $us \leq_p ws$  **and**  $us \neq \varepsilon$   
**obtains**  $p \ s \ vs \ ps$  **where**  
 $p \ us \ s \sim_{\mathcal{D}} \ vs$  **and**  $vs \in \text{lists } \mathcal{C}$   
**and**  $s \leq_p \text{concat } (us^{-1} \triangleright (ws \cdot ws))$  **and**  $p \leq_s \text{concat } ws$  — extendable  
**and**  $ps \cdot vs \leq_p ws' \cdot ws'$  **and**  $\text{concat } ps \cdot p = z$

**proof**–

**have**  $ws' \cdot ws' \in \text{lists } \mathcal{C}$   
**using**  $\langle ws' \in \text{lists } \mathcal{C} \rangle$  **by** *inlists*  
**have**  $\text{concat } us \neq \varepsilon$   
**using**  $\langle us \neq \varepsilon \rangle$  **unfolding** *concat-eq-emp-conv*[*OF pref-in-lists*[*OF*  $\langle us \leq_p ws \rangle$ ]  
 $\langle ws \in \text{lists } \mathcal{C} \rangle$ ].  
**have**  $|\text{concat } ws'| = |\text{concat } ws|$   
**using** *lenarg*[*OF conjug, unfolded lenmorph*] **by** *linarith*  
**have**  $z \cdot \text{concat } (ws \cdot ws) = \text{concat } (ws' \cdot ws') \cdot z$   
**unfolding** *rassoc concat-morph conjug*[*symmetric*] **unfolding** *lassoc cancel-right*  
**using** *conjug*.  
**hence**  $\text{concat } (ws' \cdot ws') \leq_p z \cdot \text{concat } (ws \cdot ws)$   
**by** *blast*  
**have**  $z \cdot \text{concat } ws \leq_p \text{concat } (ws' \cdot ws')$   
**unfolding** *concat-morph conjug pref-cancel-conv* **using** *eq-le-pref*[*OF conjug*  
 $\langle |z| \leq |\text{concat } ws'| \rangle$ ].  
**from** *prefixE*[*OF pref-shorten*[*OF pref-concat-pref*[*OF*  $\langle us \leq_p ws \rangle$ ] *this*], *unfolded*  
*rassoc*]  
**obtain**  $su$  **where** *fac-u*[*symmetric*]:  $\text{concat } (ws' \cdot ws') = z \cdot \text{concat } us \cdot su$ .

**from** *fac-fac-interpE*[*OF fac-u*  $\langle \text{concat } us \neq \varepsilon \rangle$ ]  
**obtain**  $ps \ ss' \ p \ s \ vs$  **where**  $p \ (\text{concat } us) \ s \sim_{\mathcal{I}} \ vs$  **and**  
 $ps \cdot vs \cdot ss' = ws' \cdot ws'$  **and**  $\text{concat } ps \cdot p = z$  **and**  $s \cdot \text{concat } ss' = su$ .  
**note** *fac-interpD*[*OF*  $\langle p \ (\text{concat } us) \ s \sim_{\mathcal{I}} \ vs \rangle$ ]

```

let ?ss = us-1>(ws · ws)
have us · ?ss = ws · ws
  using <us ≤p ws> by auto

have ps · vs ≤p ws' · ws'
  unfolding <ps · vs · ss' = ws' · ws'>[symmetric] lassoc using triv-pref.

hence vs ∈ lists C
  using <ws' ∈ lists C>
  by inlists

have s ≤p concat ?ss
  using <concat (ws' · ws') ≤p z · concat (ws · ws)>
  unfolding arg-cong[OF <ps · vs · ss' = ws' · ws'>, of concat, symmetric] <concat
ps · p = z>[symmetric]
  arg-cong[OF <us · ?ss = ws · ws>, of concat, symmetric]
  unfolding concat-morph rassoc pref-cancel-conv
  <p · concat us · s = concat vs>[symmetric]
  using append-prefixD by auto

have |p| ≤ |concat ws|
  using <|z| ≤ |concat ws'|>[folded lenarg[OF <concat ps · p = z>], unfolded
<|concat ws'| = |concat ws|>]
  by simp

with eqd[reversed, OF conjug[folded <concat ps · p = z>, unfolded lassoc, sym-
metric] this]
have p ≤s concat ws
  by blast

have disjoint: p · concat us' ≠ concat vs' if us' ≤p us vs' ≤p vs for us' vs'
proof
  have us' ≤p ws · ws
    using <us ≤p ws> <us' ≤p us> by auto
  have ps · vs' ≤p ws' · ws'
    using <vs' ≤p vs> <ps · vs ≤p ws' · ws'> pref-trans same-prefix-prefix by metis
  assume p · concat us' = concat vs'
  hence z · concat us' = concat (ps · vs')
    unfolding concat-morph <concat ps · p = z>[symmetric] rassoc cancel.
  thus False
  using shift-disjoint[OF <ws ∈ lists C> <ws' ∈ lists C> <z ∉ ⟨C⟩>
  <z · concat ws = concat ws' · z> <us' ≤p ws · ws>[folded pow-list-2] <ps · vs'
≤p ws' · ws'>[folded pow-list-2]] by fast
qed
from disjoint[of ε ε]
have p ≠ ε by blast
have s ≠ ε
  using <p · concat us · s = concat vs> disjoint by auto

```

**from** *disj-interpI*[*OF*  $\langle p \text{ (concat } us) s \sim_{\mathcal{I}} vs \rangle$ ] *disjoint*  
**have**  $p \text{ us } s \sim_{\mathcal{D}} vs$   
**by** *blast*

**from** *that*[*OF* *this*  $\langle vs \in \text{lists } \mathcal{C} \rangle$   
 $\langle s \leq p \text{ concat } ?ss \rangle \langle p \leq s \text{ concat } ws \rangle \langle ps \cdot vs \leq p \text{ ws}' \cdot ws' \rangle \langle \text{concat } ps \cdot p = z \rangle$  ]  
**show** *thesis*.  
**qed**

The conditions are in particular met by imprimitivity witnesses

**lemma** *imprim-witness-shift*:

**assumes**  $ws \in \text{lists } \mathcal{C}$  **and** *primitive*  $ws$  **and**  $\neg$  *primitive*  $(\text{concat } ws)$   
**obtains**  $z \ n$  **where**  $\text{concat } ws = z^{\textcircled{a}} n$   $z \notin \langle \mathcal{C} \rangle$  **and**  
 $z \cdot \text{concat } ws = \text{concat } ws \cdot z$  **and**  $|z| < |\text{concat } ws|$  **and**  $2 \leq n$

**proof**–

**have**  $\text{concat } ws \neq \varepsilon$   
**using**  $\langle \text{primitive } ws \rangle$  *emp-concat-emp*[*OF*  $\langle ws \in \text{lists } \mathcal{C} \rangle$ ] *emp-not-prim* **by**  
*blast*

**obtain**  $z \ n$  **where** [*symmetric*]:  $z^{\textcircled{a}} n = \text{concat } ws$  **and**  $2 \leq n$   
**using** *not-prim-primroot-expE*[*OF*  $\langle \neg \text{primitive } (\text{concat } ws) \rangle$ ] **by** *metis*

**hence**  $z \neq \varepsilon$   
**using**  $\langle \text{concat } ws \neq \varepsilon \rangle$  **by** *force*

**have**  $z \notin \langle \mathcal{C} \rangle$

**proof**

**assume**  $z \in \langle \mathcal{C} \rangle$

**then obtain**  $zs$  **where**  $zs \in \text{lists } \mathcal{C}$  **and**  $\text{concat } zs = z$

**using** *hull-concat-lists0* **by** *blast*

**from** *is-code*[*OF*  $\langle ws \in \text{lists } \mathcal{C} \rangle$ ] *pow-in-lists*[*OF*  $\langle zs \in \text{lists } \mathcal{C} \rangle$ ],

*unfolded concat-pow-list*  $\langle \text{concat } ws = z^{\textcircled{a}} n \rangle \langle \text{concat } zs = z \rangle$ , *of*  $n$

**show** *False*

**using**  $\langle \text{primitive } ws \rangle$   $\langle 2 \leq n \rangle$  *pow-nemp-imprim* **by** *blast*

**qed**

**have**  $|z| < |\text{concat } ws|$

**unfolding** *lenarg*[*OF*  $\langle \text{concat } ws = z^{\textcircled{a}} n \rangle$ , *unfolded lenmorph pow-len*]

**using** *nemp-len*[*OF*  $\langle z \neq \varepsilon \rangle$ ]  $\langle 2 \leq n \rangle$  **by** *simp*

**from** *that*[*OF*  $\langle \text{concat } ws = z^{\textcircled{a}} n \rangle \langle z \notin \langle \mathcal{C} \rangle \rangle$  - *this*  $\langle 2 \leq n \rangle$ ]

**show** *thesis*

**unfolding**  $\langle \text{concat } ws = z^{\textcircled{a}} n \rangle$  *pow-comm* **by** *blast*

**qed**

**end**

## 0.6 Covered uniform square

**lemma** *cover-xy-xxx*: **assumes**  $|x| = |y|$  **and**  $p \cdot x \cdot y \cdot s = x \cdot x \cdot x$   
**shows**  $x = y$   
**using** *append-assoc* *assms(1)* *assms(2)* *eq-le-pref* *le-refl* *long-pref* *lq-triv* *prefI*  
*pref-comm-eq'* **by** *metis*

**lemma** *cover-xy-yyy*: **assumes**  $|x| = |y|$  **and**  $eq: p \cdot x \cdot y \cdot s = y \cdot y \cdot y$   
**shows**  $x = y$   
**using** *cover-xy-xxx*[*reversed*, *unfolded rassoc*, *OF*  $\langle |x| = |y| \rangle$ ][*symmetric*] *eq*, *sym-metric*].

**lemma** *cover-xy-xyx*: **assumes**  $|x| = |y|$  **and**  $s \neq \varepsilon$  **and**  $eq: p \cdot x \cdot y \cdot s = x \cdot x \cdot y$   
**shows**  $x = y$

**proof**–

**have**  $|p| < |x|$   
**using** *lenarg*[*OF* *eq*] *nemp-len*[*OF*  $\langle s \neq \varepsilon \rangle$ ] **unfolding** *lenmorph* **by** *linarith*  
**then obtain**  $t$  **where**  $x: x = p \cdot t$  **and**  $t \neq \varepsilon$   
**using** *eqd*[*OF* *eq*] **by** *force*  
**from** *eq*[*unfolded this rassoc cancel*]  
**have**  $p \cdot t = t \cdot p$   
**by** *mismatch*  
**hence**  $x \leq_p t \cdot x$   
**unfolding**  $x$  **by** *auto*  
**from** *eq*[*unfolded x*]  
**have**  $y \leq_p t \cdot y$   
**using**  $\langle p \cdot t = t \cdot p \rangle$   $\langle p \cdot t \cdot y \cdot s = t \cdot p \cdot t \cdot y \rangle$  *pref-cancel'* *suf-marker-per-root*  
*triv-pref* **by** *metis*  
**show**  $x = y$   
**using** *same-len-nemp-root-eq*[*OF*  $\langle x \leq_p t \cdot x \rangle$   
 $\langle y \leq_p t \cdot y \rangle$   $\langle t \neq \varepsilon \rangle$  -  $\langle |x| = |y| \rangle$ ]  $\langle |x| = |y| \rangle$  **by** *force*

**qed**

**lemma** *cover-xy-xyy*: **assumes**  $|x| = |y|$  **and**  $p \neq \varepsilon$  **and**  $eq: p \cdot x \cdot y \cdot s = x \cdot y \cdot y$

**shows**  $x = y$

**using** *cover-xy-xyx*[*reversed*, *unfolded rassoc*, *OF* *assms(1)*][*symmetric*] *assms(2)*  
*eq*].

**lemma** *cover-xy-yyx*: **assumes**  $|x| = |y|$  **and**  $eq: p \cdot x \cdot y \cdot s = y \cdot y \cdot x$

**shows**  $x = y$

**proof**–

**have**  $|p| \leq |y|$   
**using** *lenarg*[*OF* *eq*] **unfolding** *lenmorph* **by** *linarith*  
**then obtain**  $t$  **where**  $y: y = p \cdot t$   
**using** *eqd*[*OF* *eq*] **by** *force*  
**from** *eqd-eq*[*OF* -  $\langle |x| = |y| \rangle$ ][*unfolded y swap-len*[*of p*], *unfolded rassoc*] *eq*[*unfolded this rassoc cancel*]  
**have**  $x: x = t \cdot p$  **by** *blast*

**from**  $eq[unfolding\ x\ y\ rassoc\ cancel]$   
**have**  $p \cdot t = t \cdot p$   
**by** *mismatch*  
**thus**  $x = y$   
**unfolding**  $x\ y..$   
**qed**

**lemma** *cover-xy-yxx*: **assumes**  $|x| = |y|$  **and**  $eq: p \cdot x \cdot y \cdot s = y \cdot x \cdot x$   
**shows**  $x = y$   
**using** *cover-xy-yxx[reversed, unfolded rassoc, OF assms(1)[symmetric] eq]..*

**lemma** *cover-xy-xyx*: **assumes**  $|x| = |y|$  **and**  $p \neq \varepsilon$  **and**  $s \neq \varepsilon$  **and**  $eq: p \cdot x \cdot y \cdot s = x \cdot y \cdot x$   
**shows**  $\neg primitive\ (x \cdot y)$   
**proof**  
**assume**  $primitive\ (x \cdot y)$   
**have**  $p \cdot (x \cdot y) \cdot (s \cdot y) = (x \cdot y) \cdot (x \cdot y)$   
**unfolding** *lassoc eq[unfolding lassoc]..*  
**from** *prim-overlap-sqE[OF <primitive (x · y)> this]*  
**show** *False*  
**using**  $\langle p \neq \varepsilon \rangle \langle s \neq \varepsilon \rangle$  **by** *blast*  
**qed**

