Myhill-Nerode Theorem for (Nominal) G-Automata

Cárolos Laméris

March 17, 2025

Abstract

This work formalizes the Myhill-Nerode theorems for G-automata and nominal G-automata. The Myhill-Nerode theorem for (nominal) G-automata states that given an orbit finite (nominal) alphabet A and a G-language $L \subseteq A^*$ the following are equivalent:

- The set of equivalence classes of $L/\equiv_{\mathtt{MN}}$,with respect to the Myhill-Nerode equivalence relation, $\equiv_{\mathtt{MN}}$, is orbit finite.
- L is recognized by a deterministic (nominal) G-automaton with an orbit finite set of states.

The proofs formalized are based on those from [1].

Contents

1 Myhill-Nerode Theorem for G-automata **1**

1.1 Extending Group Actions 2

1.2 Equivariance and Quotient Actions 8

1.3 Basic (G)-Automata Theory 19

1.4 Syntactic Automaton 38

1.5 Proving the Myhill-Nerode Theorem for G-Automata 53

1.6 Proving the standard Myhill-Nerode Theorem 65

2 Myhill-Nerode Theorem for Nominal G-Automata **69**

2.1 Data Symmetries, Supports and Nominal Actions 69

2.2 Proving the Myhill-Nerode Theorem for Nominal G-Automata 75

1 Myhill-Nerode Theorem for G-automata

We prove the Myhill-Nerode Theorem for G-automata / nominal G-automata following the proofs from [1] (The standard Myhill-Nerode theorem is also proved, as a special case of the G-Myhill-Nerode theorem). Concretely, we formalize the following results from [1]: lemmas: 3.4, 3.5, 3.6, 3.7, 4.8, 4.9; proposition: 5.1; theorems: 3.8 (Myhill-Nerode for G-automata), 5.2 (Myhill-Nerode for nominal G-automata).

Throughout this document, we maintain the following convention for isar proofs: If we `obtain` some term `t` for which some result holds, we name it `H_t`. An assumption which is an induction hypothesis is named `A_IH` Assumptions start with an `"A"` and intermediate results start with a `"H"`. Typically we just name them via indexes, i.e. as `A_i` and `H_j`. When encountering nested isar proofs we add an index for how nested the assumption / intermediate result is. For example if we have an isar proof in an isar proof in an isar proof, we would name assumptions of the most nested proof `A3_i`.

theory *Nominal-Myhill-Nerode*
imports
Main
HOL.Groups
HOL.Relation
HOL.Fun
HOL−Algebra.Group-Action
HOL−Algebra.Elementary-Groups

begin

`GMN_simps` will contain selection of lemmas / definitions is updated through out the document.

named-theorems *GMN-simps*
lemmas *GMN-simps*

We will use the ⋆-symbol for the set of words of elements of a set, $A^\star$, the induced group action on the set of words $\phi^\star$ and for the extended transition function $\delta^\star$, thus we introduce the map `star` and apply `adhoc_overloading` to get the notation working in all three situations:

consts *star* :: *'typ1* ⇒ *'typ2* (‹-⋆› [*1000*] *999*)

adhoc-overloading
star ⇌ *lists*

We use ⊙ to convert between the definition of group actions via group homomoprhisms and the more standard infix group action notation. We deviate from [1] in that we consider left group actions, rather than right group actions:

definition
make-op :: (*'grp* ⇒ *'X* ⇒ *'X*) ⇒ *'grp* ⇒ *'X* ⇒ *'X* (**infixl** ‹(⊙ı)› *70*)
where (⊙ $_\varphi$) ≡ (λ*g*. (λ*x*. φ *g* *x*))

lemmas *make-op-def* [*simp*, *GMN-simps*]

1.1 Extending Group Actions

The following lemma is used for a proof in the locale `alt_grp_act`:

lemma *pre-image-lemma*:

$[\![S \subseteq T;\ x \in T \wedge f \in Bij\ T;\ (restrict\ f\ S)\ `\ S = S;\ f\ x \in S]\!] \Longrightarrow x \in S$
apply (*clarsimp simp add*: *extensional-def subset-eq Bij-def bij-betw-def restrict-def inj-on-def*)
by (*metis imageE*)

The locale `alt_grp_act` is just a renaming of the locale `group_action`. This was done to obtain more easy to interpret type names and context variables closer to the notation of [1]:

locale *alt-grp-act* = *group-action G X* φ
for
G :: ($'grp$, $'b$) *monoid-scheme* **and**
X :: $'X$ *set* (**structure**) **and**
φ
begin

lemma *alt-grp-act-is-left-grp-act*:
shows $x \in X \Longrightarrow \mathbf{1}_G \odot_\varphi x = x$ **and**
$g \in carrier\ G \Longrightarrow h \in carrier\ G \Longrightarrow x \in X \Longrightarrow (g \otimes_G h) \odot_\varphi x = g \odot_\varphi (h \odot_\varphi x)$
proof−
assume
A-0: $x \in X$
show $\mathbf{1}_G \odot_\varphi x = x$
using *group-action-axioms*
apply (*simp add*: *group-action-def BijGroup-def*)
by (*metis A-0 id-eq-one restrict-apply'*)
next
assume
A-0: $g \in carrier\ G$ **and**
A-1: $h \in carrier\ G$ **and**
A-2: $x \in X$
show $g \otimes_G h \odot_\varphi x = g \odot_\varphi (h \odot_\varphi x)$
using *group-action-axioms*
apply (*simp add*: *group-action-def group-hom-def group-hom-axioms-def hom-def BijGroup-def*)
using *composition-rule A-0 A-1 A-2*
by *auto*
qed

definition
induced-star-map :: $('grp \Rightarrow 'X \Rightarrow 'X) \Rightarrow 'grp \Rightarrow 'X\ list \Rightarrow 'X\ list$
where *induced-star-map func* = $(\lambda g{\in}carrier\ G.\ (\lambda lst \in X^\star.\ map\ (func\ g)\ lst))$

Because the adhoc overloading is used within a locale, isues will be encountered later due to there being multiple instances of the locale `alt_grp_act` in a single context:

adhoc-overloading
star $\rightleftharpoons$ *induced-star-map*

definition
induced-quot-map ::
$'Y$ *set* ⇒ ($'grp$ ⇒ $'Y$ ⇒ $'Y$) ⇒ ($'Y$ × $'Y$) *set* ⇒ $'grp$ ⇒ $'Y$ *set* ⇒ $'Y$ *set* (‹[-]-1› *60*)
where ([*func*]$_{R\ S}$) = (λ*g*∈*carrier G*. (λ*x* ∈ (*S* // *R*). *R* ‘‘ {(*func g*) (*SOME z*. *z*∈*x*)}))

lemmas *induced-star-map-def* [*simp*, *GMN-simps*]
induced-quot-map-def [*simp*, *GMN-simps*]

lemma *act-maps-n-distrib*:
∀ *g*∈*carrier G*. ∀ *w*∈$X^\star$. ∀ *v*∈$X^\star$. ($\varphi^\star$) *g* (*w* @ *v*) = (($\varphi^\star$) *g w*) @ (($\varphi^\star$) *g v*)
by (*auto simp add*: *group-action-def group-hom-def group-hom-axioms-def hom-def*)

lemma *triv-act*:
a ∈ *X* ⟹ (φ $\mathbf{1}_G$) *a* = *a*
using *group-hom.hom-one*[*of G BijGroup X* φ] *group-BijGroup*[**where** *S* = *X*]
apply (*clarsimp simp add*: *group-action-def group-hom-def group-hom-axioms-def BijGroup-def*)
by (*metis id-eq-one restrict-apply'*)

lemma *triv-act-map*:
∀ *w*∈$X^\star$. (($\varphi^\star$) $\mathbf{1}_G$) *w* = *w*
using *triv-act*
apply *clarsimp*
apply (*rule conjI*; *rule impI*)
apply *clarify*
using *map-idI*
apply *metis*
using *group.subgroup-self group-hom group-hom.axioms*(*1*) *subgroup.one-closed*
by *blast*

proposition *lists-a-Gset*:
alt-grp-act G ($X^\star$) ($\varphi^\star$)
proof−
have *H-0*: ⋀*g*. *g* ∈ *carrier G* ⟹
restrict (*map* (φ *g*)) ($X^\star$) ∈ *carrier* (*BijGroup* ($X^\star$))
proof−
fix *g*
assume
A1-0: *g* ∈ *carrier G*
from *A1-0* **have** *H1-0*: *inj-on* (λ*x*. *if x* ∈ $X^\star$ *then map* (φ *g*) *x else undefined*) ($X^\star$)
apply (*clarsimp simp add*: *inj-on-def*)
by (*metis* (*mono-tags*, *lifting*) *inj-onD inj-prop list.inj-map-strong*)
from *A1-0* **have** *H1-1*: ⋀*y z*. ∀ *x*∈*set y*. *x* ∈ *X* ⟹ *z* ∈ *set y* ⟹ φ *g z* ∈ *X*
using *element-image*
by *blast*
have *H1-2*: (*inv* $_G$ *g*) ∈ *carrier G*

by (*meson A1-0 group.inv-closed group-hom group-hom.axioms*(*1*))
have *H1-3*: $\bigwedge x.\ x \in X^\star \Longrightarrow$
map (*comp* (φ *g*) (φ (*inv* $_G$ *g*))) *x* = *map* (φ (*g* $\otimes_G$ (*inv* $_G$ *g*))) *x*
using *alt-grp-act-axioms*
apply (*simp add*: *alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def hom-def*
BijGroup-def)
apply (*rule meta-mp*[*of* $\bigwedge x.\ x \in$ *carrier G* $\Longrightarrow \varphi\ x \in$ *Bij X*])
apply (*metis A1-0 H1-2 composition-rule in-lists-conv-set*)
by *blast*
from *H1-2* **have** *H1-4*: $\bigwedge x.\ x \in X^\star \Longrightarrow$ *map* (φ (*inv* $_G$ *g*)) *x* $\in X^\star$
using *surj-prop*
by *fastforce*
have *H1-5*: $\bigwedge y.\ \forall x{\in}$*set y*. $x \in X \Longrightarrow y \in$ *map* (φ *g*) ‘ $X^\star$
apply (*simp add*: *image-def*)
using *H1-3 H1-4*
by (*metis A1-0 group.r-inv group-hom group-hom.axioms*(*1*) *in-lists-conv-set map-idI map-map*
triv-act)
show *restrict* (*map* (φ *g*)) ($X^\star$) $\in$ *carrier* (*BijGroup* ($X^\star$))
apply (*clarsimp simp add*: *restrict-def BijGroup-def Bij-def*
extensional-def bij-betw-def)
apply (*rule conjI*)
using *H1-0*
apply *simp*
using *H1-1 H1-5*
by (*auto simp add*: *image-def*)
qed
have *H-1*: $\bigwedge x\ y.\ [\![x \in$ *carrier G*; $y \in$ *carrier G*; $x \otimes_G y \in$ *carrier G*$]\!] \Longrightarrow$
restrict (*map* (φ (*x* $\otimes_G$ *y*))) ($X^\star$) =
restrict (*map* (φ *x*)) ($X^\star$) $\otimes_{BijGroup\ (X^\star)}$
restrict (*map* (φ *y*)) ($X^\star$)
proof−
fix *x y*
assume
A1-0: $x \in$ *carrier G* **and**
A1-1: $y \in$ *carrier G* **and**
A1-2: $x \otimes_G y \in$ *carrier G*
have *H1-0*: $\bigwedge z.\ z \in$ *carrier G* $\Longrightarrow$
bij-betw (λx. *if* $x \in X^\star$ *then map* (φ *z*) *x else undefined*) ($X^\star$) ($X^\star$)
using ‹$\bigwedge g.\ g \in$ *carrier G* $\Longrightarrow$ *restrict* (*map* (φ *g*)) ($X^\star$) $\in$ *carrier* (*BijGroup* ($X^\star$))›
by (*auto simp add*: *BijGroup-def Bij-def bij-betw-def inj-on-def*)
from *A1-1* **have** *H1-1*: $\bigwedge$*lst*. *lst* $\in X^\star \Longrightarrow$ (*map* (φ *y*)) *lst* $\in X^\star$
by (*metis group-action.surj-prop group-action-axioms lists-image rev-image-eqI*)
have *H1-2*: $\bigwedge a.\ a \in X^\star \Longrightarrow$ *map* (λxb.
if xb $\in X$
then φ *x* ((φ *y*) *xb*)
else undefined) *a* = *map* (φ *x*) (*map* (φ *y*) *a*)

```
      by auto
    have H1-3: (λxa. if xa ∈ X⋆ then map (φ (x ⊗G y)) xa else undefined) =
    compose (X⋆) (λxa. if xa ∈ X⋆ then map (φ x) xa else undefined)
    (λx. if x ∈ X⋆ then map (φ y) x else undefined)
      using alt-grp-act-axioms
      apply (clarsimp simp add: compose-def alt-grp-act-def group-action-def
          group-hom-def group-hom-axioms-def hom-def BijGroup-def restrict-def)
      using A1-0 A1-1 H1-2 H1-1 bij-prop0
      by auto
    show restrict (map (φ (x ⊗G y))) (X⋆) =
  restrict (map (φ x)) (X⋆) ⊗BijGroup (X⋆)
  restrict (map (φ y)) (X⋆)
      apply (clarsimp simp add: restrict-def BijGroup-def Bij-def extensional-def)
      apply (simp add: H1-3)
      using A1-0 A1-1 H1-0
      by auto
  qed
  show alt-grp-act G (X⋆) (φ⋆)
   apply (clarsimp simp add: alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def)
    apply (intro conjI)
    using group-hom group-hom-def
      apply (auto)[1]
     apply (simp add: group-BijGroup)
    apply (clarsimp simp add: hom-def)
    apply (intro conjI; clarify)
     apply (rule H-0)
     apply simp
    apply (rule conjI; rule impI)
     apply (rule H-1)
      apply simp+
    apply (rule meta-mp[of ⋀x y. x ∈ carrier G ⟹
    y ∈ carrier G ⟹ x ⊗G y ∈ carrier G])
     apply blast
   by (meson group.subgroup-self group-hom group-hom.axioms(1) subgroup.m-closed)
qed
end

lemma alt-group-act-is-grp-act [simp, GMN-simps]:
  alt-grp-act = group-action
  using alt-grp-act-def
  by blast

lemma prod-group-act:
  assumes
    grp-act-A: alt-grp-act G A φ and
    grp-act-B: alt-grp-act G B ψ
  shows alt-grp-act G (A×B) (λg∈carrier G. λ(a, b) ∈ (A × B). (φ g a, ψ g b))
  apply (simp add: alt-grp-act-def group-action-def group-hom-def)
  apply (intro conjI)
```

subgoal
using *grp-act-A grp-act-B*
by (*auto simp add*: *alt-grp-act-def group-action-def group-hom-def*)
subgoal
using *grp-act-A grp-act-B*
by (*auto simp add*: *alt-grp-act-def group-action-def group-hom-def group-BijGroup*)
apply (*clarsimp simp add*: *group-hom-axioms-def hom-def BijGroup-def*)
apply (*intro conjI*; *clarify*)
subgoal for *g*
apply (*clarsimp simp add*: *Bij-def bij-betw-def inj-on-def restrict-def extensional-def*)
apply (*intro conjI*)
using *grp-act-A*
apply (*simp add*: *alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def BijGroup-def hom-def Pi-def compose-def Bij-def bij-betw-def inj-on-def*)
using *grp-act-B*
apply (*simp add*: *alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def BijGroup-def hom-def Pi-def compose-def Bij-def bij-betw-def inj-on-def*)
apply (*rule meta-mp*[*of* φ *g* $\in$ *Bij A* $\wedge$ ψ *g* $\in$ *Bij B*])
apply (*clarsimp simp add*: *Bij-def bij-betw-def*)
using *grp-act-A grp-act-B*
apply (*simp add*: *alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def BijGroup-def hom-def Pi-def Bij-def*)
using *grp-act-A grp-act-B*
apply (*clarsimp simp add*: *compose-def restrict-def image-def alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def BijGroup-def hom-def Pi-def Bij-def bij-betw-def*)
apply (*rule subset-antisym*)
apply *blast*+
by (*metis alt-group-act-is-grp-act group-action.bij-prop0 grp-act-A grp-act-B*)
apply (*intro conjI*; *intro impI*)
apply (*clarify*)
apply (*intro conjI*; *intro impI*)
apply (*rule conjI*)
subgoal for *x y*
apply *unfold-locales*
apply (*clarsimp simp add*: *Bij-def compose-def restrict-def bij-betw-def*)
apply (*rule extensionalityI*[**where** $A = A \times B$])
apply (*clarsimp simp add*: *extensional-def*)
using *grp-act-A grp-act-B*
apply (*simp add*: *alt-grp-act-def group-action-def group-hom-def group-hom-axioms-def BijGroup-def hom-def Pi-def Bij-def compose-def extensional-def*)
apply (*simp add*: *fun-eq-iff*; *rule conjI*; *rule impI*)
using *group-action.composition-rule*[*of G A* φ] *group-action.composition-rule*[*of G B* ψ] *grp-act-A grp-act-B*
apply *force*
by *blast*

apply (*simp add*: ‹$\bigwedge g.\ g \in carrier\ G \Longrightarrow (\lambda(a,\ b)\in A \times B.\ (\varphi\ g\ a,\ \psi\ g\ b)) \in Bij\ (A \times B)$›)
apply (*simp add*: ‹*Group.group G*› *group.subgroup-self subgroup.m-closed*)
by (*simp add*: ‹$\bigwedge g.\ g \in carrier\ G \Longrightarrow (\lambda(a,\ b)\in A \times B.\ (\varphi\ g\ a,\ \psi\ g\ b)) \in Bij\ (A \times B)$›)+

1.2 Equivariance and Quotient Actions

locale *eq-var-subset* = *alt-grp-act G X* φ
for
G :: (*'grp*, *'b*) *monoid-scheme* **and**
X :: *'X set* (**structure**) **and**
φ +
fixes
Y
assumes
is-subset: $Y \subseteq X$ **and**
is-equivar: $\forall g\in carrier\ G.\ (\varphi\ g)$ ‘ $Y = Y$

lemma (**in** *alt-grp-act*) *eq-var-one-direction*:
$\bigwedge Y.\ Y \subseteq X \Longrightarrow \forall g\in carrier\ G.\ (\varphi\ g)$ ‘ $Y \subseteq Y \Longrightarrow$ *eq-var-subset G X* φ *Y*
proof−
fix *Y*
assume
A-0: $Y \subseteq X$ **and**
A-1: $\forall g\in carrier\ G.\ (\varphi\ g)$ ‘ $Y \subseteq Y$
have *H-0*: $\bigwedge g.\ g \in carrier\ G \Longrightarrow (inv_G\ g) \in carrier\ G$
by (*meson group.inv-closed group-hom group-hom.axioms*(*1*))
hence *H-1*: $\bigwedge g\ y.\ g \in carrier\ G \Longrightarrow y \in Y \Longrightarrow (\varphi\ (inv_G\ g))\ y \in Y$
using *A-1*
by (*simp add*: *image-subset-iff*)
have *H-2*: $\bigwedge g\ y.\ g \in carrier\ G \Longrightarrow y \in Y \Longrightarrow \varphi\ g\ ((\varphi\ (inv_G\ g))\ y) = y$
by (*metis A-0 bij-prop1 orbit-sym-aux subsetD*)
show *eq-var-subset G X* φ *Y*
apply (*simp add*: *eq-var-subset-def eq-var-subset-axioms-def*)
apply (*intro conjI*)
apply (*simp add*: *group-action-axioms*)
apply (*rule A-0*)
apply (*clarify*)
apply (*rule subset-antisym*)
using *A-1*
apply *simp*
apply (*simp add*: *image-def*)
apply (*rule subsetI*)
apply *clarify*
using *H-1 H-2*
by *metis*
qed

