Combinatorics on Words formalized
Two Generated Word Monoids Intersection

Štěpán Holub
Štěpán Starosta

April 18, 2024

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

Contents

1 Binary Intersection Formalized **2**

1.1 Blocks and intersection 7

1.2 Simple blocks 10

1.3 At least one block 11

1.4 Infinite case 15

1.4.1 Description of coincidence blocks 15

1.5 Description of the basis 18

1.6 Intersection 25

References **34**

theory *Two-Generated-Word-Monoids-Intersection*
imports *Combinatorics-Words.Equations-Basic Combinatorics-Words.Binary-Code-Morphisms Combinatorics-Words-Graph-Lemma.Glued-Codes*
begin

The characterization of intersection of binary languages formalized here is due to [1].

Chapter 1

Binary Intersection Formalized

locale *binary-codes-coincidence-two-generators* = *binary-codes-coincidence* +
assumes *two-coins*: $\exists\ r\ s\ r'\ s'.\ g\ r =_m h\ s \wedge g\ r' =_m h\ s' \wedge (r,s) \neq (r',s')$

begin

lemma *criticalE'*:
obtains $p\ q\ r1\ s1\ r2\ s2$ **where**
$g\ p \cdot \alpha_g = h\ q \cdot \alpha_h$ **and**
$g\ (p \cdot r1) = h\ (q \cdot s1)$ **and**
$g\ (p \cdot r2) = h\ (q \cdot s2)$ **and**
$r1 \neq \varepsilon$ **and** $r2 \neq \varepsilon$ **and**
$hd\ r1 \neq hd\ r2$
proof–
obtain $r\ s\ r'\ s'$ **where** $g\ r =_m h\ s$ **and** $g\ r' =_m h\ s'$ **and** $(r,s) \neq (r',s')$
using *two-coins* **by** *blast*
note *eqs* = *min-coinD*[*OF* ‹$g\ r =_m h\ s$›] *min-coinD*[*OF* ‹$g\ r' =_m h\ s'$›]
have $s \cdot s' \neq s' \cdot s$
proof
assume $s \cdot s' = s' \cdot s$
from *arg-cong*[*OF this*, *of h*]
have $g\ (r \cdot r') = g\ (r' \cdot r)$
unfolding *g.morph h.morph* **using** ‹$g\ r = h\ s$› ‹$g\ r' = h\ s'$› **by** *argo*
from *g.code-morph-code*[*OF this*]
have $r \cdot r' = r' \cdot r$.
from *ruler-eq*[*OF* ‹$r \cdot r' = r' \cdot r$›] *ruler-eq*[*OF* ‹$s \cdot s' = s' \cdot s$›]
have $s \leq p\ s' \Longrightarrow r \leq p\ r'$ **and** $s' \leq p\ s \Longrightarrow r' \leq p\ r$
using ‹$g\ r = h\ s$› ‹$g\ r' = h\ s'$› *g.pref-morph-pref-eq h.pref-mono* **by** *metis*+
hence $(r,\ s) = (r',\ s')$
using ‹$s \leq p\ s' \vee s' \leq p\ s$› ‹$g\ r =_m h\ s$› ‹$g\ r' =_m h\ s'$› *npI*
unfolding *min-coin-def* **by** *metis*
thus *False*
using ‹$(r,\ s) \neq (r',\ s')$› **by** *blast*

qed
hence $h\ (s \cdot s') \cdot \alpha_h \neq h\ (s' \cdot s) \cdot \alpha_h$
unfolding *cancel-right* **using** *h.code-morph-code* **by** *blast*
hence $\neg\ h\ (s \cdot s') \cdot \alpha_h \bowtie h\ (s' \cdot s) \cdot \alpha_h$
unfolding *h.morph* **using** *comm-comp-eq-conv comp-prefs-comp* **by** *metis*
hence h_m: $h\ (s \cdot s') \cdot \alpha_h \cdot \alpha \wedge_p h(s' \cdot s) \cdot \alpha_h \cdot \alpha = h\ (s \cdot s' \wedge_p s' \cdot s) \cdot \alpha_h$
using *lcp-ext-right-conv*[*of* $h\ (s \cdot s') \cdot \alpha_h\ h\ (s' \cdot s) \cdot \alpha_h\ \alpha\ \alpha$]
h.bin-code-lcp[*symmetric*] **unfolding** *h.bin-code-lcp*[*symmetric*] *rassoc* **by** *blast*

let $?p = r \cdot r' \wedge_p r' \cdot r$
let $?q = s \cdot s' \wedge_p s' \cdot s$
let $?r1 = ?p^{-1>}(r \cdot r')$
let $?r2 = ?p^{-1>}(r' \cdot r)$
let $?s1 = ?q^{-1>}(s \cdot s')$
let $?s2 = ?q^{-1>}(s' \cdot s)$

from *eqs*
have $g\ (r \cdot r') \cdot \alpha_g = h\ (s \cdot s') \cdot \alpha_h \cdot \alpha$ **and**
$g\ (r' \cdot r) \cdot \alpha_g = h\ (s' \cdot s) \cdot \alpha_h \cdot \alpha$
unfolding *g.morph h.morph lcp-diff cancel-right* **by** *auto*

hence $g\ ?p \cdot \alpha_g = h\ ?q \cdot \alpha_h$
unfolding *g.bin-code-lcp*[*symmetric*] h_m[*symmetric*] **by** *argo*

have $g\ (?p \cdot ?r1) = h\ (?q \cdot ?s1)$
unfolding *lq-pref*[*OF lcp-pref*] *g.morph h.morph* ‹$g\ r = h\ s$› ‹$g\ r' = h\ s'$›**..**
have $g\ (?p \cdot ?r2) = h\ (?q \cdot ?s2)$
unfolding *lq-pref*[*OF lcp-pref'*] *g.morph h.morph* ‹$g\ r = h\ s$› ‹$g\ r' = h\ s'$›**..**

have $r \cdot r' \neq r' \cdot r$
proof
assume $r \cdot r' = r' \cdot r$
from *arg-cong*[*OF this*, *of g*]
have $h\ (s \cdot s') = h\ (s' \cdot s)$
unfolding *g.morph h.morph* **using** ‹$g\ r = h\ s$› ‹$g\ r' = h\ s'$› **by** *argo*
from *h.code-morph-code*[*OF this*] ‹$s \cdot s' \neq s' \cdot s$›
show *False* **by** *blast*
qed
from ‹$r \cdot r' \neq r' \cdot r$›
have $\neg\ r \cdot r' \bowtie r' \cdot r$
using *comm-comp-eq* **by** *blast*

from *that*[*OF* ‹$g\ ?p \cdot \alpha_g = h\ ?q \cdot \alpha_h$› ‹$g\ (?p \cdot ?r1) = h\ (?q \cdot ?s1)$›
‹$g\ (?p \cdot ?r2) = h\ (?q \cdot ?s2)$›] *lcp-mismatch-lq*[*OF* ‹$\neg\ r \cdot r' \bowtie r' \cdot r$›]
show *thesis*
by *blast*
qed

lemma *alphas-suf*: $\alpha_h \leq s\ \alpha_g$

proof−
  from *criticalE'*
  obtain $p\ q$ **where** $g\ p \cdot \alpha_g = h\ q \cdot \alpha_h$ **by** *meson*
  from *eqd*[*reversed*, *OF this*[*symmetric*] *alphas-len*]
  show $\alpha_h \leq s\ \alpha_g$ **by** *blast*
qed

lemma *c-def*: $\mathrm{c} \cdot \alpha_h = \alpha_g$
  unfolding *critical-overflow-def* **using** *rq-suf*[*OF alphas-suf*]**.**

lemma *marked-version-solution-conv*: $g_m\ r = h_m\ s \longleftrightarrow g\ r \cdot \mathrm{c}\ = \mathrm{c} \cdot\ \ h\ s$
  unfolding *cancel-right*[*of* $g\ r \cdot \mathrm{c}\ \alpha_h\ \ \mathrm{c} \cdot h\ s$, *symmetric*] *c-def rassoc*
  g.marked-version-conjugates[*symmetric*] *h.marked-version-conjugates*[*symmetric*]
  unfolding *lassoc c-def cancel***..**

lemma *criticalE*:
  obtains $p\ q\ r1\ s1\ r2\ s2$ **where**
    $\alpha_g \cdot g_m\ p = \alpha_h \cdot h_m\ q$ **and**
    $\bigwedge p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \Longrightarrow p \leq p\ p' \wedge q \leq p\ q'$ **and**
    $g_m\ (r1 \cdot p) = h_m\ (s1 \cdot q)$ **and**
    $g_m\ (r2 \cdot p) = h_m\ (s2 \cdot q)$ **and**
    $r1 \cdot p \neq \varepsilon$ **and** $r2 \cdot p \neq \varepsilon$ **and**
    $hd\ (r1 \cdot p) \neq hd\ (r2 \cdot p)$
proof−
  from *criticalE'*
  obtain $p'\ q'\ r1\ s1\ r2\ s2$ **where**
    $g\ p' \cdot \alpha_g = h\ q' \cdot \alpha_h$ **and**
    $g\ (p' \cdot r1) = h\ (q' \cdot s1)$ **and**
    $g\ (p' \cdot r2) = h\ (q' \cdot s2)$ **and**
    $r1 \neq \varepsilon$ **and** $r2 \neq \varepsilon$ **and**
    $hd\ r1 \neq hd\ r2$**.**
  from *this*(*1*)[*folded g.marked-version-conjugates h.marked-version-conjugates*]
  have $\alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q'$**.**
  from *min-completionE*[*OF this*]
  obtain $p\ q$ **where** $\alpha_g \cdot g_m\ p = \alpha_h \cdot h_m\ q$ **and** $\bigwedge p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q'$
$\Longrightarrow p \leq p\ p' \wedge q \leq p\ q'$
    by *blast*
  show *thesis*
  proof(*rule*)
    show $\alpha_g \cdot g_m\ p = \alpha_h \cdot h_m\ q$ **by** *fact*
    hence $g\ p \cdot \mathrm{c} = h\ q$
      unfolding *g.marked-version-conjugates h.marked-version-conjugates* **unfolding** *c-def*[*symmetric*] *lassoc cancel-right***.**
    from ‹$g\ (p' \cdot r1) = h\ (q' \cdot s1)$›[*unfolded g.morph h.morph*]
    have $g\ r1 = \mathrm{c} \cdot h\ s1$
     unfolding ‹$g\ p' \cdot \alpha_g = h\ q' \cdot \alpha_h$›[*unfolded c-def*[*symmetric*] *lassoc cancel-right*, *symmetric*] *rassoc cancel***.**
    show $g_m\ (r1 \cdot p) = h_m\ (s1 \cdot q)$
      unfolding *marked-version-solution-conv g.morph h.morph rassoc* ‹$g\ p \cdot \mathrm{c} = h$

q› ‹$g\ r1 = \mathrm{c} \cdot h\ s1$›..
  from ‹$g\ (p' \cdot r2) = h\ (q' \cdot s2)$›[*unfolded g.morph h.morph*]
  have $g\ r2 = \mathrm{c} \cdot h\ s2$
   unfolding ‹$g\ p' \cdot \alpha_g = h\ q' \cdot \alpha_h$›[*unfolded c-def*[*symmetric*] *lassoc cancel-right*, *symmetric*] *rassoc cancel*.
  show $g_m\ (r2 \cdot p) = h_m\ (s2 \cdot q)$
   unfolding *marked-version-solution-conv g.morph h.morph rassoc* ‹$g\ p \cdot \mathrm{c} = h\ q$› ‹$g\ r2 = \mathrm{c} \cdot h\ s2$›..
  show $r1 \cdot p \neq \varepsilon$
   using ‹$r1 \neq \varepsilon$› **by** *blast*
  show $r2 \cdot p \neq \varepsilon$
   using ‹$r2 \neq \varepsilon$› **by** *blast*
  show $hd\ (r1 \cdot p) \neq hd\ (r2 \cdot p)$
   using ‹$hd\ r1 \neq hd\ r2$› ‹$r1 \neq \varepsilon$› ‹$r2 \neq \varepsilon$› **by** *simp*
  show $\bigwedge p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \Longrightarrow p \leq p\ p' \wedge q \leq p\ q'$ **by** *fact*
 qed
qed

Defining the beginning block

definition *beginning-block* :: *binA list* $*$ *binA list* **where**
 $beginning\text{-}block = (SOME\ pair.\ \alpha_g \cdot g_m\ (fst\ pair) = \alpha_h \cdot h_m\ (snd\ pair) \wedge$
 $(\forall\ p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \longrightarrow (fst\ pair) \leq p\ p' \wedge (snd\ pair) \leq p\ q'))$

definition *fst-beginning-block* (p) **where**
 fst-beginning-block $\equiv$ *fst beginning-block*
definition *snd-beginning-block* (q) **where**
 snd-beginning-block $\equiv$ *snd beginning-block*

lemma *begin-block*: $\alpha \cdot g_m\ p = h_m\ q$ **and**
 begin-block-min: $\alpha \cdot g_m\ p' = h_m\ q' \Longrightarrow p \leq p\ p' \wedge q \leq p\ q'$
proof−
 from *criticalE*
 obtain *pa qa r1 s1 r2 s2* **where**
  $\alpha_g \cdot g_m\ pa = \alpha_h \cdot h_m\ qa$ **and**
  $(\bigwedge p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \Longrightarrow pa \leq p\ p' \wedge qa \leq p\ q')$ **and**
  $g_m\ (r1 \cdot pa) = h_m\ (s1 \cdot qa)$ **and** $g_m\ (r2 \cdot pa) = h_m\ (s2 \cdot qa)$ **and**
  $r1 \cdot pa \neq \varepsilon$ **and** $r2 \cdot pa \neq \varepsilon$ **and** $hd\ (r1 \cdot pa) \neq hd\ (r2 \cdot pa)$ **by** *blast*
 hence $*$: $\alpha_g \cdot g_m\ (fst\ (pa,\ qa)) = \alpha_h \cdot h_m\ (snd\ (pa,\ qa)) \wedge$
  $(\forall p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \longrightarrow fst\ (pa,\ qa) \leq p\ p' \wedge snd\ (pa,\ qa) \leq p\ q')$
  unfolding *fst-conv snd-conv* **by** *fast*
 let $?P = \lambda\ pair.\ (\alpha_g \cdot g_m\ (fst\ pair) = \alpha_h \cdot h_m\ (snd\ pair) \wedge$
  $(\forall\ p'\ q'.\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \longrightarrow (fst\ pair) \leq p\ p' \wedge (snd\ pair) \leq p\ q'))$
 from *someI*[*of ?P*, *OF* $*$]
 have *pq*: $\alpha_g \cdot g_m\ p = \alpha_h \cdot h_m\ q\ \alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q' \Longrightarrow p \leq p\ p' \wedge q \leq p\ q'$
  unfolding *fst-beginning-block-def snd-beginning-block-def beginning-block-def*
  by *blast+*
 show $\alpha \cdot g_m\ p = h_m\ q$ **and** $\alpha \cdot g_m\ p' = h_m\ q' \Longrightarrow p \leq p\ p' \wedge q \leq p\ q'$
  using *pq* **unfolding** *lcp-diff*[*symmetric*] *rassoc cancel*.
qed

lemma *begin-block-conjug-conv*:
  assumes $r \cdot p = p \cdot r'$ **and** $s \cdot q = q \cdot s'$
  shows $g\ r = h\ s \longleftrightarrow g_m\ r' = h_m\ s'$
  unfolding *solution-marked-version-conv*
proof–
  have $\alpha \cdot g_m\ r = h_m\ s \cdot \alpha \longleftrightarrow \alpha \cdot g_m\ r \cdot g_m\ p = h_m\ s \cdot \alpha \cdot g_m\ p$
    unfolding *lassoc cancel-right*..
  also have ... $\longleftrightarrow \alpha \cdot g_m\ p \cdot g_m\ r' = h_m\ q \cdot h_m\ s'$
    unfolding *begin-block gm.morph*[*symmetric*] *hm.morph*[*symmetric*] *assms*..
  also have ... $\longleftrightarrow g_m\ r' = h_m\ s'$
    unfolding *lassoc begin-block cancel*..
  finally show $(\alpha \cdot g_m\ r = h_m\ s \cdot \alpha) = (g_m\ r' = h_m\ s')$.
qed

lemma *solution-ext-conv*: $g\ r = h\ s \longleftrightarrow \alpha \cdot g_m\ (r \cdot p) = h_m\ (s \cdot q)$
  unfolding *gm.morph hm.morph lassoc begin-block*[*symmetric*] *cancel-right solution-marked-version-conv*..