**lemma** *cover-xy-yxy*: **assumes**  $|x| = |y|$  **and**  $p \neq \varepsilon$  **and**  $\langle s \neq \varepsilon \rangle$  **and**  $eq: p \cdot x \cdot y \cdot s = y \cdot x \cdot y$   
**shows**  $\neg primitive\ (x \cdot y)$   
**using** *cover-xy-yxy[reversed, unfolded rassoc, OF assms(1)[symmetric] assms(3) assms(2) eq].*

**lemma** *cover-xy-three*: **assumes**  $|ws| = 3$   $ws \in lists\ \{x,y\}$   $|x| = |y|$   
 $p \cdot (x \cdot y) \cdot s = concat\ ws$   $p \neq \varepsilon$   $s \neq \varepsilon$   
**shows**  $\neg primitive\ (x \cdot y) \wedge (ws = [x,y,x] \vee ws = [y,x,y])$   
**proof** (*cases*  $x = y$ )  
**assume**  $x = y$   
**hence**  $\sim primitive\ (x \cdot y)$   
**using** *eq-append-not-prim by blast*  
**moreover** **have**  $ws = [x,y,x] \vee ws = [y,x,y]$   
**using**  $\langle ws \in lists\ \{x,y\} \rangle$  *sing-pow-exp[of ws x, unfolded <|ws| = 3>]*  
**unfolding**  $\langle x = y \rangle$  *pow-list-3 by auto*  
**ultimately** **show** *?thesis*  
**by** *blast*

**next**  
**assume**  $x \neq y$   
**with** *assms*  
**show**  $\neg primitive\ (x \cdot y) \wedge (ws = [x,y,x] \vee ws = [y,x,y])$   
**proof**(*list-inspection, simp-all*)  
**assume**  $p \cdot x \cdot y \cdot s = x \cdot x \cdot x$   
**from** *cover-xy-xxx[OF <|x| = |y|> this]*  
**show** *False*

```

    using  $\langle x \neq y \rangle$  by blast
next
  assume  $p \cdot x \cdot y \cdot s = x \cdot x \cdot y$ 
  from cover-xy-xyx[OF  $\langle |x| = |y| \rangle \langle s \neq \varepsilon \rangle$  this]
  show False
    using  $\langle x \neq y \rangle$  by blast
next
  assume  $p \cdot x \cdot y \cdot s = x \cdot y \cdot x$ 
  from cover-xy-xyx[OF  $\langle |x| = |y| \rangle \langle p \neq \varepsilon \rangle \langle s \neq \varepsilon \rangle$  this]
  show  $\neg$  primitive  $(x \cdot y)$ 
    by blast
next
  assume  $p \cdot x \cdot y \cdot s = x \cdot y \cdot y$ 
  from cover-xy-xyy[OF  $\langle |x| = |y| \rangle \langle p \neq \varepsilon \rangle$  this]
  show False
    using  $\langle x \neq y \rangle$  by blast
next
  assume  $p \cdot x \cdot y \cdot s = y \cdot x \cdot x$ 
  from cover-xy-yxx[OF  $\langle |x| = |y| \rangle$  this]
  show False
    using  $\langle x \neq y \rangle$  by blast
next
  assume  $p \cdot x \cdot y \cdot s = y \cdot x \cdot y$ 
  from cover-xy-yxy[OF  $\langle |x| = |y| \rangle \langle p \neq \varepsilon \rangle \langle s \neq \varepsilon \rangle$  this]
  show  $\neg$  primitive  $(x \cdot y)$ 
    using  $\langle x \neq y \rangle$  by blast
next
  assume  $p \cdot x \cdot y \cdot s = y \cdot y \cdot x$ 
  from cover-xy-yyx[OF  $\langle |x| = |y| \rangle$  this]
  show False
    using  $\langle x \neq y \rangle$  by blast
next
  assume  $p \cdot x \cdot y \cdot s = y \cdot y \cdot y$ 
  from cover-xy-yyy[OF  $\langle |x| = |y| \rangle$  this]
  show False
    using  $\langle x \neq y \rangle$  by blast
qed
qed

```

**lemma bin-uniform-len:** assumes  $ws \in \text{lists } \{x, y\} \ |x| = |y|$   
 shows  $|\text{concat } ws| = |ws| * |x|$   
 using *assms* by (induct *ws*, simp-all) blast

**theorem uniform-square-interp:** assumes  $x \cdot y \neq y \cdot x$  and  $|x| = |y|$  and  $vs \in \text{lists } \{x, y\}$   
 and  $p \ (x \cdot y) \ s \sim_{\mathcal{I}} vs$  and  $p \neq \varepsilon$   
 shows  $\neg$  primitive  $(x \cdot y)$  and  $vs = [x, y, x] \vee vs = [y, x, y]$   
 proof-  
 note *fac-interpD*[OF  $\langle p \ (x \cdot y) \ s \sim_{\mathcal{I}} vs \rangle$ ]

**have**  $vs \neq \varepsilon$   
**using**  $\langle p \cdot (x \cdot y) \cdot s = \text{concat } vs \rangle$  *assms(5)* **by force**  
**have**  $|p| < |x|$   
**using** *prefix-length-less*[*OF*  $\langle p < p \text{ hd } vs \rangle$ ] *lists-hd-in-set*[*OF*  $\langle vs \neq \varepsilon \rangle \langle vs \in \text{lists } \{x,y\} \rangle$ ]  
 $\langle |x| = |y| \rangle$   
**by fastforce**  
**have**  $|s| < |x|$   
**using** *suffix-length-less*[*OF*  $\langle s < s \text{ last } vs \rangle$ ]  $\langle |x| = |y| \rangle$  *lists-hd-in-set*[*reversed*,  
*OF*  $\langle vs \neq \varepsilon \rangle \langle vs \in \text{lists } \{x,y\} \rangle$ ]  
**by fastforce**  
**have**  $|\text{concat } vs| = |x| * |vs|$   
**using** *bin-uniform-len*[*OF* *assms(3,2)*] **by simp**  
**note** *leneq* = *lenarg*[*OF*  $\langle p \cdot (x \cdot y) \cdot s = \text{concat } vs \rangle$ , *unfolded this lenmorph*  $\langle |x|$   
 $= |y| \rangle$  [*symmetric*]]  
**hence**  $2 * |x| < |x| * |vs|$  **and**  $|x| * |vs| < |x| * 4$   
**using**  $\langle |p| < |x| \rangle \langle |s| < |x| \rangle$  *nemp-len*[*OF*  $\langle p \neq \varepsilon \rangle$ ] **by linarith+**  
**hence**  $|vs| = 3$   
**by force**  
**hence**  $s \neq \varepsilon$   
**using** *leneq*  $\langle |p| < |x| \rangle$  **by force**  
  
**show**  $\neg \text{primitive } (x \cdot y) \vee vs = [x,y,x] \vee vs = [y,x,y]$   
**using** *cover-xy-three*[*OF*  $\langle |vs| = 3 \rangle \langle vs \in \text{lists } \{x,y\} \rangle \langle |x| = |y| \rangle \langle p \cdot (x \cdot y) \cdot s$   
 $= \text{concat } vs \rangle \langle p \neq \varepsilon \rangle \langle s \neq \varepsilon \rangle$  ]  
**by blast+**  
**qed**

### 0.6.1 Primitivity (non)preserving uniform binary codes

**theorem** *bin-uniform-prim-morph*:

**assumes**  $x \cdot y \neq y \cdot x$  **and**  $|x| = |y|$  **and** *primitive*  $(x \cdot y)$   
**and**  $ws \in \text{lists } \{x,y\}$  **and**  $2 \leq |ws|$   
**shows** *primitive*  $ws \iff \text{primitive } (\text{concat } ws)$

**proof** (*standard*, *rule ccontr*)

**assume**  $\langle \text{primitive } ws \rangle$  **and**  $\langle \neg \text{primitive } (\text{concat } ws) \rangle$   
**from** *bin-prim-long-pref*[*OF*  $\langle ws \in \text{lists } \{x,y\} \rangle \langle \text{primitive } ws \rangle \langle 2 \leq |ws| \rangle$ ]  
**obtain**  $ws'$  **where**  $ws \sim ws' [x, y] \leq_p ws'$ .  
**have**  $ws' \in \text{lists } \{x,y\}$   
**using** *conjug-in-lists'*[*OF*  $\langle ws \sim ws' \rangle \langle ws \in \text{lists } \{x,y\} \rangle$ ].  
**have** *primitive*  $ws'$   
**using** *prim-conjug*[*OF*  $\langle \text{primitive } ws \rangle \langle ws \sim ws' \rangle$ ].  
**have**  $\neg \text{primitive } (\text{concat } ws')$   
**using** *conjug-concat-prim-iff*  $\langle \neg \text{primitive } (\text{concat } ws) \rangle \langle ws \sim ws' \rangle$  **by auto**  
**interpret** *code*  $\{x,y\}$   
**using** *bin-code-code*[*OF*  $\langle x \cdot y \neq y \cdot x \rangle$ ].

**have**  $[x,y] \neq \varepsilon$  **by blast**

**from** *imprim-witness-shift*[*OF*  $\langle ws' \in \text{lists } \{x, y\} \rangle \langle \text{primitive } ws' \rangle \langle \neg \text{primitive } (\text{concat } ws') \rangle$ ]  
**obtain**  $z \ n$  **where**  $\text{concat } ws' = z^{\textcircled{a}} \ n \ z \notin \{\{x, y\}\} \ z \cdot \text{concat } ws' = \text{concat } ws' \cdot z \ |z| < |\text{concat } ws'|$ .  
**from** *shift-interp*[*OF*  $\langle ws' \in \text{lists } \{x, y\} \rangle \langle ws' \in \text{lists } \{x, y\} \rangle \text{this}(2-3) \text{less-imp-le}[*OF* *this*(4)]  $\langle [x, y] \leq_p ws' \rangle \langle [x, y] \neq \varepsilon \rangle$   
**obtain**  $p \ s \ vs \ ps$  **where**  $p \ [x, y] \ s \sim_{\mathcal{D}} \ vs \ vs \in \text{lists } \{x, y\} \ s \leq_p \text{concat } ([x, y]^{-1} \cdot (ws' \cdot ws'))$   
 $p \leq_s \text{concat } ws' \ ps \cdot vs \leq_p ws' \cdot ws' \text{concat } ps \cdot p = z$ .  
**from** *uniform-square-interp*(1)[*OF*  $\langle x \cdot y \neq y \cdot x \rangle \langle |x| = |y| \rangle \langle vs \in \text{lists } \{x, y\} \rangle -$   
 $-]$   
 $\langle \text{primitive } (x \cdot y) \rangle \text{disj-interpD0}$ [*OF* *this*(1), *simplified*] *disj-interp-nemp*(1)[*OF* *this*(1)]  
**show** *False*  
**by** *blast*  
**qed** (*simp add: prim-concat-prim*)$

— A stronger version is implied by the following lemma.

**lemma** *bin-uniform-imprim*: **assumes**  $x \cdot y \neq y \cdot x$  **and**  $|x| = |y|$  **and**  $\neg \text{primitive } (x \cdot y)$

**shows** *primitive x*

**proof**—

**have**  $x \cdot y \neq \varepsilon$  **and**  $x \neq \varepsilon$  **and**  $y \neq \varepsilon$

**using**  $\langle x \cdot y \neq y \cdot x \rangle$  **by** *blast+*

**from** *not-prim-expE*[*OF*  $\langle \neg \text{primitive } (x \cdot y) \rangle \langle x \cdot y \neq \varepsilon \rangle$ ]

**obtain**  $z \ k$  **where** *primitive z* **and**  $2 \leq k$  **and**  $z^{\textcircled{a}} k = x \cdot y$ .

**hence**  $0 < k$

**by** *simp*

**from** *split-pow*[*OF*  $\langle z^{\textcircled{a}} k = x \cdot y \rangle$  [*symmetric*]  $\langle 0 < k \rangle \langle y \neq \varepsilon \rangle$ ]

**obtain**  $u \ v \ l \ m$  **where** [*symmetric*]:  $z = u \cdot v$  **and**  $v \neq \varepsilon \ x = (u \cdot v)^{\textcircled{a}} l \cdot u \ y = (v \cdot u)^{\textcircled{a}} m \cdot v \ k = l + m + 1$ .