The following lemmas are used for proofs in the locale `eq_var_rel`:

lemma *some-equiv-class-id*:
  $[\![$*equiv X R*; $w \in X$ // R; $x \in w]\!] \Longrightarrow R$ ‘‘ $\{x\} = R$ ‘‘ $\{SOME\ z.\ z \in w\}$
  by (*smt* (*verit*) *Eps-cong equiv-Eps-in equiv-class-eq-iff quotient-eq-iff*)

lemma *nested-somes*:
  $[\![$*equiv X R*; $w \in X$ // $R]\!] \Longrightarrow (SOME\ z.\ z \in w) = (SOME\ z.\ z \in R$‘‘$\{(SOME\ z'.\ z' \in w)\})$
  by (*metis proj-Eps proj-def*)

locale *eq-var-rel* = *alt-grp-act G X* φ
  for
    G :: ($'grp$, $'b$) *monoid-scheme* **and**
    X :: $'X$ *set* (**structure**) **and**
    φ +
  fixes R
  assumes
    is-subrel:
    $R \subseteq X \times X$ **and**
    is-eq-var-rel:
    $\bigwedge a\ b.\ (a,\ b) \in R \Longrightarrow \forall g \in carrier\ G.\ (g \odot_{\varphi} a,\ g \odot_{\varphi} b) \in R$
begin

lemma *is-eq-var-rel'* [*simp*, *GMN-simps*]:
  $\bigwedge a\ b.\ (a,\ b) \in R \Longrightarrow \forall g \in carrier\ G.\ ((\varphi\ g)\ a,\ (\varphi\ g)\ b) \in R$
  using *is-eq-var-rel*
  by *auto*

lemma *is-eq-var-rel-rev*:
  $a \in X \Longrightarrow b \in X \Longrightarrow g \in carrier\ G \Longrightarrow (g \odot_{\varphi} a,\ g \odot_{\varphi} b) \in R \Longrightarrow (a,\ b) \in R$
proof −
  assume
    A-0: $(g \odot_{\varphi} a,\ g \odot_{\varphi} b) \in R$ **and**
    A-1: $a \in X$ **and**
    A-2: $b \in X$ **and**
    A-3: $g \in carrier\ G$
  have *H-0*: *group-action G X* φ **and**
    H-1: $R \subseteq X \times X$ **and**
    H-2: $\bigwedge a\ b\ g.\ (a,\ b) \in R \Longrightarrow g \in carrier\ G \Longrightarrow (\varphi\ g\ a,\ \varphi\ g\ b) \in R$
    by (*simp add*: *group-action-axioms is-subrel*)+
  from *H-0* **have** *H-3*: *group G*
    by (*auto simp add*: *group-action-def group-hom-def*)
  have *H-4*: $(\varphi\ (inv_G\ g)\ (\varphi\ g\ a),\ \varphi\ (inv_G\ g)\ (\varphi\ g\ b)) \in R$
    apply (*rule H-2*)
    using *A-0* **apply** *simp*
    by (*simp add*: *A-3 H-3*)
  from *H-3 A-3* **have** *H-5*: $(inv_G\ g) \in carrier\ G$
    by *auto*
  hence *H-6*: $\bigwedge e.\ e \in X \Longrightarrow \varphi\ (inv_G\ g)\ (\varphi\ g\ e) = \varphi\ ((inv_G\ g) \otimes_G g)\ e$
    using *H-0 A-3 group-action.composition-rule*

```
    by fastforce
  hence H-7: ⋀e. e ∈ X ⟹ φ (inv_G g) (φ g e) = φ 1_G e
    using H-3 A-3 group.l-inv
    by fastforce
  hence H-8: ⋀e. e ∈ X ⟹ φ (inv_G g) (φ g e) = e
    using H-0
    by (simp add: A-3 group-action.orbit-sym-aux)
  thus (a, b) ∈ R
    using A-1 A-2 H-4
    by simp
qed

lemma equiv-equivar-class-some-eq:
  assumes
    A-0: equiv X R and
    A-1: w ∈ X // R and
    A-2: g ∈ carrier G
  shows ([ φ ]_R) g w = R `` {(SOME z'. z' ∈ φ g ` w)}
proof-
  obtain z where H-z: w = R `` {z} ∧ z ∈ w
    by (metis A-0 A-1 equiv-class-self quotientE)
  have H-0: ⋀x. (φ g z, x) ∈ R ⟹ x ∈ φ g ` {y. (z, y) ∈ R}
  proof-
    fix y
    assume
      A1-0: (φ g z, y) ∈ R
    obtain y' where H2-y': y' = φ (inv_G g) y ∧ y' ∈ X
      using eq-var-rel-axioms
      apply (clarsimp simp add: eq-var-rel-def group-action-def group-hom-def)
     by (meson A-0 eq-var-rel-axioms A-2 A1-0 equiv-class-eq-iff eq-var-rel.is-eq-var-rel
          group.inv-closed element-image)
    from A-1 A-2 H2-y' have H2-0: φ g y' = y
      apply (clarsimp simp add: eq-var-rel-def eq-var-rel-axioms-def)
      using A-2 A1-0 group-action.bij-prop1[where G = G and E = X and φ =
φ]
        by (metis A-0 alt-group-act-is-grp-act alt-grp-act-axioms equiv-class-eq-iff
orbit-sym-aux)
    from A-1 A-2 A1-0 have H2-1: (z, y') ∈ R
      by (metis H2-0 A-0 A-2 H2-y' H-z equiv-class-eq-iff is-eq-var-rel-rev
          quotient-eq-iff make-op-def)
    thus y ∈ φ g ` {v. (z, v) ∈ R}
      using H2-0
      by (auto simp add: image-def)
  qed
  have H-1: φ g ` (R `` {z}) = R `` {φ g z}
    apply (clarsimp simp add: Relation.Image-def)
    apply (rule subset-antisym; simp add: Set.subset-eq; rule allI; rule impI)
    using eq-var-rel-axioms A-2 eq-var-rel.is-eq-var-rel
     apply simp
```

```
    by (simp add: H-0)
  have H-2: φ g ‘ w ∈ X // R
    using eq-var-rel-axioms A-1 A-2 H-1
    by (metis A-0 H-z equiv-class-eq-iff quotientI quotient-eq-iff element-image)
  thus ([φ]_R) g w = R ‘‘ {SOME z'. z' ∈ φ g ‘ w}
    using A-0 A-1 A-2
    apply (clarsimp simp add: Image-def)
    apply (intro subset-antisym)
     apply (clarify)
   using A-0 H-z imageI insert-absorb insert-not-empty some-in-eq some-equiv-class-id

     apply (smt (verit) A-1 Eps-cong Image-singleton-iff equiv-Eps-in)
    apply (clarify)
     by (smt (verit) Eps-cong equiv-Eps-in image-iff in-quotient-imp-closed quo-
tient-eq-iff)
qed

lemma ec-er-closed-under-action:
  assumes
    A-0: equiv X R and
    A-1: g ∈ carrier G and
    A-2: w ∈ X//R
  shows φ g ‘ w ∈ X // R
proof−
  obtain z where H-z: R ‘‘ {z} = w ∧ z ∈ X
    by (metis A-2 quotientE)
  have H-0: equiv X R ⟹ g ∈ carrier G ⟹ w ∈ X // R ⟹
      {y. (φ g z, y) ∈ R} ⊆ {y. ∃x. (z, x) ∈ R ∧ y = φ g x}
  proof (clarify)
    fix x
    assume
      A1-0: equiv X R and
      A1-1: g ∈ carrier G and
      A1-2: w ∈ X // R and
      A1-3: (φ g z, x) ∈ R
    obtain x' where H2-x': x = φ g x' ∧ x' ∈ X
      using group-action-axioms
      by (metis A1-1 is-subrel A1-3 SigmaD2 group-action.bij-prop1 subsetD)
    thus ∃y. (z, y) ∈ R ∧ x = φ g y
      using is-eq-var-rel-rev[where g = g and a = z and b = x'] A1-3
     by (auto simp add: eq-var-rel-def eq-var-rel-axioms-def A1-1 A1-2 group-action-axioms
H-z
          H2-x')
  qed
  have H-1: φ g ‘ R ‘‘ {z} = R ‘‘ {φ g z}
    using A-0 A-1 A-2
    apply (clarsimp simp add: eq-var-rel-axioms-def eq-var-rel-def
        Image-def image-def)
    apply (intro subset-antisym)
```

apply (*auto*)[*1*]
by (*rule H-0*)
thus φ *g* ' *w* $\in$ *X* // *R*
using *H-1 H-z*
by (*metis A-1 quotientI element-image*)
qed

The following lemma corresponds to the first part of lemma 3.5 from [1]:

lemma *quot-act-wd*:
$[\![$*equiv X R*; $x \in X$; $g \in$ *carrier G*$]\!] \Longrightarrow g \odot_{[\varphi]_R} (R$ '' $\{x\}) = (R$ '' $\{g \odot_\varphi x\})$
apply (*clarsimp simp add*: *eq-var-rel-def eq-var-rel-axioms-def*)
apply (*rule conjI*; *rule impI*)
apply (*smt* (*verit*, *best*) *Eps-cong Image-singleton-iff eq-var-rel.is-eq-var-rel'* *eq-var-rel-axioms equiv-Eps-in equiv-class-eq*)
by (*simp add*: *quotientI*)+

The following lemma corresponds to the second part of lemma 3.5 from [1]:

lemma *quot-act-is-grp-act*:
equiv X R $\Longrightarrow$ *alt-grp-act G* (*X* // *R*) ($[\varphi]_R$)
proof−
assume *A-0*: *equiv X R*
have *H-0*: $\bigwedge x$. *Group.group G* $\Longrightarrow$
Group.group (*BijGroup X*) $\Longrightarrow$
$R \subseteq X \times X \Longrightarrow$
$\varphi \in$ *carrier G* $\to$ *carrier* (*BijGroup X*) $\Longrightarrow$
$\forall x \in$*carrier G*. $\forall y \in$*carrier G*. φ ($x \otimes_G y$) = φ $x \otimes_{BijGroup\ X} \varphi$ $y \Longrightarrow$
$x \in$ *carrier G* $\Longrightarrow$ ($\lambda xa \in X$ // *R*. *R* '' $\{\varphi$ *x* (*SOME z*. $z \in xa$)$\}$) $\in$ *carrier* (*BijGroup* (*X* // *R*))
proof−
fix *g*
assume
A1-0: *Group.group G* **and**
A1-1: *Group.group* (*BijGroup X*) **and**
A1-2: $\varphi \in$ *carrier G* $\to$ *carrier* (*BijGroup X*) **and**
A1-3: $\forall x \in$*carrier G*. $\forall y \in$*carrier G*. φ ($x \otimes_G y$) = φ $x \otimes_{BijGroup\ X} \varphi$ y **and**
A1-4: $g \in$ *carrier G*
have *H-0*: *group-action G X* φ
apply (*clarsimp simp add*: *group-action-def group-hom-def group-hom-axioms-def*)
apply (*simp add*: *A1-0 A1-1*)+
apply (*simp add*: *hom-def*)
apply (*rule conjI*)
using *A1-2*
apply *blast*
by (*simp add*: *A1-3*)
have *H1-0*: $\bigwedge x\ y$. $[\![x \in X$ // *R*; $y \in X$ // *R*; *R* '' $\{\varphi$ *g* (*SOME z*. $z \in x$)$\}$ =
R '' $\{\varphi$ *g* (*SOME z*. $z \in y$)$\}]\!] \Longrightarrow x \subseteq y$
proof (*clarify*; *rename-tac a*)
fix *x y a*

assume
A2-0: $x \in X$ // R **and**
A2-1: $y \in X$ // R **and**
A2-2: R `` $\{\varphi\ g\ (SOME\ z.\ z \in x)\} = R$ `` $\{\varphi\ g\ (SOME\ z.\ z \in y)\}$ **and**
A2-3: $a \in x$
obtain b **where** *H2-b*: R `` $\{b\} = y \wedge b \in X$
by (*metis A2-1 quotientE*)
obtain a' b' **where** *H2-a'-b'*: $a' \in x \wedge b' \in y \wedge R$ `` $\{\varphi\ g\ a'\} = R$ `` $\{\varphi\ g\ b'\}$
by (*metis A-0 A2-1 A2-2 A2-3 equiv-Eps-in some-eq-imp*)
from *H2-a'-b'* **have** *H2-2*: $(\varphi\ g\ a',\ \varphi\ g\ b') \in R$
by (*metis A-0 A1-4 A2-1 Image-singleton-iff eq-var-rel.is-eq-var-rel' eq-var-rel-axioms quotient-eq-iff*)
hence *H2-0*: $(\varphi\ (inv_G\ g)\ (\varphi\ g\ a'),\ \varphi\ (inv_G\ g)\ (\varphi\ g\ b')) \in R$
by (*simp add*: *A1-0 is-eq-var-rel A1-4*)
have *H2-1*: $a' \in X \wedge b' \in X$
using *A-0 A2-0 A2-1 H2-a'-b' in-quotient-imp-subset*
by *blast*
hence *H2-2*: $(a',\ b') \in R$
using *H2-0*
by (*metis A1-4 H-0 group-action.orbit-sym-aux*)
have *H2-3*: $(a,\ a') \in R$
by (*meson A-0 A2-0 A2-3 H2-a'-b' quotient-eq-iff*)
hence *H2-4*: $(b',\ a) \in R$
using *H2-2*
by (*metis A-0 A2-0 A2-1 A2-3 H2-a'-b' quotient-eqI quotient-eq-iff*)
thus $a \in y$
by (*metis A-0 A2-1 H2-a'-b' in-quotient-imp-closed*)
qed
have *H1-1*: $\bigwedge x.\ x \in X$ // $R \Longrightarrow \exists xa{\in}X$ // $R.\ x = R$ `` $\{\varphi\ g\ (SOME\ z.\ z \in xa)\}$
proof −
fix x
assume
A2-0: $x \in X$ // R
have *H2-0*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow R$ `` $\{e\} \subseteq R$ `` $\{\varphi\ g\ (\varphi\ (inv_G\ g)\ e)\}$
proof (*rule subsetI*)
fix e y
assume
A3-0: R `` $\{e\} \in X$ // R **and**
A3-1: $y \in R$ `` $\{e\}$
have *H3-0*: $y \in X$
using *A3-1 is-subrel*
by *blast*
from *H-0* **have** *H3-1*: $\varphi\ g\ (\varphi\ (inv_G\ g)\ y) = y$
by (*metis* (*no-types*, *lifting*) *A1-0 A1-4 H3-0 group.inv-closed group.inv-inv group-action.orbit-sym-aux*)
from *A3-1* **have** *H3-2*: $(e,\ y) \in R$
by *simp*
hence *H3-3*: $((\varphi\ (inv_G\ g)\ e),\ (\varphi\ (inv_G\ g)\ y)) \in R$

using *is-eq-var-rel A1-4 A1-0*
by *simp*
hence *H3-4*: $(\varphi\ g\ (\varphi\ (inv_G\ g)\ e),\ \varphi\ g\ (\varphi\ (inv_G\ g)\ y)) \in R$
using *is-eq-var-rel A1-4 A1-0*
by *simp*
hence *H3-5*: $(\varphi\ g\ (\varphi\ (inv_G\ g)\ e),\ y) \in R$
using *H3-1*
by *simp*
thus $y \in R$ `` $\{\varphi\ g\ (\varphi\ (inv_G\ g)\ e)\}$
by *simp*
qed
hence *H2-1*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow R$ `` $\{e\} = R$ `` $\{\varphi\ g\ (\varphi\ (inv_G\ g)\ e)\}$
by (*metis A-0 proj-def proj-in-iff equiv-class-eq-iff subset-equiv-class*)
have *H2-2*: $\bigwedge e\ f.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow R$ `` $\{f\} \in X$ // $R \Longrightarrow$
R `` $\{e\} = R$ `` $\{f\} \Longrightarrow \forall f' \in R$ `` $\{f\}.\ R$ `` $\{e\} = R$ `` $\{f'\}$
by (*metis A-0 Image-singleton-iff equiv-class-eq*)
have *H2-3*: $x \in X$ // $R \Longrightarrow \exists\, e \in X.\ x = R$ `` $\{e\}$
by (*meson quotientE*)
have *H2-4*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow R$ `` $\{e\} = R$ `` $\{\varphi\ g\ (\varphi\ (inv_G\ g)\ e)\}$ $\wedge$
$(\varphi\ (inv_G\ g)\ e) \in R$ `` $\{\varphi\ (inv_G\ g)\ e\}$
by (*metis A1-0 A1-4 A-0 H2-1 Image-singleton-iff element-image equiv-Eps-in equiv-class-eq-iff*
group.inv-closed)
have *H2-5*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow \forall\, z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\}.\ (\varphi\ (inv_G\ g)\ e,\ z) \in R$
by *simp*
hence *H2-6*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow$
$\forall\, z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\}.\ (\varphi\ g\ (\varphi\ (inv_G\ g)\ e),\ \varphi\ g\ z) \in R$
using *is-eq-var-rel' A1-4 A1-0*
by *blast*
hence *H2-7*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow \forall\, z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\}.\ (e,\ \varphi\ g\ z) \in R$
using *H2-1*
by *blast*
hence *H2-8*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow \forall\, z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\}.\ R$ `` $\{e\} = R$ `` $\{\varphi\ g\ z\}$
by (*meson A-0 equiv-class-eq-iff*)
have *H2-9*: $\bigwedge e.\ R$ `` $\{e\} \in X$ // $R \Longrightarrow$
R `` $\{e\} = R$ `` $\{\varphi\ g\ (SOME\ z.\ z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\})\}$
proof−
fix *e*
assume
A3-0: R `` $\{e\} \in X$ // R
show R `` $\{e\} = R$ `` $\{\varphi\ g\ (SOME\ z.\ z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\})\}$
apply (*rule someI2*[**where** $Q = \lambda z.\ R$ `` $\{e\} = R$ `` $\{\varphi\ g\ z\}$ **and**
$P = \lambda z.\ z \in R$ `` $\{\varphi\ (inv_G\ g)\ e\}$ **and** $a = \varphi\ (inv_G\ g)\ e$])
using *A3-0 H2-4*

