Combinatorics on Words formalized Graph Lemma

Štěpán Holub
Štěpán Starosta

December 14, 2021

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

Contents

theory *Graph-Lemma*
imports *Combinatorics-Words.Submonoids*

begin

Chapter 1

Graph Lemma

The Graph Lemma is an important tool for gaining information about systems of word equations. It yields an upper bound on the rank of the solution, that is, on the number of factors into all images of unknowns can be factorized. The most straightforward application is showing that a system of equations admits periodic solutions only, which in particular holds for any nontrivial equation over two words.

The name refers to a graph whose vertices are the unknowns of the system, and edges connect front letters of the left- and right- hand sides of equations. The bound mentioned above is then the number of connected components of the graph.

We formalize the algebraic proof from [1]

Let C be a set of generators, and b its distinguished element. We define the set of all products that do not start with b.

inductive-set *no-head* :: *'a list set* ⇒ *'a list* ⇒ *'a list set*
  for *C b* **where**
  emp-in-no-head[*simp*]: $\varepsilon \in$ *no-head C b*
  | $u \in C \Longrightarrow u \neq b \Longrightarrow u \in$ *no-head C b*
  | $u \neq \varepsilon \Longrightarrow u \in$ *no-head C b* $\Longrightarrow v \in \langle C \rangle \Longrightarrow u \cdot v \in$ *no-head C b*

The set is a submonoid of $\langle C \rangle$

lemma *no-head-sub*: *no-head C b* $\subseteq \langle C \rangle$
proof
  fix *u* **assume** $u \in$ *no-head C b*
  show $u \in \langle C \rangle$
  proof (*induction rule*: *no-head.induct*[*OF* ‹$u \in$ *no-head C b*›], *auto*)
    case (*3 u v*)
    then show $u \cdot v \in \langle C \rangle$
      using *hull-closed* **by** *blast*
  qed
qed

lemma *no-head-closed*: $\langle$*no-head C b*$\rangle$ = *no-head C b*
proof(*intro equalityI*)
show *no-head C b* $\subseteq$ $\langle$*no-head C b*$\rangle$ **by** (*simp add*: *genset-sub*)
next
show $\langle$*no-head C b*$\rangle$ $\subseteq$ *no-head C b*
proof
fix *x* **assume** *x* $\in$ $\langle$*no-head C b*$\rangle$
thus *x* $\in$ *no-head C b*
proof (*induction rule*: *hull.induct*, *simp*)
case (*prod-cl w1 w2*)
then show *w1* · *w2* $\in$ *no-head C b*
using *no-head.intros*(*3*)[*OF* - ‹*w1* $\in$ *no-head C b*›]
in-mono[*OF no-head-sub*, *of w2*] **by** *fastforce*
qed
qed
qed

We are interested mainly in the situation when C is a code.

context *code*
begin

We characterize the set *no-head* in terms of the decomposition of its (nonempty) elements: the first factor is not b

lemma *no-head-hd*: **assumes** *u* $\in$ *no-head* $\mathcal{C}$ *b* **and** *u* $\neq$ ε **shows** *hd* (*Dec* $\mathcal{C}$ *u*) $\neq$ *b*
using ‹*u* $\neq$ ε›
proof(*induct rule*: *no-head.induct*[*OF* ‹*u* $\in$ *no-head* $\mathcal{C}$ *b*›], *simp*)
case (*2 u*)
then show *?case*
using *code-el-dec* **by** *auto*
next
case (*3 u v*)
have *Dec* $\mathcal{C}$ *u* $\neq$ ε **and** *u* $\in$ $\langle\mathcal{C}\rangle$ **and** *v* $\in$ $\langle\mathcal{C}\rangle$
using *dec-nemp no-head-sub 3.hyps* **by** *blast+*
show *hd* (*Dec* $\mathcal{C}$ *u* · *v*) $\neq$ *b*
using *3.hyps*(*3*)[*OF* ‹*u* $\neq$ ε›] *hd-append2*[*OF* ‹*Dec* $\mathcal{C}$ *u* $\neq$ ε›, *of Dec* $\mathcal{C}$ *v*]
unfolding *code-dec-morph*[*OF* ‹ *u* $\in$ $\langle\mathcal{C}\rangle$› ‹*v* $\in$ $\langle\mathcal{C}\rangle$›] **by** *simp*
qed