**have**  $u \cdot v \neq v \cdot u$

**using**  $\langle x \cdot y \neq y \cdot x \rangle$  **unfolding**  $\langle x = (u \cdot v)^{\textcircled{a}} l \cdot u \rangle \langle y = (v \cdot u)^{\textcircled{a}} m \cdot v \rangle$

*shifts unfolding pow-add* [*symmetric*] *add.commute* [*of m*] **by** *force*

**have**  $u \neq \varepsilon$  **and**  $v \neq \varepsilon$  **and**  $u \neq v$

**using**  $\langle u \cdot v \neq v \cdot u \rangle$  **by** *blast+*

**have**  $m = l$  **and**  $|u| = |v|$

**using** *almost-equal-equal*[*OF* *nemp-len-not0*[*OF*  $\langle u \neq \varepsilon \rangle$ ] *nemp-len-not0*[*OF*  $\langle v \neq \varepsilon \rangle$ , *of l m*] *lenarg*[*OF*  $\langle x = (u \cdot v)^{\textcircled{a}} l \cdot u \rangle$ , *unfolded*  $\langle |x| = |y| \rangle$ , *unfolded* *lenarg*[*OF*  $\langle y = (v \cdot u)^{\textcircled{a}} m \cdot v \rangle$ ]]

**unfolding** *lenmorph pow-len lenarg*[*OF*  $\langle u \cdot v = z \rangle$ , *symmetric*] **by** *algebra+*

**from**  $\langle k = l + m + 1 \rangle$  [*folded Suc-eq-plus1*, *symmetric*]

**have**  $l \neq 0$

**using**  $\langle 2 \leq k \rangle$  [*folded*  $\langle \text{Suc}(l+m) = k \rangle$ , *unfolded*  $\langle m = l \rangle$ ] **by** *force*

**let**  $?w = [u, v]^{\textcircled{a}} l \cdot [u]$

**have**  $?w \in \text{lists } \{u, v\}$

**by** (*induct l*, *simp-all*)

**have**  $2 \leq |?w|$

**using**  $\langle l \neq 0 \rangle$  **unfolding** *lenmorph pow-len* **by** *fastforce*  
**have** *concat ?w = x*  
**using**  $\langle x = (u \cdot v) @ l \cdot u \rangle$  **by** (*simp add: concat-pow-list*)  
**from** *bin-uniform-prim-morph*[*OF*  $\langle u \cdot v \neq v \cdot u \rangle \langle |u| = |v| \rangle \langle \text{primitive } z \rangle$ ][*folded*  
 $\langle u \cdot v = z \rangle \langle ?w \in \text{lists } \{u, v\} \rangle \langle 2 \leq |?w| \rangle$ ]  
**show** *primitive x*  
**unfolding**  $\langle \text{concat } ?w = x \rangle$  **using** *alternate-prim*[*OF*  $\langle u \neq v \rangle$ ] **by** *blast*  
**qed**

**theorem** *bin-uniform-prim-morph'*:

**assumes**  $x \cdot y \neq y \cdot x$  **and**  $|x| = |y|$  **and** *primitive*  $(x \cdot y) \vee \neg \text{primitive } x \vee \neg$   
*primitive y*  
**and**  $ws \in \text{lists } \{x, y\}$  **and**  $2 \leq |ws|$   
**shows** *primitive ws*  $\longleftrightarrow$  *primitive (concat ws)*  
**using** *bin-uniform-prim-morph*[*OF* *assms(1-2) - assms(4-5)*] *bin-uniform-imprim*[*OF*  
*assms(1-2)*]  
*bin-uniform-imprim*[*OF* *assms(1-2)[symmetric]*, *unfolded conjug-prim-iff'*][*of*  
*y*]  
*assms(3)* **by** *blast*

## 0.7 The main theorem

### 0.7.1 Imprimitive words with single y

If the shorter word occurs only once, the result is straightforward from the parametric solution of the Lyndon-Schutzenberger equation.

**lemma** *bin-imprim-single-y*:

**assumes** *non-comm*:  $x \cdot y \neq y \cdot x$  **and**  
 $ws \in \text{lists } \{x, y\}$  **and**  
 $|y| \leq |x|$  **and**  
 $2 \leq \text{count-list } ws \ x$  **and**  
 $\text{count-list } ws \ y < 2$  **and**  
*primitive ws* **and**  
 $\neg \text{primitive (concat ws)}$   
**shows**  $ws \sim [x, x, y]$  **and** *primitive x* **and** *primitive y*

**proof**–

**have**  $x \neq y$   
**using** *non-comm* **by** *blast*  
**have**  $\text{count-list } ws \ y \neq 0$   
**proof**

**assume**  $\text{count-list } ws \ y = 0$

**with**  $\langle ws \in \text{lists } \{x, y\} \rangle$

**have**  $ws \in \text{lists } \{x\}$

**unfolding** *count-list-0-iff* **by** *blast*

**from** *prim-exp-one*[*OF*  $\langle \text{primitive } ws \rangle$  *this*][*unfolded sing-lists-exp-count*]

**show** *False*

**using**  $\langle 2 \leq \text{count-list } ws \ x \rangle$  **by** *simp*

qed

hence  $\text{count-list ws } y = 1$

using  $\langle \text{count-list ws } y < 2 \rangle$  by *linarith*

from *this bin-count-one-conjug*[*OF*  $\langle ws \in \text{lists } \{x,y\} \rangle$  - *this*]

have  $ws \sim [x]^{\textcircled{a}} \text{count-list ws } x \cdot [y]$

using *non-comm (1)* by *metis*

from *conjug-concat-prim-iff*[*OF this*]

have  $\neg \text{primitive } (x^{\textcircled{a}} (\text{count-list ws } x) \cdot y)$

using  $\langle \neg \text{primitive } (\text{concat ws}) \rangle$  by *simp*

from *not-prim-primroot-expE*[*OF this*]

obtain  $z \ l$  where [*symmetric*]:  $z^{\textcircled{a}} l = x^{\textcircled{a}} (\text{count-list ws } x) \cdot y^{\textcircled{a}} 1$  and  $2 \leq l$

unfolding *pow-list-1*.

interpret *LS-len-le x y count-list ws x 1 l z*

by (*unfold-locales*)

(*use*  $\langle 2 \leq \text{count-list ws } x \rangle \langle x \cdot y \neq y \cdot x \rangle \langle |y| \leq |x| \rangle$

$\langle x^{\textcircled{a}} \text{count-list ws } x \cdot y^{\textcircled{a}} 1 = z^{\textcircled{a}} l \rangle \langle 2 \leq l \rangle$  in *force*)+

from *case-j2k1*[*OF*  $\langle 2 \leq \text{count-list ws } x \rangle$  *refl*]

have *primitive x and primitive y and count-list ws x = 2* by *blast+*

with  $\langle ws \sim [x]^{\textcircled{a}} \text{count-list ws } x \cdot [y] \rangle$  [*unfolded this(3) pow-list-2 append-Cons append-Nil*]

show *primitive x and primitive y and ws ~ [x,x,y]*

by *simp-all*

qed

## 0.7.2 Conjugate words

lemma *bin-imprim-not-conjug*:

assumes  $ws \in \text{lists } \{x,y\}$  and

$x \cdot y \neq y \cdot x$  and

$2 \leq |ws|$  and

*primitive ws* and

$\neg \text{primitive } (\text{concat ws})$

shows  $\neg x \sim y$

proof

assume  $x \sim y$

hence  $|x| = |y|$  by *force*

from *bin-uniform-prim-morph*[*OF*  $\langle x \cdot y \neq y \cdot x \rangle$  *this* -  $\langle ws \in \text{lists } \{x,y\} \rangle \langle 2 \leq |ws| \rangle$ ]

have  $\neg \text{primitive } (x \cdot y)$

using  $\langle \text{primitive ws} \rangle \langle \neg \text{primitive } (\text{concat ws}) \rangle$  by *blast*

```

from Lyndon-Schutzenberger-conjug[OF  $\langle x \sim y \rangle$  this]
show False
  using  $\langle x \cdot y \neq y \cdot x \rangle$  by blast
qed

```

### 0.7.3 Square factor of the longer word and both words primitive (was all\_assms)

The main idea of the proof is as follows: Imprimitivity of the concatenation yields (at least) two overlapping factorizations into  $\{x, y\}$ . Due to the presence of the square  $x \cdot x$ , these two can be synchronized, which yields that the situation coincides with the canonical form.

**lemma** *bin-imprim-primitive*:

```

assumes  $x \cdot y \neq y \cdot x$ 
  and primitive  $x$  and primitive  $y$ 
  and  $|y| \leq |x|$ 
  and  $ws \in \text{lists } \{x, y\}$ 
  and primitive  $ws$  and  $\neg$  primitive (concat  $ws$ )
  and  $[x, x] \leq_f ws \cdot ws$ 
shows  $ws \sim [x, x, y]$ 
proof—
  — Preliminaries
  have  $x \neq y$ 
    using assms(1) by blast
  have  $|ws| \neq 1$ 
    using len-one-concat-in[OF  $\langle ws \in \text{lists } \{x, y\} \rangle$ ]  $\langle \neg$  primitive (concat  $ws$ )  $\rangle$ 
     $\langle$  primitive  $x$   $\rangle$   $\langle$  primitive  $y$   $\rangle$ 
    by blast
  with prim-nemp[OF  $\langle$  primitive  $ws$   $\rangle$ , THEN nemp-le-len]
  have  $2 \leq |ws|$ 
    by auto
  hence  $|[x, x]| \leq |ws|$ 
    by force
  have  $\neg x \sim y$ 
    by (rule bin-imprim-not-conjug) fact+
  have primitive  $[x, x, y]$ 
    using  $\langle x \neq y \rangle$  by primitivity-inspection
  have concat  $[x, x] = x \cdot x$ 
    by simp
  interpret  $xy$ : binary-code  $x y$ 
    using  $\langle x \cdot y \neq y \cdot x \rangle$  by (unfold-locales)

  — Rotate  $ws$  in order to obtain a list with a prefix  $[x \cdot x]$ 
  obtain  $ws'$  where  $ws \sim ws' [x, x] \leq_p ws'$ 

```

**using** *rotate-into-pos-sq*[of  $\varepsilon [x,x]$  - *thesis, unfolded emp-simps*,  $OF \langle [x, x] \leq f$   
 $ws \cdot ws \rangle$

$le0 \langle |[x, x]| \leq |ws| \rangle$  **by** *blast*

**have**  $ws' \in lists \{x,y\}$  **and** *primitive*  $ws'$  **and**  $\neg$  *primitive* ( $concat \ ws'$ )  
**using** *conjug-in-lists'*[ $OF \langle ws \sim ws' \rangle \langle ws \in lists \{x, y\} \rangle$ ]  
*prim-conjug*[ $OF \langle primitive \ ws \rangle \langle ws \sim ws' \rangle$ ]  
 $\langle \neg$  *primitive* ( $concat \ ws \rangle$ )[*unfolded conjug-concat-prim-iff*[ $OF \langle ws \sim ws' \rangle$ ]].  
**have**  $2 \leq |ws'|$  **and**  $[x,x] \neq \varepsilon$  **and**  $ws' \neq \varepsilon$   
**using**  $\langle [x,x] \leq p \ ws' \rangle$  **unfolding** *prefix-def* **by** *auto*  
**have**  $concat \ ws' \neq \varepsilon$   
**using**  $\langle primitive \ x \rangle \langle [x,x] \leq p \ ws' \rangle$  **by** (*fastforce simp add: prefix-def*)  
**have**  $ws' \cdot ws' \cdot ws' \in lists \{x, y\}$  **and**  $ws' \cdot ws' \in lists \{x, y\}$   
**using**  $\langle ws' \in lists \{x,y\} \rangle$  **by** *inlists*

— The core of the proof

**have**  $ws' = [x,x,y]$

**proof**(*rule ccontr*)

**assume**  $ws' \neq [x,x,y]$

**from** *xy.imprim-witness-shift*[ $OF \langle ws' \in lists \{x,y\} \rangle \langle primitive \ ws' \rangle \langle \neg$  *primitive*  
( $concat \ ws' \rangle$ )]

**obtain**  $z \ n$  **where** *con-ws*:  $concat \ ws' = z \text{ }^\circledast \ n$  **and**  $z \notin \langle \{x, y\} \rangle$  **and**  $z \cdot concat$   
 $ws' = concat \ ws' \cdot z$

**and**  $|z| < |concat \ ws'|$  **and**  $2 \leq n$ .

**have**  $0 < n$

**using**  $\langle 2 \leq n \rangle$  **by** *simp*

**from** *xy.shift-interp*[ $OF \langle ws' \in lists \{x,y\} \rangle \langle ws' \in lists \{x,y\} \rangle \langle z \notin \langle \{x, y\} \rangle \rangle \langle z$   
 $\cdot concat \ ws' = concat \ ws' \cdot z \rangle$

*less-imp-le*[ $OF \langle |z| < |concat \ ws'| \rangle \langle [x,x] \leq p \ ws' \rangle \langle [x,x] \neq \varepsilon \rangle$ ]

**obtain**  $p \ s \ vs \ ps$  **where** *dis*:  $p [x,x] \ s \sim_{\mathcal{D}} \ vs$  **and**  $\langle vs \in lists \{x, y\} \rangle$  **and**

$s \leq p \ concat \ ([x,x]^{-1} \langle ws' \cdot ws' \rangle)$  **and**  $p \leq s \ concat \ ws'$  **and**  $ps \cdot vs \leq p \ ws' \cdot ws'$

**and**  $concat \ ps \cdot p = z$ .

**from** *disj-interp-nemp(1)*[ $OF \ this(1)$ ]

**have**  $p \neq \varepsilon$  **by** *simp*

**have**  $p \cdot concat \ p1 \neq concat \ p2$  **if**  $p1 \leq p [x, x]$  **and**  $p2 \leq p \ vs$  **for**  $p1 \ p2$

**using**  $\langle p [x,x] \ s \sim_{\mathcal{D}} \ vs \rangle$  *disj-interpD1* **that** **by** *blast*

**have**  $ps \in lists \{x,y\}$

**using**  $\langle ps \cdot vs \leq p \ ws' \cdot ws' \rangle \langle ws' \in lists \{x,y\} \rangle \langle ws' \cdot ws' \in lists \{x, y\} \rangle$

*append-prefixD* *pref-in-lists* **by** *metis*

**have**  $vs \in lists \{x,y\}$

**using**  $\langle ws' \in lists \{x,y\} \rangle$  *pref-in-lists*[ $OF \langle ps \cdot vs \leq p \ ws' \cdot ws' \rangle$ ] **by** *inlists*