apply *blast*
using *A3-0 H2-8*
by *auto*
qed
have *H2-10*: $\forall\, e.\ (R \text{ `` } \{e\} \in X\ //\ R \longrightarrow$
$(R \text{ `` } \{e\} = R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in R \text{ `` } \{\varphi\ (inv_G\ g)\ e\})\}))$
using *H2-9*
by *auto*
hence *H2-11*: $\forall\, e.\ (R \text{ `` } \{e\} \in X\ //\ R \longrightarrow$
$(\exists\, xa {\in} X\ //\ R.\ R \text{ `` } \{e\} = R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in xa)\}))$
using *H2-8*
apply *clarsimp*
by (*smt* (*verit*, *best*) *A-0 H2-3 H2-5 H2-4 equiv-Eps-in equiv-class-eq-iff quotientI*)
have *H2-12*: $\bigwedge x.\ x \in X\ //\ R \Longrightarrow \exists\, e{\in}X.\ x = R \text{ `` } \{e\}$
by (*meson quotientE*)
have *H2-13*: $\bigwedge x.\ x \in X\ //\ R \Longrightarrow \exists\, xa{\in}X\ //\ R.\ x = R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in xa)\}$
using *H2-11 H2-12*
by *blast*
show $\exists\, xa{\in}X\ //\ R.\ x = R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in xa)\}$
by (*simp add*: *A2-0 H2-13*)
qed
show $(\lambda x{\in}X\ //\ R.\ R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in x)\}) \in carrier\ (BijGroup\ (X\ //\ R))$
apply (*clarsimp simp add*: *BijGroup-def Bij-def bij-betw-def*)
apply (*clarsimp simp add*: *inj-on-def*)
apply (*rule conjI*)
apply (*clarsimp*)
apply (*rule subset-antisym*)
apply (*simp add*: *H1-0*)
apply (*simp add*: ‹$\bigwedge y\ x.\ [\![x \in X\ //\ R;$
$y \in X\ //\ R;\ R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in x)\} = R \text{ `` } \{\varphi\ g\ (SOME\ z.\ z \in y)\}]\!]$
$\Longrightarrow x \subseteq y$›)
apply (*rule subset-antisym*; *clarify*)
subgoal for *x y*
by (*metis A-0 is-eq-var-rel′ A1-4 Eps-cong equiv-Eps-preserves equiv-class-eq-iff quotientI*)
apply (*clarsimp simp add*: *Set.image-def*)
by (*simp add*: *H1-1*)
qed
have *H-1*: $\bigwedge x\ y.\ [\![Group.group\ G;\ Group.group\ (BijGroup\ X);\ R \subseteq X \times X;$
$\varphi \in carrier\ G \to carrier\ (BijGroup\ X);$
$\forall\, x{\in}carrier\ G.\ \forall\, y{\in}carrier\ G.\ \varphi\ (x \otimes_G y) = \varphi\ x \otimes_{BijGroup\ X} \varphi\ y;$
$x \in carrier\ G;\ y \in carrier\ G;\ x \otimes_G y \in carrier\ G]\!] \Longrightarrow$
$(\lambda xa{\in}X\ //\ R.\ R \text{ `` } \{(\varphi\ x \otimes_{BijGroup\ X} \varphi\ y)\ (SOME\ z.\ z \in xa)\}) =$
$(\lambda xa{\in}X\ //\ R.\ R \text{ `` } \{\varphi\ x\ (SOME\ z.\ z \in xa)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x{\in}X\ //\ R.\ R \text{ `` } \{\varphi\ y\ (SOME\ z.\ z \in x)\})$
proof −

fix $x\ y$
assume
$A1\text{-}1$: $Group.group\ G$ **and**
$A1\text{-}2$: $Group.group\ (BijGroup\ X)$ **and**
$A1\text{-}3$: $\varphi \in carrier\ G \to carrier\ (BijGroup\ X)$ **and**
$A1\text{-}4$: $\forall x{\in}carrier\ G.\ \forall y{\in}carrier\ G.\ \varphi\ (x \otimes_G y) = \varphi\ x \otimes_{BijGroup\ X} \varphi\ y$ **and**
$A1\text{-}5$: $x \in carrier\ G$ **and**
$A1\text{-}6$: $y \in carrier\ G$ **and**
$A1\text{-}7$: $x \otimes_G y \in carrier\ G$
have $H1\text{-}0$: $\bigwedge w{::}'X\ set.\ w \in X\ //\ R \Longrightarrow$
$R\ ``\ \{(\varphi\ x \otimes_{BijGroup\ X} \varphi\ y)\ (SOME\ z.\ z \in w)\} =$
$((\lambda v{\in}X\ //\ R.\ R\ ``\ \{\varphi\ x\ (SOME\ z.\ z \in v)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x{\in}X\ //\ R.\ R\ ``\ \{\varphi\ y\ (SOME\ z.\ z \in x)\}))\ w$
proof−
fix w
assume
$A2\text{-}0$: $w \in X\ //\ R$
have $H2\text{-}4$: $\varphi\ y\ `\ w \in X\ //\ R$
using *ec-er-closed-under-action*[**where** $w = w$ **and** $g = y$]
by (*clarsimp simp add*: *group-hom-axioms-def hom-def A-0 A1-1 A1-2*
is-eq-var-rel′ A1-3 A1-4
A1-6 A2-0)
hence $H2\text{-}1$: $R\ ``\ \{(\varphi\ x \otimes_{BijGroup\ X} \varphi\ y)\ (SOME\ z.\ z \in w)\} =$
$R\ ``\ \{\varphi\ (x \otimes_G y)\ (SOME\ z.\ z \in w)\}$
using *A1-4 A1-5 A1-6*
by *auto*
also have $H2\text{-}2$: $... = R\ ``\ \{SOME\ z.\ z \in \varphi\ (x \otimes_G y)\ `\ w\}$
using *A1-7 equiv-equivar-class-some-eq*[**where** $w = w$ **and** $g = x \otimes_G y$]
by (*clarsimp simp add*: *A1-7 A-0 A2-0 group-action-def group-hom-def*
group-hom-axioms-def
hom-def)
also have $H2\text{-}3$: $... = R\ ``\ \{SOME\ z.\ z \in \varphi\ x\ `\ \varphi\ y\ `\ w\}$
apply (*rule meta-mp*[*of* $\neg(\exists x.\ x \in w \land x \notin X)$])
using *A1-1 is-eq-var-rel′ A1-3 A1-4 A1-5 A1-6 A2-0*
apply (*clarsimp simp add*: *image-def BijGroup-def restrict-def compose-def*
Pi-def)
apply (*smt* (*verit*) *Eps-cong*)
apply (*clarify*)
using *A-0 A2-0 in-quotient-imp-subset*
by *auto*
also have $H2\text{-}5$: $... = R\ ``\ \{\varphi\ x\ (SOME\ z.\ z \in \varphi\ y\ `\ w)\}$
using *equiv-equivar-class-some-eq*[**where** $w = \varphi\ y\ `\ w$ **and** $g = x$]
apply (*clarsimp simp add*: *A-0 group-action-def group-hom-def group-hom-axioms-def*
hom-def)
by (*simp add*: *A1-1 A1-2 is-eq-var-rel′ A1-3 A1-4 A1-5 H2-4*)
also have $H2\text{-}6$: $... = R\ ``\ \{\varphi\ x\ (SOME\ z.\ z \in R``\{(SOME\ z'.\ z' \in \varphi\ y\ `$
$w)\})\}$
using *H2-4 nested-somes*[**where** $w = \varphi\ y\ `\ w$ **and** $X = X$ **and** $R = R$] *A-0*
by *presburger*

also have *H2-7*: ... = R `` $\{\varphi\ x\ (SOME\ z.\ z \in R$ `` $\{\varphi\ y\ (SOME\ z'.\ z' \in w)\})\}$
using *equiv-equivar-class-some-eq*[**where** $g = y$ **and** $w = w$] *H2-6*
by (*simp add*: *A-0 group-action-def group-hom-def group-hom-axioms-def hom-def A1-1 A1-2 is-eq-var-rel′ A1-3 A1-4 A2-0 A1-6*)
also have *H2-9*: ... = $((\lambda v{\in}X\ //\ R.\ R$ `` $\{\varphi\ x\ (SOME\ z.\ z \in v)\}) \otimes_{BijGroup\ (X\ //\ R)}$ $(\lambda x{\in}X\ //\ R.\ R$ `` $\{\varphi\ y\ (SOME\ z.\ z \in x)\}))\ w$
proof−
have *H3-0*: $\bigwedge u.\ R$ `` $\{\varphi\ y\ (SOME\ z.\ z \in w)\} \in X\ //\ R \Longrightarrow u \in carrier\ G \Longrightarrow$ $(\lambda v{\in}X\ //\ R.\ R$ `` $\{\varphi\ u\ (SOME\ z.\ z \in v)\}) \in Bij\ (X\ //\ R)$
proof −
fix *u*
assume
A4-0: R `` $\{\varphi\ y\ (SOME\ z.\ z \in w)\} \in X\ //\ R$ **and**
A4-1: $u \in carrier\ G$
have *H4-0*: $\forall g \in carrier\ G.$ $(\lambda x{\in}X\ //\ R.\ R$ `` $\{\varphi\ g\ (SOME\ z.\ z \in x)\}) \in carrier\ (BijGroup\ (X\ //\ R))$
by (*simp add*: *A-0 A1-1 A1-2 A1-3 A1-4 H-0 is-subrel*)
thus $(\lambda v{\in}X\ //\ R.\ R$ `` $\{\varphi\ u\ (SOME\ z.\ z \in v)\}) \in Bij\ (X\ //\ R)$
by (*auto simp add*: *BijGroup-def A4-1*)
qed
have *H3-1*: R `` $\{\varphi\ y\ (SOME\ z.\ z \in w)\} \in X\ //\ R$
proof−
have *H4-0*: $\varphi\ y$ ` $w \in X\ //\ R$
using *ec-er-closed-under-action*
by (*simp add*: *H2-4*)
hence *H4-1*: R `` $\{(SOME\ z.\ z \in \varphi\ y$ ` $w)\} = \varphi\ y$ ` w
apply (*clarsimp simp add*: *image-def*)
apply (*rule subset-antisym*)
using *A-0 equiv-Eps-in in-quotient-imp-closed*
apply *fastforce*
using *A-0 equiv-Eps-in quotient-eq-iff*
by *fastforce*
have *H4-2*: R `` $\{\varphi\ y\ (SOME\ z.\ z \in w)\} = R$ `` $\{(SOME\ z.\ z \in \varphi\ y$ ` $w)\}$
using *equiv-equivar-class-some-eq*[**where** $g = y$ **and** $w = w$]
by (*metis A-0 A2-0 H4-0 H4-1 equiv-Eps-in imageI some-equiv-class-id*)
from *H4-0 H4-1 H4-2* **show** R `` $\{\varphi\ y\ (SOME\ z.\ z \in w)\} \in X\ //\ R$
by *auto*
qed
show *?thesis*
apply (*rule meta-mp*[*of* R `` $\{\varphi\ y\ (SOME\ z.\ z \in w)\} \in X\ //\ R$])
apply (*rule meta-mp*[*of* $\forall u \in carrier\ G.$ $(\lambda v{\in}X\ //\ R.\ R$ `` $\{\varphi\ u\ (SOME\ z.\ z \in v)\}) \in Bij\ (X\ //\ R)$])
using *A2-0 A1-5 A1-6*
apply (*simp add*: *BijGroup-def compose-def*)
apply *clarify*
by (*simp add*: *H3-0 H3-1*)+

qed
finally show R '' $\{(\varphi\ x \otimes_{BijGroup\ X} \varphi\ y)\ (SOME\ z.\ z \in w)\} =$
$((\lambda v{\in}X\ //\ R.\ R$ '' $\{\varphi\ x\ (SOME\ z.\ z \in v)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x{\in}X\ //\ R.\ R$ '' $\{\varphi\ y\ (SOME\ z.\ z \in x)\}))\ w$
by *simp*
qed
have *H1-1*: $\bigwedge w{::}'X\ set.\ w \notin X\ //\ R \Longrightarrow$
$((\lambda v{\in}X\ //\ R.\ R$ '' $\{\varphi\ x\ (SOME\ z.\ z \in v)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x{\in}X\ //\ R.\ R$ '' $\{\varphi\ y\ (SOME\ z.\ z \in x)\}))\ w = undefined$
proof −
fix *w*
assume
A2-0: $w \notin X\ //\ R$
have *H2-0*: $\bigwedge u.\ u{\in}\ carrier\ G \Longrightarrow (\lambda v{\in}X\ //\ R.\ R$ '' $\{\varphi\ u\ (SOME\ z.\ z \in v)\})$
$\in Bij\ (X\ //\ R)$
using *H-0*
apply (*clarsimp simp add*: *A-0 A1-1 A1-2 is-eq-var-rel′ A1-3 A1-4 is-subrel*)
by (*simp add*: *BijGroup-def*)
hence *H2-1*: $(\lambda x'{\in}X\ //\ R.\ R$ '' $\{\varphi\ y\ (SOME\ z.\ z \in x')\}) \in Bij\ (X\ //\ R)$
using *A1-6*
by *auto*
from *H2-0* **have** *H2-2*: $(\lambda x'{\in}X\ //\ R.\ R$ '' $\{\varphi\ x\ (SOME\ z.\ z \in x')\}) \in Bij$
$(X\ //\ R)$
by (*simp add*: *A1-5*)
thus $((\lambda v{\in}X\ //\ R.\ R$ '' $\{\varphi\ x\ (SOME\ z.\ z \in v)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x{\in}X\ //\ R.\ R$ '' $\{\varphi\ y\ (SOME\ z.\ z \in x)\}))\ w = undefined$
using *H2-1 H2-2*
by (*auto simp add*: *BijGroup-def compose-def A2-0*)
qed
from *H1-0 H1-1* **have** $\bigwedge w.\ (\lambda xa{\in}X\ //\ R.\ R$ '' $\{(\varphi\ x \otimes_{BijGroup\ X} \varphi\ y)\ (SOME$
$z.\ z \in xa)\})\ w =$
$((\lambda xa{\in}X\ //\ R.\ R$ '' $\{\varphi\ x\ (SOME\ z.\ z \in xa)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x'{\in}X\ //\ R.\ R$ '' $\{\varphi\ y\ (SOME\ z.\ z \in x')\}))\ w$
by *auto*
thus $(\lambda xa{\in}X\ //\ R.\ R$ '' $\{(\varphi\ x \otimes_{BijGroup\ X} \varphi\ y)\ (SOME\ z.\ z \in xa)\}) =$
$(\lambda xa{\in}X\ //\ R.\ R$ '' $\{\varphi\ x\ (SOME\ z.\ z \in xa)\}) \otimes_{BijGroup\ (X\ //\ R)}$
$(\lambda x{\in}X\ //\ R.\ R$ '' $\{\varphi\ y\ (SOME\ z.\ z \in x)\})$
by (*simp add*: *restrict-def*)
qed
show *?thesis*
apply (*clarsimp simp add*: *group-action-def group-hom-def*)
using *eq-var-rel-axioms*
apply (*clarsimp simp add*: *eq-var-rel-def eq-var-rel-axioms-def*
group-action-def group-hom-def)
apply (*rule conjI*)
apply (*simp add*: *group-BijGroup*)
apply (*clarsimp simp add*: *group-hom-axioms-def hom-def*)
apply (*intro conjI*)

```
  apply (rule funcsetI; simp)
  apply (simp add: H-0)
 apply (clarify; rule conjI; intro impI)
  apply (simp add: H-1)
 by (auto simp add: group.is-monoid monoid.m-closed)
qed
end

locale eq-var-func = GA-0: alt-grp-act G X φ + GA-1: alt-grp-act G Y ψ
  for
    G :: ('grp, 'b) monoid-scheme and
    X :: 'X set and
    φ and
    Y :: 'Y set and
    ψ +
  fixes
    f :: 'X ⇒ 'Y
  assumes
    is-ext-func-bet:
    f ∈ (X →E Y) and
    is-eq-var-func:
    ⋀a g. a ∈ X ⟹ g ∈ carrier G ⟹ f (g ⊙φ a) = g ⊙ψ (f a)
begin

lemma is-eq-var-func' [simp]:
  a ∈ X ⟹ g ∈ carrier G ⟹ f (φ g a) = ψ g (f a)
  using is-eq-var-func
  by auto

end

lemma G-set-equiv:
  alt-grp-act G A φ ⟹ eq-var-subset G A φ A
  by (auto simp add: eq-var-subset-def eq-var-subset-axioms-def group-action-def
    group-hom-def group-hom-axioms-def hom-def BijGroup-def Bij-def bij-betw-def)
```

1.3 Basic (G)-Automata Theory

```
locale language =
  fixes A :: 'alpha set and
    L
  assumes
    is-lang: L ⊆ A⋆

locale G-lang = alt-grp-act G A φ + language A L
  for
    G :: ('grp, 'b) monoid-scheme and
    A :: 'alpha set (structure) and
    φ L +
```

```
  assumes
    L-is-equivar:
    eq-var-subset G (A⋆) (induced-star-map φ) L
begin
lemma G-lang-is-lang[simp]: language A L
  by (simp add: language-axioms)
end

sublocale G-lang ⊆ language
  by simp

fun give-input :: ('state ⇒ 'alpha ⇒ 'state) ⇒ 'state ⇒ 'alpha list ⇒ 'state
  where give-input trans-func s Nil = s
  |   give-input trans-func s (a#as) = give-input trans-func (trans-func s a) as

adhoc-overloading
  star ⇌ give-input

locale det-aut =
  fixes
    labels :: 'alpha set and
    states :: 'state set and
    init-state :: 'state and
    fin-states :: 'state set and
    trans-func :: 'state ⇒ 'alpha ⇒ 'state (‹δ›)
  assumes
    init-state-is-a-state:
    init-state ∈ states and
    fin-states-are-states:
    fin-states ⊆ states and
    trans-func-ext:
    (λ(state, label). trans-func state label) ∈ (states × labels) →E states
begin

lemma trans-func-well-def:
  ⋀state label. state ∈ states ⟹ label ∈ labels ⟹ (δ state label) ∈ states
  using trans-func-ext
  by auto

lemma give-input-closed:
  input ∈ (labels⋆) ⟹ s ∈ states ⟹ (δ⋆) s input ∈ states
  apply (induction input arbitrary: s)
  by (auto simp add: trans-func-well-def)

lemma input-under-concat:
  w ∈ labels⋆ ⟹ v ∈ labels⋆ ⟹ (δ⋆) s (w @ v) = (δ⋆) ((δ⋆) s w) v
  apply (induction w arbitrary: s)
  by auto
```

lemma *eq-pres-under-concat*:
assumes
$w \in labels^\star$ **and**
$w' \in labels^\star$ **and**
$s \in states$ **and**
$(\delta^\star)\ s\ w = (\delta^\star)\ s\ w'$
shows $\forall\, v \in labels^\star.\ (\delta^\star)\ s\ (w\ @\ v) = (\delta^\star)\ s\ (w'\ @\ v)$
using *input-under-concat*[**where** $w = w$ **and** $s = s$] *input-under-concat*[**where** $w = w'$ **and** $s = s$] *assms*
by *auto*

lemma *trans-to-charact*:
$\bigwedge s\ a\ w.\ [\![s \in states;\ a \in labels;\ w \in labels^\star;\ s = (\delta^\star)\ i\ w]\!] \Longrightarrow (\delta^\star)\ i\ (w\ @\ [a]) = \delta\ s\ a$
proof−
fix $s\ a\ w$
assume
A-0: $s \in states$ **and**
A-1: $a \in labels$ **and**
A-2: $w \in labels^\star$ **and**
A-3: $s = (\delta^\star)\ i\ w$
have *H-0*: $trans\text{-}func\ s\ a = (\delta^\star)\ s\ [a]$
by *auto*
from *A-2 A-3 H-0* **have** *H-1*: $(\delta^\star)\ s\ [a] = (\delta^\star)\ ((\delta^\star)\ i\ w)\ [a]$
by *simp*
from *A-1 A-2* **have** *H-2*: $(\delta^\star)\ ((\delta^\star)\ i\ w)\ [a] = (\delta^\star)\ i\ (w\ @\ [a])$
using *input-under-concat*
by *force*
show $(\delta^\star)\ i\ (w\ @\ [a]) = \delta\ s\ a$
using *A-1 H-0 A-3 H-1 H-2*
by *force*
qed

end

locale *aut-hom* = *Aut0*: *det-aut* $A\ S_0\ i_0\ F_0\ \delta_0$ + *Aut1*: *det-aut* $A\ S_1\ i_1\ F_1\ \delta_1$ **for**
A :: *'alpha set* **and**
S_0 :: *'states-0 set* **and**
i_0 **and** F_0 **and** δ_0 **and**
S_1 :: *'states-1 set* **and**
i_1 **and** F_1 **and** δ_1 +
fixes f :: *'states-0* ⇒ *'states-1*
assumes
hom-is-ext:
$f \in S_0 \rightarrow_E S_1$ **and**
pres-init:
$f\ i_0 = i_1$ **and**
pres-final:
$s \in F_0 \longleftrightarrow f\ s \in F_1 \wedge s \in S_0$ **and**