lemma *b-not-in-no-head*: **assumes** *b* $\in$ $\mathcal{C}$ **shows** *b* $\notin$ *no-head* $\mathcal{C}$ *b*
using ‹*b* $\in$ $\mathcal{C}$› *code-el-dec emp-not-in-code no-head-hd* **by** *fastforce*

lemma *hd-no-head*: **assumes** *u* $\in$ $\langle\mathcal{C}\rangle$ **and** *hd* (*Dec* $\mathcal{C}$ *u*) $\neq$ *b* **shows** *u* $\in$ *no-head* $\mathcal{C}$ *b*
proof(*cases u* $\neq$ ε)
assume *u* $\neq$ ε
have *Dec* $\mathcal{C}$ *u* $\neq$ ε

using *dec-nemp'*[*OF* ‹$u \neq \varepsilon$› ‹$u \in \langle \mathcal{C} \rangle$›].
have $u = hd\ (Dec\ \mathcal{C}\ u) \cdot concat\ (tl\ (Dec\ \mathcal{C}\ u))$
using *concat.simps*(2)[*of hd* (*Dec* $\mathcal{C}$ u) *tl*(*Dec* $\mathcal{C}$ u)]
unfolding *hd-Cons-tl*[*OF* ‹*Dec* $\mathcal{C}$ $u \neq \varepsilon$›] *decI*[*OF* ‹$u \in \langle \mathcal{C} \rangle$›].
have $hd\ (Dec\ \mathcal{C}\ u) \#\ tl\ (Dec\ \mathcal{C}\ u) \in lists\ \mathcal{C}$ **and** $hd\ (Dec\ \mathcal{C}\ u) \in \mathcal{C}$
using ‹*Dec* $\mathcal{C}$ $u \neq \varepsilon$› ‹$u \in \langle \mathcal{C} \rangle$› *dec-dom' lists-hd* **by** *auto blast*
have $concat\ (tl\ (decompose\ \mathcal{C}\ u)) \in \langle \mathcal{C} \rangle$
using *concat-tl*[*of hd* (*Dec* $\mathcal{C}$ u) *tl*(*Dec* $\mathcal{C}$ u) $\mathcal{C}$, *OF* ‹$hd\ (Dec\ \mathcal{C}\ u) \#\ tl\ (Dec\ \mathcal{C}\ u) \in lists\ \mathcal{C}$›].
have $hd\ (Dec\ \mathcal{C}\ u) \neq \varepsilon$ **and** $hd\ (Dec\ \mathcal{C}\ u) \in no\text{-}head\ \mathcal{C}\ b$
using *no-head.intros*(2)[*OF* ‹$hd\ (Dec\ \mathcal{C}\ u) \in \mathcal{C}$› ‹$hd\ (Dec\ \mathcal{C}\ u) \neq b$›] ‹$hd\ (Dec\ \mathcal{C}\ u) \in \mathcal{C}$› *emp-not-in-code* **by** *auto*
from *no-head.intros*(3)[*OF this* ‹$concat\ (tl\ (decompose\ \mathcal{C}\ u)) \in \langle \mathcal{C} \rangle$›]
show $u \in no\text{-}head\ \mathcal{C}\ b$
unfolding *sym*[*OF* ‹$u = hd\ (Dec\ \mathcal{C}\ u) \cdot concat\ (tl\ (Dec\ \mathcal{C}\ u))$›].
qed *simp*

corollary $no\text{-}head\ \mathcal{C}\ b = \{u \in \langle \mathcal{C} \rangle.\ u = \varepsilon \vee hd\ (Dec\ \mathcal{C}\ u) \neq b\}$
proof(*intro equalityI subsetI*, *standard*)
fix x **assume** $x \in no\text{-}head\ \mathcal{C}\ b$
thus $x \in \langle \mathcal{C} \rangle \wedge (x = \varepsilon \vee hd\ (Dec\ \mathcal{C}\ x) \neq b)$
using *no-head-hd no-head-sub* **by** *blast*
next
fix x **assume** $x \in \{u \in \langle \mathcal{C} \rangle.\ u = \varepsilon \vee hd\ (Dec\ \mathcal{C}\ u) \neq b\}$
thus $x \in no\text{-}head\ \mathcal{C}\ b$
using *hd-no-head no-head.simps* **by** *blast*
qed

end

The set *no-head* is generated by the following set.