**have**  $[x,x]^{-1} \langle ws' \cdot ws' \rangle \in lists \{x,y\}$

**using**  $\langle ws' \in lists \{x,y\} \rangle$  **by** *inlists*

**have**  $p \ x \cdot x \ s \sim_{\mathcal{I}} \ vs$

**using** *disj-interpD0*[ $OF \langle p [x,x] \ s \sim_{\mathcal{D}} \ vs \rangle$ ] **by** *simp*

```

interpret square-interp-ext x y p s vs
proof (rule square-interp-ext.intro[OF square-interp.intro, unfolded square-interp-ext-axioms-def])
  show  $(\exists pe. pe \in \langle \{x, y\} \rangle \wedge p \leq_s pe) \wedge (\exists se. se \in \langle \{x, y\} \rangle \wedge s \leq_p se)$ 
    using  $\langle s \leq_p \text{concat} ([x,x]^{-1} \langle ws' \cdot ws' \rangle) \rangle \langle p \leq_s \text{concat} ws' \rangle$ 
       $\langle [x, x]^{-1} \langle ws' \cdot ws' \rangle \in \text{lists } \{x, y\} \rangle \langle ws' \in \text{lists } \{x, y\} \rangle$  concat-in-hull' by
meson
  show  $\neg \varrho x \sim \varrho y$ 
    using  $\langle \neg x \sim y \rangle$  unfolding prim-primroot[OF  $\langle \text{primitive } y \rangle$ ] prim-primroot[OF
 $\langle \text{primitive } x \rangle$ ].
  show  $p \text{ Ref } \{ \varrho x, y \} [x, x] s \sim_{\mathcal{D}} vs$ 
    unfolding prim-primroot[OF  $\langle \text{primitive } x \rangle$ ] using dis xy.code-ref-list[of
 $[x,x]$ ] by force
  qed fact+

```

— Establishing the connection between  $ws' = [x,x,y]$  and  $z = xp$ .

```

define xp where xp = x · p

have concat [x,x,y] = xp · xp
  by (simp add: xxy-root xp-def)

hence  $ws' \cdot [x,x,y] \neq [x,x,y] \cdot ws'$ 
  using comm-prim[OF  $\langle \text{primitive } ws' \rangle \langle \text{primitive } [x,x,y] \rangle \langle ws' \neq [x,x,y] \rangle$ ] by
force

```

```

have  $z \cdot xp \neq xp \cdot z$ 
proof
  assume  $z \cdot xp = xp \cdot z$ 
  from comm-pow-comm[symmetric, OF this[symmetric], of 2,
    THEN comm-pow-comm, of n, unfolded pow-list-2]
  have  $z^{\textcircled{n}} \cdot xp \cdot xp = xp \cdot xp \cdot z^{\textcircled{n}}$ 
    unfolding rassoc.
  hence concat  $ws' \cdot \text{concat } [x,x,y] = \text{concat } [x,x,y] \cdot \text{concat } ws'$ 
    unfolding con-ws  $\langle \text{concat } [x,x,y] = xp \cdot xp \rangle$  rassoc by simp
  from xy.is-code[OF - - this[folded concat-morph]]
  have  $ws' \cdot [x, x, y] = [x,x,y] \cdot ws'$ 
    using append-in-lists  $\langle ws' \in \text{lists } \{x,y\} \rangle$  by simp
  thus False
  using  $\langle ws' \cdot [x, x, y] \neq [x,x,y] \cdot ws' \rangle$  by fastforce
qed

```

```

then interpret binary-code z xp
  by (unfold-locales)

```

```

have  $\neg \text{concat } (ws' \cdot [x, x, y]) \bowtie \text{concat } ([x, x, y] \cdot ws')$ 
proof (rule notI)
  assume  $\text{concat } (ws' \cdot [x, x, y]) \bowtie \text{concat } ([x, x, y] \cdot ws')$ 
  from comm-comp-eq[OF this[unfolded concat-morph], unfolded  $\langle \text{concat } [x,x,y] \rangle$ ]

```

```

=  $xp \cdot xp \rangle \text{con-}ws]$ 
  have  $z \text{ }^{\textcircled{a}} n \cdot xp \text{ }^{\textcircled{a}} \text{Suc}(\text{Suc } 0) = xp \text{ }^{\textcircled{a}} \text{Suc}(\text{Suc } 0) \cdot z \text{ }^{\textcircled{a}} n$ 
    unfolding pow-Suc pow-zero emp-simps rassoc.
  from comm-drop-exps[OF - - this]
  show False
    using  $\langle z \cdot xp \neq xp \cdot z \rangle \langle 2 \leq n \rangle$  by force
qed

```

- How the  $xp/z$  mismatch is reflected by mismatch in lists  $x,y$ ?
- Looking at the first occurrence of  $z$ :

```

define lcp-ws where  $lcp-ws = ws' \cdot [x,x,y] \wedge_p [x,x,y] \cdot ws'$ 

have  $lcp-ws \in \text{lists } \{x,y\}$ 
  unfolding lcp-ws-def by inlists

have  $lcp-xp-z: \text{concat } (ws' \cdot [x,x,y]) \wedge_p \text{concat } ([x,x,y] \cdot ws') = \text{bin-lcp } z (x \cdot p)$ 
  unfolding concat-morph con-ws  $\langle \text{concat } [x,x,y] = xp \cdot xp \rangle$  pow-add[symmetric]
  using bin-lcp-pows[OF  $\langle 0 < n \rangle$ , of 2]
  unfolding pow-list-2 pow-pos[OF  $\langle 0 < n \rangle$ ] rassoc xp-def by force

have  $(\text{concat } lcp-ws) \cdot \text{bin-lcp } x y = \text{bin-lcp } z (x \cdot p)$ 
proof (rule xy.bin-code-lcp-concat'  $[OF - - \langle \neg \text{concat } (ws' \cdot [x, x, y]) \bowtie \text{concat } ([x, x, y] \cdot ws') \rangle$ , folded lcp-ws-def, unfolded lcp-xp-z, symmetric])
  show  $ws' \cdot [x, x, y] \in \text{lists } \{x, y\}$  and  $[x, x, y] \cdot ws' \in \text{lists } \{x, y\}$ 
    by inlists
qed

```

- Looking at the second occurrence of  $z$ :

```

define  $ws''$  where  $ws'' = ps \cdot [x,y]$ 
define  $lcp-ws'$  where  $lcp-ws' = ws' \cdot ws'' \wedge_p ws'' \cdot ws'$ 

have  $lcp-ws' \in \text{lists } \{x,y\}$ 
  unfolding lcp-ws'-def
  using  $\langle ps \in \text{lists } \{x, y\} \rangle \langle ws' \in \text{lists } \{x, y\} \rangle$  ws''-def by inlists

have  $\text{concat } ws'' = z \cdot xp$ 
  unfolding ws''-def xp-def using  $\langle \text{concat } ps \cdot p = z \rangle$  xy-root by fastforce

have  $ws' \cdot ws'' \neq ws'' \cdot ws'$ 
proof
  assume  $ws' \cdot ws'' = ws'' \cdot ws'$ 
  from arg-cong[OF this, of concat, unfolded concat-morph con-ws
     $\langle \text{concat } ws'' = z \cdot xp \rangle$ ,
    unfolded lassoc pow-comm, unfolded rassoc cancel]
  show False
    using  $\langle z \cdot xp \neq xp \cdot z \rangle$  comm-drop-exp'  $[OF - \langle 0 < n \rangle]$  by blast
qed

```

**have**  
 $lcp\text{-}xp\text{-}z'$ :  $concat (ws' \cdot ws'') \wedge_p concat (ws'' \cdot ws') = z \cdot bin\text{-}lcp\ z (x \cdot p)$   
**unfolding**  $concat\text{-}morph\ con\text{-}ws \langle concat\ ws'' = z \cdot xp \rangle\ pow\text{-}Suc$   
**unfolding**  $lcp\text{-}ext\text{-}left[symmetric]$   $bin\text{-}lcp\text{-}def\ shifts$   
**unfolding**  $rassoc\ lcp\text{-}ext\text{-}left\ cancel$   
**using**  $bin\text{-}lcp\text{-}pows[OF \langle 0 < n \rangle, of\ 1\ \varepsilon\ z^{\otimes}(n-1),\ unfolded\ pow\text{-}list\text{-}1,\ folded\ pow\text{-}pos[OF \langle 0 < n \rangle]]$   
**unfolding**  $bin\text{-}lcp\text{-}def\ xp\text{-}def\ rassoc\ emp\text{-}simps$  **by**  $linarith$

**have**  $z \cdot bin\text{-}lcp\ z (x \cdot p) = concat (lcp\text{-}ws') \cdot bin\text{-}lcp\ x\ y$   
**unfolding**  $lcp\text{-}xp\text{-}z'[symmetric]$   $lcp\text{-}ws'\text{-}def$   
**proof** ( $rule\ xy.\ bin\text{-}code\text{-}lcp\text{-}concat'$ )  
**show**  $ws' \cdot ws'' \in lists\ \{x,\ y\}$   
**unfolding**  $ws''\text{-}def$  **using**  $\langle ws' \cdot ws' \cdot ws' \in lists\ \{x,\ y\} \rangle \langle ps \in lists\ \{x,\ y\} \rangle$   
**by**  $inlists$   
**thus**  $ws'' \cdot ws' \in lists\ \{x,\ y\}$   
**by**  $inlists$   
**show**  $\neg concat (ws' \cdot ws'') \bowtie concat (ws'' \cdot ws')$   
**unfolding**  $concat\text{-}morph\ con\text{-}ws \langle concat\ ws'' = z \cdot xp \rangle\ pow\text{-}pos[OF \langle 0 < n \rangle]$   
**unfolding**  $rassoc\ comp\text{-}cancel$   
**unfolding**  $lassoc\ pow\text{-}pos[OF \langle 0 < n \rangle, symmetric]$   $pow\text{-}pos2[OF \langle 0 < n \rangle, symmetric]$   
 $comm\text{-}comp\text{-}eq\text{-}conv$   
**using**  $comm\text{-}drop\text{-}exp'[OF\ \langle 0 < n \rangle, of\ n\ z\ xp]$   $non\text{-}comm$  **by**  $argo$   
**qed**

**have**  $concat\ lcp\text{-}ws' = z \cdot concat\ lcp\text{-}ws$   
**unfolding**  $cancel\text{-}right[of\ concat\ lcp\text{-}ws' bin\text{-}lcp\ x\ y\ z \cdot concat\ lcp\text{-}ws, symmetric]$   
**unfolding**  $rassoc[of\ z] \langle concat (lcp\text{-}ws) \cdot bin\text{-}lcp\ x\ y = bin\text{-}lcp\ z (x \cdot p) \rangle \langle z \cdot bin\text{-}lcp\ z (x \cdot p) = concat (lcp\text{-}ws') \cdot bin\text{-}lcp\ x\ y \dots$

**have**  $lcp\text{-}ws \leq_p ws' \cdot [x, x, y]$   
**unfolding**  $lcp\text{-}ws\text{-}def$  **using**  $longest\text{-}common\text{-}prefix\text{-}prefix1$ .  
**have**  $lcp\text{-}ws \neq ws' \cdot [x, x, y]$   
**unfolding**  $lcp\text{-}ws\text{-}def\ lcp\text{-}pref\text{-}conv$   
**using**  $\langle ws' \cdot [x, x, y] \neq [x, x, y] \cdot ws' \rangle\ pref\text{-}comm\text{-}eq$  **by**  $blast$   
**have**  $lcp\text{-}ws \leq_p ws' \cdot [x, x]$   
**using**  $spref\text{-}butlast\text{-}pref[OF \langle lcp\text{-}ws \leq_p ws' \cdot [x, x, y] \rangle \langle lcp\text{-}ws \neq ws' \cdot [x, x, y] \rangle]$   
**unfolding**  $butlast\text{-}append$  **by**  $simp$   
**from**  $prefixE[OF\ pref\text{-}prolong[OF\ this\ \langle [x, x] \leq_p ws' \rangle]]$   
**obtain**  $ws''_1$  **where**  $ws' \cdot ws' \cdot ws' = lcp\text{-}ws \cdot ws''_1$  **using**  $rassoc$  **by**  $metis$

**have**  $ws' \cdot ps \cdot [x, y] \leq_p ws' \cdot ps \cdot [x, y, x]$   
**by**  $simp$   
**from**  $pref\text{-}trans[OF\ pref\text{-}trans[OF\ longest\text{-}common\text{-}prefix\text{-}prefix1\ this]]$   
**have**  $lcp\text{-}ws' \leq_p ws' \cdot ws' \cdot ws'$   
**unfolding**  $lcp\text{-}ws'\text{-}def\ ws''\text{-}def$  **using**  $\langle ps \cdot ws \leq_p ws' \cdot ws' \rangle [unfolded\ cover\text{-}xyx, unfolded\ pref\text{-}cancel\text{-}conv]$

**unfolding** *pref-cancel-conv*[*symmetric, of ps · [x,y,x]* *ws' · ws' ws'*] **by** *blast*  
**from** *prefixE*[*OF this*]

**obtain**  $ws''_2$  **where**  $ws' · ws' · ws' = lcp-ws' · ws''_2$ .

**have**  $concat\ lcp-ws' · concat\ ws''_1 = z · concat(lcp-ws) · concat\ ws''_1$

**unfolding** *lassoc*  $\langle concat\ lcp-ws' = z · concat\ lcp-ws \rangle ..$

**also have**  $... = z · concat\ (ws' · ws' · ws')$

**unfolding** *rassoc*  $\langle ws' · ws' · ws' = lcp-ws · ws''_1 \rangle concat-morph ..$

**also have**  $... = concat\ (ws' · ws' · ws') · z$

**unfolding** *concat-morph con-ws pow-add*[*symmetric*]

*pow-Suc*[*symmetric*] *pow-Suc2*[*symmetric*] ..