Both block exist

lemma *both-blocks*: *marked.blockP c*
proof–
  from *criticalE*
  obtain p' q' *r1 s1 r2 s2*
    where $\alpha_g \cdot g_m\ p' = \alpha_h \cdot h_m\ q'$
      $g_m\ (r1 \cdot p') = h_m\ (s1 \cdot q')$ $g_m\ (r2 \cdot p') = h_m\ (s2 \cdot q')$ $r1 \cdot p' \neq \varepsilon$ $r2 \cdot p' \neq \varepsilon$ $hd\ (r1 \cdot p') \neq hd\ (r2 \cdot p')$.
  let $?ua = r1 \cdot p'$ **let** $?va = s1 \cdot q'$ **let** $?ub = r2 \cdot p'$ **let** $?vb = s2 \cdot q'$
  obtain *ea fa* **where** $g_m\ (ea) =_m h_m\ (fa)$ **and** $hd\ ea = hd\ ?ua$
    using *marked.min-coin-prefE*[*OF* ‹$g_m\ (?ua) = h_m\ (?va)$› ‹$?ua \neq \varepsilon$›].
  obtain *eb fb* **where** $g_m\ (eb) =_m h_m\ (fb)$ **and** $hd\ eb = hd\ ?ub$
    using *marked.min-coin-prefE*[*OF* ‹$g_m\ ?ub = h_m\ ?vb$› ‹$?ub \neq \varepsilon$›].
  from *bin-neq-induct*[*OF* ‹$hd\ ?ua \neq hd\ ?ub$›[*folded* ‹$hd\ ea = hd\ ?ua$› ‹$hd\ eb = hd\ ?ub$›] *marked.block-ex*[*OF* ‹$g_m\ ea =_m h_m\ fa$›] *marked.block-ex*[*OF* ‹$g_m\ eb =_m h_m\ fb$›]]
  show *marked.blockP c*.
qed

notation *marked.suc-fst* ($\mathfrak{e}$) **and**
    marked.suc-snd ($\mathfrak{f}$)

lemma *sucs-eq*: $g_m\ (\mathfrak{e}\ \tau) = h_m\ (\mathfrak{f}\ \tau)$
  using *marked.blocks-eq both-blocks* **by** *blast*

sublocale *marked*: *two-binary-marked-blocks* g_m h_m
  by *unfold-locales* (*use both-blocks* **in** *fast*)

1.1 Blocks and intersection

Every solution has a block decomposition. However, not all block combinations yield a solution. This motivates the following definition.

definition *coin-block* **where** *coin-block* $\tau \ \equiv p \leq s\ p \cdot (\mathfrak{e}\ \tau) \wedge q \leq s\ q \cdot (\mathfrak{f}\ \tau)$

theorem *char-coincidence*:
$g\ r = h\ s \longleftrightarrow (\exists\ \tau.\ coin\text{-}block\ \tau \wedge r = (p \cdot \mathfrak{e}\ \tau)^{<-1}p \wedge s = (q \cdot \mathfrak{f}\ \tau)^{<-1}q)$ (**is** $g\ r = h\ s \longleftrightarrow ?Q$)
proof
assume $g\ r = h\ s$
hence $p \leq p\ r \cdot p$ **and** $q \leq p\ s \cdot q$
unfolding *solution-ext-conv* **using** *begin-block-min* **by** *blast+*
from *lq-pref*[*OF this*(*1*), *symmetric*] *lq-pref*[*OF this*(*2*), *symmetric*]
have $r \cdot p = p \cdot p^{-1>}(r \cdot p)$ **and** $s \cdot q = q \cdot q^{-1>}(s \cdot q)$**.**
hence $g_m\ (p^{-1>}(r \cdot p)) = h_m\ (q^{-1>}(s \cdot q))$
using ‹$g\ r = h\ s$› *begin-block-conjug-conv*[*of* $r\ p^{-1>}\ (r \cdot p)\ s\ q^{-1>}\ (s \cdot q)$]
by *fast*
from *marked.block-decomposition*[*OF this*]
obtain τ **where** *gsuc*: $\mathfrak{e}\ \tau = p^{-1>}(r \cdot p)$ **and** *hsuc*: $\mathfrak{f}\ \tau = q^{-1>}(s \cdot q)$**.**
note $lq = lq\text{-}pref$[*OF* ‹$p \leq p\ r \cdot p$›] *lq-pref*[*OF* ‹$q \leq p\ s \cdot q$›]
have r: $r = (p \cdot \mathfrak{e}\ \tau)^{<-1}p$ **and** s: $s = (q \cdot \mathfrak{f}\ \tau)^{<-1}q$
unfolding ‹$\mathfrak{e}\ \tau = p^{-1>}(r \cdot p)$› ‹$\mathfrak{f}\ \tau = q^{-1>}(s \cdot q)$›*lq rq-triv* **by** *simp-all*
have *coin-block* τ
unfolding *coin-block-def gsuc hsuc lq* **using** *triv-suf* **by** *blast+*
thus *?Q*
using *s r* **by** *blast*
next
assume *?Q*
then obtain τ **where** $p \leq s\ p \cdot (\mathfrak{e}\ \tau)$ **and** $q \leq s\ q \cdot (\mathfrak{f}\ \tau)$
and r: $r = (p\cdot\ (\mathfrak{e}\ \tau))^{<-1}p$ **and** s: $s = (q\cdot(\mathfrak{f}\ \tau))^{<-1}q$ **unfolding** *coin-block-def* **by** *blast*
hence *gp*: $g_m\ p \cdot g_m\ (\mathfrak{e}\ \tau) = g_m\ ((p\cdot(\mathfrak{e}\ \tau))^{<-1}p) \cdot g_m\ p$
unfolding *gm.morph*[*symmetric*] *rq-suf*[*OF* ‹$p \leq s\ p \cdot (\mathfrak{e}\ \tau)$›] **by** *blast*
have *hq*: $h_m\ q \cdot h_m\ (\mathfrak{f}\ \tau) = h_m\ ((q\cdot(\mathfrak{f}\ \tau))^{<-1}q) \cdot h_m\ q$
unfolding *hm.morph*[*symmetric*] *rq-suf*[*OF* ‹$q \leq s\ q \cdot (\mathfrak{f}\ \tau)$›] **by** *blast*
from *this*
show $g\ r = h\ s$
unfolding *begin-block*[*symmetric*] *sucs-eq*[*symmetric*] *rassoc gp*
unfolding *lassoc cancel-right*
unfolding *solution-marked-version-conv*
unfolding *r s***.**
qed

theorem *char-coincidence′*:
$g\ r = h\ s \longleftrightarrow (g_m\ (p^{-1>}(r \cdot p)) = h_m\ (q^{-1>}(s \cdot q)) \wedge p \leq p\ r \cdot p \wedge q \leq p\ s \cdot q)$ (**is** $g\ r = h\ s \longleftrightarrow ?Q$)
proof
assume $g\ r = h\ s$

from *this*[*unfolded char-coincidence coin-block-def*]
obtain *e f* **where** g_m $e = h_m$ f $p \le s$ $p \cdot e$ $q \le s$ $q \cdot f$ $r = (p \cdot e)^{<-1}p$ $s = (q \cdot f)^{<-1}q$
using *sucs-eq* **by** *blast*
have $r \cdot p = p \cdot e$ **and** $s \cdot q = q \cdot f$
unfolding ‹$r = (p \cdot e)^{<-1}p$› *rq-suf*[*OF* ‹$p \le s$ $p \cdot e$›] ‹$s = (q \cdot f)^{<-1}q$› *rq-suf*[*OF* ‹$q \le s$ $q \cdot f$›] **by** *blast+*
hence $e = p^{-1>}(r \cdot p)$ **and** $f = q^{-1>}(s \cdot q)$
using *lq-triv* **by** *fastforce+*
from ‹g_m $e = h_m$ f›[*unfolded this*]
show *?Q*
using *triv-pref* ‹$r \cdot p = p \cdot e$› ‹$s \cdot q = q \cdot f$› **by** *blast*
next
assume *?Q*
hence g_m $(p^{-1>}(r \cdot p)) = h_m$ $(q^{-1>}(s \cdot q))$ **and** $p \le p$ $r \cdot p$ **and** $q \le p$ $s \cdot q$
by *blast+*
from *this*(*1*)
show g $r = h$ s
unfolding *begin-block-conjug-conv*[*of r* $p^{-1>}(r \cdot p)$ *s* $q^{-1>}(s \cdot q)$, *OF lq-pref*[*symmetric*] *lq-pref*[*symmetric*], *OF* ‹$p \le p$ $r \cdot p$› ‹$q \le p$ $s \cdot q$›]**.**
qed

theorem *coincidence-eq-blocks*: $\mathfrak{C}$ g $h = \{((p \cdot \mathfrak{e}\ \tau)^{<-1}p,(q \cdot \mathfrak{f}\ \tau)^{<-1}q) \mid \tau.$ *coin-block* $\tau\}$
unfolding *coincidence-set-def*
using *pairs-extensional'*[*OF char-coincidence*]**.**

lemma
minblock0: g_m $(\mathfrak{e}\ \mathfrak{a}) =_m h_m$ $(\mathfrak{f}\ \mathfrak{a})$ **and**
minblock1: g_m $(\mathfrak{e}\ \mathfrak{b}) =_m h_m$ $(\mathfrak{f}\ \mathfrak{b})$ **and**
hdblock0: *hd* $(\mathfrak{e}\ \mathfrak{a})$ = *bina* **and**
hdblock1: *hd* $(\mathfrak{e}\ \mathfrak{b})$ = *binb*
using *marked.blockP-D both-blocks marked.blockP-D-hd* **by** *blast+*

definition $\mathcal{T}$ **where** $\mathcal{T} \equiv \{\tau$. *coin-block* $\tau\}$

lemma $\mathcal{T}$*-def'*: $\tau \in \mathcal{T} \longleftrightarrow$ *coin-block* τ
unfolding $\mathcal{T}$*-def mem-Collect-eq***..**

Properties of the set of coincidence blocks

lemma $\mathcal{T}$*-closed*: **assumes** *coin-block* τ_1 **and** *coin-block* τ_2
shows *coin-block* $(\tau_1 \cdot \tau_2)$
proof−
from *assms*
have $p \le s$ $p \cdot \mathfrak{e}\ \tau_2$ **and** $p \le s$ $p \cdot \mathfrak{e}\ \tau_1$ **and**
$q \le s$ $q \cdot \mathfrak{f}\ \tau_2$ **and** $q \le s$ $q \cdot \mathfrak{f}\ \tau_1$
unfolding *coin-block-def* **by** *blast+*
from *suf-prolong*[*OF this*(*1*−*2*), *unfolded rassoc*] *suf-prolong*[*OF this*(*3*−*4*), *unfolded rassoc*]

show *coin-block* $(\tau_1{\cdot}\tau_2)$
unfolding *coin-block-def marked.sucs.h.morph marked.sucs.g.morph* **by** *blast*
qed

lemma *emp-block*: *coin-block* ε
unfolding *coin-block-def marked.sucs.g.emp-to-emp marked.sucs.h.emp-to-emp*
by *force*

lemma $\mathcal{T}$*-hull*: $\langle\mathcal{T}\rangle = \mathcal{T}$
proof (*intro hull-I*)
show $\varepsilon \in \mathcal{T}$
unfolding $\mathcal{T}$*-def'* *coin-block-def marked.sucs.g.emp-to-emp marked.sucs.h.emp-to-emp*
by *force*
show $\bigwedge x\ y.\ x \in \mathcal{T} \Longrightarrow y \in \mathcal{T} \Longrightarrow x \cdot y \in \mathcal{T}$
unfolding $\mathcal{T}$*-def'* **using** $\mathcal{T}$*-closed*.
qed

lemma $\mathcal{T}$*-pref*: *coin-block* $\tau_1 \Longrightarrow$ *coin-block* $(\tau_1 \cdot \tau_2) \Longrightarrow$ *coin-block* τ_2
using *suf-prod-suf*[*of* $p\ p \cdot \mathfrak{e}\ \tau_1\ \mathfrak{e}\ \tau_2$]
suf-prod-suf[*of* $q\ q \cdot \mathfrak{f}\ \tau_1\ \mathfrak{f}\ \tau_2$]
unfolding *coin-block-def marked.sucs.g.morph marked.sucs.h.morph rassoc*
by *blast*