pres-trans:
$s_0 \in S_0 \Longrightarrow a \in A \Longrightarrow f\ (\delta_0\ s_0\ a) = \delta_1\ (f\ s_0)\ a$
begin

lemma *hom-translation*:
$input \in (A^\star) \Longrightarrow s \in S_0 \Longrightarrow (f\ ((\delta_0{}^\star)\ s\ input)) = ((\delta_1{}^\star)\ (f\ s)\ input)$
apply (*induction input arbitrary*: *s*)
by (*auto simp add*: *Aut0.trans-func-well-def pres-trans*)

lemma *recognise-same-lang*:
$input \in A^\star \Longrightarrow ((\delta_0{}^\star)\ i_0\ input) \in F_0 \longleftrightarrow ((\delta_1{}^\star)\ i_1\ input) \in F_1$
using *hom-translation*[**where** *input* = *input* **and** $s = i_0$]
apply (*clarsimp simp add*: *Aut0.init-state-is-a-state pres-init pres-final*)
apply (*induction input*)
apply (*clarsimp simp add*: *Aut0.init-state-is-a-state*)
using *Aut0.give-input-closed Aut0.init-state-is-a-state*
by *blast*

end

locale *aut-epi* = *aut-hom* +
assumes
is-epi: f ‘ $S_0 = S_1$

locale *det-G-aut* =
is-aut: *det-aut* $A\ S\ i\ F\ \delta$ +
labels-a-G-set: *alt-grp-act* $G\ A\ \varphi$ +
states-a-G-set: *alt-grp-act* $G\ S\ \psi$ +
accepting-is-eq-var: *eq-var-subset* $G\ S\ \psi\ F$ +
init-is-eq-var: *eq-var-subset* $G\ S\ \psi\ \{i\}$ +
trans-is-eq-var: *eq-var-func* $G\ S \times A$
$\lambda g{\in}carrier\ G.\ \lambda(s, a) \in (S \times A).\ (\psi\ g\ s, \varphi\ g\ a)$
$S\ \psi\ (\lambda(s, a) \in (S \times A).\ \delta\ s\ a)$
for A :: *'alpha set* (**structure**) **and**
S :: *'states set* **and**
$i\ F\ \delta$ **and**
G :: (*'grp*, *'b*) *monoid-scheme* **and**
$\varphi\ \psi$
begin

adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*

lemma *give-input-eq-var*:
eq-var-func G
$(A^\star \times S)\ (\lambda g{\in}carrier\ G.\ \lambda(w, s) \in (A^\star \times S).\ ((\varphi^\star)\ g\ w, \psi\ g\ s))$
$S\ \psi$
$(\lambda(w, s) \in (A^\star \times S).\ (\delta^\star)\ s\ w)$
proof−

have *H-0*: $\bigwedge a\ w\ s\ g.$
$(\bigwedge s.\ s \in S \Longrightarrow (\varphi^\star)\ g\ w \in A^\star \wedge \psi\ g\ s \in S \Longrightarrow$
$(\delta^\star)\ (\psi\ g\ s)\ ((\varphi^\star)\ g\ w) = \psi\ g\ ((\delta^\star)\ s\ w)) \Longrightarrow$
$s \in S \Longrightarrow$
$g \in carrier\ G \Longrightarrow$
$a \in A \Longrightarrow \forall x{\in}set\ w.\ x \in A \Longrightarrow \psi\ g\ s \in S \Longrightarrow \forall x{\in}set\ ((\varphi^\star)\ g\ (a\ \#\ w)).\ x$
$\in A \Longrightarrow$
$(\delta^\star)\ (\psi\ g\ s)\ ((\varphi^\star)\ g\ (a\ \#\ w)) = \psi\ g\ ((\delta^\star)\ (\delta\ s\ a)\ w)$
proof−
fix *a w s g*
assume
A-IH: $(\bigwedge s.\ s \in S \Longrightarrow$
$(\varphi^\star)\ g\ w \in A^\star \wedge \psi\ g\ s \in S \Longrightarrow$
$(\delta^\star)\ (\psi\ g\ s)\ ((\varphi^\star)\ g\ w) = \psi\ g\ ((\delta^\star)\ s\ w))$ **and**
A-0: $s \in S$ **and**
A-1: $\psi\ g\ s \in S$ **and**
A-2: $\forall x{\in}set\ ((\varphi^\star)\ g\ (a\ \#\ w)).\ x \in A$ **and**
A-3: $g \in carrier\ G$ **and**
A-4: $a \in A$ **and**
A-5: $\forall x{\in}set\ w.\ x \in A$
have *H-0*: $((\varphi^\star)\ g\ (a\ \#\ w)) = (\varphi\ g\ a)\ \#\ (\varphi^\star)\ g\ w$
using *A-4 A-5 A-3*
by *auto*
hence *H-1*: $(\delta^\star)\ (\psi\ g\ s)\ ((\varphi^\star)\ g\ (a\ \#\ w))$
$= (\delta^\star)\ (\psi\ g\ s)\ ((\varphi\ g\ a)\ \#\ (\varphi^\star)\ g\ w)$
by *simp*
have *H-2*: ... $= (\delta^\star)\ ((\delta^\star)\ (\psi\ g\ s)\ [\varphi\ g\ a])\ ((\varphi^\star)\ g\ w)$
using *is-aut.input-under-concat*
by *simp*
have *H-3*: $(\delta^\star)\ (\psi\ g\ s)\ [\varphi\ g\ a] = \psi\ g\ (\delta\ s\ a)$
using *trans-is-eq-var.eq-var-func-axioms A-4 A-5 A-0 A-1 A-3* **apply** (*clarsimp*
simp del:
GMN-simps simp add: *eq-var-func-def eq-var-func-axioms-def make-op-def*)
apply (*rule meta-mp*[*of* $\psi\ g\ s \in S \wedge \varphi\ g\ a \in A \wedge s \in S \wedge a \in A$])
apply *presburger*
apply (*clarify*)
using *labels-a-G-set.element-image*
by *presburger*
have *H-4*: $(\delta^\star)\ (\psi\ g\ (\delta\ s\ a))\ ((\varphi^\star)\ g\ w) = \psi\ g\ ((\delta^\star)\ (\delta\ s\ a)\ w)$
apply (*rule A-IH*[**where** *s1* $= \delta\ s\ a$])
subgoal
using *A-4 A-5 A-0*
by (*auto simp add*: *is-aut.trans-func-well-def*)
using *A-4 A-5 A-0 A-3* ‹$\delta\ s\ a \in S$› *states-a-G-set.element-image*
by (*metis A-2 Cons-in-lists-iff H-0 in-listsI*)
show $(\delta^\star)\ (\psi\ g\ s)\ ((\varphi^\star)\ g\ (a\ \#\ w)) = \psi\ g\ ((\delta^\star)\ (\delta\ s\ a)\ w)$
using *H-0 H-1 H-2 H-3 H-4*
by *presburger*
qed

show *?thesis*
apply (*subst eq-var-func-def*)
apply (*subst eq-var-func-axioms-def*)
apply (*rule conjI*)
apply (*rule prod-group-act*[**where** $G = G$ **and** $A = A^\star$ **and** $\varphi = (\varphi^\star)$
and $B = S$ **and** $\psi = \psi$])
using *labels-a-G-set.lists-a-Gset*
apply *blast*
apply (*simp add*: *states-a-G-set.group-action-axioms*)
apply (*rule conjI*)
apply (*simp add*: *states-a-G-set.group-action-axioms*)
apply (*rule conjI*)
apply (*subst extensional-funcset-def*)
apply (*subst restrict-def*)
apply (*subst Pi-def*)
apply (*subst extensional-def*)
apply (*auto simp add*: *in-listsI is-aut.give-input-closed*)[*1*]
apply (*subst restrict-def*)
apply (*clarsimp simp del*: *GMN-simps simp add*: *make-op-def*)
apply (*rule conjI*; *intro impI*)
subgoal for *w s g*
apply (*induction w arbitrary*: *s*)
apply *simp*
apply (*clarsimp simp del*: *GMN-simps*)
by (*simp add*: *H-0 del*: *GMN-simps*)
apply *clarsimp*
by (*metis* (*no-types*, *lifting*) *image-iff in-lists-conv-set labels-a-G-set.surj-prop list.set-map*
states-a-G-set.element-image)
qed

definition
accepted-words :: *'alpha list set*
where *accepted-words* = $\{w.\ w \in A^\star \wedge ((\delta^\star)\ i\ w) \in F\}$

lemma *induced-g-lang*:
G-lang G A φ *accepted-words*
proof−
have *H-0*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \wedge (\delta^\star)\ i\ w \in F \Longrightarrow map\ (\varphi\ g)\ w \in A^\star$
apply (*clarsimp*)
using *labels-a-G-set.element-image*
by *blast*
have *H-1*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow (\delta^\star)\ i\ w \in F \Longrightarrow (\delta^\star)\ i\ (map\ (\varphi\ g)\ w) \in F$
proof −
fix *g w*
assume
A-0: $g \in carrier\ G$ **and**

```
      A-1: w ∈ A⋆ and
      A-2: (δ⋆) i w ∈ F
    have H1-0: ψ g ((δ⋆) i w) ∈ F
      using accepting-is-eq-var.eq-var-subset-axioms
        A-0 A-2 accepting-is-eq-var.is-equivar
      by blast
    have H1-1: ψ g i = i
      using init-is-eq-var.eq-var-subset-axioms A-0
        init-is-eq-var.is-equivar
      by auto
    have H1-2: ⋀w g. ⟦g ∈ carrier G; w ∈ A⋆; (δ⋆) i w ∈ F⟧ ⟹ (φ⋆) g w ∈ A⋆
      using H-0
      by auto
    from A-1 have H1-3: w ∈ A⋆
      by auto
    show (δ⋆) i (map (φ g) w) ∈ F
      using give-input-eq-var A-0 A-1 H1-1 H1-3
     apply (clarsimp simp del: GMN-simps simp add: eq-var-func-def eq-var-func-axioms-def
          make-op-def)
      using A-2 H1-0 is-aut.init-state-is-a-state H1-2
      by (smt (verit, best) H1-3 labels-a-G-set.induced-star-map-def restrict-apply)
  qed
  show ?thesis
    apply (clarsimp simp del: GMN-simps simp add: G-lang-def accepted-words-def
G-lang-axioms-def)
    apply (rule conjI)
    using labels-a-G-set.alt-grp-act-axioms
     apply (auto)[1]
    apply (intro conjI)
     apply (simp add: language.intro)
    apply (rule alt-grp-act.eq-var-one-direction)
    using labels-a-G-set.alt-grp-act-axioms labels-a-G-set.lists-a-Gset
      apply blast
     apply (clarsimp )
    apply (clarsimp)
    by (simp add: H-0 H-1 in-listsI)
qed
end

locale reach-det-aut =
  det-aut A S i F δ
  for A :: 'alpha set (structure) and
    S :: 'states set and
    i F δ +
  assumes
    is-reachable:
    s ∈ S ⟹ ∃ input ∈ A⋆. (δ⋆) i input = s

locale reach-det-G-aut =
```

$det\text{-}G\text{-}aut\ A\ S\ i\ F\ \delta\ G\ \varphi\ \psi$ + $reach\text{-}det\text{-}aut\ A\ S\ i\ F\ \delta$
for A :: *'alpha set* (**structure**) **and**
S :: *'states set* **and**
i **and** F **and** δ **and**
G :: (*'grp*, *'b*) *monoid-scheme* **and**
$\varphi\ \psi$
begin

To avoid duplicate variant of "star":

no-adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*
end

sublocale *reach-det-G-aut* ⊆ *reach-det-aut*
using *reach-det-aut-axioms*
by *simp*

locale *G-aut-hom* = *Aut0*: $reach\text{-}det\text{-}G\text{-}aut\ A\ S_0\ i_0\ F_0\ \delta_0\ G\ \varphi\ \psi_0$ +
Aut1: $reach\text{-}det\text{-}G\text{-}aut\ A\ S_1\ i_1\ F_1\ \delta_1\ G\ \varphi\ \psi_1$ +
hom-f: $aut\text{-}hom\ A\ S_0\ i_0\ F_0\ \delta_0\ S_1\ i_1\ F_1\ \delta_1\ f$ +
eq-var-f: $eq\text{-}var\text{-}func\ G\ S_0\ \psi_0\ S_1\ \psi_1\ f$ **for**
A :: *'alpha set* **and**
S_0 :: *'states-0 set* **and**
i_0 **and** F_0 **and** δ_0 **and**
S_1 :: *'states-1 set* **and**
i_1 **and** F_1 **and** δ_1 **and**
G :: (*'grp*, *'b*) *monoid-scheme* **and**
$\varphi\ \psi_0\ \psi_1\ f$

locale *G-aut-epi* = *G-aut-hom* +
assumes
is-epi: f ‘ $S_0 = S_1$

locale *det-aut-rec-lang* = $det\text{-}aut\ A\ S\ i\ F\ \delta$ + $language\ A\ L$
for A :: *'alpha set* (**structure**) **and**
S :: *'states set* **and**
$i\ F\ \delta\ L$ +
assumes
is-recognised:
$w \in L \longleftrightarrow w \in A^\star \wedge ((\delta^\star)\ i\ w) \in F$

locale *det-G-aut-rec-lang* = $det\text{-}G\text{-}aut\ A\ S\ i\ F\ \delta\ G\ \varphi\ \psi$ + $det\text{-}aut\text{-}rec\text{-}lang\ A\ S\ i\ F\ \delta\ L$
for A :: *'alpha set* (**structure**) **and**
S :: *'states set* **and**
$i\ F\ \delta$ **and**
G :: (*'grp*, *'b*) *monoid-scheme* **and**
$\varphi\ \psi\ L$
begin

lemma *lang-is-G-lang*: *G-lang G A φ L*
proof−
  have *H0*: *L* = *accepted-words*
    apply (*simp add*: *accepted-words-def*)
    apply (*subst is-recognised* [*symmetric*])
    by *simp*
  show *G-lang G A φ L*
    apply (*subst H0*)
    apply (*rule det-G-aut.induced-g-lang*[*of A S i F δ G φ ψ*])
    by (*simp add*: *det-G-aut-axioms*)
qed

To avoid ambiguous parse trees:

no-notation *trans-is-eq-var.GA-0.induced-quot-map* (‹[-]-1› *60*)
no-notation *states-a-G-set.induced-quot-map* (‹[-]-1› *60*)

end

locale *reach-det-aut-rec-lang* = *reach-det-aut A S i F δ* + *det-aut-rec-lang A S i F δ L*
  for *A* :: *'alpha set* **and**
    S :: *'states set* **and**
    i F δ **and**
    L :: *'alpha list set*

locale *reach-det-G-aut-rec-lang* = *det-G-aut-rec-lang A S i F δ G φ ψ L* +
  reach-det-G-aut A S i F δ G φ ψ
  for *A* :: *'alpha set* **and**
    S :: *'states set* **and**
    i F δ **and**
    G :: (*'grp*, *'b*) *monoid-scheme* **and**
    φ ψ **and**
    L :: *'alpha list set*

sublocale *reach-det-G-aut-rec-lang* ⊆ *det-G-aut-rec-lang*
  apply (*simp add*: *det-G-aut-rec-lang-def*)
  using *reach-det-G-aut-rec-lang-axioms*
  by (*simp add*: *det-G-aut-axioms det-aut-rec-lang-axioms*)

locale *det-G-aut-recog-G-lang* = *det-G-aut-rec-lang A S i F δ G φ ψ L* + *G-lang G A φ L*
  for *A* :: *'alpha set* (**structure**) **and**
    S :: *'states set* **and**
    i F δ **and**
    G :: (*'grp*, *'b*) *monoid-scheme* **and**
    φ ψ **and**
    L :: *'alpha list set*

sublocale *det-G-aut-rec-lang* ⊆ *det-G-aut-recog-G-lang*
apply (*simp add*: *det-G-aut-recog-G-lang-def*)
apply (*rule conjI*)
apply (*simp add*: *det-G-aut-rec-lang-axioms*)
by (*simp add*: *lang-is-G-lang*)

locale *reach-det-G-aut-rec-G-lang* = *reach-det-G-aut-rec-lang* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ L$
+ *G-lang* $G\ A\ \varphi\ L$
for A :: *'alpha set* (**structure**) **and**
S :: *'states set* **and**
$i\ F\ \delta$ **and**
G :: (*'grp*, *'b*) *monoid-scheme* **and**
$\varphi\ \psi\ L$

sublocale *reach-det-G-aut-rec-lang* ⊆ *reach-det-G-aut-rec-G-lang*
apply (*simp add*: *reach-det-G-aut-rec-G-lang-def*)
apply (*rule conjI*)
apply (*simp add*: *reach-det-G-aut-rec-lang-axioms*)
by (*simp add*: *lang-is-G-lang*)

lemma (**in** *reach-det-G-aut*)
reach-det-G-aut-rec-lang $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi$ *accepted-words*
apply (*clarsimp simp del*: *simp add*: *reach-det-G-aut-rec-lang-def*
det-G-aut-rec-lang-def det-aut-rec-lang-axioms-def)
apply (*intro conjI*)
apply (*simp add*: *det-G-aut-axioms*)
apply (*clarsimp simp add*: *reach-det-G-aut-axioms accepted-words-def reach-det-aut-rec-lang-def*)
apply (*simp add*: *det-aut-rec-lang-def det-aut-rec-lang-axioms.intro is-aut.det-aut-axioms*
language-def)
by (*simp add*: *reach-det-G-aut-axioms*)

lemma (**in** *det-G-aut*) *action-on-input*:
$\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow \psi\ g\ ((\delta^\star)\ i\ w) = (\delta^\star)\ i\ ((\varphi^\star)\ g\ w)$
proof−
fix $g\ w$
assume
A-0: $g \in carrier\ G$ **and**
A-1: $w \in A^\star$
have *H-0*: $(\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w) = (\delta^\star)\ i\ ((\varphi^\star)\ g\ w)$
using *A-0 init-is-eq-var.is-equivar*
by *fastforce*
have *H-1*: $\psi\ g\ ((\delta^\star)\ i\ w) = (\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w)$
using *A-0 A-1 give-input-eq-var*
apply (*clarsimp simp del*: *GMN-simps simp add*: *eq-var-func-axioms-def eq-var-func-def*
make-op-def)
apply (*rule meta-mp*[*of* $((\varphi^\star)\ g\ w) \in A^\star \wedge \psi\ g\ i \in S$])
using *is-aut.init-state-is-a-state A-1*
apply *presburger*
using *det-G-aut-axioms*

apply (*clarsimp simp add*: *det-G-aut-def*)
apply (*rule conjI*; *rule impI*; *rule conjI*)
using *labels-a-G-set.element-image*
apply *fastforce*
using *is-aut.init-state-is-a-state states-a-G-set.element-image*
by *blast+*
show ψ *g* $((\delta^\star)\ i\ w) = (\delta^\star)\ i\ ((\varphi^\star)\ g\ w)$
using *H-0 H-1*
by *simp*
qed

definition (**in** *det-G-aut*)
reachable-states :: *'states set* (‹S_{reach}›)
where $S_{reach} = \{s\ .\ \exists\ w \in A^\star.\ (\delta^\star)\ i\ w = s\}$