**also have**  $... = concat\ lcp-ws' · concat\ ws''_2 · z$

**unfolding**  $\langle ws' · ws' · ws' = lcp-ws' · ws''_2 \rangle concat-morph\ rassoc ..$

**finally have**  $concat\ ws''_1 = concat\ ws''_2 · z$

**unfolding** *cancel*.

**from** *xy.stability*[*of concat ws''\_2 concat lcp-ws z,*

*folded*  $\langle concat\ ws''_1 = concat\ ws''_2 · z \rangle \langle concat\ lcp-ws' = z · concat\ lcp-ws \rangle]$

**have**  $z \in \langle \{x, y\} \rangle$

**using**  $\langle ws' · ws' · ws' = lcp-ws · ws''_1 \rangle \langle ws' · ws' · ws' = lcp-ws' · ws''_2 \rangle \langle ws' · ws' · ws' \in lists\ \{x, y\} \rangle$

*append-in-lists-dest append-in-lists-dest' concat-in-hull'* **by** *metis*

**thus** *False*

**using**  $\langle z \notin \langle \{x, y\} \rangle \rangle$  **by** *blast*

**qed**

**thus**  $ws \sim [x, x, y]$

**using**  $\langle ws \sim ws' \rangle$  **by** *blast*

**qed**

#### 0.7.4 Obtaining primitivity with two squares (refining)

**lemma** *bin-imprim-both-squares-prim*:

**assumes**  $x · y \neq y · x$

**and**  $ws \in lists\ \{x, y\}$

**and** *primitive ws* **and**  $\neg primitive\ (concat\ ws)$

**and**  $[x, x] \leq_f ws · ws$

**and**  $[y, y] \leq_f ws · ws$

**and** *primitive x* **and** *primitive y*

**shows** *False*

**proof**–

**have**  $x \neq y$  **using**  $\langle x · y \neq y · x \rangle$

**by** *blast*

**from** *bin-imprim-primitive*[*OF*  $\langle x · y \neq y · x \rangle \langle primitive\ x \rangle \langle primitive\ y \rangle$

$\langle ws \in lists\ \{x, y\} \rangle \langle primitive\ ws \rangle \langle \neg primitive\ (concat\ ws) \rangle \langle [x, x] \leq_f ws · ws \rangle]$

*bin-imprim-primitive*[*OF*  $\langle x · y \neq y · x \rangle$ [*symmetric*]  $\langle primitive\ y \rangle \langle primitive\ x \rangle$

$\langle ws \in lists\ \{x, y\} \rangle$ [*unfolded insert-commute*[*of x*]]  $\langle primitive\ ws \rangle \langle \neg primitive\ (concat\ ws) \rangle$

$\langle [y, y] \leq_f ws · ws \rangle]$

```

have  $ws \sim [x, x, y] \vee ws \sim [y, y, x]$ 
  using  $\langle x \cdot y \neq y \cdot x \rangle$ 
  by force
hence  $|ws| = 3$ 
  using conjug-len by force
note $[simp] = sublist-code(3)$ 
from  $\langle |ws| = 3 \rangle \langle ws \in lists\ \{x,y\} \rangle \langle x \neq y \rangle$ 
   $\langle [x, x] \leq_f ws \cdot ws \rangle \langle [y, y] \leq_f ws \cdot ws \rangle$ 
show False
  by list-inspection simp-all
qed

```

**lemma** *bin-imprim-both-squares*:

```

assumes  $x \cdot y \neq y \cdot x$ 
  and  $ws \in lists\ \{x, y\}$ 
  and primitive  $ws$  and  $\neg primitive\ (concat\ ws)$ 
  and  $[x, x] \leq_f ws \cdot ws$ 
  and  $[y, y] \leq_f ws \cdot ws$ 
shows False
proof (rule bin-imprim-both-squares-prim)
  have  $x \neq \varepsilon$  and  $y \neq \varepsilon$  and  $x \neq y$ 
    using  $\langle x \cdot y \neq y \cdot x \rangle$  by blast+
  let  $?R = \lambda x. [x]^\circledast (e_\circledast x)$ 
  define  $ws'$  where  $ws' = Ref_\circledast ws$ 
  show  $\circledast x \cdot \circledast y \neq \circledast y \cdot \circledast x$ 
    using  $\langle x \cdot y \neq y \cdot x \rangle [unfolded\ comm-primroot-conv'[of\ x\ y]]$ .
  have  $[simp]: a = x \vee a = y \implies [a]^\circledast e_\circledast a \in lists\ \{\circledast x, \circledast y\}$  for  $a$ 
    using insert-iff sing-pow-lists $[of\ -\ \{\circledast x, \circledast y\}]$  by metis
  show  $ws' \in lists\ \{\circledast x, \circledast y\}$ 
    unfolding  $ws'-def\ root-ref-def$  using  $\langle ws \in lists\ \{x,y\} \rangle$ 
    by (induction ws, simp-all)

```

— The primitivity of  $ws'$  is obtained from the fact that the decompositions into roots is a primitive morphism

```

interpret binary-code  $x\ y$ 
  using  $\langle x \cdot y \neq y \cdot x \rangle$  by unfold-locales
note $[simp] = sublist-code(3)$ 
have  $|ws| \leq 3 \implies ws \in lists\ \{x,y\} \implies x \neq y \implies [x, x] \leq_f ws \cdot ws \implies [y, y] \leq_f ws \cdot ws \implies False$ 
  by list-inspection simp-all
from  $this[OF\ -\ \langle ws \in lists\ \{x,y\} \rangle \langle x \neq y \rangle \langle [x, x] \leq_f ws \cdot ws \rangle \langle [y, y] \leq_f ws \cdot ws \rangle]$ 
  roots-prim-morph $[OF\ \langle ws \in lists\ \{x,y\} \rangle - \langle primitive\ ws \rangle]$ 
show primitive  $ws'$ 
  unfolding  $ws'-def\ root-ref-def$  by fastforce

show  $\neg primitive\ (concat\ ws')$ 
  unfolding  $ws'-def\ root-ref-refines$  by fact

```

```

have  $Ref_\circledast [x,x] \leq_f ws' \cdot ws'$  and  $Ref_\circledast [y,y] \leq_f ws' \cdot ws'$ 

```

**unfolding** *ws'-def root-ref-def*  
**using** *concat-mono-fac*[*OF map-mono-sublist*[*OF*  $\langle [x,x] \leq f \text{ ws} \cdot \text{ws} \rangle$ ]]  
*concat-mono-fac*[*OF map-mono-sublist*[*OF*  $\langle [y,y] \leq f \text{ ws} \cdot \text{ws} \rangle$ ]]  
**unfolding** *concat-morph map-append*.

**have**  $\text{Suc} (\text{Suc} (e_\rho x + e_\rho x - 2)) = e_\rho x + e_\rho x$   
**using** *Suc-minus2 primroot-exp-nemp*[*OF*  $\langle x \neq \varepsilon \rangle$ ] **by** *simp*  
**have**  $\text{Ref}_\rho [x,x] = [\rho x]^\textcircled{\text{R}} (\text{Suc} (e_\rho x - 1) + \text{Suc} (e_\rho x - 1))$   
**unfolding** *Suc-minus-pos*[*OF primroot-exp-nemp*[*OF*  $\langle x \neq \varepsilon \rangle$ ]]  
*root-ref-def* **by** (*simp add: pow-add*)  
**hence**  $[\rho x, \rho x] \leq f \text{Ref}_\rho [x,x]$   
**by** *auto*  
**thus**  $[\rho x, \rho x] \leq f \text{ws}' \cdot \text{ws}'$   
**using** *fac-trans*[*OF* -  $\langle \text{Ref}_\rho [x,x] \leq f \text{ws}' \cdot \text{ws}' \rangle$ ] **by** *blast*

**have**  $\text{Suc} (\text{Suc} (e_\rho y + e_\rho y - 2)) = e_\rho y + e_\rho y$   
**using** *Suc-minus2 primroot-exp-nemp*[*OF*  $\langle y \neq \varepsilon \rangle$ ] **by** *simp*  
**have**  $\text{Ref}_\rho [y,y] = [\rho y]^\textcircled{\text{R}} (\text{Suc} (e_\rho y - 1) + \text{Suc} (e_\rho y - 1))$   
**unfolding** *Suc-minus-pos*[*OF primroot-exp-nemp*[*OF*  $\langle y \neq \varepsilon \rangle$ ]] *root-ref-def*  
**by** (*simp add: pow-add*)  
**hence**  $[\rho y, \rho y] \leq f \text{Ref}_\rho [y,y]$   
**by** *auto*  
**thus**  $[\rho y, \rho y] \leq f \text{ws}' \cdot \text{ws}'$   
**using** *fac-trans*[*OF* -  $\langle \text{Ref}_\rho [y,y] \leq f \text{ws}' \cdot \text{ws}' \rangle$ ] **by** *blast*

**show** *primitive* ( $\rho x$ ) **and** *primitive* ( $\rho y$ )  
**using** *primroot-prim*  $\langle x \neq \varepsilon \rangle \langle y \neq \varepsilon \rangle$  **by** *blast+*  
**qed**

### 0.7.5 Obtaining the square of the longer word (gluing)

**lemma** *bin-imprim-longer-twice*:

— 1. If there are both squares, then contradiction; 2. If a square is missing: a) if  $y$  appears once: the positive conclusion b) if  $y$  appears twice, then gluing preserves presence of the longer word at least twice (because both appear twice) and induction yields  $[x',x',y']$  where  $y'$  is a suffix of  $x'$ , a contradiction with primitivity of words of the form  $xyxy$ ;

**assumes**  $x \cdot y \neq y \cdot x$   
**and**  $\text{ws} \in \text{lists } \{x, y\}$   
**and**  $|y| \leq |x|$   
**and** *count-list*  $\text{ws } x \geq 2$   
**and** *primitive*  $\text{ws}$  **and**  $\neg \text{primitive} (\text{concat } \text{ws})$   
**shows**  $\text{ws} \sim [x,x,y] \wedge \text{primitive } x \wedge \text{primitive } y$   
**using** *assms proof* (*induction*  $|\text{ws}|$  *arbitrary: x y ws rule: less-induct*)  
**case** *less*  
**then show** *?case*  
**proof** (*cases*)  
**assume**  $[x, x] \leq f \text{ws} \cdot \text{ws} \wedge [y, y] \leq f \text{ws} \cdot \text{ws}$   
**with** *bin-imprim-both-squares*[*OF*  $\langle x \cdot y \neq y \cdot x \rangle \langle \text{ws} \in \text{lists } \{x,y\} \rangle \langle \text{primitive} \text{ws} \rangle$ ]

```

ws › ⟨ ¬ primitive (concat ws) ›
  have False by blast
  thus ?case by blast
next
assume missing-sq: ¬ ([x, x] ≤f ws · ws ∧ [y, y] ≤f ws · ws)
then show ?case
proof (cases)
  assume count-list ws y < 2
  with bin-imprim-single-y[OF less.prem(1-4) this less.prem(5-6)]
  show ws ~ [x,x,y] ∧ primitive x ∧ primitive y
    by blast
next
assume ¬ count-list ws y < 2 hence 2 ≤ count-list ws y by simp

```

— Missing square and two y's allow gluing

```

define x' where x' = (if ¬ [x, x] ≤f ws · ws then x else y)
define y' where y' = (if ¬ [x, x] ≤f ws · ws then y else x)

have {x', y'} = {x, y}
  by (simp add: doubleton-eq-iff x'-def y'-def)
note cases = disjE[OF this[unfolded doubleton-eq-iff]]
have ¬ [x', x'] ≤f ws · ws
  using missing-sq x'-def by presburger
have count-list ws x' ≥ 2 and count-list ws y' ≥ 2
  unfolding x'-def y'-def using ⟨2 ≤ count-list ws x⟩ ⟨2 ≤ count-list ws y⟩
by presburger+
have x' · y' ≠ y' · x'
  by (rule cases, simp-all add: ⟨x · y ≠ y · x⟩ ⟨x · y ≠ y · x⟩[symmetric])
have x' ≠ ε and x' ≠ y' and x' · y' ≠ y'
  using ⟨x' · y' ≠ y' · x'⟩ by auto

```

— rotating last if necessary for successful gluing

```

note prim-nemp[OF ⟨primitive ws⟩]
hence rot: last ws = x' ⟹ hd ws = x' ⟹ butlast ws · [x',x'] · tl ws = ws ·
ws
  using append-butlast-last-id hd-tl hd-word rassoc by metis
from this[THEN fact']
have last ws = x' ⟹ hd ws ≠ x'
  using ⟨¬ [x', x'] ≤f ws · ws⟩ by blast
define ws' where ws' = (if last ws ≠ x' then ws else tl ws · [hd ws])
have cond: ws' = ε ∨ last ws' ≠ x' — gluing condition
  unfolding ws'-def using ⟨last ws = x' ⟹ hd ws ≠ x'⟩ by simp
have ws' ~ ws
  unfolding ws'-def using ⟨ws ≠ ε⟩ by fastforce
hence counts': count-list ws' x' ≥ 2 count-list ws' y' ≥ 2
by (simp-all add: ⟨2 ≤ count-list ws x'⟩ ⟨2 ≤ count-list ws y'⟩ count-list-conjug)

```

— verify induction assumptions of the glued word

```

let ?ws = glue x' ws'

```

```

have c1: |?ws| < |ws|
  using len-glue[OF cond] conjug-len[OF <ws' ~ ws>] <count-list ws' x' ≥ 2>
by linarith
hence c2: (x' · y') · y' ≠ y' · x' · y'
  using <x' · y' ≠ y' · x'> by force

have ws' ≤f ws · ws
  using conjugE[OF <ws' ~ ws>] rassoc sublist-appendI by metis
hence ¬ [x', x'] ≤f ws'
  using <¬ [x', x'] ≤f ws · ws> by blast
have ws' ∈ lists {x', y'}
  using conjug-in-lists[OF <ws' ~ ws>] <ws ∈ lists {x, y}> [folded <{x', y'} =
{x, y}>]].
have c3: ?ws ∈ lists {x' · y', y'}
  using single-bin-glue-in-lists[OF cond] <¬ [x', x'] ≤f ws'> <ws' ∈ lists {x', y'}>.

have c4: 2 ≤ count-list (glue x' ws') (x' · y')
  using <2 ≤ count-list ws' x'>
  unfolding count-list-single-bin-glue(1)[OF <x' ≠ ε>] <x' ≠ y'> cond <ws' ∈
lists {x', y'}> <¬ [x', x'] ≤f ws'>.

from <primitive ws> [folded conjug-prim-iff[OF <ws' ~ ws>]]
have c5: primitive (glue x' ws')
  using prim-bin-glue [OF <ws' ∈ lists {x', y'}>] <x' ≠ ε> cond] by blast

have count-list ws' x' ≥ 2
  using <count-list ws x ≥ 2> <count-list ws y ≥ 2> <{x', y'} = {x, y}>
  count-list-conjug[OF <ws' ~ ws>] x'-def by metis

have concat (glue x' ws') = concat ws'
  by (simp add: cond)
have c6: ¬ primitive (concat (glue x' ws'))
  unfolding <concat (glue x' ws') = concat ws'> using <¬ primitive (concat
ws)> <ws' ~ ws>
  conjug-concat-conjug prim-conjug by metis
  — The claim holds by induction
from less.hyps[OF c1 c2 c3 - c4 c5 c6]
have glue x' ws' ~ [x' · y', x' · y', y'] by simp
  — Which is impossible after gluing
from prim-xyxyy[OF <x' · y' ≠ y' · x'>] conjug-prim-iff[OF conjug-concat-conjug[OF
this]]
have False
  using <¬ primitive (concat (glue x' ws'))> by simp
thus ?case by blast
qed
qed
qed