inductive-set *no-head-gen* :: *'a list set* ⇒ *'a list* ⇒ *'a list set*
for C b **where**
$u \in C \Longrightarrow u \neq b \Longrightarrow u \in no\text{-}head\text{-}gen\ C\ b$
| $u \in no\text{-}head\text{-}gen\ C\ b \Longrightarrow\ u \cdot b \in no\text{-}head\text{-}gen\ C\ b$

lemma *no-head-gen-set*: $no\text{-}head\text{-}gen\ C\ b = \{z \cdot b^{@}k\ |z\ k.\ z \in C \wedge z \neq b\}$
proof(*intro equalityI subsetI*)
fix x **assume** $x \in no\text{-}head\text{-}gen\ C\ b$
hence $\exists\ z\ k.\ z \in C \wedge z \neq b \wedge x = z \cdot b^{@}k$
proof (*rule no-head-gen.induct*)
fix u **assume** $u \in C$ **and** $u \neq b$
show $\exists z\ k.\ z \in C \wedge z \neq b \wedge u = z \cdot b^{@}k$
using ‹$u \in C$› ‹$u \neq b$› *pow-zero* **by** *blast*
next
fix u **assume** $u \in no\text{-}head\text{-}gen\ C\ b$ **and** $\exists z\ k.\ z \in C \wedge z \neq b \wedge u = z \cdot b^{@}k$
thus $\exists z\ k.\ z \in C \wedge z \neq b \wedge u \cdot b = z \cdot b^{@}k$
using *pow-Suc2-list append.assoc* **by** *metis*
qed

thus $x \in \{z \cdot b^{@}k \mid z\ k.\ z \in C \wedge z \neq b\}$
by *auto*
next
fix x **assume** $x \in \{z \cdot b\ ^{@}\ k \mid z\ k.\ z \in C \wedge z \neq b\}$
then obtain $z\ k$ **where** $z \in C$ **and** $z \neq b$ **and** $x = z{\cdot}b^{@}k$ **by** *blast*
then show $x \in$ *no-head-gen* $C\ b$
proof(*induct k arbitrary*: x)
case *0*
then show *?case*
by (*simp add*: ‹$z \in C$› ‹$z \neq b$› *no-head-gen.intros*(*1*))
next
case (*Suc k*)
from *this*(*1*)[*OF this*(*2*) *this*(*3*), *of* $z \cdot b\ ^{@}\ k$,
THEN no-head-gen.intros(*2*), *unfolded rassoc*,
folded pow-Suc2-list[*of b k*] ‹$x = z{\cdot}b^{@}Suc\ k$›]
show *?case*
by *blast*
qed
qed

lemma *no-head-genE*: **assumes** $u \in$ *no-head-gen* $C\ b$
obtains $z\ k$ **where** $z \in C$ **and** $z \neq b$ **and** $u = z \cdot b^{@}k$
proof(*induct rule*: *no-head-gen.induct*[*OF assms*])
case (*1 u*)
show *?case*
using *1.prems*[*OF 1.hyps*, *of 0*] **by** *simp*
next
case (*2 u*)
have $(z \cdot b^{@}k) \cdot b = z \cdot b^{@}(Suc\ k)$ **for** $z\ k$
by (*simp add*: *power-commutes*)
then show *?case*
using *2.prems 2.hyps*(*2*) **by** *blast*
qed

context *code*
begin

We show that this indeed is a set of generators

lemma *emp-not-in-Cb*: $\varepsilon \notin$ *no-head-gen* $\mathcal{C}\ b$
by (*simp add*: *emp-not-in-code no-head-gen-set*)