Translation from blocks to the intersection

lemma *translate-coin-blocks-to-intersection*:
$(h \circ (\lambda\ x.\ (q \cdot x)^{<-1}q) \circ \mathfrak{f})$ ‘ $\mathcal{T}$ = *range* $g \cap$ *range* h
unfolding *coin-set-inter-snd*[*of* $h\ g$, *unfolded coincidence-eq-blocks*, *symmetric*]
$\mathcal{T}$*-def*
proof–
have $(h \circ snd)$ ‘ $\{(F\ x,\ G\ x) \mid x.\ \textit{coin-block}\ x\} = \{h\ (G\ x) \mid x.\ \textit{coin-block}\ x\}$
for $F\ G$ :: *binA list* $\Rightarrow$ *binA list*
by (*standard*, *standard*, *auto*, *force*)
note *rule1* = *this*[*of* $\lambda\tau.\ (p \cdot \mathfrak{e}\ \tau)^{<-1}p\ \lambda\tau.\ (q \cdot \mathfrak{f}\ \tau)^{<-1}q$]
have $(h \circ I \circ \mathfrak{f})$ ‘ $\{x\ .\ \textit{coin-block}\ x\} = \{h\ (I\ (\mathfrak{f}\ x)) \mid x.\ \textit{coin-block}\ x\}$ **for** I
by *rule auto*
from *this*[*of* $(\lambda x.\ (q \cdot x)^{<-1}q)$, *folded rule1*]
show $(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})$ ‘ *Collect coin-block* =
$(h \circ snd)$ ‘ $\{((p \cdot \mathfrak{e}\ \tau)^{<-1}p\ ,\ (q \cdot \mathfrak{f}\ \tau)^{<-1}q\ ) \mid \tau.\ \textit{coin-block}\ \tau\}$.
qed

lemma *translation-blocks-inj*:
inj-on $(h \circ (\lambda\ x.\ (q \cdot x)^{<-1}q) \circ \mathfrak{f})\ \langle\mathcal{T}\rangle$
proof
fix $x\ y$ **assume** $x \in \langle\mathcal{T}\rangle$ **and** $y \in \langle\mathcal{T}\rangle$
hence $q \le s\ q \cdot \mathfrak{f}\ x$ **and** $q \le s\ q \cdot \mathfrak{f}\ y$ **unfolding** $\mathcal{T}$*-def'* $\mathcal{T}$*-hull coin-block-def* **by** *blast+*
assume $(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ x = (h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ y$
hence $h\ ((q \cdot \mathfrak{f}\ x)^{<-1}q) = h\ ((q \cdot \mathfrak{f}\ y)^{<-1}q)$
by *simp*

from *h.code-morph-code*[*OF this*] *rq-suf*[*OF* ‹$q \leq s\ q \cdot \mathfrak{f}\ x$›] *rq-suf*[*OF* ‹$q \leq s\ q \cdot \mathfrak{f}\ y$›]
have $\mathfrak{f}\ x = \mathfrak{f}\ y$
unfolding *cancel*[*of* $q\ \mathfrak{f}\ x\ \mathfrak{f}\ y$, *symmetric*] **by** *argo*
thus $x = y$
using *marked.sucs.h.code-morph-code* **by** *blast*
qed

lemma *translation-blocks-morph-on*: *morphism-on* $(h \circ (\lambda\ x.\ (q \cdot x)^{<-1}q) \circ \mathfrak{f})\ \mathcal{T}$
proof
fix $x\ y$ **assume** $x \in \langle\mathcal{T}\rangle$ **and** $y \in \langle\mathcal{T}\rangle$
hence $q \leq s\ q \cdot \mathfrak{f}\ x$ **and** $q \leq s\ q \cdot \mathfrak{f}\ y$
unfolding $\mathcal{T}$*-hull* $\mathcal{T}$*-def'* *coin-block-def* **by** *blast+*
show $(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ (x \cdot y) =$
$(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ x\cdot\ (h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ y$
unfolding *comp-apply h.morph*[*symmetric*] *rq-reassoc*[*OF* ‹$q \leq s\ q \cdot \mathfrak{f}\ y$›] *lassoc* *rq-suf*[*OF* ‹$q \leq s\ q \cdot \mathfrak{f}\ x$›]
unfolding *rassoc marked.sucs.h.morph*..
qed

interpretation *morphism-on* $(h \circ (\lambda\ x.\ (q \cdot x)^{<-1}q) \circ \mathfrak{f})\ \mathcal{T}$
using *translation-blocks-morph-on*.

theorem *inter-basis*: $\mathfrak{B}$ (*range* $g \cap$ *range* h) $= (h \circ (\lambda\ x.\ (q \cdot x)^{<-1}q) \circ \mathfrak{f})$ ‘ $(\mathfrak{B}\ \mathcal{T})$
using *inj-basis-to-basis*[*OF translation-blocks-inj*, *unfolded* $\mathcal{T}$*-hull*]
translate-coin-blocks-to-intersection **by** *presburger*

1.2 Simple blocks

If both letters are blocks, the situation is easy

theorem *simple-blocks*: **assumes** $\bigwedge$ *a. coin-block* [*a*] **shows**
coin-block τ
by (*induct* τ, *simp add*: *emp-block*)
(*use assms* $\mathcal{T}$*-closed*[*OF assms*] *hd-word* **in** *force*)

theorem *simple-blocks-UNIV*: ($\bigwedge$ *a. coin-block* [*a*]) $\Longrightarrow \mathcal{T} =$ *UNIV*
using *simple-blocks* $\mathcal{T}$*-def'* **by** *auto*

theorem *simple-blocks-basis*: **assumes** $\bigwedge$*a. coin-block* [*a*]
shows $\mathfrak{B}\ \mathcal{T} = \{\mathfrak{a},\ \mathfrak{b}\}$
using *basis-of-hull*[*of* $\{\mathfrak{a},\mathfrak{b}\}$] *code.code-is-basis*[*OF bin-basis-code*]
unfolding *bin-basis-generates simple-blocks-UNIV*[*OF assms*, *symmetric*]
by *argo*

1.3 At least one block

At least one letter – the last one – is a block

lemma *last-letter-fst-suf*: **assumes** *coin-block* $(z \cdot [c])$
shows $p <s\ \mathfrak{e}\ [c]$
proof–
from *assms*
have $p \leq s\ p \cdot \mathfrak{e}\ (z \cdot [c])$ **and** $q \leq s\ q \cdot \mathfrak{f}\ (z \cdot [c])$
unfolding *coin-block-def* **by** *blast+*
hence $p \bowtie_s \mathfrak{e}\ [c]$ **and** $q \bowtie_s \mathfrak{f}\ [c]$
unfolding *marked.sucs.g.morph marked.sucs.h.morph lassoc* **using** *ruler-suf''*
by *blast+*
have $\neg\ \mathfrak{e}\ [c] \leq s\ p$
proof
assume $\mathfrak{e}\ [c] \leq s\ p$
hence $g_m\ (\mathfrak{e}\ [c]) \leq s\ \alpha \cdot g_m\ p$
using *gm.suf-mono suf-ext* **by** *blast*
hence $h_m\ (\mathfrak{f}\ [c]) \leq s\ h_m\ q$
unfolding *begin-block sucs-eq*.
hence $\mathfrak{f}\ [c] \leq s\ q$
using *hm.suf-mono*
‹$q \bowtie_s \mathfrak{f}\ [c]$›[*unfolded suf-comp-or*] *hm.code-morph-code suffix-order.antisym*
by *metis*
have $\alpha \cdot g_m\ (p^{<-1}\mathfrak{e}\ [c] \cdot \mathfrak{e}\ [c]) = h_m\ (q^{<-1}\mathfrak{f}[c] \cdot \mathfrak{f}[c])$
unfolding *rq-suf*[*OF* ‹$\mathfrak{e}\ [c] \leq s\ p$›] *rq-suf*[*OF* ‹$\mathfrak{f}\ [c] \leq s\ q$›] *begin-block*[*symmetric*]..
hence $\alpha \cdot g_m\ (p^{<-1}\mathfrak{e}\ [c]) = h_m\ (q^{<-1}\mathfrak{f}[c])$
unfolding *gm.morph hm.morph marked.block-eq*[*OF both-blocks*] *lassoc cancel-right*.
from *conjunct1*[*OF begin-block-min*[*OF this*]]
have $\mathfrak{e}\ [c] = \varepsilon$
using *rq-suf*[*OF* ‹$\mathfrak{e}\ [c] \leq s\ p$›] *same-prefix-nil* **by** *metis*
thus *False*
using *marked.sucs.g.sing-to-nemp* **by** *blast*
qed
thus $p <s\ \mathfrak{e}\ [c]$
unfolding *strict-suffix-def* **using** ‹$p \bowtie_s \mathfrak{e}\ [c]$›[*unfolded suf-comp-or*]
by *metis*
qed

lemma *rich-block-suf-fst'*:
assumes *coin-block* $(z \cdot [1-c] \cdot [c]^{@} Suc\ i)$
shows *gm.bin-code-lcs* $\cdot\ g_m\ p \leq s\ g_m\ (\mathfrak{e}\ ([1-c]\cdot[c]^{@} Suc\ i))$
proof–
from *last-letter-fst-suf assms*[*unfolded pow-Suc' lassoc*]
have $p <s\ \mathfrak{e}\ [c]$
by *blast*
hence $\mathfrak{e}\ [c] = [c] \cdot tl\ (\mathfrak{e}\ [c])$
using *marked.blockP-D-hd*[*OF both-blocks*[*of c*]] *hd-tl*[*OF marked.sucs.g.sing-to-nemp*]
by *metis*

then obtain p' **where** $\mathfrak{e}\ [c] = [c] \cdot p' \cdot p$
  using *ssufE*[*OF* ‹$p <s\ \mathfrak{e}\ [c]$›] *ssuf-tl-suf suffix-def* **by** *metis*
hence $*$: $\mathfrak{e}([1{-}c] \cdot [c]^{@}Suc\ i) = \mathfrak{e}\ ([1{-}c] \cdot [c]^{@}i) \cdot [c] \cdot p' \cdot p$
  unfolding *pow-Suc′ marked.sucs.g.morph* **by** *force*
have *f1*: $[c] \le f\ \mathfrak{e}\ ([1 - c] \cdot [c]\ ^{@}\ i) \cdot [c] \cdot p'$
  by *fast*
have $[1 - c] \le f\ ([1 - c] \cdot tl\ (\mathfrak{e}\ [1 - c])) \cdot \mathfrak{e}\ ([c]\ ^{@}\ i) \cdot [c] \cdot p'$
  unfolding *rassoc* **by** *blast*
from *this*[*unfolded hd-tl*[*OF marked.sucs.g.sing-to-nemp*, *of* $1{-}c$, *unfolded marked.blockP-D-hd*[*OF both-blocks*[*of* $1{-}c$]]]]
have *f2*: $[1{-}c] \le f\ \mathfrak{e}\ ([1 - c] \cdot [c]\ ^{@}\ i) \cdot [c] \cdot p'$
  unfolding *marked.sucs.g.morph rassoc*.
from *marked.revs.g.bin-lcp-pref′′*[*reversed*, *OF f1 f2*, *unfolded g.marked-lcs*] *g.marked-lcs*
show $gm.bin\text{-}code\text{-}lcs \cdot g_m\ p \le s\ g_m\ (\mathfrak{e}\ ([1{-}c]{\cdot}[c]^{@}Suc\ i))$
  unfolding $*$ *gm.morph lassoc suf-cancel-conv lcp-diff*[*symmetric*] **by** *simp*
qed

lemma *rich-block-suf-fst*:
  assumes *coin-block* $(z \cdot [1{-}c] \cdot [c]^{@}Suc\ i)$
  shows $\alpha \cdot g_m\ (p) \le s\ g_m\ (\mathfrak{e}\ ([1{-}c]{\cdot}[c]^{@}Suc\ i))$
  using *rich-block-suf-fst′*[*OF assms*]
  using *g.marked-lcs lcp-diff*[*symmetric*] *rassoc*
  using *suf-extD* **by** *metis*

lemma *rich-block-suf-snd′*:
  assumes *coin-block* $(z \cdot [1{-}c] \cdot [c]^{@}Suc\ i)$
  shows $\alpha_h \cdot h_m\ q \le s\ h_m\ (\mathfrak{f}\ ([1{-}c]{\cdot}[c]^{@}Suc\ i))$
using *rich-block-suf-fst′*[*OF assms*, *unfolded marked.suc-eq′*[*OF both-blocks*] *g.marked-lcs rassoc*]
  unfolding *lcp-diff*[*symmetric*] *rassoc begin-block*
  using *suf-extD* **by** *blast*

lemma *rich-block-suf-snd*:
  assumes *coin-block* $(z \cdot [1{-}c] \cdot [c]^{@}Suc\ i)$
  shows $q \le s\ \mathfrak{f}\ ([1{-}c]{\cdot}[c]^{@}Suc\ i)$
proof(*rule ccontr*)
  assume *notsuf*: $\neg\ q \le s\ \mathfrak{f}\ ([1 - c] \cdot [c]\ ^{@}\ Suc\ i)$
  from *conjunct2*[*OF assms*[*unfolded coin-block-def*]]
  have $q \le s\ (q \cdot \mathfrak{f}\ z) \cdot \mathfrak{f}\ ([1 - c] \cdot [c]\ ^{@}\ Suc\ i)$
    unfolding *marked.sucs.h.morph rassoc*.
  note *ruler* = *suf-ruler*[*OF this triv-suf*]
  from *this*
  have $\mathfrak{f}\ ([1 - c] \cdot [c]\ ^{@}\ Suc\ i) <s\ q$
    using *notsuf suffix-order.less-le-not-le* **by** *blast*
  from *hm.ssuf-mono*[*OF this*] *rich-block-suf-fst*[*OF assms*, *unfolded marked.suc-eq′*[*OF both-blocks*] *begin-block*]
  show *False* **by** *force*
qed

lemma *last-letter-block*: **assumes** *coin-block* $(z \cdot [c])$
shows *coin-block* $[c]$
proof (*cases*)
assume $z \in [c]*$
from *sing-pow-exp*[*OF this*]
obtain i **where** $z = [c]^{@}i$
by *blast*
have $z \cdot [c] = [c]^{@}Suc\ i$
unfolding ‹$z = [c]^{@}i$› *pow-Suc′*..
have $\mathfrak{e}\ (z \cdot [c]) = (\mathfrak{e}\ [c])^{@}Suc\ i$ **and** $\mathfrak{f}\ (z \cdot [c]) = (\mathfrak{f}\ [c])^{@}Suc\ i$
unfolding ‹$z \cdot [c] = [c]^{@}Suc\ i$› *marked.sucs.g.pow-morph marked.sucs.h.pow-morph*
by *simp-all*
from ‹*coin-block* $(z \cdot [c])$›[*unfolded coin-block-def this*]
show *coin-block* $[c]$
unfolding *coin-block-def* **using** *per-drop-exp-rev*[*OF zero-less-Suc*] **by** *metis*
next
assume $z \notin [c]*$
from *distinct-letter-in-suf*[*OF this*]
obtain $t\ z'\ b$ **where** z: $z = z' \cdot [b] \cdot [c]^{@}t$ **and** $b \neq c$
unfolding *suffix-def* **by** *metis*
have $p <s\ \mathfrak{e}\ [c]$
using *last-letter-fst-suf*[*OF* ‹*coin-block* $(z \cdot [c])$›].
from *ssufD*[*OF this*, *unfolded suffix-def*]
obtain p' **where** $p' \cdot p = \mathfrak{e}[c]$ **and** $p' \neq \varepsilon$ **by** *force*
hence $hd\ p' = c$
using *marked.blockP-D-hd*[*OF both-blocks*[*of c*]] *hd-append2*[*OF* ‹$p' \neq \varepsilon$›, *of p*]
by *argo*
hence $\mathfrak{e}[c] = [c] \cdot tl\ p' \cdot p$
unfolding ‹$p' \cdot p = \mathfrak{e}[c]$›[*symmetric*] **using** *hd-tl*[*OF* ‹$p' \neq \varepsilon$›] **by** *simp*
show *coin-block* $[c]$
proof(*cases*)
assume $q \leq s\ q \cdot \mathfrak{f}\ [c]$
thus *coin-block* $[c]$
unfolding *coin-block-def* **using** *ssufD1*[*OF ssuf-ext*[*OF* ‹$p <s\ \mathfrak{e}\ [c]$›]] **by** *blast*
next — the other option leads to a contradiction
write
marked.sucs.h.bin-morph-mismatch-suf ($\mathfrak{d}$) **and**
marked.sucs.h.bin-code-lcs ($\beta_{\mathfrak{h}}$) **and**
hm.bin-code-lcs (β_H) **and**
gm.bin-code-lcs (β_G) **and**
g.bin-code-lcs (β_g)
assume $\neg\ q \leq s\ q \cdot \mathfrak{f}\ [c]$
— suffix of q
hence $\neg\ q \leq s\ q \cdot \mathfrak{f}\ ([c]^{@}\ Suc\ t)$
unfolding *marked.sucs.h.pow-morph* **using** *per-drop-exp′*[*reversed*] **by** *blast*
hence $\neg\ q \leq s\ \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]^{@}Suc\ t)$
using *suf-prolong-per-root*[*OF - marked.sucs.revs.h.bin-lcp-pref-all*[*reversed*], *of q* $[c]^{@}Suc\ t$] **by** *blast*