definition (**in** *det-G-aut*)
reachable-trans :: *'states* ⇒ *'alpha* ⇒ *'states* (‹δ_{reach}›)
where $\delta_{reach}\ s\ a = (\lambda(s', a') \in S_{reach} \times A.\ \delta\ s'\ a')\ (s, a)$

definition (**in** *det-G-aut*)
reachable-action :: *'grp* ⇒ *'states* ⇒ *'states* (‹ψ_{reach}›)
where $\psi_{reach}\ g\ s = (\lambda(g', s') \in carrier\ G \times S_{reach}.\ \psi\ g'\ s')\ (g, s)$

lemma (**in** *det-G-aut*) *reachable-action-is-restict*:
$\bigwedge g\ s.\ g \in carrier\ G \Longrightarrow s \in S_{reach} \Longrightarrow \psi_{reach}\ g\ s = \psi\ g\ s$
by (*auto simp add*: *reachable-action-def reachable-states-def*)

lemma (**in** *det-G-aut-rec-lang*) *reach-det-aut-is-det-aut-rec-L*:
reach-det-G-aut-rec-lang $A\ S_{reach}\ i\ (F \cap S_{reach})\ \delta_{reach}\ G\ \varphi\ \psi_{reach}\ L$
proof−
have *H-0*: $(\lambda(x, y).\ \delta_{reach}\ x\ y) \in S_{reach} \times A \rightarrow_E S_{reach}$
proof−
have *H1-0*: $(\lambda(x, y).\ \delta\ x\ y) \in extensional\ (S \times A)$
using *is-aut.trans-func-ext*
by (*simp add*: *PiE-iff*)
have *H1-1*: $(\lambda(s', a') \in S_{reach} \times A.\ \delta\ s'\ a') \in extensional\ (S_{reach} \times A)$
using *H1-0*
by *simp*
have *H1-2*: $(\lambda(s', a') \in S_{reach} \times A.\ \delta\ s'\ a') = (\lambda(x, y).\ \delta_{reach}\ x\ y)$
by (*auto simp add*: *reachable-trans-def*)
show $(\lambda(x, y).\ \delta_{reach}\ x\ y) \in S_{reach} \times A \rightarrow_E S_{reach}$
apply (*clarsimp simp add*: *PiE-iff*)
apply (*rule conjI*)
apply (*clarify*)
using *reachable-trans-def*
apply (*simp add*: *reachable-states-def*)[*1*]
apply (*metis Cons-in-lists-iff append-Nil2 append-in-lists-conv is-aut.give-input-closed is-aut.init-state-is-a-state is-aut.trans-to-charact*)
using *H1-1 H1-2*

by *simp*
qed
have *H-1*: $\bigwedge g.\ g \in carrier\ G \Longrightarrow$
($\bigwedge s.\ \psi_{reach}\ g\ s = (if\ s \in S_{reach}\ then\ case\ (g,\ s)\ of\ (x,\ xa) \Rightarrow \psi\ x\ xa\ else$ *undefined*)) $\Longrightarrow$
bij-betw ($\psi_{reach}\ g$) $S_{reach}\ S_{reach}$
proof−
fix *g*
assume
A1-0: $g \in carrier\ G$ **and**
A1-1: ($\bigwedge s.\ \psi_{reach}\ g\ s =$
($if\ s \in S_{reach}$
then case (*g*, *s*) *of* (*x*, *xa*) $\Rightarrow \psi\ x\ xa$
else undefined))
have *H1-0*: $\bigwedge r.\ r \in S_{reach} \Longrightarrow (\psi_{reach}\ g)\ r \in S_{reach}$
using *A1-0*
apply (*clarsimp simp add*: *reachable-states-def reachable-action-def*)
apply (*rule meta-mp*[*of* $\bigwedge w.\ w \in A^\star \Longrightarrow ((\varphi^\star)\ g\ w) \in A^\star$])
using *action-on-input*[**where** $g = g$]
apply (*metis in-listsI*)
by (*metis alt-group-act-is-grp-act group-action.element-image labels-a-G-set.lists-a-Gset*)
have *H1-1*: $\bigwedge f\ T\ U.\ bij\text{-}betw\ f\ T\ T \Longrightarrow f\ \text{`}\ U = U \Longrightarrow U \subseteq T \Longrightarrow bij\text{-}betw$ (*restrict f U*) *U U*
apply (*clarsimp simp add*: *bij-betw-def inj-on-def image-def*)
by (*meson in-mono*)
have *H1-2*: $\psi_{reach}\ g = restrict\ (\psi\ g)\ S_{reach}$
using *reachable-action-def A1-0*
by (*auto simp add*: *restrict-def*)
have *H1-3*: *bij-betw* (ψ *g*) *S S* $\Longrightarrow (\psi_{reach}\ g)\ \text{`}\ S_{reach} = S_{reach}$
$\Longrightarrow S_{reach} \subseteq S \Longrightarrow$ *bij-betw* ($\psi_{reach}\ g$) $S_{reach}\ S_{reach}$
by (*metis H1-2 bij-betw-imp-inj-on inj-on-imp-bij-betw inj-on-restrict-eq inj-on-subset*)
have *H1-4*: $\bigwedge w\ s.\ s = (\delta^\star)\ i\ w \Longrightarrow$
$\forall x \in set\ w.\ x \in A \Longrightarrow$
$\exists x.\ (\exists w \in A^\star.\ (\delta^\star)\ i\ w = x) \wedge (\delta^\star)\ i\ w = \psi_{reach}\ g\ x$
proof−
fix *w s*
assume
A2-0: $\forall x \in set\ w.\ x \in A$ **and**
A2-1: $s = (\delta^\star)\ i\ w$
have *H2-0*: ($inv\ _G\ g) \in carrier\ G$
apply (*rule meta-mp*[*of group G*])
using *A1-0*
apply *simp*
using *det-G-aut-rec-lang-axioms*
by (*auto simp add*: *det-G-aut-rec-lang-def*
det-aut-rec-lang-axioms-def det-G-aut-def group-action-def group-hom-def)
have *H2-1*: $\psi\ (inv\ _G\ g)\ s = (\delta^\star)\ i\ ((\varphi^\star)\ (inv\ _G\ g)\ w)$
apply (*simp del*: *GMN-simps add*: *A2-1*)
apply (*rule action-on-input*[**where** $g = (inv\ _G\ g)$ **and** $w = w$])

using *H2-0 A2-0*
by *auto*
have *H2-2*: $((\varphi^\star)\ (inv\ _G\ g)\ w) \in A^\star$
using *A2-0 H2-0 det-G-aut-rec-lang-axioms*
apply (*clarsimp*)
using *labels-a-G-set.surj-prop list.set-map*
by *fastforce*
have *H2-3*: $\exists\, w{\in}A^\star.\ (\delta^\star)\ i\ w = \psi\ (inv\ _G\ g)\ s$
by (*metis H2-1 H2-2*)
from *H2-3* **have** *H2-4*: $\psi\ (inv\ _G\ g)\ s \in S_{reach}$
by (*simp add*: *reachable-states-def*)
have *H2-5*: $\psi_{reach}\ g\ (\psi\ (inv\ _G\ g)\ s) = \psi\ g\ (\psi\ (inv\ _G\ g)\ s)$
apply (*rule reachable-action-is-restict*)
using *A1-0 H2-4*
by *simp+*
have *H2-6*: $(\delta^\star)\ i\ w = \psi_{reach}\ g\ (\psi\ (inv\ _G\ g)\ s)$
apply (*simp add*: *H2-5 A2-1*)
by (*metis A1-0 A2-0 in-listsI A2-1 H2-5 is-aut.give-input-closed is-aut.init-state-is-a-state states-a-G-set.bij-prop1 states-a-G-set.orbit-sym-aux*)
show $\exists\, x.\ (\exists\, w{\in}A^\star.\ (\delta^\star)\ i\ w = x) \wedge (\delta^\star)\ i\ w = \psi_{reach}\ g\ x$
using *H2-3 H2-6*
by *blast*
qed
show *bij-betw* $(\psi_{reach}\ g)\ S_{reach}\ S_{reach}$
apply (*rule H1-3*)
apply (*simp add*: *A1-0 bij-betw-def states-a-G-set.inj-prop states-a-G-set.surj-prop*)
apply (*clarsimp simp add*: *image-def H1-0*)
apply (*rule subset-antisym*; *simp add*: *Set.subset-eq*; *clarify*)
using *H1-0*
apply *auto*[*1*]
subgoal for *s*
apply (*clarsimp simp add*: *reachable-states-def*)
by (*simp add*: *H1-4*)
apply (*simp add*: *reachable-states-def Set.subset-eq*; *rule allI*; *rule impI*)
using *is-aut.give-input-closed is-aut.init-state-is-a-state*
by *auto*
qed
have *H-2*: *group G*
using *det-G-aut-rec-lang-axioms*
by (*auto simp add*: *det-G-aut-rec-lang-def det-G-aut-def group-action-def group-hom-def*)
have *H-3*: $\bigwedge g.\ g \in carrier\ G \Longrightarrow \psi_{reach}\ g \in carrier\ (BijGroup\ S_{reach})$
subgoal for *g*
using *reachable-action-def*[**where** $g = g$]
apply (*simp add*: *BijGroup-def Bij-def extensional-def*)
by (*simp add*: *H-1*)
done
have *H-4*: $\bigwedge x\, y.\ x \in carrier\ G \Longrightarrow y \in carrier\ G \Longrightarrow \psi_{reach}\ (x \otimes_G y) = \psi_{reach}\ x \otimes_{BijGroup\ S_{reach}}$

ψ_{reach} *y*

proof −
fix *g h*
assume
A1-0: $g \in$ *carrier G* **and**
A1-1: $h \in$ *carrier G*
have *H1-0*: $\bigwedge g$. $g \in$ *carrier G* $\Longrightarrow \psi_{reach}$ *g* = *restrict* (ψ *g*) S_{reach}
using *reachable-action-def*
by (*auto simp add*: *restrict-def*)
from *H1-0* **have** *H1-1*: ψ_{reach} ($g \otimes_G h$) = *restrict* (ψ ($g \otimes_G h$)) S_{reach}
by (*simp add*: *A1-0 A1-1 H-2 group.subgroup-self subgroup.m-closed*)
have *H1-2*: ψ_{reach} $g \otimes_{BijGroup\ S_{reach}} \psi_{reach}$ h =
(*restrict* (ψ *g*) S_{reach}) $\otimes_{BijGroup\ S_{reach}}$
(*restrict* (ψ *h*) S_{reach})
using *A1-0 A1-1 H1-0*
by *simp*
have *H1-3*: $\bigwedge g$. $g \in$ *carrier G* $\Longrightarrow \psi_{reach}$ $g \in$ *carrier* (*BijGroup* S_{reach})
by (*simp add*: *H-3*)
have *H1-4*: $\bigwedge x\ y$. $x \in$*carrier G* $\Longrightarrow$ $y \in$*carrier G* $\Longrightarrow \psi$ ($x \otimes_G y$) = ψ x $\otimes_{BijGroup\ S}$ ψ y
using *det-G-aut-axioms*
by (*simp add*: *det-G-aut-def group-action-def group-hom-def group-hom-axioms-def hom-def*)
hence *H1-5*: ψ ($g \otimes_G h$) = ψ $g \otimes_{BijGroup\ S} \psi$ h
using *A1-0 A1-1*
by *simp*
have *H1-6*: (λx. *if* $x \in S_{reach}$
then if (*if* $x \in S_{reach}$
then ψ *h x*
else undefined) $\in S_{reach}$
then ψ *g* (*if* $x \in S_{reach}$
then ψ *h x*
else undefined)
else undefined
else undefined) =
(λx. *if* $x \in S_{reach}$
then ψ *g* (ψ *h x*)
else undefined)
apply (*rule meta-mp*[*of* $\bigwedge x$. $x \in S_{reach} \Longrightarrow$ (ψ *h x*) $\in S_{reach}$])
using *H1-3*[**where** *g1* = *h*] *A1-1 H1-0*
by (*auto simp add*: *A1-1 BijGroup-def Bij-def bij-betw-def*)
have *H1-7*: ... = (λx. *if* $x \in S_{reach}$
then if $x \in S$
then ψ *g* (ψ *h x*)
else undefined
else undefined)
apply (*clarsimp simp add*: *reachable-states-def*)
by (*metis is-aut.give-input-closed is-aut.init-state-is-a-state*)
have *H1-8*: (*restrict* (ψ *g*) S_{reach}) $\otimes_{BijGroup\ S_{reach}}$ (*restrict* (ψ *h*) S_{reach}) =

restrict (ψ ($g \otimes_G h$)) S_{reach}
apply (*rule meta-mp*[*of* $\bigwedge g.\ g \in carrier\ G \Longrightarrow restrict\ (\psi\ g)\ S_{reach} \in Bij\ S_{reach} \wedge$ $\psi\ g \in Bij\ S$])
apply (*clarsimp simp add*: *H1-5 BijGroup-def*; *intro conjI*; *intro impI*)
subgoal
using *A1-0 A1-1*
apply (*clarsimp simp add*: *compose-def restrict-def*)
by (*simp add*: *H1-6 H1-7*)
apply (*simp add*: *A1-0 A1-1*)+
subgoal for *g*
using *H1-3*[**where** $g1 = g$] *H1-0*[*of g*]
by (*simp add*: *BijGroup-def states-a-G-set.bij-prop0*)
done
show $\psi_{reach}\ (g \otimes_G h) =$
$\psi_{reach}\ g \otimes_{BijGroup\ S_{reach}} \psi_{reach}\ h$
by (*simp add*: *H1-1 H1-2 H1-8*)
qed
have *H-5*: $\bigwedge w'\ w\ g.\ g \in carrier\ G \Longrightarrow$
$(\delta^\star)\ i\ w \in F \Longrightarrow \forall x{\in}set\ w.\ x \in A \Longrightarrow (\delta^\star)\ i\ w' = (\delta^\star)\ i\ w \Longrightarrow \forall x{\in}set\ w'.\ x \in A \Longrightarrow$
$\exists w'{\in}\ A^\star.\ (\delta^\star)\ i\ w' = \psi\ g\ ((\delta^\star)\ i\ w)$
proof −
fix $w'\ w\ g$
assume
A1-0: $g \in carrier\ G$ **and**
A1-1: $(\delta^\star)\ i\ w \in F$ **and**
A1-2: $\forall x{\in}set\ w.\ x \in A$ **and**
A1-3: $(\delta^\star)\ i\ w' = (\delta^\star)\ i\ w$ **and**
A1-4: $\forall x{\in}set\ w.\ x \in A$
from *A1-1 A1-2* **have** *H1-0*: $((\delta^\star)\ i\ w) \in S_{reach}$
using *reachable-states-def*
by *auto*
have *H1-1*: $\psi\ g\ ((\delta^\star)\ i\ w) = ((\delta^\star)\ i\ ((\varphi^\star)\ g\ w))$
using *give-input-eq-var*
apply (*clarsimp simp add*: *eq-var-func-def eq-var-func-axioms-def simp del*: *GMN-simps*)
using *A1-0 A1-2 action-on-input*
by *blast*
have *H1-2*: $(\varphi^\star)\ g\ w \in A^\star$
using *A1-0 A1-2*
by (*metis in-listsI alt-group-act-is-grp-act group-action.element-image labels-a-G-set.lists-a-Gset*)
show $\exists wa{\in}A^\star.\ (\delta^\star)\ i\ wa = \psi\ g\ ((\delta^\star)\ i\ w)$
by (*metis H1-1 H1-2*)
qed
have *H-6*: *alt-grp-act G* $S_{reach}\ \psi_{reach}$
apply (*clarsimp simp add*: *group-action-def group-hom-def group-hom-axioms-def hom-def*)

apply (*intro conjI*)
apply (*simp add*: *H-2*)
subgoal
by (*simp add*: *group-BijGroup*)
apply *clarify*
apply (*simp add*: *H-3*)
by (*simp add*: *H-4*)
have *H-7*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow (\delta^\star)\ i\ w \in F \Longrightarrow \forall x{\in}set\ w.\ x \in A \Longrightarrow$
$\exists x.\ x \in F \wedge (\exists w{\in}A^\star.\ (\delta^\star)\ i\ w = x) \wedge (\delta^\star)\ i\ w = \psi\ g\ x$
proof−
fix *g w*
assume
A1-0: $g \in carrier\ G$ **and**
A1-1: $(\delta^\star)\ i\ w \in F$ **and**
A1-2: $\forall x{\in}set\ w.\ x \in A$
have *H1-0*: $(inv\ _G\ g) \in carrier\ G$
by (*meson A1-0 group.inv-closed group-hom.axioms*(*1*) *labels-a-G-set.group-hom*)
have *H1-1*: $((\delta^\star)\ i\ w) \in S_{reach}$
using *A1-1 A1-2 reachable-states-def*
by *auto*
have *H1-2*: $\psi_{reach}\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w) = \psi\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w)$
apply (*rule reachable-action-is-restict*)
using *H1-0 H1-1*
by *auto*
have *H1-3*: $\psi_{reach}\ g\ (\psi\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w)) = ((\delta^\star)\ i\ w)$
by (*smt* (*verit*) *A1-0 H1-1 H-6 H1-2*
alt-group-act-is-grp-act group-action.bij-prop1 group-action.orbit-sym-aux)
have *H1-4*: $\psi\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w) \in F$
using *A1-1 H1-0 accepting-is-eq-var.is-equivar*
by *blast*
have *H1-5*: $\psi\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w) \in F \wedge (\delta^\star)\ i\ w = \psi\ g\ (\psi\ (inv\ _G\ g)\ ((\delta^\star)\ i$
$w))$
using *H1-4 H1-3 A1-0 A1-1 H1-0 H1-1 reachable-action-is-restict*
by (*metis H-6 alt-group-act-is-grp-act*
group-action.element-image)
have *H1-6*: $\psi\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w) = ((\delta^\star)\ i\ ((\varphi^\star)\ (inv\ _G\ g)\ w))$
using *give-input-eq-var*
apply (*clarsimp simp add*: *eq-var-func-def eq-var-func-axioms-def simp del*:
GMN-simps)
using *A1-2 H1-0 action-on-input*
by *blast*
have *H1-7*: $(\varphi^\star)\ (inv\ _G\ g)\ w \in A^\star$
by (*metis A1-2 in-listsI H1-0 alt-group-act-is-grp-act group-action.element-image*
labels-a-G-set.lists-a-Gset)
thus $\exists x.\ x \in F \wedge (\exists w{\in}A^\star.\ (\delta^\star)\ i\ w = x) \wedge (\delta^\star)\ i\ w = \psi\ g\ x$
using *H1-5 H1-6 H1-7*
by *metis*
qed
have *H-8*: $\bigwedge r\ a\ g\ .\ r \in S_{reach} \Longrightarrow a \in A \Longrightarrow \psi_{reach}\ g\ r \in S_{reach} \wedge \varphi\ g\ a \in$

```
A ⟹ g ∈ carrier G ⟹
        δreach (ψreach g r) (φ g a) = ψreach g (δreach r a)
 proof−
  fix r a g
  assume
   A1-0: r ∈ Sreach and
   A1-1: a ∈ A and
   A1-2: ψreach g r ∈ Sreach ∧ φ g a ∈ A and
   A1-3: g ∈ carrier G
  have H1-0: r ∈ S ∧ ψ g r ∈ S
   apply (rule conjI)
   subgoal
    using A1-0
    apply (clarsimp simp add: reachable-states-def)
    by (simp add: in-listsI is-aut.give-input-closed is-aut.init-state-is-a-state)
   using ‹r ∈ S› A1-3 states-a-G-set.element-image
   by blast
  have H1-1: ⋀a b g . a ∈ S ∧ b ∈ A ⟹ g ∈ carrier G ⟹
    (if ψ g a ∈ S ∧ φ g b ∈ A then δ (ψ g a) (φ g b) else undefined) =
    ψ g (δ a b)
   using det-G-aut-axioms A1-0 A1-1 A1-3
  apply (clarsimp simp add: det-G-aut-def eq-var-func-def eq-var-func-axioms-def)

   by presburger+
  hence H1-2: ψ g (δ r a) = (δ (ψ g r) (φ g a))
   using H1-1[where a1 = r and b1 = a and g1 = g] H1-0 A1-1 A1-2 A1-3
   by simp
  have H1-3: ⋀a w. a ∈ A ⟹ w ∈ A⋆ ⟹ ∃w′∈ A⋆. (δ⋆) i w′ = δ ((δ⋆) i w) a
  proof −
   fix a w
   assume
    A2-0: a ∈ A and
    A2-1: w ∈ A⋆
   have H2-0: (w @ [a]) ∈ A⋆ ∧ (w @ [a]) ∈ A⋆ ⟹ (δ⋆) i (w @ [a]) = δ ((δ⋆)
i w) a
    by (simp add: is-aut.give-input-closed is-aut.trans-to-charact
      is-aut.init-state-is-a-state)
   show ∃w′∈A⋆. (δ⋆) i w′ = δ ((δ⋆) i w) a
    using H2-0
    apply clarsimp
    by (metis A2-0 A2-1 append-in-lists-conv lists.Cons lists.Nil)
  qed
  have H1-4: ψreach g (δreach r a) = ψ g (δ r a)
   apply (clarsimp simp add: reachable-action-def reachable-trans-def)
   using A1-0 A1-1 A1-3 H1-0 H1-3
   using reachable-states-def by fastforce
  have H1-5:ψ g r = ψreach g r
   using A1-0 A1-3
   by (auto simp add: reachable-action-def)
```