lemma *no-head-sub'*: $b \in \mathcal{C} \Longrightarrow$ *no-head-gen* $\mathcal{C}\ b \subseteq$ *no-head* $\mathcal{C}\ b$
proof
fix u **assume** $b \in \mathcal{C}$ $u \in$ *no-head-gen* $\mathcal{C}\ b$
show $u \in$ *no-head* $\mathcal{C}\ b$
proof (*induction rule*: *no-head-gen.induct*[*OF* ‹$u \in$ *no-head-gen* $\mathcal{C}\ b$›], *simp add*: *no-head.intros*(*2*))
case (*2 u*)
show $u \cdot b \in$ *no-head* $\mathcal{C}\ b$

```
    using no-head.intros(3)[OF - ‹u ∈ no-head C b› gen-in[OF ‹b ∈ C›]]
     2.hyps emp-not-in-Cb by blast
  qed
qed

lemma no-head-generates0: assumes v ∈ ⟨C⟩ shows
  u ≠ ε ⟶ u ∈ ⟨no-head-gen C b⟩ ⟶ u · v ∈ ⟨no-head-gen C b⟩
proof (induct arbitrary: u rule: hull.induct[OF ‹v ∈ ⟨C⟩›], simp)
  case (2 w1 w2)
  then show ?case
  proof(cases w1 = b)
    assume w1 ≠ b
    show ?thesis
      using 2.hyps(1) emp-not-in-code  no-head-gen.intros(1)[OF ‹w1 ∈ C› ‹w1 ≠
b›,THEN gen-in]
          2.hyps(3)[of w1] hull-closed[of u no-head-gen C b w1· w2] by blast
  next
    assume w1 = b
    show ?thesis
    proof (standard, standard)
      assume u ≠ ε and u ∈ ⟨no-head-gen C b⟩
      hence dec-nemp: Dec (no-head-gen C b) u ≠ ε
        using dec-nemp' by blast
      from concat-butlast-last[OF this]
      have u-w1: u · w1 = concat (butlast (Dec (no-head-gen C b) u)) · (last (Dec
(no-head-gen C b) u) · w1)
        unfolding decI[OF ‹u ∈ ⟨no-head-gen C b⟩›] by simp
         from dec-dom'[OF ‹u ∈ ⟨no-head-gen C b⟩›] append-butlast-last-id[OF
dec-nemp]
       have con-but: concat (butlast (Dec (no-head-gen C b) u)) ∈ ⟨no-head-gen C
b⟩ and last-in: last (Dec (no-head-gen C b) u) ∈ no-head-gen C b
        using append-in-lists-conv[of butlast (Dec (no-head-gen C b) u) [last (Dec
(no-head-gen C b) u)] no-head-gen C b]
          concat-in-hull'[of butlast (Dec (no-head-gen C b) u) no-head-gen C b] by
auto
      hence last (Dec (no-head-gen C b) u) · w1 ∈ no-head-gen C b
       unfolding ‹w1 = b› using  no-head-gen.intros(2)[of last (Dec (no-head-gen
C b) u) C b] by blast
      from this[THEN  gen-in,THEN[2] hull-closed, OF con-but]
      have u · w1 ∈ ⟨no-head-gen C b⟩
        unfolding u-w1.
      from 2.hyps(3)[rule-format, OF - this, unfolded rassoc]
      show u · w1 · w2 ∈ ⟨no-head-gen C b⟩
        using  pref-nemp[OF ‹u≠ε›] by blast
    qed
  qed
qed
```

theorem *no-head-generates*: **assumes** $b \in \mathcal{C}$ **shows** $\langle$*no-head-gen* $\mathcal{C}$ $b\rangle =$ *no-head* $\mathcal{C}$ b
proof (*intro equalityI*)
show $\langle$*no-head-gen* $\mathcal{C}$ $b\rangle \subseteq$ *no-head* $\mathcal{C}$ b
using *hull-mon*[*OF no-head-sub'*[*OF* ‹$b \in \mathcal{C}$›]] **unfolding** *no-head-closed*.
show *no-head* $\mathcal{C}$ $b \subseteq \langle$*no-head-gen* $\mathcal{C}$ $b\rangle$
proof (*intro subsetI*)
fix x **assume** $x \in$ *no-head* $\mathcal{C}$ b
show $x \in \langle$*no-head-gen* $\mathcal{C}$ $b\rangle$
by (*induct rule*: *no-head.induct*[*OF* ‹$x \in$ *no-head* $\mathcal{C}$ b›],*auto*, *simp add*: *gen-in no-head-gen.intros*(*1*),*simp add*: *no-head-generates0*)
qed
qed