— analysis of q
have $q \le s\ q \cdot \mathfrak{f}(z' \cdot [b] \cdot [c]^{@} Suc\ t)$
using ‹*coin-block* $(z \cdot [c])$›
unfolding *z coin-block-def rassoc pow-Suc′* **by** *blast*
note *per-exp-pref*[*reversed*, *OF this*, *of 2*, *unfolded pow-two*]
hence *suf1*: $q \le s\ q \cdot \mathfrak{f}\ (z' \cdot [b]) \cdot \mathfrak{f}\ ([c]\ ^{@}\ Suc\ t \cdot z' \cdot [b]) \cdot \mathfrak{f}\ ([c]\ ^{@}\ Suc\ t)$
unfolding *marked.sucs.h.morph rassoc*.
have $[\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}} \le s\ \mathfrak{f}\ ([c]^{@} Suc\ t \cdot z') \cdot \mathfrak{f}\ [b]$
by (*rule marked.sucs.revs.h.bin-lcp-mismatch-pref-all-set*[*reversed*])
(*unfold bin-neq-swap*[*OF* ‹$b \neq c$›], *simp*)
from *this*[*folded marked.sucs.h.morph lassoc*, *unfolded suffix-def*]
obtain *zs* **where** $\mathfrak{f}\ ([c]\ ^{@}\ Suc\ t \cdot z' \cdot [b]) = zs \cdot [\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}}$
by *blast*
have *suf2*: $[\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]^{@} Suc\ t) \le s\ q \cdot \mathfrak{f}\ (z' \cdot [b]) \cdot \mathfrak{f}\ ([c]\ ^{@}\ Suc\ t \cdot z' \cdot [b]) \cdot \mathfrak{f}\ ([c]\ ^{@}\ Suc\ t)$
unfolding ‹$\mathfrak{f}\ ([c]\ ^{@}\ Suc\ t \cdot z' \cdot [b]) = zs \cdot [\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}}$›
using *triv-suf*[*of* $[\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]\ ^{@}\ Suc\ t)\ q \cdot \mathfrak{f}\ (z' \cdot [b]) \cdot zs$] **unfolding** *rassoc*.
have $q \bowtie_s [\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]^{@} Suc\ t)$
using *ruler*[*reversed*, *OF suf1 suf2*] **unfolding** *suf-comp-or*.
with ‹$\neg\ q \le s\ \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]^{@} Suc\ t)$›
have $[\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]^{@} Suc\ t) \le s\ q$
unfolding *suf-comp-or hd-word*[*symmetric*] *suffix-Cons* **using** *suffix-order.eq-refl*[*OF sym*, *of q*] **by** *blast*
from *suffixE*[*OF this*]
obtain q' **where** *q-factors*: $q = q' \cdot [\mathfrak{d}\ b] \cdot \beta_{\mathfrak{h}} \cdot \mathfrak{f}\ ([c]\ ^{@}\ Suc\ t)$.

— length of $\beta_H \wedge_p \alpha$
— 1. inequality
from *marked.lcs-fst-suf-snd*
have $\beta_G \le s\ \beta_H \cdot h_m\ \beta_{\mathfrak{h}}$.
from *suf-len*[*OF this*, *unfolded lenmorph*]
have *ineq1*: $|\beta_G| \le |\beta_H| + |h_m\ \beta_{\mathfrak{h}}|$
using *lenarg*[*OF lcp-diff*, *unfolded lenmorph*] **by** *linarith*
— 2. inequality
from *begin-block*[*unfolded q-factors*, *unfolded pow-Suc′ marked.sucs.h.morph hm.morph*, *folded sucs-eq*[*of* $[c]$], *unfolded* ‹$\mathfrak{c}[c] = [c] \cdot tl\ p' \cdot p$› *gm.morph lassoc cancel-right*, *unfolded rassoc*]
have $\alpha = h_m\ q' \cdot h_m\ [\mathfrak{d}\ b] \cdot h_m\ \beta_{\mathfrak{h}} \cdot h_m\ (\mathfrak{f}\ ([c]\ ^{@}\ t)) \cdot g_m\ [c] \cdot g_m\ (tl\ p')$.
from *lenarg*[*OF this*] *lenarg*[*OF lcp-diff*]
have *ineq2*: $|h_m\ [\mathfrak{d}\ b]| + |g_m\ [c]| + |h_m\ \beta_{\mathfrak{h}}| \le |\alpha_g|$
unfolding *lenmorph* **by** *linarith*
— conclusions
have *concl1*: $|h_m\ [\mathfrak{d}\ b]| + |g_m\ [c]| \le |\alpha_g|$
using *ineq2* **by** *linarith*
have *concl2*: $|h_m\ [\mathfrak{d}\ b]| + |g_m\ [c]| \le |\beta_H|$
using *ineq1 ineq2 lenarg*[*OF g.marked-lcs*, *unfolded lenmorph*]
by *linarith*
from *suf-comp-monotone*[*OF marked.suf-comp-lcs*] *sufI*[*OF g.marked-lcs*[*symmetric*]]

have $\alpha_g \bowtie_s \beta_H$
by *blast*
have *concl*: $|h_m\ [\mathfrak{d}\ b]| + |g_m\ [c]| \leq |\alpha_g \wedge_s \beta_H|$
by (*rule disjE*[*OF* ‹$\alpha_g \bowtie_s \beta_H$›[*unfolded suf-comp-or*], *unfolded lcs-suf-conv*[*symmetric*] *lcs-sym*[*of* β_H]])
(*use concl1 concl2* **in** *argo*)+

— two periods of $\alpha_g \wedge_s \beta_H$
have $\alpha_g \leq s\ \alpha_g \cdot g_m\ [c]$
unfolding *g.marked-version-conjugates* **by** *blast*
hence *per1*: $\alpha_g \wedge_s \beta_H \leq s\ (\alpha_g \wedge_s \beta_H) \cdot g_m\ [c]$
using *lcs-suf suf-keeps-root* **by** *blast*
have $\beta_H \leq s\ \beta_H \cdot h_m\ [\mathfrak{d}\ b]$
using *marked.revs.h.bin-lcp-pref-all*[*reversed*]**.**
hence *per2*: $\alpha_g \wedge_s \beta_H \leq s\ (\alpha_g \wedge_s \beta_H) \cdot h_m\ [\mathfrak{d}\ b]$
using *lcs-suf′ suf-keeps-root* **by** *blast*
from *two-pers*[*reversed*, *OF per2 per1 concl*]
have $g_m\ [c] \cdot h_m\ [\mathfrak{d}\ b] = h_m\ [\mathfrak{d}\ b] \cdot g_m\ [c]$**.**

from *marked.comm-sings-block*[*OF this*]
obtain n **where** $\mathfrak{f}\ [c] = [\mathfrak{d}\ b]\ ^{@}\ Suc\ n$ **by** *blast*
from *marked.sucs.h.sing-pow-mismatch-suf*[*OF* ‹$\mathfrak{f}\ [c] = [\mathfrak{d}\ b]\ ^{@}\ Suc\ n$›]
‹$b \neq c$› *marked.sucs.h.bin-mismatch-suf-inj*
have *False*
unfolding *inj-on-def* **by** *blast*
thus *coin-block* $[c]$
by *blast*
qed
qed

end

1.4 Infinite case

locale *binary-codes-coincidence-infinite* = *binary-codes-coincidence-two-generators*
for *a1* +
assumes *non-block*: ¬ *coin-block* [*a1*]

begin

1.4.1 Description of coincidence blocks

lemma *swap-coin-block*: *coin-block* [*1−a1*]
proof−
obtain $u\ v$ **where** $g\ u =_m h\ v$
using *coin-ex* **by** *blast*
from *min-coinD*[*OF this*, *unfolded char-coincidence*]
obtain τ **where** *coin-block* τ **and** $u = (p \cdot \mathfrak{e}\ \tau)^{<-1}p$
by *blast*

from *conjunct1*[*OF min-coinD'*[*OF* ‹$g\ u =_m h\ v$›], *unfolded this*(*2*)]
have $\tau \neq \varepsilon$
using *rq-self*[*of p*] *marked.sucs.g.emp-to-emp emp-simps*(*1*) **by** *metis*
from *append-butlast-last-id*[*OF this*]
have *coin-block* [*last* τ]
using ‹*coin-block* τ› *last-letter-block* **by** *metis*
with *non-block*
show *coin-block* [*1−a1*]
by (*cases rule*: *bin-swap-exhaust*[*of last* τ *a1*]) *simp-all*
qed

definition *coincidence-exponent* (t) **where**
coincidence-exponent = (*LEAST x*. ($q \le s\ q \cdot \mathfrak{f}([a1] \cdot [1{-}a1]^{@} Suc\ x)$))

lemma *q-nemp*: $q \neq \varepsilon$
proof (*rule notI*)
assume $q = \varepsilon$
with *coin-block-def*
marked.ne-g[*OF suf-of-emp*[*OF begin-block*[*unfolded hm.emp-to-emp'*[*OF this*]]]]
non-block
show *False* **by** *blast*
qed

lemma *p-suf*: $p <s\ \mathfrak{e}\ [1{-}a1]$
using *last-letter-fst-suf*[*of* ε, *unfolded emp-simps*, *OF swap-coin-block*]**.**

lemma *coin-exp*: *coin-block* ($[a1] \cdot [1{-}a1]^{@} Suc\ t$) **and**
coin-exp-min: $j \le t \Longrightarrow \neg$ *coin-block* ($[a1] \cdot [1{-}a1]^{@} j$)
proof−
have $|q| \le |\mathfrak{f}\ ([1{-}a1]^{@}|q|)|$
using *long-pow marked.sucs.h.pow-morph marked.sucs.h.sing-to-nemp* **by** *metis*
moreover have $q \le s\ q \cdot \mathfrak{f}\ ([1{-}a1]^{@}|q|)$
unfolding *marked.sucs.h.pow-morph* **using** *conjunct2*[*OF swap-coin-block*[*unfolded*
coin-block-def]] **using** *per-exp-suf* **by** *blast*
ultimately have $q \le s\ \mathfrak{f}\ ([a1] \cdot [1{-}a1]^{@}|q|)$
unfolding *marked.sucs.h.morph* **using** *suf-prod-le suf-ext* **by** *blast*
from *LeastI*[*of* λ *x*. ($q \le s\ q \cdot \mathfrak{f}([a1] \cdot [1{-}a1]^{@} Suc\ x)$),
folded coincidence-exponent-def, *of* $|q| - 1$] *suf-ext*[*OF this*, *of q*]
have $q \le s\ q \cdot \mathfrak{f}\ ([a1] \cdot [1 - a1]\ ^{@}\ Suc\ t)$
unfolding *Suc-minus*[*OF nemp-len*[*OF q-nemp*]] **by** *blast*
thus *coin-block* ($[a1] \cdot [1{-}a1]^{@} Suc\ t$)
unfolding *pow-Suc' marked.sucs.g.morph coin-block-def*
using *suf-ext*[*OF ssufD1*[*OF p-suf*], *of* $p \cdot \mathfrak{e}\ [a1] \cdot \mathfrak{e}\ ([1 - a1]\ ^{@}\ t)$, *unfolded*
rassoc] **by** *blast*
next
fix j **assume** $j \le t$
show $\neg$ *coin-block* ($[a1] \cdot [1{-}a1]^{@} j$)
proof (*cases* $j = 0$, *simp add*: *non-block*)
assume $j \neq 0$

hence $j - 1 < t$ **and** $t \neq 0$
using ‹$j \leq t$› ‹$j \neq 0$› **by** *simp-all*
thus $\neg$ *coin-block* $([a1] \cdot [1{-}a1]^{@}j)$
using *not-less-Least*[*of* $j - 1$ λ x. $q \leq s$ $q \cdot \mathfrak{f}$ $([a1] \cdot [1 - a1]$ $^{@}$ *Suc* $x)$, *folded coincidence-exponent-def*]
unfolding *coin-block-def Suc-minus*[*OF* ‹$j \neq 0$›] **by** *linarith*
qed
qed

lemma *exp-min*: $\neg$ $q \leq s$ $\mathfrak{f}$ $[1{-}a1]^{@}t$
proof (*cases* $t = 0$, *simp add*: *q-nemp*)
assume $t \neq 0$
hence $t - 1 < t$ **by** *simp*
show *?thesis*
using *not-less-Least*[*of* $t - 1$ λ m. $q \leq s$ $q \cdot \mathfrak{f}$ $([a1] \cdot [1 - a1]$ $^{@}$ *Suc* $m)$, *folded coincidence-exponent-def*, *OF* ‹$t - 1 < t$›, *unfolded marked.sucs.h.morph Suc-minus*[*OF* ‹$t \neq 0$›]]
unfolding *marked.sucs.h.pow-morph* **using** *suf-ext* **by** *metis*
qed

lemma *q-suf-conv*: $q \leq s$ $\mathfrak{f}$ $([a1] \cdot [1{-}a1]^{@}$*Suc* $k)$ $\longleftrightarrow$ $t \leq k$
proof
have *psuf'*: $p \leq s$ $p \cdot \mathfrak{e}$ $([a1] \cdot [1 - a1]$ $^{@}$ *Suc* $k)$ **for** k
unfolding *pow-Suc'* **using** *marked.sucs.g.morph ssufD1*[*OF p-suf*] *suffix-appendI* **by** *metis*
assume $q \leq s$ $\mathfrak{f}$ $([a1] \cdot [1 - a1]$ $^{@}$ *Suc* $k)$
hence $\neg$ *Suc* k $\leq t$
using *coin-exp-min*[*of Suc k*] *psuf'*[*of k*] *suf-ext*[*of q - q*] **unfolding** *coin-block-def* **by** *blast*
thus $t \leq k$
by *linarith*
next
assume $t \leq k$
have $q \leq s$ $q \cdot \mathfrak{f}[1{-}a1]$
using *coin-block-def swap-coin-block* **by** *blast*
have $q \leq s$ $\mathfrak{f}$ $[a1] \cdot \mathfrak{f}$ $([1{-}a1]^{@}$*Suc* $t)$
using *coin-exp rich-block-suf-snd*[*of* ε $1 - a1$ t, *unfolded emp-simps binA-simps*] **unfolding** *marked.sucs.h.morph* **by** *blast*
from *suf-prolong*[*OF per-exp-suf*[*OF* ‹$q \leq s$ $q \cdot \mathfrak{f}[1{-}a1]$›, *folded marked.sucs.h.pow-morph*] *this*, *of* $k{-}t$, *folded marked.sucs.h.morph lassoc*, *folded add-exps*[*of* $[1{-}a1]$ *Suc* t]]
show $q \leq s$ $\mathfrak{f}$ $([a1] \cdot [1{-}a1]^{@}$*Suc* $k)$
using ‹$t \leq k$› **by** *fastforce*
qed

lemma *coin-block-with-bad-letter*: **assumes** $a1 \in$ *set* w
shows *coin-block* $w \longleftrightarrow [1{-}a1]^{@}$*Suc* $t \leq s$ w
proof–
obtain i b **where** $[b] \cdot [1{-}a1]^{@}i \leq s$ w **and** $b \neq 1{-}a1$
using *distinct-letter-in-suf*[*of* w $1{-}a1$, *OF neq-set-not-root*[*OF bin-swap-neq*,