```

lemma bin-imprim-both-twice:

```

assumes  $x \cdot y \neq y \cdot x$ 
and  $ws \in \text{lists } \{x, y\}$ 
and  $\text{count-list } ws \ x \geq 2$ 
and  $\text{count-list } ws \ y \geq 2$ 
and  $\text{primitive } ws$  and  $\neg \text{primitive } (\text{concat } ws)$ 
shows False
proof-
have  $x \neq y$ 
using  $\langle x \cdot y \neq y \cdot x \rangle$  by blast
from  $\text{bin-imprim-longer-twice}[OF \text{ assms}(1-2) - \text{assms}(3) \text{ assms}(5-6)]$ 
 $\text{bin-imprim-longer-twice}[OF \text{ assms}(1)[\text{symmetric}] \text{ assms}(2)[\text{unfolded insert-commute}[of$ 
 $x]] - \text{assms}(4) \text{ assms}(5-6)]$ 
have  $or: ws \sim [x, x, y] \vee ws \sim [y, y, x]$  by linarith
thus False
proof (rule disjE)
assume  $ws \sim [x, x, y]$ 
from  $\langle \text{count-list } ws \ y \geq 2 \rangle$   $[\text{unfolded count-list-conjug}[OF \text{ this}]$ 
show False
using  $\langle x \neq y \rangle$  by force
next
assume  $ws \sim [y, y, x]$ 
from  $\langle \text{count-list } ws \ x \geq 2 \rangle$   $[\text{unfolded count-list-conjug}[OF \text{ this}]$ 
show False
using  $\langle x \neq y \rangle$  by force
qed
qed

```

## 0.8 Examples

```

lemma  $x \neq \varepsilon \implies \varepsilon (x \cdot x) \varepsilon \sim_{\mathcal{I}} [x, x]$ 
unfolding factor-interpretation-def
by simp

```

```

lemma assumes  $x = [(0::\text{nat}), 1, 0, 1, 0]$  and  $y = [1, 0, 0, 1]$ 
shows  $[0, 1] (x \cdot x) [1, 0] \sim_{\mathcal{I}} [x, y, x]$ 
unfolding factor-interpretation-def assms by (simp add: suffix-def)

```

## 0.9 Primitivity non-preserving binary code

In this section, we give the final form of imprimitive words over a given binary code  $\{x, y\}$ . We start with a lemma, then we show that the only possibility is that such word is conjugate with  $x^{\textcircled{a}} j \cdot y^{\textcircled{a}} k$ .

```

lemma bin-imprim-expsE-y: assumes  $x \cdot y \neq y \cdot x$  and
 $ws \in \text{lists } \{x, y\}$  and
 $2 \leq |ws|$  and
 $\text{primitive } ws$  and
 $\neg \text{primitive } (\text{concat } ws)$  and

```

*count-list ws y = 1*  
**obtains**  $j\ k$  **where**  $1 \leq j\ 1 \leq k\ j = 1 \vee k = 1$   
 $ws \sim [x]^{\textcircled{a}j} \cdot [y]^{\textcircled{a}k}$   
**proof**–  
**have**  $x \neq y$  **using**  $\langle x \cdot y \neq y \cdot x \rangle$  **by** *blast*  
**obtain**  $j1\ j2$  **where**  $[x]^{\textcircled{a}j1} \cdot [y] \cdot [x]^{\textcircled{a}j2} = ws$   
**using** *bin-count-one-decompose*[*OF*  $\langle ws \in \text{lists } \{x,y\} \rangle \langle x \neq y \rangle \langle \text{count-list } ws\ y = 1 \rangle$ ].  
**have**  $1 \leq j2 + j1$   
**using**  $\langle [x]^{\textcircled{a}j1} \cdot [y] \cdot [x]^{\textcircled{a}j2} = ws \rangle \langle 2 \leq |ws| \rangle$  **not-less-eq-eq** **by** *fastforce*  
**have**  $ws \sim [x]^{\textcircled{a}(j2+j1)} \cdot [y]^{\textcircled{a}1}$   
**using** *conjugI'*[*of*  $[x]^{\textcircled{a}j1} \cdot [y] \cdot [x]^{\textcircled{a}j2}$ ]  
**unfolding**  $\langle [x]^{\textcircled{a}j1} \cdot [y] \cdot [x]^{\textcircled{a}j2} = ws \rangle$  [*symmetric*] *pow-add rassoc pow-list-1*.  
**from** *that*[*OF*  $\langle 1 \leq j2 + j1 \rangle$  - - *this*]  
**show** *?thesis*  
**by** *blast*  
**qed**

**lemma** *bin-imprim-expsE*: **assumes**  $x \cdot y \neq y \cdot x$  **and**  
 $ws \in \text{lists } \{x,y\}$  **and**  
 $2 \leq |ws|$  **and**  
*primitive ws* **and**  
 $\neg \text{primitive } (\text{concat } ws)$   
**obtains**  $j\ k$  **where**  $1 \leq j\ 1 \leq k\ j = 1 \vee k = 1$   
 $ws \sim [x]^{\textcircled{a}j} \cdot [y]^{\textcircled{a}k}$   
**proof**–  
**note**  $\langle ws \in \text{lists } \{x,y\} \rangle$  [*unfolded insert-commute*[*of x*]]

**from**  $\langle ws \in \text{lists } \{x,y\} \rangle$   
*sing-lists-exp-len*[*of ws y*]  
*prim-exp-one*[*OF*  $\langle \text{primitive } ws \rangle$ , *of*  $|ws|$   $[y]$ ]  
**have** *count-list ws x*  $\neq 0$   
**using**  $\langle 2 \leq |ws| \rangle$  **unfolding** *count-list-0-iff* **by** *force*

**from**  $\langle ws \in \text{lists } \{y,x\} \rangle$   
*sing-lists-exp-len*[*of ws x*]  
*prim-exp-one*[*OF*  $\langle \text{primitive } ws \rangle$ , *of*  $|ws|$   $[x]$ ]  
**have** *count-list ws y*  $\neq 0$   
**using**  $\langle 2 \leq |ws| \rangle$  **unfolding** *count-list-0-iff* **by** *fastforce*

**consider** *count-list ws x = 1* | *count-list ws y = 1*  
**using** *bin-imprim-both-twice*[*OF*  $\langle x \cdot y \neq y \cdot x \rangle \langle ws \in \text{lists } \{x,y\} \rangle$  - -  
 $\langle \text{primitive } ws \rangle \langle \neg \text{primitive } (\text{concat } ws) \rangle$   
 $\langle \text{count-list } ws\ x \neq 0 \rangle \langle \text{count-list } ws\ y \neq 0 \rangle$   
**unfolding** *One-less-Two-le-iff*[*symmetric*] *less-one*[*symmetric*] **by** *fastforce*  
**thus** *thesis*  
**proof**(*cases*)  
**assume**  $\langle \text{count-list } ws\ x = 1 \rangle$   
**from** *bin-imprim-expsE*-*y*[*reversed*, *OF*  $\langle x \cdot y \neq y \cdot x \rangle \langle ws \in \text{lists } \{y, x\} \rangle \langle 2 \leq$

```

|ws|
  ⟨primitive ws⟩ ⟨¬ primitive (concat ws)⟩ ⟨count-list ws x = 1⟩
  show thesis
  using that by metis
next
  assume ⟨count-list ws y = 1⟩
  from bin-imprim-expsE-y[OF ⟨x · y ≠ y · x⟩ ⟨ws ∈ lists {x, y}⟩ ⟨2 ≤ |ws|⟩
    ⟨primitive ws⟩ ⟨¬ primitive (concat ws)⟩ ⟨count-list ws y = 1⟩]
  show ?thesis
  using that.
qed
qed

```

### 0.9.1 The target theorem

Given a binary code  $\{x, y\}$  such that there is a primitive factorisation  $ws$  over it whose concatenation is imprimitive, we finally show that there are integers  $j$  and  $k$  (depending only on  $\{x, y\}$ ) such that any other such factorisation  $ws'$  is conjugate to  $[x]^{\textcircled{j}} \cdot [y]^{\textcircled{k}}$ .

**theorem** *bin-imprim-code*: **assumes**  $x \cdot y \neq y \cdot x$  **and**  $ws \in \text{lists } \{x, y\}$  **and**  $2 \leq |ws|$  **and** *primitive ws* **and**  $\neg \text{primitive } (\text{concat } ws)$   
**obtains**  $j$   $k$  **where**  $1 \leq j$  **and**  $1 \leq k$  **and**  $j = 1 \vee k = 1$   
 $\wedge ws. ws \in \text{lists } \{x, y\} \implies 2 \leq |ws| \implies$   
 $(\text{primitive } ws \wedge \neg \text{primitive } (\text{concat } ws) \longleftrightarrow ws \sim [x]^{\textcircled{j}} \cdot [y]^{\textcircled{k}})$  **and**  
 $|y| \leq |x| \implies 2 \leq j \implies j = 2 \wedge \text{primitive } x \wedge \text{primitive } y$  **and**  
 $|y| \leq |x| \implies 2 \leq k \implies j = 1 \wedge \text{primitive } x$

**proof**–

```

obtain  $j$   $k$  where  $1 \leq j$   $1 \leq k$   $j = 1 \vee k = 1$ 
 $ws \sim [x]^{\textcircled{j}} \cdot [y]^{\textcircled{k}}$ 
using bin-imprim-expsE[OF ⟨x · y ≠ y · x⟩]
using assms by metis

```

```

have  $\neg \text{primitive } (x^{\textcircled{j}} \cdot y^{\textcircled{k}})$ 
using ⟨¬ primitive (concat ws)⟩
unfolding concat-morph concat-pow-list-single
conjug-prim-iff[OF conjug-concat-conjug[OF ⟨ws ∼ [x]Ⓢ j · [y]Ⓢ k⟩]].

```

```

from not-prim-primroot-expE[OF this]
obtain  $z$   $l$  where [symmetric]:  $z^{\textcircled{l}} = x^{\textcircled{j}} \cdot y^{\textcircled{k}}$  and  $2 \leq l$ .

```

```

show thesis
proof (rule that[of  $j$   $k$  ])
  show  $1 \leq j$   $1 \leq k$   $j = 1 \vee k = 1$  by fact+

```

```

  fix  $ws'$ 
  assume hyps:  $ws' \in \text{lists } \{x, y\}$   $2 \leq |ws'|$ 
  show  $\text{primitive } ws' \wedge \neg \text{primitive } (\text{concat } ws') \longleftrightarrow ws' \sim [x]^{\textcircled{j}} \cdot [y]^{\textcircled{k}}$ 
  proof

```

```

assume primitive ws'  $\wedge$   $\neg$  primitive (concat ws')
hence prems: primitive ws'  $\neg$  primitive (concat ws') by blast+
obtain j' k' where  $1 \leq j'$   $1 \leq k'$   $j' = 1 \vee k' = 1$ 
  ws'  $\sim$  [x]@j' · [y]@k'
  using bin-imprim-expsE[OF  $\langle x \cdot y \neq y \cdot x \rangle$  hyps prems].

have  $\neg$  primitive (x@j' · y@k')
  using  $\langle \neg$  primitive (concat ws') $\rangle$ 
  unfolding concat-morph concat-pow-list-single
    conjug-prim-iff[OF conjug-concat-conjug[OF  $\langle ws' \sim [x]^{\textcircled{a}} j' \cdot [y]^{\textcircled{a}} k' \rangle$ ]].

have j = j' k = k'
  using LS-unique[OF  $\langle x \cdot y \neq y \cdot x \rangle$ 
    - -  $\langle \neg$  primitive (x@j · y@k) $\rangle$ 
    - -  $\langle \neg$  primitive (x@j' · y@k') $\rangle$ ]  $\langle 1 \leq j \rangle$   $\langle 1 \leq k \rangle$   $\langle 1 \leq j' \rangle$   $\langle 1 \leq k' \rangle$  by
force+

  show ws'  $\sim$  [x]@j · [y]@k
  unfolding  $\langle j = j' \rangle$   $\langle k = k' \rangle$  by fact
next
assume ws'  $\sim$  [x]@j · [y]@k
note conjug-trans[OF  $\langle ws \sim [x]^{\textcircled{a}} j \cdot [y]^{\textcircled{a}} k \rangle$  conjug-sym[OF this]]
from prim-conjug[OF  $\langle$ primitive ws $\rangle$  this]
   $\langle \neg$  primitive (concat ws) $\rangle$ [unfolded conjug-concat-prim-iff[OF  $\langle ws \sim ws' \rangle$ ]]
  show primitive ws'  $\wedge$   $\neg$  primitive (concat ws') by blast
qed
next
assume |y|  $\leq$  |x|
interpret LS-len-le x y j k l z
by unfold-locales fact+

assume  $2 \leq j$ 
with jk-small
have k = 1 by fastforce
from case-j2k1[OF  $\langle 2 \leq j \rangle$  this]
show j = 2  $\wedge$  primitive x  $\wedge$  primitive y
  by blast
next
assume |y|  $\leq$  |x|
interpret LS-len-le x y j k l z
  by unfold-locales fact+

assume  $2 \leq k$ 
show j = 1  $\wedge$  primitive x
  using  $\langle 2 \leq k \rangle$   $\langle j = 1 \vee k = 1 \rangle$  case-j1k2-primitive by auto
qed
qed