hence *H1-6*: $\psi\ g\ r \in S_{reach}$
using *A1-2*
by *simp*
have *H1-7*: $\delta_{reach}\ (\psi_{reach}\ g\ r)\ (\varphi\ g\ a) = \delta\ (\psi\ g\ r)\ (\varphi\ g\ a)$
using *A1-0 A1-1 A1-2 A1-3*
by (*auto simp del*: *simp add*:*reachable-trans-def reachable-action-def*)
show $\delta_{reach}\ (\psi_{reach}\ g\ r)\ (\varphi\ g\ a) = \psi_{reach}\ g\ (\delta_{reach}\ r\ a)$
using *H1-2 H1-4 H1-7*
by *auto*
qed
have *H-9*: $\bigwedge a\ w\ s.\ [\![(\bigwedge s.\ s \in S_{reach} \Longrightarrow (\delta^\star)\ s\ w = (\delta_{reach}{}^\star)\ s\ w);$
$a \in A \wedge (\forall x{\in}set\ w.\ x \in A);\ s \in S_{reach}]\!] \Longrightarrow (\delta^\star)\ (\delta\ s\ a)\ w = (\delta_{reach}{}^\star)$
$(\delta_{reach}\ s\ a)\ w$
proof−
fix *a w s*
assume
A1-IH: $(\bigwedge s.\ s \in S_{reach} \Longrightarrow (\delta^\star)\ s\ w = (\delta_{reach}{}^\star)\ s\ w)$ **and**
A1-0: $a \in A \wedge (\forall x{\in}set\ w.\ x \in A)$ **and**
A1-1: $s \in S_{reach}$
have *H1-0*: $\delta_{reach}\ s\ a = \delta\ s\ a$
using *A1-1*
apply (*clarsimp simp add*: *reachable-trans-def*)
apply (*rule meta-mp*[*of det-aut A S i F* δ])
using *det-aut.trans-func-ext*[**where** *labels* = *A* **and** *states* = *S* **and**
init-state = *i* **and** *fin-states* = *F* **and** *trans-func* = δ]
apply (*simp add*: *extensional-def*)
by (*auto simp add*: *A1-0*)
show $(\delta^\star)\ (\delta\ s\ a)\ w = (\delta_{reach}{}^\star)\ (\delta_{reach}\ s\ a)\ w$
apply (*simp add*: *H1-0*)
apply (*rule A1-IH*[**where** *s1* = δ *s a*])
using *A1-0 A1-1*
apply (*simp add*: *reachable-states-def*)
by (*metis Cons-in-lists-iff append-Nil2 append-in-lists-conv is-aut.give-input-closed*
is-aut.init-state-is-a-state is-aut.trans-to-charact)
qed
show *?thesis*
apply (*clarsimp simp del*: *GMN-simps simp add*: *reach-det-G-aut-rec-lang-def*
det-G-aut-rec-lang-def det-G-aut-def det-aut-def)
apply (*intro conjI*)
subgoal
apply (*simp add*: *reachable-states-def*)
by (*meson give-input.simps*(*1*) *lists.Nil*)
apply (*simp add*: *H-0*)
using *labels-a-G-set.alt-grp-act-axioms*
apply (*auto*)[*1*]
apply (*rule H-6*)
subgoal
apply (*clarsimp simp add*: *eq-var-subset-def eq-var-subset-axioms-def*)
apply (*rule conjI*)

using *H-6*
apply (*auto*)[*1*]
apply (*simp del*: *add*: *reachable-states-def*)[*1*]
apply (*clarify*; *rule subset-antisym*; *simp add*: *Set.subset-eq*; *clarify*)
apply (*rule conjI*)
subgoal for *g* - *w*
apply (*clarsimp simp add*: *reachable-action-def reachable-states-def*)
using *accepting-is-eq-var.is-equivar*
by *blast*
subgoal for *g* - *w*
apply (*clarsimp simp add*: *reachable-action-def reachable-states-def*)
apply (*rule conjI*; *clarify*)
apply (*auto*)[*2*]
by (*simp add*: *H-5*)
apply (*clarsimp simp add*: *reachable-states-def Int-def reachable-action-def*)
apply (*clarsimp simp add*: *image-def*)
by (*simp add*: *H-7*)
subgoal
apply (*clarsimp simp add*: *eq-var-subset-def*)
apply (*rule conjI*)
using *H-6*
apply (*auto*)[*1*]
apply (*clarsimp simp add*: *eq-var-subset-axioms-def*)
apply (*simp add*: ‹$i \in S_{reach}$›)
apply (*simp add*: *reachable-action-def*)
using ‹$i \in S_{reach}$› *init-is-eq-var.is-equivar*
by *fastforce*
subgoal
apply (*clarsimp simp add*: *eq-var-func-def eq-var-func-axioms-def*)
apply (*intro conjI*)
using *H-6 alt-grp-act.axioms*
labels-a-G-set.group-action-axioms prod-group-act labels-a-G-set.alt-grp-act-axioms
apply *blast*
using *H-6*
apply (*auto*)[*1*]
apply (*rule funcsetI*; *clarsimp*)
subgoal for *s a*
apply (*clarsimp simp add*: *reachable-states-def reachable-trans-def*)
by (*metis Cons-in-lists-iff append-Nil2 append-in-lists-conv in-listsI*
is-aut.give-input-closed is-aut.init-state-is-a-state is-aut.trans-to-charact)
apply (*intro allI*; *clarify*; *rule conjI*; *intro impI*)
apply (*simp add*: *H-8*)
using *G-set-equiv H-6 eq-var-subset.is-equivar*
labels-a-G-set.element-image
by *fastforce*
apply (*rule meta-mp*[*of* $\bigwedge w\ s.\ w \in A^\star \Longrightarrow s \in S_{reach} \Longrightarrow (\delta^\star)\ s\ w = (\delta_{reach}{}^\star)$
$s\ w$])
subgoal
using *det-G-aut-rec-lang-axioms*

apply (*clarsimp simp add*: *det-aut-rec-lang-axioms-def det-aut-rec-lang-def det-G-aut-rec-lang-def det-aut-def*)
apply (*intro conjI*)
using ‹$i \in S_{reach}$›
apply *blast*
using *H-0*
apply *blast*
by (*metis* (*mono-tags*, *lifting*) ‹$i \in S_{reach}$› *mem-Collect-eq reachable-states-def*)
subgoal for *w s*
apply (*induction w arbitrary*: *s*)
apply (*clarsimp*)
apply (*simp add*: *in-lists-conv-set*)
by (*simp add*: *H-9*)
apply (*clarsimp simp add*: *reach-det-G-aut-def det-G-aut-def det-aut-def*)
apply (*intro conjI*)
apply (*simp add*: ‹$i \in S_{reach}$›)
apply (*simp add*: *H-0*)
apply (*simp add*: *labels-a-G-set.group-action-axioms*)
using ‹*alt-grp-act G* S_{reach} ψ_{reach}›
apply (*auto*)[*1*]
apply (*simp add*: ‹*eq-var-subset G* S_{reach} ψ_{reach} $(F \cap S_{reach})$›)
apply (*simp add*: ‹*eq-var-subset G* S_{reach} ψ_{reach} $\{i\}$›)
using ‹*eq-var-func G* $(S_{reach} \times A)$ $(\lambda g{\in}carrier\ G.\ \lambda(s, a){\in}S_{reach} \times A.\ (\psi_{reach}\ g\ s, \varphi\ g\ a))$
S_{reach} ψ_{reach} $(\lambda(x, y){\in}S_{reach} \times A.\ \delta_{reach}\ x\ y)$›
apply *blast*
apply (*simp add*: *reach-det-aut-axioms-def reach-det-aut-def reachable-states-def*)
apply (*rule meta-mp*[*of* $\bigwedge s\ input.\ s \in S_{reach} \Longrightarrow input \in A^\star \Longrightarrow (\delta_{reach}{}^\star)\ s\ input = (\delta^\star)\ s\ input$])
using ‹$i \in S_{reach}$›
apply (*metis* (*no-types*, *lifting*) ‹$(\bigwedge w\ s.\ [\![w \in A^\star; s \in S_{reach}]\!] \Longrightarrow (\delta^\star)\ s\ w = (\delta_{reach}{}^\star)\ s\ w) \Longrightarrow$ *det-aut-rec-lang A* S_{reach} i $(F \cap S_{reach})$ δ_{reach} L› *det-aut-rec-lang-def*
reachable-states-def)
by (*simp add*: ‹$\bigwedge w\ s.\ [\![w \in A^\star; s \in S_{reach}]\!] \Longrightarrow (\delta^\star)\ s\ w = (\delta_{reach}{}^\star)\ s\ w$›)
qed

1.4 Syntactic Automaton

context *language* **begin**

definition
rel-MN :: ($'$*alpha list* × $'$*alpha list*) *set* (‹$\equiv_{MN}$›)
where *rel-MN* = $\{(w, w') \in (A^\star)\times(A^\star).\ (\forall v \in A^\star.\ (w\ @\ v) \in L \longleftrightarrow (w'\ @\ v) \in L)\}$

lemma *MN-rel-equival*:
equiv $(A^\star)$ *rel-MN*
by (*auto simp add*: *rel-MN-def equiv-def refl-on-def sym-def trans-def*)

abbreviation
MN-equiv
where *MN-equiv* ≡ $A^\star$ // *rel-MN*

definition
alt-natural-map-MN :: *'alpha list* ⇒ *'alpha list set* (‹[-]$_{MN}$›)
where $[w]_{MN}$ = *rel-MN* `` {*w*}

definition
MN-trans-func :: (*'alpha list set*) ⇒ *'alpha* ⇒ *'alpha list set* (‹δ_{MN}›)
where *MN-trans-func W' a'* =
(λ(*W*,*a*) ∈ *MN-equiv* × *A*. *rel-MN* `` {(*SOME w*. *w* ∈ *W*) @ [*a*]}) (*W'*, *a'*)

abbreviation
MN-init-state
where *MN-init-state* ≡ $[Nil{::}'alpha\ list]_{MN}$

abbreviation
MN-fin-states
where *MN-fin-states* ≡ {*v*. ∃ *w*∈*L*. *v* = $[w]_{MN}$}

lemmas
alt-natural-map-MN-def [*simp*, *GMN-simps*]
MN-trans-func-def [*simp*, *GMN-simps*]
end

context *G-lang* **begin**
adhoc-overloading
star ⇌ *induced-star-map*

lemma *MN-quot-act-wd*:
$w' \in [w]_{MN} \Longrightarrow \forall\, g \in carrier\ G.\ (g \odot_{\varphi^\star} w') \in [g \odot_{\varphi^\star} w]_{MN}$
proof−
assume *A-0*: $w' \in [w]_{MN}$
have *H-0*: ⋀*g*. ⟦(*w*, *w'*) ∈ $\equiv_{MN}$; *g* ∈ *carrier G*; *group-hom G* (*BijGroup A*) φ;
group-hom G (*BijGroup* ($A^\star$)) (λ*g*∈*carrier G*. *restrict* (*map* (φ *g*)) ($A^\star$)); *L* ⊆ $A^\star$;
∀ *g*∈*carrier G*. *map* (φ *g*) ` (*L* ∩ $A^\star$) ∪ (λ*x*. *undefined*) ` (*L* ∩ {*x*. *x* ∉ $A^\star$}) = *L*;
∀ *x*∈*set w*. *x* ∈ *A*; *w'* ∈ $A^\star$⟧ ⟹ (*map* (φ *g*) *w*, *map* (φ *g*) *w'*) ∈ $\equiv_{MN}$
proof−
fix *g*
assume
A1-0: (*w*, *w'*) ∈ $\equiv_{MN}$ **and**
A1-1: *g* ∈ *carrier G* **and**
A1-2: *group-hom G* (*BijGroup A*) φ **and**
A1-3: *group-hom G* (*BijGroup* ($A^\star$)) (λ*g*∈*carrier G*. *restrict* (*map* (φ *g*)) ($A^\star$)) **and**
A1-4: *L* ⊆ $A^\star$ **and**

A1-5: $\forall\, g{\in}carrier\ G.$
$map\ (\varphi\ g)$ ‘ $(L \cap A^\star) \cup (\lambda x.\ undefined)$ ‘ $(L \cap \{x.\ x \notin A^\star\}) = L$ **and**
A1-6: $\forall\, x{\in}set\ w.\ x \in A$ **and**
A1-7: $w' \in A^\star$
have *H1-0*: $\bigwedge v\ w\ w'.\ [\![g \in carrier\ G;\ group\text{-}hom\ G\ (BijGroup\ A)\ \varphi;$
$group\text{-}hom\ G\ (BijGroup\ (A^\star))\ (\lambda g{\in}carrier\ G.\ restrict\ (map\ (\varphi\ g))\ (A^\star));$
$L \subseteq A^\star;\ \forall\, g{\in}carrier\ G.$
$\{y.\ \exists\, x{\in}L \cap A^\star.\ y = map\ (\varphi\ g)\ x\} \cup \{y.\ y = undefined \wedge (\exists\, x.\ x \in L \wedge x \notin A^\star)\} = L;$
$\forall\, x{\in}set\ w.\ x \in A;\ \forall\, v{\in}A^\star.\ (w\ @\ v \in L) = (w'\ @\ v \in L);\ \forall\, x{\in}set\ w'.\ x \in A;\ \forall\, x{\in}set\ v.\ x \in A;$
$map\ (\varphi\ g)\ w\ @\ v \in L]\!] \Longrightarrow map\ (\varphi\ g)\ w'\ @\ v \in L$
proof −
fix $v\ w\ w'$
assume
A2-0: $g \in carrier\ G$ **and**
A2-1: $L \subseteq A^\star$ **and**
A2-2: $group\text{-}hom\ G\ (BijGroup\ A)\ \varphi$ **and**
A2-3: $group\text{-}hom\ G\ (BijGroup\ (A^\star))\ (\lambda g{\in}carrier\ G.\ restrict\ (map\ (\varphi\ g))\ (A^\star))$ **and**
A2-4: $\forall\, g{\in}carrier\ G.\ \{y.\ \exists\, x{\in}L \cap A^\star.\ y = map\ (\varphi\ g)\ x\} \cup$
$\{y.\ y = undefined \wedge (\exists\, x.\ x \in L \wedge x \notin A^\star)\} = L$ **and**
A2-5: $\forall\, x{\in}set\ w.\ x \in A$ **and**
A2-6: $\forall\, x{\in}set\ w'.\ x \in A$ **and**
A2-7: $\forall\, v{\in}A^\star.\ (w\ @\ v \in L) = (w'\ @\ v \in L)$ **and**
A2-8: $\forall\, x{\in}set\ v.\ x \in A$ **and**
A2-9: $map\ (\varphi\ g)\ w\ @\ v \in L$
have *H2-0*: $\forall\, g{\in}carrier\ G.\ \{y.\ \exists\, x{\in}L \cap A^\star.\ y = map\ (\varphi\ g)\ x\} = L$
using *A2-1 A2-4 subset-eq*
by (*smt* (*verit*, *ccfv-SIG*) *Collect-mono-iff sup.orderE*)
hence *H2-1*: $\forall\, g{\in}carrier\ G.\ \{y.\ \exists\, x{\in}L.\ y = map\ (\varphi\ g)\ x\} = L$
using *A2-1 inf.absorb-iff1*
by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong*)
hence *H2-2*: $\forall\, g{\in}carrier\ G.\forall\, x{\in}L.\ map\ (\varphi\ g)\ x \in L$
by *auto*
from *A2-2* **have** *H2-3*: $\forall\, h{\in}carrier\ G.\ \forall\, a{\in}A.\ (\varphi\ h)\ a \in A$
by (*auto simp add*: *group-hom-def BijGroup-def group-hom-axioms-def hom-def Bij-def*
bij-betw-def)
from *A2-8* **have** *H2-4*: $v \in A^\star$
by (*simp add*: *in-listsI*)
hence *H2-5*: $\forall\, h{\in}carrier\ G.\ map\ (\varphi\ h)\ v \in A^\star$
using *H2-3*
by *fastforce*
hence *H2-6*: $\forall\, h{\in}carrier\ G.\ (w\ @\ (map\ (\varphi\ h)\ v) \in L) = (w'\ @\ (map\ (\varphi\ h)\ v) \in L)$
using *A2-7*
by *force*
hence *H2-7*: $(w\ @\ (map\ (\varphi\ (inv_G\ g))\ v) \in L) = (w'\ @\ (map\ (\varphi\ (inv_G\ g))\ v)$