OF assms]].
have $b = a1$
using *bin-neq-swap'''*[*OF* ‹$b \neq 1{-}a1$›, *unfolded binA-simps*].
from ‹$[b] \cdot [1{-}a1]^{@} i \leq s\ w$›[*unfolded this*, *unfolded suffix-def*]
obtain w' **where** w: $w = w' \cdot [a1] \cdot [1{-}a1]^{@} i$ **by** *blast*
show *?thesis*
proof(*cases*)
assume $i = 0$
have $\neg\ [1 - a1]\ ^{@}\ Suc\ t \leq s\ w' \cdot [a1]$
unfolding *pow-Suc'* **using** *bin-swap-neq*[*of a1*]
by *simp*
then show *coin-block* $w \longleftrightarrow [1{-}a1]^{@} Suc\ t \leq s\ w$
unfolding w ‹$i = 0$› *cow-simps* **using** *last-letter-block non-block* **by** *meson*
next
assume $i \neq 0$
have *psuf*: $p \leq s\ p \cdot \mathfrak{e}\ (w' \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ k)$ **for** k
unfolding *pow-Suc'* **using** *marked.sucs.g.morph ssufD1*[*OF p-suf*] *suffix-appendI* **by** *metis*
have *psuf'*: $p \leq s\ p \cdot \mathfrak{e}\ ([a1] \cdot [1 - a1]\ ^{@}\ Suc\ k)$ **for** k
unfolding *pow-Suc'* **using** *marked.sucs.g.morph ssufD1*[*OF p-suf*] *suffix-appendI* **by** *metis*
have *equiv1*: *coin-block* $(w' \cdot [a1] \cdot [1{-}a1]^{@} Suc\ k) \longleftrightarrow q \leq s\ \mathfrak{f}\ ([a1] \cdot [1{-}a1]^{@} Suc\ k)$ **for** k
proof
show *coin-block* $(w' \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ k) \Longrightarrow q \leq s\ \mathfrak{f}\ ([a1] \cdot [1 - a1]\ ^{@}\ Suc\ k)$
using *rich-block-suf-snd*[*of* w' $1 - a1$ k] *suffix-append* **unfolding** *binA-simps marked.sucs.h.morph* **by** *blast*
show $q \leq s\ \mathfrak{f}\ ([a1] \cdot [1 - a1]\ ^{@}\ Suc\ k) \Longrightarrow$ *coin-block* $(w' \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ k)$
unfolding *coin-block-def marked.sucs.h.morph* **using** *psuf suffix-appendI* **by** *metis*
qed
have $t \leq k \longleftrightarrow [1{-}a1]^{@} Suc\ t \leq s\ w' \cdot [a1] \cdot [1{-}a1]^{@} Suc\ k$ **for** k
using *sing-exp-pref-iff*[*reversed*, *OF bin-swap-neq'*, *symmetric*, *of Suc t Suc k* $1{-}a1$, *unfolded Suc-le-mono binA-simps rassoc*].
from *equiv1*[*unfolded q-suf-conv this*, *of* $i{-}1$, *unfolded Suc-minus*[*OF* ‹$i \neq 0$›], *folded* w]
show *?thesis*.
qed
qed

1.5 Description of the basis

The infinite part of the basis

inductive-set $\mathcal{W}$ :: *binA list set* **where**
$[a1] \cdot [1{-}a1]^{@} Suc\ t\ \in \mathcal{W}$
$|\ \tau \in \mathcal{W} \Longrightarrow\ i \leq t \Longrightarrow [a1] \cdot [1{-}a1]^{@} i \cdot \tau \in \mathcal{W}$

lemma $\mathcal{W}$-*nemp*: $x \in \mathcal{W} \Longrightarrow x \neq \varepsilon$
  by (*rule* $\mathcal{W}$.*cases*[*of* x $x \neq \varepsilon$], *simp-all*)

lemma $\mathcal{W}$-*nemp'*: $x \in (\{[1 - a1]\} \cup \mathcal{W}) \Longrightarrow x \neq \varepsilon$
  using $\mathcal{W}$-*nemp* **by** *blast*

lemma $\mathcal{W}$-*hd*: $x \in \mathcal{W} \Longrightarrow$ *hd* $x = a1$
  by (*induction* x *rule*: $\mathcal{W}$.*induct*, *simp-all*)

lemma $\mathcal{W}$-*set*: $x \in \mathcal{W} \Longrightarrow a1 \in$ *set* x
  using $\mathcal{W}$-*hd* $\mathcal{W}$-*nemp* *hd-in-set* **by** *blast*

lemma $\mathcal{W}$-*butlast-hd-tl*: $x \in \mathcal{W} \Longrightarrow$ *butlast* $x = [a1] \cdot$ *butlast* (*tl* x)
  by (*induction* x *rule*: $\mathcal{W}$.*induct*, *auto*)

lemma $\mathcal{W}$-*suf*: $x \in \mathcal{W} \Longrightarrow [a1] \cdot [1{-}a1]^{@}$*Suc* $t \leq s$ x
  by (*induction* x *rule*: $\mathcal{W}$.*induct*, *simp-all add*: *suffix-Cons suffix-append*)

lemma $\mathcal{W}$-*fac*: $x \in \mathcal{W} \Longrightarrow \neg [1{-}a1]^{@}$*Suc* $t \leq f$ *butlast* x
proof (*induction* x *rule*: $\mathcal{W}$.*induct*)
  show $\neg [1 - a1]$ $^{@}$ *Suc* $t \leq f$ *butlast* ($[a1] \cdot [1 - a1]$ $^{@}$ *Suc* t)
    using *fac-len-eq*[*of* $[1 - a1]$ $^{@}$ *Suc* t *butlast* ($[a1] \cdot [1 - a1]$ $^{@}$ *Suc* t)]
    unfolding *pow-Suc'* *lassoc butlast-snoc sing-pow-len lenmorph sing-len*
    unfolding *pow-comm*[*of* $[1{-}a1]$] *add.commute*[*of* t] *cancel-right*
    using *bin-swap-neq* **by** *fast*
  fix x' i
  assume $x' \in \mathcal{W}$ **and** *notf*: $\neg [1 - a1]$ $^{@}$ *Suc* $t \leq f$ *butlast* x' **and** $i \leq t$
  show $\neg [1 - a1]$ $^{@}$ *Suc* $t \leq f$ *butlast* ($[a1] \cdot [1 - a1]$ $^{@}$ $i \cdot x'$)
  proof
    assume $[1 - a1]$ $^{@}$ *Suc* $t \leq f$ *butlast* ($[a1] \cdot [1 - a1]$ $^{@}$ $i \cdot x'$)
    hence $[1 - a1]$ $^{@}$ *Suc* $t \leq f$ $[a1] \cdot [1 - a1]$ $^{@}$ $i \cdot$ *butlast* x'
      unfolding *lassoc butlast-append* **using** $\mathcal{W}$-*nemp*[*OF* ‹$x' \in \mathcal{W}$›] **by** *force*
    then obtain *pp ss* **where** *fac*: $pp \cdot [1 - a1]$ $^{@}$ *Suc* $t \cdot ss = ([a1] \cdot [1 - a1]$ $^{@}$ $i) \cdot$ *butlast* x' **unfolding** *rassoc* **by** *fast*
    from *notf eqd*[*OF this*[*symmetric*]]
    have $\neg |[a1] \cdot [1 - a1]$ $^{@}$ $i| \leq |pp|$
      unfolding *fac-def* **by** *metis*
    hence $|pp| \leq i$
      unfolding *lenmorph* **by** *simp*
    have $pp \neq \varepsilon$
      using *fac emp-simps*(2) *bin-swap-neq*[*of* $a1$] **unfolding** *pow-Suc rassoc* **by** *force*
    have *Suc* $i < |pp| +$ *Suc* t **and** *Suc* $i - |pp| <$ *Suc* t
      using *nemp-len*[*OF* ‹$pp \neq \varepsilon$›] ‹$i \leq t$› **by** *linarith+*
    have ($pp \cdot [1 - a1]$ $^{@}$ *Suc* $t \cdot ss$)!(*Suc* i) $= a1$
      unfolding *fac* $\mathcal{W}$-*butlast-hd-tl*[*OF* ‹$x' \in \mathcal{W}$›]
      using *nth-append-length*[*of* $[a1] \cdot [1 - a1]$ $^{@}$ i $a1$] **unfolding** *lenmorph sing-pow-len sing-len swap-len* **by** *force*

from *this*[*unfolded lassoc nth-append*[*of - ss*]]
have $(pp \cdot [1 - a1]^{@} \ Suc\ t)!(Suc\ i) = a1$
unfolding *lenmorph sing-pow-len* **using** ‹$Suc\ i < |pp| + Suc\ t$› **by** *presburger*
from *this*[*unfolded nth-append*]
have $([1 - a1]^{@} \ Suc\ t)!(Suc\ i - |pp|) = a1$
using ‹$|pp| \le i$› **by** *force*
thus *False*
unfolding *sing-pow-nth*[*OF* ‹$Suc\ i - |pp| < Suc\ t$›]
using *bin-swap-neq* **by** *blast*
qed
qed

lemma *pref-code-$\mathcal{W}$*: *pref-code* $(\{[1-a1]\} \cup \mathcal{W})$
proof
show *nemp*: $\varepsilon \notin \{[1 - a1]\} \cup \mathcal{W}$
using *$\mathcal{W}$-nemp* **by** *auto*
show $u = v$ **if** *u-in*: $u \in \{[1 - a1]\} \cup \mathcal{W}$ **and** *v-in*: $v \in \{[1 - a1]\} \cup \mathcal{W}$ **and** $u \le p\ v$ **for** $u\ v$
proof (*rule bin-swap-exhaust*[*of hd u a1*])
assume $hd\ u = 1 - a1$
hence $u = [1-a1]$
using *u-in*[*unfolded Un-def mem-Collect-eq*]
$\mathcal{W}$-hd[*of u*] *bin-swap-neq′* **by** *blast*
from *sing-pref-hd*[*OF* ‹$u \le p\ v$›[*unfolded this*]]
have $hd\ v = 1 - a1$**.**
hence $v = [1-a1]$
using *v-in*[*unfolded Un-def mem-Collect-eq*]
$\mathcal{W}$-hd[*of v*] *bin-swap-neq′* **by** *blast*
with ‹$u = [1-a1]$›
show $u = v$
by *simp*
next
assume $hd\ u = a1$
have $u \neq \varepsilon\ v \neq \varepsilon$
using *nemp u-in v-in* **by** *blast+*
from *pref-hd-eq*[*OF* ‹$u \le p\ v$› ‹$u \neq \varepsilon$›]
have $hd\ v = a1$
using ‹$hd\ u = a1$› **by** *simp*
from *u-in* ‹$hd\ u = a1$› *bin-swap-neq*[*of a1*]
have $u \in \mathcal{W}$
unfolding *Un-def mem-Collect-eq* **using** *singletonD*[*of u* $[1-a1]$] *list.sel*(*1*)[*of* $1-a1\ \varepsilon$] **by** *metis*
from ‹$hd\ v = a1$› *v-in* ‹$hd\ u = a1$› *bin-swap-neq*[*of a1*]
have $v \in \mathcal{W}$
unfolding *Un-def mem-Collect-eq* **using** *singletonD*[*of v* $[1-a1]$] *list.sel*(*1*)[*of* $1-a1\ \varepsilon$] **by** *metis*
from *$\mathcal{W}$-suf*[*OF* ‹$u \in \mathcal{W}$›]
have $[1 - a1]^{@} \ Suc\ t \le s\ u$
using *suf-extD* **by** *blast*

hence $\neg\ u \leq p\ butlast\ v$
using $\mathcal{W}$-*fac*[*OF* ‹$v \in \mathcal{W}$›] **unfolding** *fac-def suffix-def prefix-def* **by** *fastforce*
with ‹$u \leq p\ v$›
show $u = v$
using *spref-butlast-pref* **by** *blast*
qed
qed

lemma $\mathcal{W}$-*coin-blocks*:
assumes $x \in \{[1 - a1]\} \cup \mathcal{W}$ **shows** $x \in \mathcal{T}$
proof–
consider $x = [1{-}a1] \mid x \in \mathcal{W}$
using ‹$x \in \{[1 - a1]\} \cup \mathcal{W}$› **by** *blast*
thus $x \in \mathcal{T}$
proof (*cases*)
assume $x = [1{-}a1]$
show $x \in \mathcal{T}$
unfolding $\mathcal{T}$-*def′* ‹$x = [1{-}a1]$› **using** *swap-coin-block*.
next
assume $x \in \mathcal{W}$
show $x \in \mathcal{T}$
unfolding $\mathcal{T}$-*def′* *coin-block-with-bad-letter*[*OF* $\mathcal{W}$-*set*[*OF* ‹$x \in \mathcal{W}$›]] **using** *suf-extD*[*OF* $\mathcal{W}$-*suf*[*OF* ‹$x \in \mathcal{W}$›]].
qed
qed