Moreover, the generating set *no-head-gen* is a code

lemma *lists-no-head-sub*: $b \in \mathcal{C} \Longrightarrow us \in$ *lists* (*no-head-gen* $\mathcal{C}$ b) $\Longrightarrow us \in$ *lists* $\langle\mathcal{C}\rangle$
using *no-head-sub'* *no-head-sub* **by** *blast*

lemma *ref-hd*: **assumes** $z \in \mathcal{C}$ **and** $b \in \mathcal{C}$ **and** $z \neq b$ **and** $vs \in$ *lists* (*no-head-gen* $\mathcal{C}$ b)
shows *refine* $\mathcal{C}$ $((z{\cdot}b^{@}k)$ # $vs) = [z]{\cdot}[b]^{@}k \cdot$ *refine* $\mathcal{C}$ vs
proof–
have *refine* $\mathcal{C}$ $((z{\cdot}b^{@}k)$ # $vs) = (Dec\ \mathcal{C}\ (z{\cdot}b^{@}k)) \cdot$ *refine* $\mathcal{C}$ vs
using *ref-pop-hd lists-no-head-sub*[*OF* ‹$b \in \mathcal{C}$›] **by** *simp*
have $[z]{\cdot}[b]^{@}k \in$ *lists* $\mathcal{C}$
by (*induct k*, *simp add*: ‹$z \in \mathcal{C}$›, *simp add*: ‹$b \in \mathcal{C}$›)
have *concat* $([z]{\cdot}[b]^{@}k) = z \cdot b^{@}k$
using *concat-sing-pow* **by** *auto*
hence *Dec* $\mathcal{C}$ $(z{\cdot}b^{@}k) = [z]{\cdot}[b]^{@}k$
using *code.code-unique-dec*[*OF code-axioms* ‹$[z]{\cdot}[b]^{@}k \in$ *lists* $\mathcal{C}$›] **by** *auto*
thus *?thesis*
by *simp*
qed

lemma *no-head-gen-code-ind-step*:
assumes $vs \in$ *lists* (*no-head-gen* $\mathcal{C}$ b) $us \in$ *lists* (*no-head-gen* $\mathcal{C}$ b) $b \in \mathcal{C}$
and *eq*: $[b]^{@}ku \cdot$ (*refine* $\mathcal{C}$ us) $= [b]^{@}kv \cdot$ (*refine* $\mathcal{C}$ vs)
shows $ku = kv$
proof–
{fix ku :: *nat* **and** kv **and** us **and** vs **and** b
assume $kv \leq ku$ $[b]^{@}ku \cdot$ (*refine* $\mathcal{C}$ us) $= [b]^{@}kv \cdot$ (*refine* $\mathcal{C}$ vs)
$vs \in$ *lists* (*no-head-gen* $\mathcal{C}$ b) $us \in$ *lists* (*no-head-gen* $\mathcal{C}$ b) $b \in \mathcal{C}$
have *concat vs* $\in$ *no-head* $\mathcal{C}$ b
using ‹$vs \in$ *lists* (*no-head-gen* $\mathcal{C}$ b)› *no-head-generates*[*OF* ‹$b \in \mathcal{C}$›] **by** *fastforce*
have *Ref* $\mathcal{C}$ $vs =$ *Dec* $\mathcal{C}$ (*concat vs*)
using ‹$vs \in$ *lists* (*no-head-gen* $\mathcal{C}$ b)› ‹$b \in \mathcal{C}$› *code-unique-ref lists-no-head-sub* **by** *auto*

have $vs \neq \varepsilon \Longrightarrow concat\ vs \neq \varepsilon$
using *emp-not-in-Cb*[*of b*] *concat.simps*(*2*) ‹*vs* ∈ *lists* (*no-head-gen* $\mathcal{C}$ *b*)›[*unfolded lists.simps*[*of vs*]]
pref-nemp **by** *auto*
have $[b]^{@}(ku - kv) \cdot (refine\ \mathcal{C}\ us) = refine\ \mathcal{C}\ vs$
using ‹$kv \leq ku$› *eq pop-pow-cancel* ‹$[b]^{@}ku \cdot (refine\ \mathcal{C}\ us) = [b]^{@}kv \cdot (refine\ \mathcal{C}\ vs)$ ›**by** *blast*
hence $ku - kv \neq 0 \Longrightarrow hd\ (refine\ \mathcal{C}\ vs) = b \wedge vs \neq \varepsilon$
using *hd-append2*[*of* $[b]^{@}(ku - kv)$ *refine* $\mathcal{C}$ *us*] ‹$[b]^{@}(ku - kv) \cdot (refine\ \mathcal{C}\ us) = refine\ \mathcal{C}\ vs$›
hd-sing-power[*of ku − kv b*]
append-is-Nil-conv[*of* $[b]^{@}(ku - kv)$ *refine* $\mathcal{C}$ *us*] *sing-pow-empty*[*of b ku − kv*]
dec-emp[*of* $\mathcal{C}$] **by** *auto*
hence $ku = kv$
using ‹$kv \leq ku$› *no-head-hd*[*OF* ‹*concat vs* ∈ *no-head* $\mathcal{C}$ *b*›] ‹$vs \neq \varepsilon \Longrightarrow concat\ vs \neq \varepsilon$›
unfolding ‹*Ref* $\mathcal{C}$ *vs* = *Dec* $\mathcal{C}$ (*concat vs*)› **by** *auto*}
thus *?thesis* **using** *assms nat-le-linear*[*of ku kv*] **by** *metis*
qed