$\in$ *L*)
using *A2-0*
by (*meson A2-7 A2-1 append-in-lists-conv in-mono*)
have (*map* (φ *g*) *w*) $\in$ ($A^\star$)
using *A2-0 A2-2 A2-5 H2-3*
by (*auto simp add*: *group-hom-def group-hom-axioms-def hom-def Bij-Group-def Bij-def*
bij-betw-def)
hence *H2-8*: $\forall\, w \in A^\star.\ \forall\, g \in carrier\ G.\ map\ (\varphi\ (inv_G\ g))\ ((map\ (\varphi\ g)\ w)\ @\ v) =$
$w\ @\ (map\ (\varphi\ (inv_G\ g))\ v)$
using *act-maps-n-distrib triv-act-map A2-0 A2-2 A2-3 H2-4*
apply (*clarsimp*)
by (*smt* (*verit*, *del-insts*) *comp-apply group-action.intro group-action.orbit-sym-aux map-idI*)
have *H2-9*: $map\ (\varphi\ (inv_G\ g))\ ((map\ (\varphi\ g)\ w)\ @\ v) \in L$
using *A2-9 H2-1 H2-2 A2-1*
apply *clarsimp*
by (*metis A2-0 A2-2 group.inv-closed group-hom.axioms*(*1*) *list.map-comp map-append*)
hence *H2-10*: $w\ @\ (map\ (\varphi\ (inv_G\ g))\ v) \in L$
using *H2-8 A2-0*
by (*auto simp add*: *A2-5 in-listsI*)
hence *H2-11*: $w'\ @\ (map\ (\varphi\ (inv_G\ g))\ v) \in L$
using *H2-7*
by *simp*
hence *H2-12*: $map\ (\varphi\ (inv_G\ g))\ ((map\ (\varphi\ g)\ w')\ @\ v) \in L$
using *A2-0 H2-8 A2-1 subsetD*
by (*metis append-in-lists-conv*)
have *H2-13*: $\forall\, g \in carrier\ G.\ restrict\ (map\ (\varphi\ g))\ (A^\star) \in Bij\ (A^\star)$
using *alt-grp-act.lists-a-Gset*[**where** $G = G$ **and** $X = A$ **and** $\varphi = \varphi$] *A1-3*
by (*auto simp add*: *group-action-def*
group-hom-def group-hom-axioms-def Pi-def hom-def BijGroup-def)
have *H2-14*: $\forall\, g \in carrier\ G.\ restrict\ (map\ (\varphi\ g))\ L\ `\ L = L$
using *H2-2*
apply (*clarsimp simp add*: *Set.image-def*)
using *H2-1*
by *blast*
have *H2-15*: $map\ (\varphi\ g)\ w' \in A^\star$
using *A2-0 A2-1 H2-13 H2-2*
by (*metis H2-11 append-in-lists-conv image-eqI lists-image subset-eq surj-prop*)
have *H2-16*: $inv_G\ g \in carrier\ G$
by (*metis A2-0 A2-2 group.inv-closed group-hom.axioms*(*1*))
thus $map\ (\varphi\ g)\ w'\ @\ v \in L$
using *A2-0 A2-1 A2-2 H2-4 H2-12 H2-13 H2-14 H2-15 H2-16 group.inv-closed group-hom.axioms*(*1*)
alt-grp-act.lists-a-Gset[**where** $G = G$ **and** $X = A$ **and** $\varphi = \varphi$]
pre-image-lemma[**where** $S = L$ **and** $T = A^\star$ **and** $f = map\ (\varphi\ (inv_G\ g))$
and

```
        x = ((map (φ g) w′) @ v)]
      apply (clarsimp simp add: group-action-def)
     by (smt (verit, best) A2-1 FuncSet.restrict-restrict H2-14 H2-15 H2-16 H2-4
       append-in-lists-conv inf.absorb-iff2 map-append map-map pre-image-lemma
restrict-apply′
        restrict-apply′)
  qed
  show (map (φ g) w, map (φ g) w′) ∈ ≡MN
    apply (clarsimp simp add: rel-MN-def Set.image-def)
    apply (intro conjI)
    using A1-1 A1-6 group-action.surj-prop group-action-axioms
      apply fastforce
    using A1-1 A1-7 image-iff surj-prop
     apply fastforce
    apply (clarify; rule iffI)
    subgoal for v
      apply (rule H1-0[where v1 =v and w1 = w and w′1 = w′])
      using A1-0 A1-1 A1-2 A1-3 A1-4 A1-5 A1-6 A1-7
      by (auto simp add: rel-MN-def Set.image-def)
    apply (rule H1-0[where w1 = w′ and w′1 = w])
    using A1-0 A1-1 A1-2 A1-3 A1-4 A1-5 A1-6 A1-7
    by (auto simp add: rel-MN-def Set.image-def)
 qed
 show ?thesis
  using G-lang-axioms A-0
  apply (clarsimp simp add: G-lang-def G-lang-axioms-def eq-var-subset-def
     eq-var-subset-axioms-def alt-grp-act-def group-action-def)
  apply (intro conjI; clarify)
   apply (rule conjI; rule impI)
    apply (simp add: H-0)
  by (auto simp add: rel-MN-def)
qed
```

The following lemma corresponds to lemma 3.4 from [1]:

```
lemma MN-rel-eq-var:
 eq-var-rel G (A⋆) (φ⋆) ≡MN
 apply (clarsimp simp add: eq-var-rel-def alt-grp-act-def eq-var-rel-axioms-def)
 apply (intro conjI)
  apply (metis L-is-equivar alt-grp-act.axioms eq-var-subset.axioms(1) induced-star-map-def)
 using L-is-equivar
  apply (simp add: rel-MN-def eq-var-subset-def eq-var-subset-axioms-def)
  apply fastforce
 apply (clarify)
 apply (intro conjI; rule impI; rule conjI; rule impI)
   apply (simp add: in-lists-conv-set)
   apply (clarsimp simp add: rel-MN-def)
   apply (intro conjI)
    apply (clarsimp simp add: rel-MN-def)
 subgoal for w v g w′
```

using *L-is-equivar*
apply (*clarsimp simp add*: *restrict-def eq-var-subset-def eq-var-subset-axioms-def*)
by (*meson element-image*)
apply(*metis image-mono in-listsI in-mono list.set-map lists-mono subset-code*(*1*) *surj-prop*)
apply (*clarify*; *rule iffI*)
subgoal for *w v g u*
using *G-lang-axioms MN-quot-act-wd*[**where** $w = w$ **and** $w' = v$]
by (*auto simp add*: *rel-MN-def G-lang-def G-lang-axioms-def eq-var-subset-def eq-var-subset-axioms-def Set.subset-eq element-image*)
subgoal for *w v g u*
using *G-lang-axioms MN-quot-act-wd*[**where** $w = w$ **and** $w' = v$]
by (*auto simp add*: *rel-MN-def G-lang-def G-lang-axioms-def eq-var-subset-def eq-var-subset-axioms-def Set.subset-eq element-image*)
using *G-lang-axioms MN-quot-act-wd*
by (*auto simp add*: *rel-MN-def G-lang-def G-lang-axioms-def eq-var-subset-def eq-var-subset-axioms-def Set.subset-eq element-image*)

lemma *quot-act-wd-alt-notation*:
$w \in A^\star \Longrightarrow g \in carrier\ G \Longrightarrow g \odot_{[\varphi^\star]_{\equiv MN}\ A^\star} ([w]_{MN}) = ([g \odot_{\varphi^\star} w]_{MN})$
using *eq-var-rel.quot-act-wd*[**where** $G = G$ **and** $\varphi = \varphi^\star$ **and** $X = A^\star$ **and** $R = \equiv_{MN}$ **and** $x = w$ **and** $g = g$]
by (*simp del*: *GMN-simps add*: *alt-natural-map-MN-def MN-rel-eq-var MN-rel-equival*)

lemma *MN-trans-func-characterization*:
$v \in (A^\star) \Longrightarrow a \in A \Longrightarrow \delta_{MN}\ [v]_{MN}\ a = [v\ @\ [a]]_{MN}$
proof−
assume
A-0: $v \in (A^\star)$ **and**
A-1: $a \in A$
have *H-0*: $\bigwedge u.\ u \in [v]_{MN} \Longrightarrow (u\ @\ [a]) \in [v\ @\ [a]]_{MN}$
by (*auto simp add*: *rel-MN-def A-1 A-0*)
hence *H-1*: $(SOME\ w.\ (v, w) \in \equiv_{MN}) \in [v]_{MN} \Longrightarrow ((SOME\ w.\ (v, w) \in \equiv_{MN})\ @\ [a]) \in [v\ @\ [a]]_{MN}$
by *auto*
from *A-0* **have** $(v, v) \in \equiv_{MN} \wedge v \in [v]_{MN}$
by (*auto simp add*: *rel-MN-def*)
hence *H-2*: $(SOME\ w.\ (v, w) \in \equiv_{MN}) \in [v]_{MN}$
apply (*clarsimp simp add*: *rel-MN-def*)
apply (*rule conjI*)
apply (*smt* (*verit*, *ccfv-SIG*) *A-0 in-listsD verit-sko-ex-indirect*)
by (*smt* (*verit*, *del-insts*) *A-0 in-listsI tfl-some*)
hence *H-3*: $((SOME\ w.\ (v, w) \in \equiv_{MN})\ @\ [a]) \in [v\ @\ [a]]_{MN}$
using *H-1*
by *simp*
thus $\delta_{MN}\ [v]_{MN}\ a = [v\ @\ [a]]_{MN}$
using *A-0 A-1 MN-rel-equival*
apply (*clarsimp simp add*: *equiv-def*)

apply (*rule conjI*; *rule impI*)
apply (*metis MN-rel-equival equiv-class-eq*)
by (*simp add*: *A-0 quotientI*)
qed

lemma *MN-trans-eq-var-func* :
eq-var-func G
(*MN-equiv* $\times$ *A*) ($\lambda g \in$*carrier G*. $\lambda(W, a) \in$ (*MN-equiv* $\times$ *A*). (([($\varphi^\star$)]$_{\equiv MN}$ $_{A^\star}$) *g* *W*, φ *g a*))
MN-equiv ([($\varphi^\star$)]$_{\equiv MN}$ $_{A^\star}$)
($\lambda(w, a) \in$ *MN-equiv* $\times$ *A*. δ_{MN} *w a*)
proof−
have *H-0*: *alt-grp-act G MN-equiv* ([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$)
using *MN-rel-eq-var MN-rel-equival eq-var-rel.quot-act-is-grp-act*
alt-group-act-is-grp-act restrict-apply
by *fastforce*
have *H-1*: $\bigwedge$*a b g*.
a $\in$ *MN-equiv* $\Longrightarrow$
b $\in$ *A* $\Longrightarrow$
(([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$) *g a* $\in$ *MN-equiv* $\wedge$ φ *g b* $\in$ *A* $\longrightarrow$
g $\in$ *carrier G* $\longrightarrow$ δ_{MN} (([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$) *g a*) ($\varphi$ *g b*) = ([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$) *g* ($\delta_{MN}$ *a b*)) $\wedge$
((([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$) *g a* $\in$ *MN-equiv* $\longrightarrow$ φ *g b* $\notin$ *A*) $\longrightarrow$
g $\in$ *carrier G* $\longrightarrow$ *undefined* = ([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$) *g* ($\delta_{MN}$ *a b*))
proof −
fix *C a g*
assume
A1-0: *C* $\in$ *MN-equiv* **and**
A1-1: *a* $\in$ *A*
have *H1-0*: *g* $\in$ *carrier G* $\Longrightarrow$ φ *g a* $\in$ *A*
by (*meson A1-1 element-image*)
from *A1-0* **obtain** *c* **where** *H1-c*: $[c]_{MN}$ = *C* $\wedge$ *c* $\in$ $A^\star$
by (*auto simp add*: *quotient-def*)
have *H1-1*: *g* $\in$ *carrier G* $\Longrightarrow$ δ_{MN} (([$\varphi^\star$]$_{\equiv MN}$$_{A^\star}$) *g C*) ($\varphi$ *g a*) = ([$\varphi^\star$]$_{\equiv MN}$ $_{A^\star}$) *g* (δ_{MN} $[c]_{MN}$ *a*)
proof−
assume
A2-0: *g* $\in$ *carrier G*
have *H2-0*: φ *g a* $\in$ *A*
using *H1-0 A2-0*
by *simp*
have *H2-1*: ($\varphi^\star$) *g* $\in$ *Bij* ($A^\star$) **using** *G-lang-axioms lists-a-Gset A2-0*
apply (*clarsimp simp add*: *G-lang-def G-lang-axioms-def group-action-def*
group-hom-def hom-def group-hom-axioms-def BijGroup-def image-def)
by (*meson Pi-iff restrict-Pi-cancel*)
hence *H2-2*: ($\varphi^\star$) *g c* $\in$ ($A^\star$)
using *H1-c*
apply (*clarsimp simp add*: *Bij-def bij-betw-def inj-on-def Image-def image-def*)

apply (*rule conjI*; *rule impI*; *clarify*)
using *surj-prop*
apply *fastforce*
using *A2-0*
by *blast*
from *H1-c* **have** *H2-1*: $([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ (\equiv_{MN}\ \text{``}\ \{c\}) = ([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C$
by *auto*
also have *H2-2*: $([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C = [(\varphi^\star)\ g\ c]_{MN}$
using *eq-var-rel.quot-act-wd*[**where** $R = \equiv_{MN}$ **and** $G = G$ **and** $X = A^\star$ **and** $\varphi = \varphi^\star$ **and** $g = g$
and $x = c$]
by (*clarsimp simp del*: *GMN-simps simp add*: *alt-natural-map-MN-def make-op-def MN-rel-eq-var*
MN-rel-equival H1-c A2-0 H2-1)
hence *H2-3*: $\delta_{MN}\ (([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C)\ (\varphi\ g\ a) = \delta_{MN}\ ([(\varphi^\star)\ g\ c]_{MN})\ (\varphi\ g\ a)$
using *H2-2*
by *simp*
also have *H2-4*: ... $= [((\varphi^\star)\ g\ c)\ @\ [(\varphi\ g\ a)]]_{MN}$
using *MN-trans-func-characterization*[**where** $v = (\varphi^\star)\ g\ c$ **and** $a = \varphi\ g\ a$] *H1-c A2-0*
G-set-equiv H2-0 eq-var-subset.is-equivar insert-iff lists-a-Gset
by *blast*
also have *H2-5*: ... $= [(\varphi^\star)\ g\ (c\ @\ [a])]_{MN}$
using *A2-0 H1-c A1-1*
by *auto*
also have *H2-6*: ... $= ([\varphi^\star]_{\equiv_{MN}\ A^\star})\ g\ [(c\ @\ [a])]_{MN}$
apply (*rule meta-mp*[*of* $c\ @\ [a] \in A^\star$])
using *eq-var-rel.quot-act-wd*[**where** $R = \equiv_{MN}$ **and** $G = G$ **and** $X = A^\star$ **and** $\varphi = \varphi^\star$ **and** $g = g$
and $x = c\ @\ [a]$]
apply (*clarsimp simp del*: *GMN-simps simp add*: *make-op-def MN-rel-eq-var MN-rel-equival H1-c*
A2-0 H2-1)
using *H1-c A1-1*
by *auto*
also have *H2-7*: ... $= ([\varphi^\star]_{\equiv_{MN}\ A^\star})\ g\ (\delta_{MN}\ [c]_{MN}\ a)$
using *MN-trans-func-characterization*[**where** $v = c$ **and** $a = a$] *H1-c A1-1*
by *metis*
finally show $\delta_{MN}\ (([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C)\ (\varphi\ g\ a) = ([\varphi^\star]_{\equiv_{MN}\ A^\star})\ g\ (\delta_{MN}\ [c]_{MN}\ a)$
using *H2-1*
by *metis*
qed
show $(([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C \in$ *MN-equiv* $\wedge\ \varphi\ g\ a \in A \longrightarrow$
$g \in$ *carrier* $G \longrightarrow$
$\delta_{MN}\ (([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C)\ (\varphi\ g\ a) =$
$([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ (\delta_{MN}\ C\ a)) \wedge$
$((([\varphi^\star]_{\equiv_{MN}A^\star})\ g\ C \in$ *MN-equiv* $\longrightarrow \varphi\ g\ a \notin A) \longrightarrow$

$g \in$ *carrier* $G \longrightarrow$ *undefined* $= ([\varphi^\star]_{\equiv_{MN}} A^\star)\ g\ (\delta_{MN}\ C\ a))$
apply (*rule conjI*; *clarify*)
using *H1-1 H1-c*
apply *blast*
by (*metis A1-0 H1-0 H-0 alt-group-act-is-grp-act*
group-action.element-image)
qed
show *?thesis*
apply (*subst eq-var-func-def*)
apply (*subst eq-var-func-axioms-def*)
apply (*rule conjI*)
subgoal
apply (*rule prod-group-act*[**where** $G = G$ **and** $A =$ *MN-equiv* **and** $\varphi = [(\varphi^\star)]_{\equiv_{MN}}\ A^\star$
and $B = A$ **and** $\psi = \varphi$])
apply (*rule H-0*)
using *G-lang-axioms*
by (*auto simp add*: *G-lang-def G-lang-axioms-def*)
apply (*rule conjI*)
subgoal
using *MN-rel-eq-var MN-rel-equival eq-var-rel.quot-act-is-grp-act*
using *alt-group-act-is-grp-act restrict-apply*
by *fastforce*
apply (*rule conjI*)
subgoal
apply (*subst extensional-funcset-def*)
apply (*subst restrict-def*)
apply (*subst Pi-def*)
apply (*subst extensional-def*)
apply (*clarsimp*)
by (*metis MN-rel-equival append-in-lists-conv equiv-Eps-preserves lists.Cons lists.Nil*
quotientI)
apply (*subst restrict-def*)
apply (*clarsimp simp del*: *GMN-simps simp add*: *make-op-def*)
by (*simp add*: *H-1 del*: *GMN-simps*)
qed

lemma *MN-quot-act-on-empty-str*:
$\bigwedge g.\ [\![g \in$ *carrier* G; $([], x) \in \equiv_{MN}]\!] \Longrightarrow x \in$ *map* $(\varphi\ g)$ ‘ $\equiv_{MN}$ “ $\{[]\}$
proof−
fix g
assume
A-0: $g \in$ *carrier* G **and**
A-1: $([], x) \in \equiv_{MN}$
from *A-1* **have** *H-0*: $x \in (A^\star)$
by (*auto simp add*: *rel-MN-def*)
from *A-0 H-0* **have** *H-1*: $x = (\varphi^\star)\ g\ ((\varphi^\star)\ (inv_G\ g)\ x)$
by (*smt* (*verit*) *alt-grp-act-def group-action.bij-prop1 group-action.orbit-sym-aux*

```
lists-a-Gset)
  have H-2: inv G g ∈ carrier G
    using A-0 MN-rel-eq-var
   by (auto simp add: eq-var-rel-def eq-var-rel-axioms-def group-action-def group-hom-def)
  have H-3: ([], (φ⋆) (inv G g) x) ∈ ≡MN
    using A-0 A-1 H-0 MN-rel-eq-var
    apply (clarsimp simp add: eq-var-rel-def eq-var-rel-axioms-def)
    apply (rule conjI; clarify)
     apply (smt (verit, best) H-0 list.simps(8) lists.Nil)
    using H-2
    by simp
  hence H-4: ∃ y∈≡MN `` {[]}. x = map (φ g) y
    using A-0 H-0 H-1 H-2
    apply clarsimp
    by (metis H-0 Image-singleton-iff insert-iff insert-image lists-image surj-prop)
  thus x ∈ map (φ g) ` ≡MN `` {[]}
    by (auto simp add: image-def)
qed

lemma MN-init-state-equivar:
  eq-var-subset G (A⋆) (φ⋆) MN-init-state
  apply (rule alt-grp-act.eq-var-one-direction)
  using lists-a-Gset
    apply (auto)[1]
   apply (clarsimp)
  subgoal for w a
    by (auto simp add: rel-MN-def)
  apply (simp add: Set.subset-eq; clarify)
  apply (clarsimp simp add: image-def Image-def Int-def)
  apply (erule disjE)
  subgoal for g w
    using MN-rel-eq-var
    apply (clarsimp simp add: eq-var-rel-def eq-var-rel-axioms-def)
    by (metis (full-types, opaque-lifting) in-listsI list.simps(8) lists.Nil)
  by (auto simp add: ‹⋀a w. ⟦([], w) ∈ ≡MN; a ∈ set w⟧ ⟹ a ∈ A›)

lemma MN-init-state-equivar-v2:
  eq-var-subset G (MN-equiv) ([φ⋆]≡MN A⋆ ) {MN-init-state}
proof−
  have H-0: ∀ g∈carrier G. (φ⋆) g ` MN-init-state = MN-init-state ⟹
            ∀ g∈carrier G. ([φ⋆]≡MN A⋆) g MN-init-state = MN-init-state
  proof (clarify)
    fix g
    assume
      A-0: g ∈ carrier G
    have H-0: ⋀x. [x]MN = ≡MN `` {x}
      by simp
    have H-1: ([φ⋆]≡MN A⋆) g [[]]MN = [(φ⋆) g []]MN
       using eq-var-rel.quot-act-wd[where R = ≡MN and G = G and X = A⋆
```