lemma $\mathcal{W}$-*gen-T*: $\langle\{[1{-}a1]\} \cup \mathcal{W}\rangle = \mathcal{T}$
proof
from *subsetI*[*OF* $\mathcal{W}$-*coin-blocks*, *THEN hull-mono*]
show $\langle\{[1 - a1]\} \cup \mathcal{W}\rangle \subseteq \mathcal{T}$
unfolding $\mathcal{T}$-*hull*.
next
show $\mathcal{T} \subseteq \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
proof
fix x **assume** $x \in \mathcal{T}$
from *this*[*unfolded* $\mathcal{T}$-*def′*] **have** *coin-block* x.
thus $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
proof (*induction* $|x|$ *arbitrary*: x *rule*: *less-induct*)
case *less*
show *?case*
proof (*cases* $\exists\ px.\ px \neq \varepsilon \wedge px <p\ x \wedge$ *coin-block* px)
assume $\exists\ px.\ px \neq \varepsilon \wedge px <p\ x \wedge$ *coin-block* px
from *exE*[*OF this*]
obtain px **where** $px \neq \varepsilon$ **and** $px <p\ x$ **and** *coin-block* px **by** *metis*
from *spref-exE*[*OF* ‹$px <p\ x$›]
obtain sx **where** $px{\cdot}sx = x$ **and** $sx \neq \varepsilon$.
from $\mathcal{T}$-*pref*[*OF* ‹*coin-block* px› ‹*coin-block* x›[*folded* ‹$px{\cdot}sx = x$›]]
have *coin-block* sx.
have $|px| < |x|$ **and** $|sx| < |x|$

using ‹$px \cdot sx = x$› ‹$px \neq \varepsilon$› ‹$sx \neq \varepsilon$› **by** *auto*
from *less.hyps*[*OF this*(*1*) ‹*coin-block px*›]
less.hyps[*OF this*(*2*) ‹*coin-block sx*›]
show $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
using ‹$px \cdot sx = x$› **by** *auto*
next
assume *non-ex*: $\nexists\ px.\ px \neq \varepsilon \wedge px <p\ x \wedge$ *coin-block px*
show $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
proof (*cases* $a1 \in$ *set x*)
assume $a1 \notin$ *set x*
then obtain k **where** $x = [1 - a1]^{@}k$
using *bin-without-letter* **by** *blast*
thus $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
using *gen-in*[*THEN power-in*] **by** *fast*
next
assume $a1 \in$ *set x*
hence $x \neq \varepsilon$ **by** *force*
have *hd* $x = a1$
proof (*rule ccontr*)
assume *hd* $x \neq a1$
hence *hd* $x = 1{-}a1$
using *bin-neq-iff* **by** *auto*
from *non-ex swap-coin-block hd-tl*[*OF* ‹$x \neq \varepsilon$›, *unfolded this*]
have *tl* $x = \varepsilon$ **by** *blast*
from ‹$[1 - a1] \cdot$ *tl* $x = x$›[*unfolded this emp-simps*]
show *False*
using *neq-in-set-not-pow*[*OF bin-swap-neq*[*of a1*] ‹$a1 \in$ *set x*›, *of 1*, *unfolded exp-simps*] **by** *simp*
qed
define j **where** $j = (LEAST\ k.\ \neg\ [a1]\cdot[1{-}a1]^{@}Suc\ k \leq p\ x)$
hence $\neg\ [a1]\cdot[1{-}a1]^{@}(Suc\ t) <p\ x$
using *coin-exp non-ex* **by** *blast*
hence $\neg\ [a1]\cdot[1{-}a1]^{@}Suc(Suc\ t) \leq p\ x$
unfolding *pow-Suc′*[*of - Suc t*] *lassoc*
using *prefix-snocD* **by** *metis*
from *Least-le*[*of* $\lambda\ i.\ \neg\ ([a1]\cdot[1{-}a1]^{@}Suc\ i) \leq p\ x$, *OF this*, *folded j-def*]
have $j \leq Suc\ t$**.**
have $\neg\ [a1]\cdot[1{-}a1]^{@}\ Suc\ j \leq p\ x$
using *LeastI*[*of* $\lambda\ i.\ \neg\ ([a1]\cdot[1{-}a1]^{@}Suc\ i) \leq p\ x$, *OF* ‹$\neg\ ([a1]\cdot[1{-}a1]^{@}Suc(Suc\ t)) \leq p\ x$›, *folded j-def*]**.**
have $[a1]\cdot[1{-}a1]^{@}j \leq p\ x$
using *not-less-Least*[*of* $j{-}1$ $\lambda\ i.\ \neg\ ([a1]\cdot[1{-}a1]^{@}Suc\ i) \leq p\ x$]
unfolding *j-def*[*symmetric*] *not-not*
by (*cases* $j = 0$, *simp-all add*: *hd-pref*[*OF* ‹$x \neq \varepsilon$›, *unfolded* ‹*hd* $x = a1$›])
show $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
proof(*cases* $j = Suc\ t$)
assume $j = Suc\ t$
have $x = [a1]\cdot[1{-}a1]^{@}j$
using ‹$[a1]\cdot[1{-}a1]^{@}j \leq p\ x$› ‹$\neg\ [a1] \cdot [1 - a1]\ ^{@}\ Suc\ t <p\ x$›

unfolding ‹$j = Suc\ t$› **by** *force*
from $\mathcal{W}$*.intros*(*1*)[*folded* ‹$j = Suc\ t$›, *folded this*]
show $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$ **by** *auto*
next
assume $j \neq Suc\ t$
hence $j \leq t$ **using** ‹$j \leq Suc\ t$› **by** *force*
from *prefE*[*OF* ‹$[a1]\cdot[1{-}a1]^{@}j \leq p\ x$›, *unfolded rassoc*]
obtain x' **where** $x = [a1] \cdot [1{-}a1]^{@}j \cdot x'$.
with *coin-exp-min*[*OF* ‹$j \leq t$›] ‹*coin-block x*›
have $x' \neq \varepsilon$
by *auto*
from ‹$\neg\ [a1] \cdot [1{-}a1]^{@}Suc\ j \leq p\ x$› *hd-tl*[*OF this*]
have $hd\ x' = a1$
unfolding ‹$x = [a1] \cdot [1{-}a1]^{@}j \cdot x'$› *pow-Suc′ pref-cancel-conv*
using *bin-neq-iff′*[*of hd* x' *1−a1*, *unfolded binA-simps*] **by** *fastforce*
from ‹$[hd\ x'] \cdot tl\ x' = x'$›[*unfolded this*]
‹*coin-block x*›[*unfolded coin-block-with-bad-letter*[*OF* ‹$a1 \in set\ x$›]]
have $[1{-}a1]^{@}Suc\ t \leq s\ [a1] \cdot [1 - a1]\ ^{@}\ j \cdot [a1] \cdot tl\ x'$
unfolding ‹$x = [a1] \cdot [1{-}a1]^{@}j \cdot x'$› **by** *presburger*
have $a1 \notin set\ ([1{-}a1]^{@}Suc\ t)$
using *neq-in-set-not-pow*[*OF bin-swap-neq*, *of a1*] **by** *blast*
hence $\neg\ [a1] \cdot tl\ x' \leq s\ [1{-}a1]^{@}Suc\ t$
unfolding *suffix-def* **using** *Cons-eq-appendI in-set-conv-decomp* **by** *metis*
with *ruler*[*reversed*, *of* x', *OF* - ‹$[1{-}a1]^{@}Suc\ t \leq s\ x$›]
have $[1{-}a1]^{@}Suc\ t \leq s\ x'$
unfolding ‹$x = [a1] \cdot [1{-}a1]^{@}j \cdot x'$› ‹$[a1] \cdot tl\ x' = x'$› *suffix-def* **by** *fastforce*
have $a1 \in set\ x'$
using ‹$hd\ x' = a1$› ‹$x' \neq \varepsilon$› *hd-in-set* **by** *blast*
from *coin-block-with-bad-letter*[*OF this*]
have *coin-block* x'
using ‹$[1{-}a1]^{@}Suc\ t \leq s\ x'$› **by** *blast*
have $|x'| < |x|$
using *lenarg*[*OF* ‹$x = [a1] \cdot [1{-}a1]^{@}j \cdot x'$›] **unfolding** *lenmorph* **by** *simp*
from *less.hyps*[*OF this* ‹*coin-block* x'›]
obtain xs' **where** $xs' \in lists\ (\{[1 - a1]\} \cup \mathcal{W})$ **and** $concat\ xs' = x'$ **using** *hull-concat-listsE* **by** *blast*
have $xs' \neq \varepsilon$
using ‹$concat\ xs' = x'$› ‹$x' \neq \varepsilon$› *concat.simps*(*1*) **by** *blast*
from *lists-hd-in-set*[*OF this* ‹$xs' \in lists\ (\{[1 - a1]\} \cup \mathcal{W})$›]
have $hd\ xs' \in (\{[1 - a1]\} \cup \mathcal{W})$.
from $\mathcal{W}$*-coin-blocks*[*OF this*] $\mathcal{W}$*-nemp′*[*OF this*]
have *coin-block* $(hd\ xs')$ **and** $hd\ xs' \neq \varepsilon$
unfolding $\mathcal{T}$*-def′*.
have $hd\ (hd\ xs') = a1$
using *hd-concat*[*OF* ‹$xs' \neq \varepsilon$› ‹$hd\ xs' \neq \varepsilon$›, *symmetric*]
unfolding ‹$concat\ xs' = x'$› ‹$hd\ x' = a1$›.

hence $hd\ xs' \in \mathcal{W}$
using ‹$hd\ xs' \in (\{[1 - a1]\} \cup \mathcal{W})$› *bin-swap-neq*[*of a1*] *list.sel*(*1*)[*of* $1{-}a1\ \varepsilon$]
unfolding *Un-def mem-Collect-eq singleton-iff* **by** *metis*
hence $[a1] \cdot [1{-}a1]^{@}j \cdot hd\ xs' \in \mathcal{W}$
using ‹$j \le t$› $\mathcal{W}$.*intros*(*2*) **by** *blast*
with $\mathcal{W}$-*coin-blocks*
have *coin-block* $([a1] \cdot [1{-}a1]^{@}j \cdot hd\ xs')$
unfolding $\mathcal{T}$-*def*$'$ *Un-def* **by** *blast*
have $[a1] \cdot [1{-}a1]^{@}j \cdot hd\ xs' \le p\ x$
using *hd-concat-tl*[*OF* ‹$xs' \neq \varepsilon$›]
unfolding ‹$concat\ xs' = x'$› ‹$x = [a1] \cdot [1{-}a1]^{@}j \cdot x'$›
by *fastforce*
with *non-ex* ‹*coin-block* $(hd\ xs')$› ‹$hd\ xs' \neq \varepsilon$›
have $x = [a1] \cdot [1{-}a1]^{@}j \cdot hd\ xs'$
using ‹*coin-block* $([a1] \cdot [1 - a1]^{@}\ j \cdot hd\ xs')$› *strict-prefixI suf-nemp* **by** *metis*
from ‹$[a1] \cdot [1{-}a1]^{@}j \cdot hd\ xs' \in \mathcal{W}$›[*folded this*]
show $x \in \langle\{[1 - a1]\} \cup \mathcal{W}\rangle$
by *auto*
qed
qed
qed
qed
qed
qed

lemma $\mathcal{W}$-*explicit*: $\mathcal{W} = \{w \cdot [a1] \cdot [1{-}a1]^{@}Suc\ t \mid w.\ w \in \langle\{[a1] \cdot [1{-}a1]^{@}i \mid i.\ i \le t\}\rangle\}$
proof
show $\mathcal{W} \subseteq \{w \cdot [a1] \cdot [1 - a1]^{@}\ Suc\ t \mid w.\ w \in \langle\{[a1] \cdot [1 - a1]^{@}\ i \mid i.\ i \le t\}\rangle\}$
proof
fix x **assume** $x \in \mathcal{W}$
thus $x \in \{w \cdot [a1] \cdot [1 - a1]^{@}\ Suc\ t \mid w.\ w \in \langle\{[a1] \cdot [1 - a1]^{@}\ i \mid i.\ i \le t\}\rangle\}$
unfolding *mem-Collect-eq*
proof (*induction x rule*: $\mathcal{W}$.*induct*, *simp*)
case $(2\ \tau\ i)$
then obtain w **where** $\tau = w \cdot [a1] \cdot [1 - a1]^{@}\ Suc\ t$ **and** $w \in \langle\{[a1] \cdot [1 - a1]^{@}\ i \mid i.\ i \le t\}\rangle$
by *blast*
from *hull.prod-cl*[*OF - this*(*2*), *of* $[a1] \cdot [1 - a1]^{@}\ i$] ‹$i \le t$›
have $[a1] \cdot [1 - a1]^{@}\ i \cdot w \in \langle\{[a1] \cdot [1 - a1]^{@}\ i \mid i.\ i \le t\}\rangle$
unfolding *mem-Collect-eq* **by** *simp*
thus *?case*
using ‹$\tau = w \cdot [a1] \cdot [1 - a1]^{@}\ Suc\ t$› **by** *auto*
qed
qed
next
show $\{w \cdot [a1] \cdot [1 - a1]^{@}\ Suc\ t \mid w.\ w \in \langle\{[a1] \cdot [1 - a1]^{@}\ i \mid i.\ i \le t\}\rangle\} \subseteq \mathcal{W}$

proof
fix x **assume** $x \in \{w \cdot [a1] \cdot [1 - a1]^{@} \textit{Suc } t \mid w.\ w \in \langle\{[a1] \cdot [1 - a1]^{@} i \mid i.\ i \leq t\}\rangle\}$
then obtain w **where** $x = w \cdot [a1] \cdot [1 - a1]^{@} \textit{Suc } t$ **and** $w \in \langle\{[a1] \cdot [1 - a1]^{@} i \mid i.\ i \leq t\}\rangle$
unfolding *mem-Collect-eq* **by** *blast*
show $x \in \mathcal{W}$
unfolding ‹$x = w \cdot [a1] \cdot [1 - a1]^{@} \textit{Suc } t$›
by (*rule hull.induct*[*OF* ‹$w \in \langle\{[a1] \cdot [1 - a1]^{@} i \mid i.\ i \leq t\}\rangle$›], *use* $\mathcal{W}$*.intros*(*1*) **in** *force*)
(*use* $\mathcal{W}$*.intros*(*2*) **in** *force*)
qed
qed

theorem *infinite-basis*: $\mathfrak{B}\ \mathcal{T} = (\{[1{-}a1]\} \cup \mathcal{W})$
using *basis-of-hull*[*of* $\{[1{-}a1]\} \cup \mathcal{W}$]
unfolding $\mathcal{W}$*-gen-T* *code.code-is-basis*[*OF pref-code.code*, *OF pref-code-*$\mathcal{W}$]**.**

end

1.6 Intersection

lemma *bin-inter-coin-set-fst*: $\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle = ((\textit{bin-morph-of } x\ y) \circ \textit{fst})$ ‘ $\mathfrak{C}$ (*bin-morph-of* x y) (*bin-morph-of* u v)
using *bin-morph-of-range coin-set-inter-fst* **by** *metis*

lemma *bin-inter-coin-set-snd*: $\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle = ((\textit{bin-morph-of } u\ v) \circ \textit{snd})$ ‘ $\mathfrak{C}$ (*bin-morph-of* x y) (*bin-morph-of* u v)
using *bin-inter-coin-set-fst* **unfolding** *coin-set-eq***.**