```

— Formulation in terms of (binary) primitive morphism

**definition** *bin-imprim-code* **where** *bin-imprim-code*  $x\ y \equiv x \cdot y \neq y \cdot x \wedge (\neg \text{bin-prim } x\ y)$

**theorem** *bin-imprim-code'*: **assumes** *bin-imprim-code*  $x\ y$

**obtains**  $j\ k$  **where**  $1 \leq j$  **and**  $1 \leq k$  **and**  $j = 1 \vee k = 1$

$\wedge ws. ws \in \text{lists } \{x,y\} \implies 2 \leq |ws| \implies$

$(\text{primitive } ws \wedge \neg \text{primitive } (\text{concat } ws) \longleftrightarrow ws \sim [x]^{\textcircled{j}} \cdot [y]^{\textcircled{k}})$  **and**

$|y| \leq |x| \implies 2 \leq j \implies j = 2 \wedge \text{primitive } x \wedge \text{primitive } y$  **and**

$|y| \leq |x| \implies 2 \leq k \implies j = 1 \wedge \text{primitive } x$

**proof**–

**thm** *bin-imprim-code*

**obtain**  $ws$  **where**  $x \cdot y \neq y \cdot x$

**and**  $ws \in \text{lists } \{x,y\}$  **and**  $2 \leq |ws|$  **and** *primitive*  $ws$  **and**  $\neg \text{primitive } (\text{concat } ws)$

**using** *assms* **unfolding** *bin-imprim-code-def bin-prim-altdef2* **by** *metis*

**from** *bin-imprim-code[OF this]* **that**

**show** *thesis*

**by** *blast*

**qed**

**end**

**theory** *Binary-Imprimitive-Decision*

**imports**

*Binary-Code-Imprimitive.Binary-Code-Imprimitive*

**begin**

## 0.10 Upper bound of the power exponent in the canonical imprimitivity witness

**lemma** *LS-power-len-ge*:

**assumes**  $y^{\textcircled{k}} \cdot x = z^{\textcircled{l}}$

**and**  $k * |y| \geq |z| + |y| - 1$

**shows**  $x \cdot y = y \cdot x$

**proof** (*rule nemp-comm*)

**assume**  $y \neq \varepsilon$

**have**  $y^{\textcircled{k}} \leq_p z \cdot y^{\textcircled{k}}$

**using**  $\langle y^{\textcircled{k}} \cdot x = z^{\textcircled{l}} \rangle$

**by** (*blast intro!: pref-pow-root*)

**moreover** **have**  $y^{\textcircled{k}} \leq_p y \cdot y^{\textcircled{k}}$

**by** (*blast intro!: pref-pow-ext'*)

**moreover** **have**  $1 \leq \text{gcd } |z| |y|$

**using**  $\langle y \neq \varepsilon \rangle$

**by** (*simp flip: less-eq-Suc-le*)

**from** *this*  $\langle k * |y| \geq |z| + |y| - 1 \rangle$

**have**  $|z| + |y| - (\text{gcd } |z| \ |y|) \leq k * |y|$   
**by** (*rule le-trans[OF diff-le-mono2]*)  
**ultimately have**  $z \cdot y = y \cdot z$   
**unfolding** *pow-len[symmetric]* **by** (*fact per-lemma-comm*)  
**with**  $\langle y^{\textcircled{a}} k \cdot x = z^{\textcircled{a}} l \rangle$   
**show**  $x \cdot y = y \cdot x$   
**by** (*fact LS-comm*)  
**qed**

**lemma** *LS-root-len-ge:*

**assumes**  $y^{\textcircled{a}} k \cdot x = z^{\textcircled{a}} l$   
**and**  $1 \leq k$  **and**  $2 \leq l$   
**and**  $x \cdot y \neq y \cdot x$   
**shows**  $(k - 1) * |y| + 2 \leq |z|$   
**proof** (*intro leI notI*)  
**assume**  $|z| < (k - 1) * |y| + 2$   
**then have**  $|z| + |y| \leq \text{Suc } (k - 1) * |y| + 1$   
**by** *simp*  
**also have**  $\dots = k * |y| + 1$   
**using**  $\langle 1 \leq k \rangle$  **by** *simp*  
**finally have**  $k * |y| \geq |z| + |y| - 1$   
**unfolding** *le-diff-conv.*  
**from**  $\langle x \cdot y \neq y \cdot x \rangle$  *LS-power-len-ge[OF  $\langle y^{\textcircled{a}} k \cdot x = z^{\textcircled{a}} l \rangle$  this]*  
**show** *False..*  
**qed**

**lemma** *LS-root-len-le:*

**assumes**  $y^{\textcircled{a}} k \cdot x = z^{\textcircled{a}} l$   
**and**  $1 \leq k$  **and**  $2 \leq l$   
**and**  $x \cdot y \neq y \cdot x$   
**shows**  $|z| \leq |x| + |y| - 2$   
**proof** –  
**have**  $|x| + k * |y| = l * |z|$   
**using** *lenarg[OF  $\langle y^{\textcircled{a}} k \cdot x = z^{\textcircled{a}} l \rangle$ ]*  
**by** (*simp only: pow-len lenmorph add.commute[of |x|]*)  
**have**  $|z| \leq (l - 1) * |z|$   
**using** *diff-le-mono[OF  $\langle 2 \leq l, \text{ of } 1 \rangle$  by simp]*  
**also have**  $\dots = |x| + k * |y| - |z|$   
**unfolding** *diff-mult-distrib  $\langle |x| + k * |y| = l * |z| \rangle$ [symmetric]* **by** *simp*  
**also have**  $\dots \leq |x| + k * |y| - ((k - 1) * |y| + 2)$   
**using** *LS-root-len-ge[OF assms]*  
**by** (*rule diff-le-mono2*)  
**also have**  $\dots \leq |x| + |y| - 2$   
**using**  $\langle 1 \leq k \rangle$  **unfolding** *diff-diff-eq[symmetric]*  
**by** (*intro diff-le-mono*) (*simp add: le-diff-conv add.assoc diff-mult-distrib*)  
**finally show**  $|z| \leq |x| + |y| - 2.$   
**qed**

**lemma** *LS-exp-le':*

**assumes**  $y \textcircled{\small \text{a}} k \cdot x = z \textcircled{\small \text{a}} l$   
**and**  $2 \leq l$   
**and**  $x \cdot y \neq y \cdot x$   
**shows**  $k \leq (|x| - 4) \text{ div } |y| + 2$   
**proof** (*cases*  $1 \leq k$ )  
**assume**  $1 \leq k$   
**have**  $|y| > 0$   
**using**  $\langle x \cdot y \neq y \cdot x \rangle$  **by** *blast*  
**have**  $(k - 1) * |y| + 2 \leq |z|$   
**using** *LS-root-len-ge*  $\langle y \textcircled{\small \text{a}} k \cdot x = z \textcircled{\small \text{a}} l \rangle \langle 1 \leq k \rangle \langle 2 \leq l \rangle \langle x \cdot y \neq y \cdot x \rangle$ .  
**also have**  $|z| \leq |x| + |y| - 2$   
**using** *LS-root-len-le*  $\langle y \textcircled{\small \text{a}} k \cdot x = z \textcircled{\small \text{a}} l \rangle \langle 1 \leq k \rangle \langle 2 \leq l \rangle \langle x \cdot y \neq y \cdot x \rangle$ .  
**finally have**  $(k - 1) * |y| + 2 \leq |x| + |y| - 2$ .  
**then have**  $(k - 1) * |y| + 2 - |y| - 2 \leq |x| + |y| - 2 - |y| - 2$   
**by** (*intro diff-le-mono*)  
**then have**  $(k - 1) * |y| + 2 - 2 - |y| \leq |x| - (2 + 2)$   
**unfolding** *diff-commute[of - 2 |y|]* **unfolding** *diff-add-inverse2 diff-diff-eq*.  
**then have**  $(k - (1 + 1)) * |y| \leq |x| - 4$   
**unfolding** *diff-add-inverse2 nat-distrib diff-diff-eq mult-1*  
**by** *presburger*  
**then show**  $k \leq (|x| - 4) \text{ div } |y| + 2$   
**using**  $\langle |y| > 0 \rangle$   
**by** (*simp only: less-eq-div-iff-mult-less-eq one-add-one flip: le-diff-conv*)  
**qed** (*simp add: trans-le-add2*)

**lemma** *LS-exp-le*:

**assumes**  $x \cdot y \textcircled{\small \text{a}} k = z \textcircled{\small \text{a}} l$   
**and**  $2 \leq l$   
**and**  $x \cdot y \neq y \cdot x$   
**shows**  $k \leq (|x| - 4) \text{ div } |y| + 2$   
**using** *LS-exp-le'[reversed, OF*  $\langle x \cdot y \textcircled{\small \text{a}} k = z \textcircled{\small \text{a}} l \rangle \langle 2 \leq l \rangle \langle x \cdot y \neq y \cdot x \rangle$   
*symmetric]*.

**thm** *bin-imprim-expsE*

**lemma** *bin-imprim-code-witnessE*:

**assumes**  $x \cdot y \neq y \cdot x$  **and**  $|y| \leq |x|$   
**and**  $ws \in \text{lists } \{x, y\}$  **and**  $2 \leq |ws|$   
**and** *primitive ws* **and**  $\neg \text{primitive } (\text{concat } ws)$   
**obtains**  $ws \sim [x, x, y]$   
**|**  $k$  **where**  $1 \leq k$  **and**  $k \leq (|x| - 4) \text{ div } |y| + 2$   
**and**  $ws \sim [x] \cdot [y] \textcircled{\small \text{a}} k$

**proof** –

**obtain**  $j \ k$

**where**  $1 \leq j$  **and**  $1 \leq k$  **and**  $j = 1 \vee k = 1$

**and** *witness-iff*:  $\bigwedge ws. ws \in \text{lists } \{x, y\} \implies 2 \leq |ws| \implies$

$(\text{primitive } ws \wedge \neg \text{primitive } (\text{concat } ws)) \iff ws \sim [x] \textcircled{\small \text{a}} j \cdot [y] \textcircled{\small \text{a}} k)$

**and**

*j-ge*:  $|y| \leq |x| \implies 2 \leq j \implies j = 2 \wedge \text{primitive } x \wedge \text{primitive } y$  **and**

*k-ge*:  $|y| \leq |x| \implies 2 \leq k \implies j = 1 \wedge \text{primitive } x$

```

  by (fact bin-imprim-code[OF assms(1, 3 - 6)])
  have  $ws \sim [x]^{\textcircled{a} j} \cdot [y]^{\textcircled{a} k}$ 
  using witness-iff[OF  $\langle ws \in \text{lists } \{x, y\} \rangle \langle 2 \leq |ws| \rangle \langle \text{primitive } ws \rangle \langle \neg \text{primitive } (\text{concat } ws) \rangle$ ]
  by simp
  show thesis
  proof (cases)
    assume  $2 \leq j$ 
    from j-ge[OF  $\langle |y| \leq |x| \rangle \text{ this } \langle j = 1 \vee k = 1 \rangle$ ]
    have  $j = 2$  and  $k = 1$ 
    by simp-all
    then have  $ws \sim [x, x, y]$ 
    using  $\langle ws \sim [x]^{\textcircled{a} j} \cdot [y]^{\textcircled{a} k} \rangle$  by simp
    then show thesis..
  next
    assume  $\neg 2 \leq j$ 
    with  $\langle 1 \leq j \rangle$  have  $j = 1$ 
    by simp
    then have  $ws \sim [x] \cdot [y]^{\textcircled{a} k}$ 
    using  $\langle ws \sim [x]^{\textcircled{a} j} \cdot [y]^{\textcircled{a} k} \rangle$  by simp
    then have  $\neg \text{primitive } (x \cdot y^{\textcircled{a} k})$ 
    using  $\langle \neg \text{primitive } (\text{concat } ws) \rangle$  unfolding conjug-concat-prim-iff[OF  $\langle ws \sim [x] \cdot [y]^{\textcircled{a} k} \rangle$ ]
    by simp
    moreover have  $x \cdot y^{\textcircled{a} k} \neq \varepsilon$ 
    using  $\langle x \cdot y \neq y \cdot x \rangle$  by (intro notI) simp
    ultimately obtain  $z \ l$ 
    where  $2 \leq l$  and  $z^{\textcircled{a} l} = x \cdot y^{\textcircled{a} k}$ 
    by (elim not-prim-expE)
    have  $k \leq (|x| - 4) \text{ div } |y| + 2$ 
    using LS-exp-le[OF  $\langle z^{\textcircled{a} l} = x \cdot y^{\textcircled{a} k} \rangle$  [symmetric]  $\langle 2 \leq l \rangle \langle x \cdot y \neq y \cdot x \rangle$ ]
    from  $\langle 1 \leq k \rangle$  this  $\langle ws \sim [x] \cdot [y]^{\textcircled{a} k} \rangle$ 
    show thesis..
  qed
qed

```

### 0.10.1 Optimality of the exponent upper bound

lemma examples-bound-optimality:

```

  fixes  $m \ k$  and  $x \ y \ z :: \text{binA list}$ 
  assumes  $1 \leq m$  and  $k' = 0 \implies m = 1$ 
  defines  $x \equiv a \cdot b \cdot (b \cdot (a \cdot b)^{\textcircled{a} m})^{\textcircled{a} k'} \cdot b \cdot a$ 
    and  $y \equiv b \cdot (a \cdot b)^{\textcircled{a} m}$ 
    and  $z \equiv a \cdot b \cdot (b \cdot (a \cdot b)^{\textcircled{a} m})^{\textcircled{a} (k' + 1)}$ 
    and  $k \equiv k' + 2$ 