lemma *no-head-gen-code′*:
$b \in \mathcal{C} \Longrightarrow xs \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$
$\Longrightarrow ys \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b) \Longrightarrow concat\ xs = concat\ ys \Longrightarrow xs = ys$
proof (*induct xs ys rule*: *list-induct2′*, *simp*, *simp add*: *emp-not-in-Cb*, *simp add*: *emp-not-in-Cb*)
case (*4 hx xs hy ys*)
then show *?case*
proof−
have $hx \# xs \in lists\ \langle\mathcal{C}\rangle$ **and** $hy \# ys \in lists\ \langle\mathcal{C}\rangle$
using ‹$b \in \mathcal{C}$› ‹$hx \# xs \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› ‹$hy \# ys \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› *lists-no-head-sub* **by** *blast*+
have *eq*: $refine\ \mathcal{C}\ (hx\#xs) = refine\ \mathcal{C}\ \ (hy\#ys)$
using ‹$concat\ (hx \# xs) = concat\ (hy \# ys)$› ‹$hx \# xs \in lists\ \langle\mathcal{C}\rangle$› ‹$hy \# ys \in lists\ \langle\mathcal{C}\rangle$›
using *code-unique-ref* **by** *presburger*
have $hx \in (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$ **and** $hy \in (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$
using ‹$hx \# xs \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› ‹$hy \# ys \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› **by** *auto*+
then obtain *zx zy kx ky* **where** $hx = zx \cdot b^{@}kx$ **and** $hy = zy \cdot b^{@}ky\ zx \neq b\ zy \neq b\ zx \in \mathcal{C}\ zy \in \mathcal{C}$
using *no-head-genE* **by** *metis*
have *r1*: $refine\ \mathcal{C}\ (hx\#xs) = [zx] \cdot [b]^{@}kx \cdot refine\ \mathcal{C}\ xs$
using ‹$hx = zx \cdot b^{@}kx$› ‹$zx \in \mathcal{C}$› ‹$zx \neq b$› *ref-hd* ‹$b \in \mathcal{C}$› **by** *auto*
have *r2*: $refine\ \mathcal{C}\ (hy\#ys) = [zy] \cdot [b]^{@}ky \cdot refine\ \mathcal{C}\ ys$
using ‹$hy = zy \cdot b^{@}ky$› ‹$zy \in \mathcal{C}$› ‹$zy \neq b$› *ref-hd* ‹$b \in \mathcal{C}$› **by** *auto*
hence $zx = zy$
using *r1 r2 eq* **by** *auto*
hence $[b]^{@}kx \cdot refine\ \mathcal{C}\ xs = [b]^{@}ky \cdot refine\ \mathcal{C}\ ys$

using *eq r1 r2* **by** *auto*
hence $kx = ky$
using *no-head-gen-code-ind-step* ‹$b \in \mathcal{C}$› ‹$hx \# xs \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› ‹$hy \# ys \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$›
listsE **by** *metis*
hence $hx = hy$
by (*simp add*: ‹$hx = zx \cdot b^{@}kx$› ‹$hy = zy \cdot b^{@}ky$› ‹$zx = zy$›)
hence $xs = ys$ **using** *4.hyps*
using ‹$hx \# xs \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› ‹$hy \# ys \in lists\ (no\text{-}head\text{-}gen\ \mathcal{C}\ b)$› ‹$concat\ (hx \# xs) = concat\ (hy \# ys)$› ‹$b \in \mathcal{C}$› **by** *auto*
thus *?thesis*
by (*simp add*: ‹$hx = hy$›)
qed
qed

end

theorem *no-head-gen-code*:
assumes *code C* **and** $b \in C$
shows *code* $\{z \cdot b^{@}k \mid z\ k.\ z \in C \wedge z \neq b\}$
using *code.no-head-gen-code'*[*OF* ‹*code C*› ‹$b \in C$›] *code.intro*
unfolding *no-head-gen-set* **by** *blast*