theorem *bin-inter-basis*: **assumes** *binary-code* x y **and** *binary-code* u v
shows $\mathfrak{B}\ (\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle) = ((\textit{bin-morph-of } u\ v) \circ \textit{snd})$ ‘ $\mathfrak{C}_m$ (*bin-morph-of* x y) (*bin-morph-of* u v)
unfolding *bin-inter-coin-set-snd*
using *two-code-morphisms.range-inter-basis-snd*(*1*)[*OF two-code-morphisms.intro*, *OF binary-code.code-morph-of binary-code.code-morph-of*, *OF assms*, *folded coin-set-inter-snd*]
unfolding *image-comp***.**

theorem *binary-intersection-code*:
assumes *binary-code* x y **and** *binary-code* u v
shows *code* $\mathfrak{B}\ (\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle)$
using *two-code-morphisms.range-inter-code*[*OF two-code-morphisms.intro*[*OF binary-code.code-morph-of*[*OF assms*(*1*)] *binary-code.code-morph-of*[*OF assms*(*2*)]]]
unfolding *bin-morph-of-range***.**

theorem *binary-intersection*:
assumes *binary-code* x y **and** *binary-code* u v
obtains
$\mathfrak{B}\ (\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle) = \{\}$

|

β **where** $\mathfrak{B}$ $(\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle) = \{\beta\}$

|

β γ **where** $\mathfrak{B}$ $(\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle) = \{\beta,\gamma\}$

|

β γ δ t **where** $\delta \neq \varepsilon$ **and** $\gamma \cdot \beta \neq \varepsilon$ **and** $hd\ \delta \neq hd\ (\gamma \cdot \beta)$
$\mathfrak{B}$ $(\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle) = \{\beta \cdot \gamma\} \cup \{\beta \cdot (\gamma \cdot \beta)^{@}\mathsf{t} \cdot w \cdot \delta \cdot \gamma \mid w.\ w \in \langle\{\delta \cdot (\gamma \cdot \beta)^{@}i \mid i.\ i \leq \mathsf{t}\}\rangle\}$

|

β γ δ t q **where** $\delta \neq \varepsilon$**and** $\gamma \cdot \beta \neq \varepsilon$ **and** $hd\ \delta \neq hd\ (\gamma \cdot \beta)$ **and**
$1 \leq \mathsf{q} \wedge \mathsf{q} \leq \mathsf{t}$ **and**
$\mathfrak{B}$ $(\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle) = \{\beta \cdot \gamma\} \cup \{\beta \cdot (\gamma \cdot \beta)^{@}\mathsf{t} \cdot w \cdot \delta^{<-1}(\beta \cdot (\gamma \cdot \beta)^{@}(\mathsf{t}-\mathsf{q}))$
|
$w.\ w \in \langle\{\delta \cdot (\gamma \cdot \beta)^{@}i \mid i.\ i \leq \mathsf{q} - 1\}\rangle\}$

proof–
define x' **where** $x' = (if\ |bin\text{-}lcp\ u\ v| \leq |bin\text{-}lcp\ x\ y|\ then\ x\ else\ u)$
define y' **where** $y' = (if\ |bin\text{-}lcp\ u\ v| \leq |bin\text{-}lcp\ x\ y|\ then\ y\ else\ v)$
define u' **where** $u' = (if\ |bin\text{-}lcp\ u\ v| \leq |bin\text{-}lcp\ x\ y|\ then\ u\ else\ x)$
define v' **where** $v' = (if\ |bin\text{-}lcp\ u\ v| \leq |bin\text{-}lcp\ x\ y|\ then\ v\ else\ y)$
have *lcp-le*: $|bin\text{-}lcp\ u'\ v'| \leq |bin\text{-}lcp\ x'\ y'|$
unfolding *x′-def y′-def u′-def v′-def* **by** *simp*
have *int′*: $\langle\{x,y\}\rangle \cap \langle\{u,v\}\rangle = \langle\{x',y'\}\rangle \cap \langle\{u',v'\}\rangle$
unfolding *x′-def y′-def u′-def v′-def* **using** *Int-commute* **by** *force*
have *assms′*: *binary-code x′ y′ binary-code u′ v′*
using *assms* **unfolding** *x′-def y′-def u′-def v′-def* **by** *simp-all*

define *first-morphism* (*g*)
where *first-morphism* ≡ *bin-morph-of x′ y′*
define *second-morphism* (*h*)
where *second-morphism* ≡ *bin-morph-of u′ v′*
note *mdefs* = *first-morphism-def second-morphism-def*
have *ranges*: *range g* = $\langle\{x',y'\}\rangle$ *range h* = $\langle\{u',v'\}\rangle$
unfolding *mdefs bin-morph-of-range* **by** *blast+*

have *nemp*: $x' \neq \varepsilon$ $y' \neq \varepsilon$ $u' \neq \varepsilon$ $v' \neq \varepsilon$
using *assms′ binary-code.non-comm* **by** *blast+*

interpret *two-binary-code-morphisms g h*
using *two-binary-code-morphisms.intro*
unfolding *binary-code-morphism-def first-morphism-def second-morphism-def*
using *binary-code.code-morph-of assms′* **by** *blast*

interpret *two-nonerasing-morphisms g h*
using *code.two-nonerasing-morphisms-axioms*.

show *thesis*
proof (*cases*)
assume $\mathfrak{C}_m$ $g\ h = \{\}$ — simple case: coincidence set is empty
have $\langle\{x',y'\}\rangle \cap \langle\{u',v'\}\rangle = \{\varepsilon\}$

unfolding *bin-inter-coin-set-snd image-comp*[*symmetric*] *mdefs*[*symmetric*]
code.min-coin-gen-snd[*symmetric*, *unfolded* ‹$\mathfrak{C}_m$ *g h* = {}›]
by (*simp add*: *emp-gen-set*)
from *that*(*1*)[*unfolded int′ this*]
show *?thesis*
unfolding *emp-basis-iff* **by** *simp*
next
assume $\mathfrak{C}_m$ *g h* ≠ {}
then obtain *r1 s1* **where** *g r1* $=_m$ *h s1*
unfolding *min-coincidence-set-def* **by** *blast*
interpret *binary-codes-coincidence g h*
proof
show ∃ *r s*. *g r* $=_m$ *h s*
using ‹*g r1* $=_m$ *h s1*› **by** *blast*
show |*h.bin-code-lcp*| ≤ |*g.bin-code-lcp*|
unfolding *bin-morph-ofD mdefs* **using** *lcp-le*.
qed
show *thesis*
proof (*cases*)
assume $\mathfrak{C}_m$ *g h* = {(*r1*,*s1*)} — min. coincidence set contains 1 element
from *that*(*2*)[*unfolded int′*]
show *thesis*
unfolding *bin-inter-basis* [*OF assms′*, *unfolded* ‹$\mathfrak{C}_m$ *g h* = {(*r1*,*s1*)}›[*unfolded mdefs*] *image-comp*[*symmetric*]] **by** *simp*
next
assume $\mathfrak{C}_m$ *g h* ≠ {(*r1*,*s1*)} — min. coincidence set contains more than 1 element
then obtain *r2 s2* **where** (*r2*,*s2*) ∈ $\mathfrak{C}_m$ *g h* **and** (*r2*,*s2*) ≠ (*r1*,*s1*)
using ‹$\mathfrak{C}_m$ *g h* ≠ {}› **by** *auto*
from *min-coin-setD*[*OF this*(*1*)] ‹*g r1* $=_m$ *h s1*› *this*(*2*)
interpret *binary-codes-coincidence-two-generators g h*
by *unfold-locales auto*
write *g.marked-version* (g_m) **and**
h.marked-version (h_m) **and**
fst-beginning-block (*p*) **and**
snd-beginning-block (*q*) **and**
h.bin-code-lcp (α_h) **and**
marked.suc-snd ($\mathfrak{f}$)
show *thesis*
proof(*cases*)
assume ∀ *a*. *coin-block* [*a*]
hence ⋀ *a*. *coin-block* [*a*] **by** *force*
define *β* **where** $\beta = (h \circ (\lambda x.\ (q \cdot x)^{<-1} q\ ) \circ \mathfrak{f})\ \mathfrak{a}$
define *γ* **where** $\gamma = (h \circ (\lambda x.\ (q \cdot x)^{<-1} q\ ) \circ \mathfrak{f})\ \mathfrak{b}$
have *range* (*bin-morph-of x′ y′*) = ⟨{*x′*,*y′*}⟩
using *bin-morph-of-range* **by** *auto*
from *that*(*3*)[*of β γ*, *unfolded int′ β-def γ-def mdefs*[*symmetric*]]
show *thesis*
using *inter-basis*[*unfolded simple-blocks-basis*[*OF* ‹⋀ *a*. *coin-block* [*a*]›]

bin-morph-of-range]
unfolding *ranges* **by** *blast*
next
assume $\neg$ ($\forall$ *a*. *coin-block* [*a*])
then obtain *a1* **where** $\neg$ *coin-block* [*a1*] **by** *blast*
then interpret *binary-codes-coincidence-infinite g h a1*
by *unfold-locales*
write *coincidence-exponent* (t)

from *inter-basis*[*unfolded ranges infinite-basis bin-morph-of-range*, *folded Setcompr-eq-image*, *unfolded mem-Collect-eq*]
have *inter*:$\mathfrak{B}$ $(\langle\{x', y'\}\rangle \cap \langle\{u', v'\}\rangle) = \{(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ x \mid x.\ x \in \{[1 - a1]\} \cup \mathcal{W}\}$**.**

have $q \leq s\ q \cdot \mathfrak{f}\ [1 - a1]$
using *swap-coin-block*[*unfolded coin-block-def*] **by** *blast*
from *conjug-eqE*[*OF rq-suf*[*OF this*]] *conjug-emp-emp'*[*OF this*] *marked.sucs.h.sing-to-nemp*
obtain *q1 q2 k* **where** *q21*: $\mathfrak{f}\ [1 - a1] = q2 \cdot q1$ **and**
q-def: $q = (q1 \cdot q2)^{@}k \cdot q1$ **and** $q2 \neq \varepsilon$
by *metis*
have $(h\ q1 \cdot h\ q2) \cdot (h\ q1 \cdot \alpha_h) = (h\ q1 \cdot \alpha_h) \cdot (h_m\ q2 \cdot h_m\ q1)$
unfolding *rassoc h.marked-version-conjugates*[*of* $q2 \cdot q1$, *unfolded hm.morph h.morph*]**..**
from *conjug-eqE*[*OF this*] *h.nemp-to-nemp*[*OF* ‹$q2 \neq \varepsilon$›]
obtain $\beta\ \gamma\ k'$ **where** *bg*: $\beta \cdot \gamma = h\ (q1 \cdot q2)$ **and** $\gamma \cdot \beta = h_m\ (q2 \cdot q1)$ **and**
k': $\beta \cdot (\gamma \cdot \beta)^{@}k' = h\ q1 \cdot \alpha_h$ **and** $\gamma \neq \varepsilon$
unfolding *hm.morph h.morph shift-pow* **by** *blast*
have *bgb-q*: $\beta \cdot (\gamma \cdot \beta)^{@}(k' + k) = \alpha_h \cdot h_m\ q$
unfolding *add-exps lassoc* ‹$\beta \cdot (\gamma \cdot \beta)^{@}k' = h\ q1 \cdot \alpha_h$›
unfolding ‹$\gamma \cdot \beta = h_m\ (q2 \cdot q1)$› *h.marked-version-conjugates*[*symmetric*]
rassoc cancel q-def shift-pow **unfolding** *hm.morph hm.pow-morph***..**
define δ **where** $\delta = h_m\ (\mathfrak{f}\ [a1])$
have *bg-def*: $\beta \cdot \gamma = h\ ((q \cdot \mathfrak{f}\ [1 - a1])^{<-1}q\ )$
unfolding *bg q-def q21 rassoc shift-pow pow-comm*
unfolding *lassoc*[*of q1*]
unfolding *rq-triv***..**
have *bg-def'*: $(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ [1{-}a1] = \beta \cdot \gamma$
using *bg-def* **by** *simp*
have *gb-def*: $\gamma \cdot \beta = h_m\ (\mathfrak{f}\ [1 - a1])$
unfolding ‹$\gamma \cdot \beta = h_m\ (q2 \cdot q1)$› *q21***..**

have $\gamma \cdot \beta \neq \varepsilon$
using ‹$\gamma \neq \varepsilon$› **by** *blast*
have $\delta \neq \varepsilon$
unfolding *δ-def* **using** *hm.nonerasing marked.sucs.h.sing-to-nemp* **by** *blast*
have *hd* $\delta \neq$ *hd* $(\gamma \cdot \beta)$
unfolding *δ-def gb-def*
using *hm.hd-im-eq-hd-eq marked.sucs.h.bin-marked-sing marked.sucs.h.sing-to-nemp*

by *blast*

have *w-decode*: $w \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle \Longrightarrow\ h_m\ (\mathfrak{f}\ w) \in \langle\{\delta \cdot (\gamma \cdot \beta)\ ^{@}\ i\ |i.\ i \leq t\}\rangle$
for w
proof (*induct w rule*: *hull.induct*, *unfold marked.sucs.h.emp-to-emp hm.emp-to-emp*, *fast*)
case (*prod-cl w1 w2*)
then obtain i **where** $w1 = [a1] \cdot [1{-}a1]^{@} i$ **and** $i \leq t$ **by** *blast*
with *prod-cl.hyps*
show *?case*
unfolding *marked.sucs.h.morph marked.sucs.h.pow-morph hm.morph*
hm.pow-morph ‹$w1 = [a1] \cdot [1{-}a1]^{@} i$› δ*-def*[*symmetric*] *gb-def*[*symmetric*]
by *blast*
qed

have *w-decode'*: $w \in \langle\{\delta \cdot (\gamma \cdot \beta)\ ^{@}\ i\ |i.\ i \leq t\}\rangle \Longrightarrow \exists\ w'.\ w' \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle \wedge h_m\ (\mathfrak{f}\ w') = w$ **for** w
proof (*induct w rule*: *hull.induct*, *use marked.sucs.h.emp-to-emp hm.emp-to-emp* **in** *force*)
case (*prod-cl w1 w2*)
then obtain w' j **where** $w' \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle$ **and** $h_m\ (\mathfrak{f}\ w') = w2$ **and** $w1 = \delta \cdot (\gamma \cdot \beta)\ ^{@}\ j$ **and** $j \leq t$ **by** *blast*
have $([a1] \cdot [1 - a1]\ ^{@}\ j) \cdot w' \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle$
using ‹$w' \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle$› ‹$j \leq t$› **by** *blast*
moreover have $h_m\ (\mathfrak{f}\ (([a1] \cdot [1 - a1]\ ^{@}\ j) \cdot w')) = w1 \cdot w2$
unfolding *marked.sucs.h.morph marked.sucs.h.pow-morph hm.morph hm.pow-morph* ‹$h_m\ (\mathfrak{f}\ w') = w2$›
δ*-def*[*symmetric*] *gb-def*[*symmetric*] ‹$w1 = \delta \cdot (\gamma \cdot \beta)\ ^{@}\ j$›**..**
ultimately show $\exists\, w'.\ w' \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle \wedge h_m\ (\mathfrak{f}\ w') = w1 \cdot w2$ **by** *blast*
qed