```

shows  $|y| \leq |x|$  and  $x \cdot y^{\textcircled{a} k} = z \cdot z$  and  $k = (|x| - 4) \text{ div } |y| + 2$

proof -

obtain  $m'$  where  $m: m = m' + 1$

using  $\langle 1 \leq m \rangle$  using add.commute le-Suc-ex by blast

```

have x-len: |x| = k' * (2 * m + 1) + 4
  unfolding x-def by (simp add: pow-len)
have y-len: |y| = 2 * m + 1
  unfolding y-def by (simp add: pow-len)
have z-len: |z| = (k' + 1) * (2 * m + 1) + 2
  unfolding z-def by (simp add: pow-len)
show |y| ≤ |x|
  using ⟨k' = 0 ⇒ m = 1⟩ unfolding x-len y-len
  by (cases k') (simp-all add: pow-len)
show x · y @ k = z · z
  unfolding x-def y-def z-def k-def
  by comparison
show k = (|x| - 4) div |y| + 2
proof -
  have |x| - 4 = k' * |y|
    unfolding x-len y-len by simp
  have |y| ≠ 0
    unfolding y-def by blast
  have k' = (|x| - 4) div |y|
    unfolding ⟨|x| - 4 = k' * |y|⟩ nonzero-mult-div-cancel-right[OF ⟨|y| ≠ 0⟩]..
  then show k = (|x| - 4) div |y| + 2
    unfolding k-def by blast
qed
qed

```

## 0.11 Characterization of binary primitivity preserving morphisms given by a pair of words

```

lemma len-le-not-bin-primE:
  assumes |y| ≤ |x|
  and ¬ bin-prim x y
  obtains ¬ primitive (x · x · y)
  | k where 1 ≤ k and k ≤ (|x| - 4) div |y| + 2
  and ¬ primitive (x · y @ k)
proof (cases)
  assume x · y = y · x
  have ¬ primitive (x · x · y)
    using ⟨x · y = y · x⟩ ⟨|y| ≤ |x|⟩
    by (cases x ≠ ε) (intro comm-not-prim, simp-all)
  then show thesis..
next
  assume x · y ≠ y · x
  then have x ≠ y
    by blast
  obtain ws where
    ws ∈ lists {x, y} and 2 ≤ |ws| and primitive ws and ¬ primitive (concat ws)
  using ⟨¬ bin-prim x y⟩ unfolding bin-prim-concat-prim-pres-conv[OF ⟨x ≠ y⟩]
  by meson

```

**then consider**  $ws \sim [x, x, y]$   
|  $k$  **where**  $1 \leq k$  **and**  $k \leq (|x| - 4) \text{ div } |y| + 2$   
**and**  $ws \sim [x] \cdot [y]^{\textcircled{a}} k$   
**by** (*rule bin-imprim-code-witnessE*[*OF*  $\langle x \cdot y \neq y \cdot x \rangle \langle |y| \leq |x| \rangle$ ])  
**then show thesis**  
**proof** (*cases*)  
**assume**  $ws \sim [x, x, y]$   
**then have**  $\neg \text{primitive}(x \cdot x \cdot y)$   
**using**  $\langle \neg \text{primitive}(\text{concat } ws) \rangle$   
**by** (*simp add: conjug-concat-prim-iff*)  
**then show thesis..**  
**next**  
**fix**  $k$   
**assume**  $1 \leq k$  **and**  $k \leq (|x| - 4) \text{ div } |y| + 2$  **and**  $ws \sim [x] \cdot [y]^{\textcircled{a}} k$   
**have**  $\neg \text{primitive}(x \cdot y^{\textcircled{a}} k)$   
**using**  $\langle ws \sim [x] \cdot [y]^{\textcircled{a}} k \rangle \langle \neg \text{primitive}(\text{concat } ws) \rangle$   
**by** (*simp add: conjug-concat-prim-iff*)  
**from**  $\langle 1 \leq k \rangle \langle k \leq (|x| - 4) \text{ div } |y| + 2 \rangle$  *this*  
**show thesis..**  
**qed**  
**qed**

**lemma bin-prim-xyk:**  
**assumes** *bin-prim*  $x y$  **and**  $0 < k$   
**shows** *primitive*  $(x \cdot y^{\textcircled{a}} k)$   
**proof** –  
**have** *primitive*  $([x] \cdot [y]^{\textcircled{a}} k)$   
**using** *bin-prim-code*[*OF*  $\langle \text{bin-prim } x y \rangle$ ]  
**by** (*intro prim-abk*) *blast*  
**from** *bin-prim-concat-prim-pres*[*OF*  $\langle \text{bin-prim } x y \rangle$  - - *this*]  $\langle 0 < k \rangle$   
**show** *primitive*  $(x \cdot y^{\textcircled{a}} k)$   
**by** (*simp add: pow-in-lists*)  
**qed**

**lemma len-le-bin-prim-iff:**  
**assumes**  $|y| \leq |x|$   
**shows**  
 $\text{bin-prim } x y \iff \text{primitive}(x \cdot x \cdot y) \wedge (\forall k. 1 \leq k \wedge k \leq (|x| - 4) \text{ div } |y| + 2 \implies \text{primitive}(x \cdot y^{\textcircled{a}} k))$   
(is *bin-prim*  $x y \iff (?xxy \wedge ?xyk)$ )  
**proof** (*intro iffI*[*OF* - *contrapos-pp*])  
**assume** *bin-prim*  $x y$   
**show**  $?xxy \wedge ?xyk$   
**proof** (*intro conjI allI impI*)  
**show** *primitive*  $(x \cdot x \cdot y)$   
**using** *bin-prim-xyk*[*OF*  $\langle \text{bin-prim } x y \rangle$  [*symmetric*], *of 2*] *conjug-prim-iff'*  
**by** (*simp add: conjug-prim-iff'*[*of y*])  
**show** *primitive*  $(x \cdot y^{\textcircled{a}} k)$  **if**  $1 \leq k \wedge k \leq (|x| - 4) \text{ div } |y| + 2$  **for**  $k$   
**using** *bin-prim-xyk*[*OF*  $\langle \text{bin-prim } x y \rangle$ , *of k*] *conjunct1*[*OF that*]

by *fastforce*  
**qed**  
**next**  
 assume  $\neg \text{bin-prim } x \ y$   
 then consider  $\neg \text{primitive } (x \cdot x \cdot y)$   
 |  $k$  where  $1 \leq k$  and  $k \leq (|x| - 4) \text{ div } |y| + 2$   
 and  $\neg \text{primitive } (x \cdot y^{\textcircled{a}} k)$   
 by (*elim len-le-not-bin-primE*[*OF*  $\langle |y| \leq |x| \rangle$ ])  
 then show  $\neg (?xxy \wedge ?xyk)$   
 by (*cases*) *blast+*  
**qed**

**lemma** *len-eq-bin-prim-iff*:  
 assumes  $|x| = |y|$   
 shows  $\text{bin-prim } x \ y \longleftrightarrow \text{primitive } (x \cdot y)$   
**proof**  
 show  $\text{bin-prim } x \ y \implies \text{primitive } (x \cdot y)$   
 using *bin-prim-xyk*[*of* - - 1]  
 by *simp*  
 assume  $\text{primitive } (x \cdot y)$   
 then have  $x \cdot y \neq y \cdot x$   
 using *assms eq-append-not-prim* by *auto*  
 from *this bin-uniform-prim-morph*[*OF* *this*  $\langle |x| = |y| \rangle$   $\langle \text{primitive } (x \cdot y) \rangle$ ]  
 show  $\text{bin-prim } x \ y$   
 unfolding *bin-prim-altdef2*  
 by *simp*  
**qed**

**theorem** *bin-prim-iff*:  
 $\text{bin-prim } x \ y \longleftrightarrow$   
 (*if*  $|y| < |x|$   
 then  $\text{primitive } (x \cdot x \cdot y) \wedge (\forall k. 1 \leq k \wedge k \leq (|x| - 4) \text{ div } |y| + 2 \longrightarrow$   
 $\text{primitive } (x \cdot y^{\textcircled{a}} k))$   
 else *if*  $|x| < |y|$   
 then  $\text{primitive } (y \cdot y \cdot x) \wedge (\forall k. 1 \leq k \wedge k \leq (|y| - 4) \text{ div } |x| + 2 \longrightarrow$   
 $\text{primitive } (y \cdot x^{\textcircled{a}} k))$   
 else  $\text{primitive } (x \cdot y)$   
 )  
**proof** (*cases rule: linorder-cases*)  
 assume  $|x| < |y|$   
 then show *?thesis*  
 unfolding *bin-prim-commutes*[*of*  $x \ y$ ]  
 unfolding *len-le-bin-prim-iff*[*OF* *less-imp-le*[*OF*  $\langle |x| < |y| \rangle$ ]]  
 by (*simp only: if-not-P*[*OF* *less-not-sym*] *if-P*)  
**next**  
 assume  $|y| < |x|$   
 then show *?thesis*  
 unfolding *len-le-bin-prim-iff*[*OF* *less-imp-le*[*OF*  $\langle |y| < |x| \rangle$ ]]  
 by (*simp only: if-P*)

```

next
  assume  $|x| = |y|$ 
  then show ?thesis
    unfolding len-eq-bin-prim-iff[OF  $\langle |x| = |y| \rangle$ ]
    by simp
qed

```

### 0.11.1 Code equation for *bin-prim* predicate

```

context
begin

```

```

private lemma all-less-Suc-conv:  $(\forall k < n. P (Suc k)) \longleftrightarrow (\forall k \leq n. k \geq 1 \longrightarrow P k)$ 
proof (intro iffI allI impI)
  fix k
  assume  $\forall k < n. P (Suc k)$  and  $k \leq n$  and  $1 \leq k$ 
  then show P k
    by (elim allE[of - k - 1]) (simp only: Suc-diff-1)
qed simp

```

```

lemma bin-prim-iff' [code]:

```

```

  bin-prim x y  $\longleftrightarrow$ 
    (if  $|y| < |x|$ 
      then primitive  $(x \cdot x \cdot y) \wedge (\forall k < (|x| - 4) \operatorname{div} |y| + 2. \operatorname{primitive} (x \cdot y @ (Suc k)))$ 
      else if  $|x| < |y|$ 
        then bin-prim y x
        else primitive  $(x \cdot y)$ 
    )

```

```

proof (cases rule: linorder-cases)

```

```

  show  $|x| < |y| \implies ?thesis$ 
    unfolding bin-prim-commutes[of x y]
    by simp

```

```

next

```

```

  assume  $|y| < |x|$ 
  then show ?thesis
    using len-le-bin-prim-iff[OF less-imp-le[OF  $\langle |y| < |x| \rangle$ ]]
    unfolding all-less-Suc-conv[where P =  $\lambda k. \operatorname{primitive} (- \cdot - @ k)$ ]
    unfolding conj-comms[of 1 ≤ -] imp-conjL
    by (simp only: if-P)

```

```

next

```

```

  assume  $|x| = |y|$ 
  then show ?thesis
    unfolding len-eq-bin-prim-iff[OF  $\langle |x| = |y| \rangle$ ]
    by simp

```

```

qed

```

```

end

```

**value** *bin-prim* (a·b·b·a·a) b — True  
**value** *bin-prim* (a·b·b·a) b — False  
**value** *bin-prim* (a·b·b·a) (b·a·b·a·b) — False  
**value** *bin-prim* (a·b) (a·b) — False  
**value** *bin-prim* (a·b) (a·b·a·b) — False  
**value** *bin-prim* (a·b·b·a·a) (b·b·b·b·b·b) — True

## 0.12 Characterization of binary imprimitivity codes

**theorem** *bin-imprim-code-iff*:

$bin-imprim-code\ x\ y \iff x \cdot y \neq y \cdot x \wedge$   
 (if  $|y| < |x|$   
   then  $\neg primitive\ (x \cdot x \cdot y) \vee (\exists k. 1 \leq k \wedge k \leq (|x| - 4) \operatorname{div} |y| + 2 \wedge \neg$   
*primitive*  $(x \cdot y^{\textcircled{a}} k))$   
   else if  $|x| < |y|$   
   then  $\neg primitive\ (y \cdot y \cdot x) \vee (\exists k. 1 \leq k \wedge k \leq (|y| - 4) \operatorname{div} |x| + 2 \wedge \neg$   
*primitive*  $(y \cdot x^{\textcircled{a}} k))$   
   else  $\neg primitive\ (x \cdot y)$   
 )

**unfolding** *bin-imprim-code-def bin-prim-iff*

**by** (*simp only: de-Morgan-conj not-all not-imp conj-assoc flip: if-image[of - Not]*)

**value** *bin-imprim-code* (a·b·b·a·a) b — False  
**value** *bin-imprim-code* (a·b·b·a) b — True  
**value** *bin-imprim-code* (a·b·b·a) (b·a·b·a·b) — True  
**value** *bin-imprim-code* (a·b) (a·b) — False  
**value** *bin-imprim-code* (a·b) (a·b·a·b) — False  
**value** *bin-imprim-code* (a·b·b·a·a) (b·b·b·b·b·b) — False

**end**

# References

- [1] E. Barbin-Le Rest and M. Le Rest. Sur la combinatoire des codes à deux mots. *Theor. Comput. Sci.*, 41:61–80, 1985.
- [2] J. Mañuch. Defect effect of bi-infinite words in the two-element case. *Discret. Math. Theor. Comput. Sci.*, 4(2):273–290, 2001.
- [3] J.-P. Spehner. *Quelques problèmes d'extension, de conjugaison et de présentation des sous-monoïdes d'un monoïde libre*. PhD thesis, Université Paris VII, Paris, 1976.