We are now ready to prove the Graph Lemma

theorem *graph-lemma*: $\mathfrak{B}_F\ X = \{hd\ (Dec\ (\mathfrak{B}_F\ X)\ x) \mid x.\ x \in X \wedge x \neq \varepsilon\}$
proof
— the core is to show that each element of the free basis must be a head
show $\mathfrak{B}_F\ X \subseteq \{hd\ (Dec\ (\mathfrak{B}_F\ X)\ x) \mid x.\ x \in X \wedge x \neq \varepsilon\}$
proof (*rule ccontr*)
— Assume the contrary.
assume $\neg\ \mathfrak{B}_F\ X \subseteq \{hd\ (Dec\ \mathfrak{B}_F\ X\ x) \mid x.\ x \in X \wedge x \neq \varepsilon\}$
— And let b be the not-head
then obtain b **where** $b \in \mathfrak{B}_F\ X$ **and** *nohd*: $\bigwedge x.\quad x \in X \wedge x \neq \varepsilon \Longrightarrow hd\ (Dec\ (\mathfrak{B}_F\ X)\ x) \neq b$
by *blast*
interpret *code* $\mathfrak{B}_F\ X$
using *free-basis-code* **by** *auto*
— For contradiction: We have a free hull which does not contain b but contains X.
let *?Cb* = *no-head-gen* $(\mathfrak{B}_F\ X)\ b$
have *code ?Cb*
using ‹$b \in \mathfrak{B}_F\ X$› *code-def no-head-gen-code'* **by** *blast*
have $b \notin \langle ?Cb \rangle$
using ‹$b \in \mathfrak{B}_F\ X$› *b-not-in-no-head no-head-generates* **by** *blast*
have $X \subseteq \langle ?Cb \rangle$
proof
fix x **assume** $x \in X$
hence $x \in \langle \mathfrak{B}_F\ X \rangle$
using *basis-gen-hull free-hull.free-gen-in*

unfolding *free-basis-def* **by** *blast*
have $x \in X \wedge x \neq \varepsilon \Longrightarrow x \in \langle ?Cb \rangle$
using *hd-no-head*[*OF* ‹$x \in \langle \mathfrak{B}_F\ X \rangle$› *nohd*]
‹$b \in \mathfrak{B}_F\ X$› *no-head-generates* **by** *blast*
thus $x \in \langle ?Cb \rangle$
using ‹$x \in X$› **by** *blast*
qed
have $\langle X \rangle_F \subseteq \langle ?Cb \rangle$
using *free-hull-min*[*OF* ‹*code ?Cb*› ‹$X \subseteq \langle ?Cb \rangle$›].
from *this*[*unfolded subset-eq*, *rule-format*, *of b*]
show *False*
using ‹$b \in \mathfrak{B}_F\ X$› ‹$b \notin \langle ?Cb \rangle$› *basisD simp-el-el* **unfolding** *free-basis-def*
by *blast*
qed
next
— The opposite inclusion is easy
show $\{hd\ (Dec\ (\mathfrak{B}_F\ X)\ x) \mid x.\ x \in X \wedge x \neq \varepsilon\} \subseteq \mathfrak{B}_F\ X$
using *basis-gen-hull-free dec-hd genset-sub-free* **by** *blast*
qed

1.1 Binary code

We illustrate the use of the Graph Lemma in an alternative proof of the fact that two non-commuting words form a code. See also $[\![u_0 \cdot u_1 \neq u_1 \cdot u_0;\ us \in lists\ \{u_0,\ u_1\};\ vs \in lists\ \{u_0,\ u_1\};\ concat\ us = concat\ vs]\!] \Longrightarrow us = vs$ in *Combinatorics-Words.CoWBasic*.