show *thesis*
proof (*cases*)
assume $\alpha_h \cdot h_m\ q <s\ h_m\ (\mathfrak{f}\ ([1{-}a1]^{@} Suc\ t))$
from *ssuf-extD*[*OF this*[*unfolded bgb-q*[*symmetric*] *marked.sucs.h.pow-morph q21 gb-def hm.pow-morph*]]
have $k' + k < Suc\ t$
unfolding *marked.sucs.h.pow-morph*[*symmetric*] **using** *comp-pows-ssuf* **by** *blast*
have $\neg\ q \leq s\ \mathfrak{f}\ ([1{-}a1]^{@} t)$
unfolding *marked.sucs.h.pow-morph* **using** *exp-min*.
hence $t \leq k$
unfolding *marked.sucs.h.pow-morph q21 q-def shift-pow*
using *comp-pows-not-suf* **by** *blast*
hence $t = k$ **and** $k' = 0$
using ‹$k' + k < Suc\ t$› **by** *force+*

from *bgb-q*[*folded* ‹$t = k$›, *unfolded* ‹$k' = 0$›]
have $\beta \cdot (\gamma \cdot \beta)$ $^@$ $t = \alpha_h \cdot h_m\ q$ **by** *simp*
have $q \leq s\ \mathfrak{f}\ [1 - a1]$ $^@$ *Suc t*
unfolding *q-def* ‹$t = k$› *q21 pow-Suc shift-pow* **by** *force*

have $\beta = h\ q1 \cdot \alpha_h$
using ‹$\beta \cdot (\gamma \cdot \beta)^@ k' = h\ q1 \cdot \alpha_h$›[*unfolded* ‹$k' = 0$› *cow-simps*].
from *gb-def*[*unfolded q21 hm.morph this h.marked-version-conjugates*[*symmetric*]]
have $\gamma \cdot \alpha_h = h_m\ q2$
by *force*
from *h.marked-version-conjugates*[*of q2*, *folded this*]
have $h\ ((\mathfrak{f}\ [1 - a1]$ $^@$ *Suc* $t)^{<-1}q) = \alpha_h \cdot \gamma$
unfolding *q-def* ‹$t = k$› *q21 pow-Suc shift-pow rassoc rq-triv* **by** *force*

have *apply-h0*: $h\ ((q \cdot \mathfrak{f}\ (w \cdot [a1] \cdot [1 - a1]$ $^@$ *Suc* $t))^{<-1}q) = \beta \cdot (\gamma \cdot \beta)^@ t \cdot h_m\ (\mathfrak{f}\ w) \cdot \delta \cdot \gamma$ **for** w
proof−
have $\beta \cdot (\gamma \cdot \beta)^@ t \cdot h_m\ (\mathfrak{f}\ w) \cdot \delta \cdot \gamma = \alpha_h \cdot h_m\ (q \cdot \mathfrak{f}\ (w \cdot [a1])) \cdot \gamma$
unfolding *hm.morph marked.sucs.h.morph δ-def lassoc cancel-right* ‹$\beta \cdot (\gamma \cdot \beta)$ $^@$ $t = \alpha_h \cdot h_m\ q$›..
also have ... $= h\ (q \cdot \mathfrak{f}\ (w \cdot [a1])) \cdot h\ ((\mathfrak{f}\ [1 - a1]$ $^@$ *Suc* $t)^{<-1}q)$
unfolding *lassoc h.marked-version-conjugates*
unfolding ‹$h\ ((\mathfrak{f}\ [1 - a1]$ $^@$ *Suc* $t)^{<-1}q) = \alpha_h \cdot \gamma$› *rassoc*..
finally show *?thesis*
unfolding *h.morph*[*symmetric*] *marked.sucs.h.morph marked.sucs.h.pow-morph rq-reassoc*[*OF* ‹$q \leq s\ \mathfrak{f}\ [1 - a1]$ $^@$ *Suc t*›, *of* $q \cdot \mathfrak{f}\ w \cdot \mathfrak{f}\ [a1]$]
unfolding *rassoc* **by** *argo*
qed

have *inf-part-equal*:
$\{(h \circ (\lambda x.\ (q \cdot x)^{<-1}q\ ) \circ \mathfrak{f})\ (w \cdot [a1] \cdot [1 - a1]$ $^@$ *Suc* $t)$
$|w.\ w \in \langle\{[a1] \cdot [1 - a1]$ $^@$ $i\ |i.\ i \leq t\}\rangle\}$
$= \{\beta \cdot (\gamma \cdot \beta)$ $^@$ $t \cdot w \cdot \delta \cdot \gamma\ |w.\ w \in \langle\{\delta \cdot (\gamma \cdot \beta)$ $^@$ $i\ |i.\ i \leq t\}\rangle\}$ (**is** *?I* = *?E*)
proof
show *?I* ⊆ *?E*
proof
fix x **assume** $x \in$ *?I*
then obtain w **where** $x = h\ ((q \cdot \mathfrak{f}\ (w \cdot [a1] \cdot [1 - a1]$ $^@$ *Suc* $t))^{<-1}q$) **and**
$w \in \langle\{[a1] \cdot [1 - a1]$ $^@$ $i\ |i.\ i \leq t\}\rangle$
unfolding *mem-Collect-eq o-apply* **by** *blast*
from *this*(*1*)[*unfolded apply-h0*] *w-decode*[*OF this*(*2*)]
show $x \in$ *?E* **by** *blast*
qed
next
show *?E* ⊆ *?I*
proof

fix x **assume** $x \in$ *?E*
then obtain w **where** x: $x = \beta \cdot (\gamma \cdot \beta)$ $^@$ $t \cdot w \cdot \delta \cdot \gamma$ **and**
$w \in \langle\{\delta \cdot (\gamma \cdot \beta)$ $^@$ i $|i.$ $i \leq t\}\rangle$ **by** *blast*
from *w-decode'*[*OF this*(*2*)]
obtain w' **where** $w' \in \langle\{[a1] \cdot [1 - a1]$ $^@$ i $|i.$ $i \leq t\}\rangle$ **and** h_m ($\mathfrak{f}$ w') $= w$ **by** *blast*
from *x*[*folded this*(*2*), *folded apply-h0*[*of w'*]] *this*(*1*)
show $x \in$ *?I*
unfolding *o-apply* **by** *blast*
qed
qed
from *that*(*4*)[*OF* ‹$\delta \neq \varepsilon$› ‹$\gamma \cdot \beta \neq \varepsilon$› ‹*hd* $\delta \neq$ *hd* $(\gamma \cdot \beta)$›, *unfolded int'*, *of t*,
unfolded inter, *folded inf-part-equal bg-def'*, *unfolded* $\mathcal{W}$*-explicit*]
show *thesis*
by *blast*
next
assume $\neg$ $\alpha_h \cdot h_m$ q <*s* h_m ($\mathfrak{f}$ ([*1*−*a1*]$^@$*Suc t*))
note *not-suf* = *this*[*unfolded marked.sucs.h.pow-morph q21*]
have $\alpha_h \cdot h_m$ q ≤*s* $(\alpha_h \cdot h_m$ $q) \cdot h_m$ ((*q2* · *q1*) $^@$ *Suc t*)
unfolding *q-def shift-pow rassoc hm.morph*[*symmetric*] *pows-comm*[*of* - *k*]
unfolding *hm.morph lassoc suf-cancel-conv*
unfolding *rassoc hm.morph*[*symmetric*] *shift-pow*[*symmetric*]
unfolding *hm.morph lassoc suf-cancel-conv*
unfolding *h.marked-version-conjugates* **by** *blast*
from *ruler*[*reversed*, *of* $\alpha_h \cdot h_m$ q - h_m ((*q2* · *q1*) $^@$ *Suc t*), *OF* - *triv-suf*,
of $\alpha_h \cdot h_m$ q, *OF this*]
have h_m ((*q2*·*q1*)$^@$*Suc t*) ≤*s* $\alpha_h \cdot h_m$ q
unfolding *marked.sucs.h.pow-morph q21* **using** *not-suf* **by** *force*
from *this*[*unfolded q-def shift-pow hm.morph*, *unfolded hm.pow-morph*, *folded gb-def*[*unfolded q21*], *unfolded lassoc*,
folded k'[*folded h.marked-version-conjugates*], *unfolded rassoc add-exps*[*symmetric*]]
have *Suc t* $\leq k' + k$
using *comp-pows-suf'*[*OF* ‹$\gamma \neq \varepsilon$›] **by** *blast*
from *le-add-diff-inverse2*[*OF this*]
have *split*: $\beta \cdot (\gamma \cdot \beta)$ $^@$ $(k' + k) = \beta \cdot (\gamma \cdot \beta)$ $^@$ $(k' + k -$ *Suc t*$) \cdot (\gamma \cdot \beta)$ $^@$ *Suc t*
unfolding *add-exps*[*symmetric*] **by** *argo*

have $\alpha_h \cdot h_m$ $q = \beta \cdot (\gamma \cdot \beta)$ $^@$ $(k' + k)$
unfolding *q-def shift-pow hm.morph*
unfolding *hm.pow-morph gb-def*[*symmetric*, *unfolded q21*] *lassoc k'*[*folded h.marked-version-conjugates*, *symmetric*]
unfolding *rassoc add-exps*..

have *q-suf*: q ≤*s* $\mathfrak{f}$ ([*a1*] · [*1* − *a1*] $^@$ *Suc t*)
unfolding *q-suf-conv* **by** *blast*
have *q-suf'*: q ≤*s* $q \cdot \mathfrak{f}$ ((w · [*a1*]) · [*1* − *a1*] $^@$ *Suc t*) **for** w

using *suf-ext*[*OF q-suf*, *of* $q \cdot \mathfrak{f}\ w$] **unfolding** *marked.sucs.h.morph rassoc*.

note *long* = *rich-block-suf-snd′*[*of* ε *1−a1*, *unfolded emp-simps binA-simps*, *OF coin-exp*]

have *delta-suf*: $(\beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k - Suc\ t)) \leq s\ \delta$
using *long* **unfolding** *bgb-q*[*symmetric*] *δ-def marked.sucs.h.morph marked.sucs.h.pow-morph q21 hm.morph*
unfolding *hm.pow-morph gb-def*[*unfolded q21*,*symmetric*] *split*
unfolding *lassoc suf-cancel-conv*.

have *apply-h0*: $h\ ((q \cdot \mathfrak{f}\ (w \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ t))^{<-1} q)$
$= \beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k) \cdot\ h_m\ (\mathfrak{f}\ w) \cdot \delta^{<-1}(\beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k - Suc\ t))$ **for** w
unfolding *cancel-right*[*symmetric*, *of* $h\ ((q \cdot \mathfrak{f}\ (w \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ t))^{<-1}q)$ - $(\beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k - Suc\ t))$]
unfolding *rassoc rq-suf*[*OF delta-suf*]
unfolding *cancel-right*[*symmetric*, *of* - $\beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k) \cdot h_m\ (\mathfrak{f}\ w) \cdot \delta\ (\gamma \cdot \beta)\ ^{@}\ Suc\ t$]
unfolding *rassoc add-exps*[*symmetric*] ‹$k' + k - Suc\ t + Suc\ t = k' + k$› *bgb-q*[*unfolded h.marked-version-conjugates*]
unfolding *lassoc h.morph*[*symmetric*]
unfolding *rassoc rq-suf*[*OF q-suf′*, *unfolded rassoc*, *of w*]
unfolding *h.marked-version-conjugates*[*symmetric*] *hm.morph marked.sucs.h.morph*
unfolding *lassoc bgb-q* **unfolding** *rassoc δ-def gb-def hm.pow-morph marked.sucs.h.pow-morph*..

have *inf-part-equal*: $\{(h \circ (\lambda x.\ (q \cdot x)^{<-1} q\ ) \circ \mathfrak{f})\ (w \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ t)\ |w.\ w \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle\}$
$= \{\beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k) \cdot w \cdot \delta^{<-1}(\beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k - Suc\ t))\ |w.\ w \in \langle\{\delta \cdot (\gamma \cdot \beta)\ ^{@}\ i\ |i.\ i \leq t\}\rangle\}$ (**is** *?I* = *?E*)
proof
show *?I* ⊆ *?E*
proof
fix x **assume** $x \in$ *?I*
then obtain w **where** $x = h\ ((q \cdot \mathfrak{f}\ (w \cdot [a1] \cdot [1 - a1]\ ^{@}\ Suc\ t))^{<-1} q\ )$ **and**
$w \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle$
unfolding *mem-Collect-eq o-apply apply-h0* **by** *blast*

from *this*(*1*)[*unfolded apply-h0*] *w-decode*[*OF this*(*2*)]
show $x \in$ *?E* **by** *blast*
qed
next
show *?E* ⊆ *?I*
proof
fix x **assume** $x \in$ *?E*
then obtain w **where** x: $x = \beta \cdot (\gamma \cdot \beta)\ ^{@}\ (k' + k) \cdot w \cdot \delta^{<-1}(\beta \cdot (\gamma$

$\cdot\ \beta)\ ^{@}\ (k' + k - \mathit{Suc}\ t))$ **and**
$w \in \langle\{\delta \cdot (\gamma \cdot \beta)\ ^{@}\ i\ |i.\ i \leq t\}\rangle$ **by** *blast*
from *w-decode'*[*OF this*(*2*)]
obtain w' **where** $w' \in \langle\{[a1] \cdot [1 - a1]\ ^{@}\ i\ |i.\ i \leq t\}\rangle$ **and** $h_m\ (\mathfrak{f}\ w')$ $= w$ **by** *blast*
from *x*[*folded this*(*2*), *folded apply-h0*[*of w'*]] *this*(*1*)
show *x*∈ *?I*
unfolding *o-apply* **by** *blast*
qed
qed
have $1 \leq \mathit{Suc}\ t \wedge \mathit{Suc}\ t \leq k' + k$ **using** ‹$\mathit{Suc}\ t \leq k' + k$› **by** *simp*
from *that*(*5*)[*OF* ‹$\delta \neq \varepsilon$› ‹$\gamma \cdot \beta \neq \varepsilon$› ‹$\mathit{hd}\ \delta \neq \mathit{hd}\ (\gamma \cdot \beta)$› *this*]
show *thesis*
unfolding *diff-Suc-1* **unfolding** *int'* **unfolding** *inter* $\mathcal{W}$*-explicit bg-def'*[*symmetric*]
unfolding *inf-part-equal*[*symmetric*] **by** *blast*
qed
qed
qed
qed
qed

end

References

[1] J. Karhumäki. A note on intersections of free submonoids of a free monoid. *Semigroup forum*, 29:183–206, 1984.