First, we prove a lemma which is the core of the alternative proof.

lemma *non-comm-hds-neq*: **assumes** $u \cdot v \neq v \cdot u$ **shows** $hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ u) \neq hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ v)$
proof
have $u \neq \varepsilon$ **and** $v \neq \varepsilon$ **using** *assms* **by** *auto*
hence *basis*: $\mathfrak{B}_F\ \{u,v\} = \{hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ u), hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ v)\}$
using *graph-lemma*[*of* $\{u,v\}$] **by** *blast*
assume *eq*: $hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ u) = hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ v)$
hence $u \in (hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ u))*$
using *basis* **unfolding** *free-basis-def*
by (*metis basis-gen-hull free-hull.simps free-hull-hull insertI1 insert-absorb2 sing-gen*)
moreover have $v \in (hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ u))*$
using *basis eq* **unfolding** *free-basis-def*
by (*metis basis-gen-hull free-hull-hull genset-sub-free insert-absorb2 insert-subset sing-gen*)
ultimately show *False*
using *comm-root assms* **by** *blast*
qed

theorem assumes $u \cdot v \neq v \cdot u$

shows $xs \in lists\ \{u,\ v\} \Longrightarrow ys \in lists\ \{u,\ v\} \Longrightarrow concat\ xs = concat\ ys \Longrightarrow xs = ys$
proof (*induct xs ys rule*: *list-induct2′*, *simp*)
case (*2 x xs*)
then show *?case*
using *Nil-is-append-conv append-Nil assms* **by** *auto*
next
case (*3 y ys*)
then show *?case*
using *Nil-is-append-conv append-Nil assms* **by** *auto*
next
case (*4 x xs y ys*)
then show *?case*
proof−
have $u \neq \varepsilon$ **and** $v \neq \varepsilon$ **using** *assms* **by** *force*+
hence $x \neq \varepsilon$ **and** $y \neq \varepsilon$ **using** ‹$x \# xs \in lists\ \{u,\ v\}$› ‹$y \# ys \in lists\ \{u,\ v\}$› **by** *auto*
have *or-x*: $x = u \vee x = v$ **and** *or-y*: $y = u \vee y = v$ **using** ‹$x \# xs \in lists\ \{u,\ v\}$› ‹$y \# ys \in lists\ \{u,\ v\}$› **by** *auto*

have *hd-z*: $z \neq \varepsilon \Longrightarrow z \# zs \in lists\ \{u,\ v\} \Longrightarrow hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ (concat\ (z\#zs))) = hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ z)$ **for** *z zs*
proof−
assume $z \neq \varepsilon\ z \# zs \in lists\ \{u,\ v\}$
have $z \in \langle\{u,v\}\rangle_F$
using ‹$z \# zs \in lists\ \{u,\ v\}$› **by** *auto*
moreover have $concat\ zs \in \langle\{u,v\}\rangle_F$
using *concat-tl*[*OF* ‹$z \# zs \in lists\ \{u,\ v\}$›] *hull-in-free-hull*[*of* $\{u,v\}$] **by** *blast*
ultimately have $Dec\ \mathfrak{B}_F\ \{u,v\}\ (concat\ (z\#zs)) = (Dec\ \mathfrak{B}_F\ \{u,v\}\ z) \cdot (Dec\ \mathfrak{B}_F\ \{u,v\}\ (concat\ zs))$
using *free-basis-dec-morph*[*of z* $\{u,v\}$ *concat zs*] **by** *fastforce*
moreover have $Dec\ \mathfrak{B}_F\ \{u,v\}\ z \neq \varepsilon$
using ‹$z \in \langle\{u,\ v\}\rangle_F$› *basis-gen-hull-free dec-nemp′*[*OF* ‹$z \neq \varepsilon$›] **by** *blast*
ultimately show $hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ (concat\ (z\#zs))) = hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ z)$
using *hd-append* **by** *simp*
qed

have $hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ u) \neq hd\ (Dec\ \mathfrak{B}_F\ \{u,v\}\ v)$
using *non-comm-hds-neq*[*OF assms*]**.**
hence $x = y$
using *hd-z*[*OF* ‹$x \neq \varepsilon$› ‹$x \# xs \in lists\ \{u,\ v\}$› , *unfolded* ‹$concat\ (x \# xs) = concat\ (y \# ys)$› *hd-z*[*OF* ‹$y \neq \varepsilon$› ‹$y \# ys \in lists\ \{u,\ v\}$›]] *or-x or-y*
by *fastforce*
thus *?thesis*
using *4.hyps 4.prems* **by** *auto*
qed
qed

end

References

[1] J. Berstel, D. Perrin, J. Perrot, and A. Restivo. Sur le théorème du défaut. *Journal of Algebra*, 60(1):169–180, 1979.