and $\varphi = \varphi^\star$ **and** $g = g$
and $x = []$] *MN-rel-eq-var MN-rel-equival*
by (*clarsimp simp del*: *GMN-simps simp add*: *H-0 make-op-def A-0*)
from *A-0 H-1* **show** $([\varphi^\star]_{\equiv MN\,A^\star})\ g\ [[]]_{MN} = [[]]_{MN}$
by *auto*
qed
show *?thesis*
using *MN-init-state-equivar*
apply (*clarsimp simp add*: *eq-var-subset-def simp del*: *GMN-simps*)
apply (*rule conjI*)
subgoal
by (*metis MN-rel-eq-var MN-rel-equival eq-var-rel.quot-act-is-grp-act*)
apply (*clarsimp del*: *subset-antisym simp del*: *GMN-simps simp add*: *eq-var-subset-axioms-def*)
apply (*rule conjI*)
apply (*auto simp add*: *quotient-def*)[*1*]
by (*simp add*: *H-0 del*: *GMN-simps*)
qed

lemma *MN-final-state-equiv*:
eq-var-subset G (*MN-equiv*) $([\varphi^\star]_{\equiv MN\ \ A^\star})$ *MN-fin-states*
proof−
have *H-0*: $\bigwedge g\ x\ w.\ g \in carrier\ G \Longrightarrow w \in L \Longrightarrow \exists\, wa{\in}L.\ ([\varphi^\star]_{\equiv MN\,A^\star})\ g\ [w]_{MN}$
$= [wa]_{MN}$
proof−
fix *g w*
assume
A1-0: $g \in carrier\ G$ **and**
A1-1: $w \in L$
have *H1-0*: $\bigwedge v.\ v \in L \Longrightarrow (\varphi^\star)\ g\ v \in L$
using *A1-0 G-lang-axioms*
apply (*clarsimp simp add*: *G-lang-def G-lang-axioms-def eq-var-subset-def*
eq-var-subset-axioms-def)
by *blast*
hence *H1-1*: $(\varphi^\star)\ g\ w \in L$
using *A1-1*
by *simp*
from *A1-1* **have** *H1-2*: $\bigwedge v.\ v \in [w]_{MN} \Longrightarrow v \in L$
apply (*clarsimp simp add*: *rel-MN-def*)
by (*metis lists.simps self-append-conv*)
have *H1-3*: $([\varphi^\star]_{\equiv MN\,A^\star})\ g\ [w]_{MN} = [(\varphi^\star)\ g\ w]_{MN}$
using *eq-var-rel.quot-act-wd*[**where** $R = \equiv_{MN}$ **and** $G = G$ **and** $X = A^\star$
and $\varphi = \varphi^\star$ **and** $g = g$
and $x = w$] *MN-rel-eq-var MN-rel-equival G-lang-axioms*
by (*clarsimp simp add*: *A1-0 A1-1 G-lang-axioms-def G-lang-def eq-var-subset-def*
eq-var-subset-axioms-def subset-eq)
show $\exists\, wa{\in}L.\ ([\varphi^\star]_{\equiv MN\,A^\star})\ g\ [w]_{MN} = [wa]_{MN}$
using *H1-1 H1-3*
by *blast*
qed

```
  show ?thesis
    apply (rule alt-grp-act.eq-var-one-direction)
    using MN-init-state-equivar-v2 eq-var-subset.axioms(1)
      apply blast
     apply (clarsimp)
    subgoal for w
      using G-lang-axioms
     by (auto simp add: quotient-def G-lang-axioms-def G-lang-def eq-var-subset-def
          eq-var-subset-axioms-def)
    apply (simp add: Set.subset-eq del: GMN-simps; clarify)
    by (simp add: H-0 del: GMN-simps)
qed

interpretation syntac-aut :
  det-aut A MN-equiv MN-init-state MN-fin-states MN-trans-func
proof−
  have H-0: ⋀state label. state ∈ MN-equiv ⟹ label ∈ A ⟹ δ_MN state label ∈
MN-equiv
  proof −
    fix state label
    assume
      A-0: state ∈ MN-equiv and
      A-1: label ∈ A
    obtain w where H-w: state = [w]_MN ∧ w ∈ A⋆
      by (metis A-0 alt-natural-map-MN-def quotientE)
    have H-0: δ_MN [w]_MN label = [w @ [label]]_MN
     using MN-trans-func-characterization[where v = w and a = label] H-w A-1
      by simp
    have H-1: ⋀v. v ∈ A⋆ ⟹ [v]_MN ∈ MN-equiv
      by (simp add: in-listsI quotientI)
    show δ_MN state label ∈ MN-equiv
      using H-w H-0 H-1
      by (simp add: A-1)
  qed
  show det-aut A MN-equiv MN-init-state MN-fin-states δ_MN
   apply (clarsimp simp del: GMN-simps simp add: det-aut-def alt-natural-map-MN-def)
    apply (intro conjI)
      apply (auto simp add: quotient-def)[1]
    using G-lang-axioms
     apply (auto simp add: quotient-def G-lang-axioms-def G-lang-def
        eq-var-subset-def eq-var-subset-axioms-def)[1]
    apply (auto simp add: extensional-def PiE-iff simp del: MN-trans-func-def)[1]
       apply (simp add: H-0 del: GMN-simps)
    by auto
qed

corollary syth-aut-is-det-aut:
  det-aut A MN-equiv MN-init-state MN-fin-states δ_MN
  using local.syntac-aut.det-aut-axioms
```

by *simp*

lemma *give-input-transition-func*:
$w \in (A^\star) \Longrightarrow \forall\, v \in (A^\star).\ [v\ @\ w]_{MN} = (\delta_{MN}{}^\star)\ [v]_{MN}\ w$
proof −
assume
A-0: $w \in A^\star$
have *H-0*: $\bigwedge\ a\ w\ v.\ [\![a \in A;\ w \in A^\star;\ \forall\, v {\in} A^\star.\ [v\ @\ w]_{MN} = (\delta_{MN}{}^\star)\ [v]_{MN}\ w;$
$v \in A^\star]\!] \Longrightarrow$
$[v\ @\ a\ \#\ w]_{MN} = (\delta_{MN}{}^\star)\ [v]_{MN}\ (a\ \#\ w)$
proof −
fix *a w v*
assume
A1-IH: $\forall\, v {\in}\ A^\star.\ [v\ @\ w]_{MN} = (\delta_{MN}{}^\star)\ [v]_{MN}\ w$ **and**
A1-0: $a \in A$ **and**
A1-1: $v \in A^\star$ **and**
A1-2: $w \in A^\star$
from *A1-IH A1-1 A1-2* **have** *H1-1*: $[v\ @\ w]_{MN} = (\delta_{MN}{}^\star)\ [v]_{MN}\ w$
by *auto*
have *H1-2*: $[(v\ @\ [a])\ @\ w]_{MN} = (\delta_{MN}{}^\star)\ [v\ @\ [a]]_{MN}\ w$
apply (*rule meta-mp*[*of* $(v\ @\ [a]) \in (A^\star)$])
using *A1-IH A1-2 H1-1*
apply *blast*
using *A1-0 A1-1*
by *auto*
have *H1-3*: $\delta_{MN}\ [v]_{MN}\ a = [v\ @\ [a]]_{MN}$
using *MN-trans-func-characterization*[**where** $a = a$] *A1-0 A1-1*
by *auto*
hence *H1-4*: $[v\ @\ a\ \#\ w]_{MN} = (\delta_{MN}{}^\star)\ [v\ @\ [a]]_{MN}\ w$
using *H1-2*
by *auto*
also have *H1-5*: $... = (\delta_{MN}{}^\star)\ (\delta_{MN}\ [v]_{MN}\ a)\ w$
using *H1-4 H1-3 A1-1*
by *auto*
thus $[v\ @\ a\ \#\ w]_{MN} = (\delta_{MN}{}^\star)\ [v]_{MN}\ (a\ \#\ w)$
using *calculation*
by *auto*
qed
from *A-0* **show** *?thesis*
apply (*induction w*)
apply (*auto*)[*1*]
by (*simp add*: *H-0 del*: *GMN-simps*)
qed

lemma *MN-unique-init-state*:
$w \in (A^\star) \Longrightarrow [w]_{MN} = (\delta_{MN}{}^\star)\ [Nil]_{MN}\ w$
using *give-input-transition-func*[**where** $w = w$]
by (*metis append-self-conv2 lists.Nil*)

lemma *fin-states-rep-by-lang*:
$w \in A^\star \Longrightarrow [w]_{MN} \in$ *MN-fin-states* $\Longrightarrow w \in L$
proof −
assume
A-0: $w \in A^\star$ **and**
A-1: $[w]_{MN} \in$ *MN-fin-states*
from *A-1* **have** *H-0*: $\exists\, w' \in [w]_{MN}.\ w' \in L$
apply (*clarsimp*)
by (*metis A-0 MN-rel-equival equiv-class-self proj-def proj-in-iff*)
from *H-0* **obtain** w' **where** *H-w'*: $w' \in [w]_{MN} \wedge w' \in L$
by *auto*
have *H-1*: $\bigwedge v.\ v \in A^\star \Longrightarrow w'@v \in L \Longrightarrow w@v \in L$
using *H-w' A-1 A-0*
by (*auto simp add*: *rel-MN-def*)
show $w \in L$
using *H-1 H-w'*
apply *clarify*
by (*metis append-Nil2 lists.Nil*)
qed

The following lemma corresponds to lemma 3.6 from [1]:

lemma *syntactic-aut-det-G-aut*:
det-G-aut A MN-equiv MN-init-state MN-fin-states MN-trans-func G φ $([\varphi^\star]_{\equiv MN}\ A^\star)$
apply (*clarsimp simp add*: *det-G-aut-def simp del*: *GMN-simps*)
apply (*intro conjI*)
using *syth-aut-is-det-aut*
apply (*auto*)[*1*]
using *alt-grp-act-axioms*
apply (*auto*)[*1*]
using *MN-init-state-equivar-v2 eq-var-subset.axioms*(*1*)
apply *blast*
using *MN-final-state-equiv*
apply *presburger*
using *MN-init-state-equivar-v2*
apply *presburger*
using *MN-trans-eq-var-func*
by *linarith*

lemma *syntactic-aut-det-G-aut-rec-L*:
det-G-aut-rec-lang A MN-equiv MN-init-state MN-fin-states MN-trans-func G φ $([\varphi^\star]_{\equiv MN}\ A^\star)$ *L*
apply (*clarsimp simp add*: *det-G-aut-rec-lang-def det-aut-rec-lang-axioms-def det-aut-rec-lang-def simp del*: *GMN-simps*)
apply (*intro conjI*)
using *syntactic-aut-det-G-aut syth-aut-is-det-aut*
apply (*auto*)[*1*]
using *syntactic-aut-det-G-aut syth-aut-is-det-aut*

apply (*auto*)[*1*]
apply (*rule allI*; *rule iffI*)
apply (*rule conjI*)
using *L-is-equivar eq-var-subset.is-subset image-iff image-mono insert-image insert-subset*
apply *blast*
using *MN-unique-init-state L-is-equivar eq-var-subset.is-subset*
apply *blast*
using *MN-unique-init-state fin-states-rep-by-lang in-lists-conv-set*
by (*smt* (*verit*) *mem-Collect-eq*)

lemma *syntact-aut-is-reach-aut-rec-lang*:
reach-det-G-aut-rec-lang A MN-equiv MN-init-state MN-fin-states MN-trans-func G φ
$([\varphi^\star]_{\equiv_{MN}} A^\star)$ *L*
apply (*clarsimp simp del*: *GMN-simps simp add*: *reach-det-G-aut-rec-lang-def det-G-aut-rec-lang-def det-aut-rec-lang-axioms-def reach-det-G-aut-def reach-det-aut-def reach-det-aut-axioms-def det-G-aut-def det-aut-rec-lang-def*)
apply (*intro conjI*)
using *syth-aut-is-det-aut*
apply *blast*
using *alt-grp-act-axioms*
apply (*auto*)[*1*]
subgoal
using *MN-init-state-equivar-v2 eq-var-subset.axioms*(*1*)
by *blast*
using *MN-final-state-equiv*
apply *presburger*
using *MN-init-state-equivar-v2*
subgoal
by *presburger*
using *MN-trans-eq-var-func*
apply *linarith*
using *syth-aut-is-det-aut*
apply (*auto*)[*1*]
apply (*metis* (*mono-tags*, *lifting*) *G-lang.MN-unique-init-state G-lang-axioms det-G-aut-rec-lang-def det-aut-rec-lang.is-recognised syntactic-aut-det-G-aut-rec-L*)
using *syth-aut-is-det-aut*
apply (*auto*)[*1*]
using *alt-grp-act-axioms*
apply (*auto*)[*1*]
using ‹*alt-grp-act G MN-equiv* $([\varphi^\star]_{\equiv_{MN}} A^\star)$›
apply *blast*
using ‹*eq-var-subset G MN-equiv* $([\varphi^\star]_{\equiv_{MN}} A^\star)$ *MN-fin-states*›
apply *blast*
using ‹*eq-var-subset G MN-equiv* $([\varphi^\star]_{\equiv_{MN}} A^\star)$ {*MN-init-state*}›
apply *blast*
using *MN-trans-eq-var-func*
apply *blast*

using *syth-aut-is-det-aut*
apply *auto*[*1*]
by (*metis MN-unique-init-state alt-natural-map-MN-def quotientE*)
end

1.5 Proving the Myhill-Nerode Theorem for G-Automata

context *det-G-aut* **begin**
no-adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*
end

context *reach-det-G-aut-rec-lang* **begin**
adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*

definition
states-to-words :: *'states* ⇒ *'alpha list*
where $states\text{-}to\text{-}words = (\lambda s \in S.\ SOME\ w.\ w \in A^\star \wedge ((\delta^\star)\ i\ w = s))$

definition
words-to-syth-states :: *'alpha list* ⇒ *'alpha list set*
where $words\text{-}to\text{-}syth\text{-}states\ w = [w]_{MN}$

definition
induced-epi:: *'states* ⇒ *'alpha list set*
where *induced-epi* = *compose S words-to-syth-states states-to-words*

lemma *induced-epi-wd1*:
$s \in S \Longrightarrow \exists\, w.\ w \in A^\star \wedge ((\delta^\star)\ i\ w = s)$
using *reach-det-G-aut-rec-lang-axioms is-reachable*
by *auto*

lemma *induced-epi-wd2*:
$w \in A^\star \Longrightarrow w' \in A^\star \Longrightarrow (\delta^\star)\ i\ w = (\delta^\star)\ i\ w' \Longrightarrow [w]_{MN} = [w']_{MN}$
proof−
assume
A-0: $w \in A^\star$ **and**
A-1: $w' \in A^\star$ **and**
A-2: $(\delta^\star)\ i\ w = (\delta^\star)\ i\ w'$
have *H-0*: $\bigwedge v.\ v \in A^\star \Longrightarrow w\ @\ v \in L \longleftrightarrow w'\ @\ v \in L$
apply *clarify*
by (*smt* (*verit*) *A-0 A-1 A-2 append-in-lists-conv is-aut.eq-pres-under-concat is-aut.init-state-is-a-state is-lang is-recognised subsetD*)+
show $[w]_{MN} = [w']_{MN}$
apply (*simp add*: *rel-MN-def*)
using *H-0 A-0 A-1*
by *auto*
qed

lemma *states-to-words-on-final*:
  states-to-words $\in (F \to L)$
proof−
  have *H-0*: $\bigwedge x.\ x \in F \Longrightarrow x \in S \Longrightarrow (SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = x) \in L$
  proof−
    fix *s*
    assume
      A1-0: $s \in F$
    have *H1-0*: $\exists\, w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s$
      using *A1-0 is-reachable*
      by (*metis is-aut.fin-states-are-states subsetD*)
    have *H1-1*: $\bigwedge w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s \Longrightarrow w \in L$
      using *A1-0 is-recognised*
      by *auto*
    show $(SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s) \in L$
      by (*metis* (*mono-tags*, *lifting*) *H1-0 H1-1 someI-ex*)
  qed
  show *?thesis*
    apply (*clarsimp simp add*: *states-to-words-def*)
    apply (*rule conjI*; *rule impI*)
     apply (*simp add*: *H-0*)
    using *is-aut.fin-states-are-states*
    by *blast*
qed

lemma *induced-epi-eq-var*:
  eq-var-func G S ψ *MN-equiv* $([(\varphi^\star)]_{\equiv MN\ \ A^\star})$ *induced-epi*
proof−
  have *H-0*: $\bigwedge\ s\ g.\ [\![s \in S;\ g \in carrier\ G;\ \psi\ g\ s \in S]\!] \Longrightarrow$
    words-to-syth-states (*states-to-words* $(\psi\ g\ s)) =$
    $([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g$ (*words-to-syth-states* (*states-to-words s*))
  proof−
    fix *s g*
    assume
      A1-0: $s \in S$ **and**
      A1-1: $g \in carrier\ G$ **and**
      A1-2: $\psi\ g\ s \in S$
    have *H1-0*: $([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g$ (*words-to-syth-states* (*states-to-words s*)) $=$
     $[(\varphi^\star)\ g\ (SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s)]_{MN}$
      apply (*clarsimp simp del*: *GMN-simps simp add*: *words-to-syth-states-def*
          states-to-words-def A1-0)
      apply (*rule meta-mp*[*of* $(SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s) \in A^\star$])
      using *quot-act-wd-alt-notation*[**where** $w = (SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w =$
$s)$ **and** $g = g$] *A1-1*
       apply *simp*
      using *A1-0*
      by (*metis* (*mono-tags*, *lifting*) *induced-epi-wd1 some-eq-imp*)

have *H1-1*: $\bigwedge g\ s'\ w'.\ [\![s' \in S;\ w' \in A^\star;\ g \in carrier\ G;\ (\varphi^\star)\ g\ w' \in A^\star \wedge \psi\ g\ s' \in S]\!]$
$\Longrightarrow (\delta^\star)\ (\psi\ g\ s')\ ((\varphi^\star)\ g\ w') = \psi\ g\ ((\delta^\star)\ s'\ w')$
using *give-input-eq-var*
apply (*clarsimp simp del*: *GMN-simps simp add*: *eq-var-func-axioms-def eq-var-func-def*
make-op-def)
by (*meson in-listsI*)
have *H1-2*: $\{w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = \psi\ g\ s\} =$
$\{w'.\ \exists\, w \in A^\star.\ (\varphi^\star)\ g\ w = w' \wedge (\delta^\star)\ i\ w = s\}$
proof (*rule subset-antisym*; *clarify*)
fix w'
assume
A2-0: $(\delta^\star)\ i\ w' = \psi\ g\ s$ **and**
A2-1: $\forall\, x{\in}set\ w'.\ x \in A$
have *H2-0*: $(inv\ _G\ g) \in carrier\ G$
by (*meson A1-1 group.inv-closed group-hom.axioms*(*1*) *states-a-G-set.group-hom*)
have *H2-1*: $(\varphi^\star)\ g\ ((\varphi^\star)\ (inv\ _G\ g)\ w') = w'$
by (*smt* (*verit*) *A1-1 A2-1 alt-group-act-is-grp-act group-action.bij-prop1*
group-action.orbit-sym-aux in-listsI labels-a-G-set.lists-a-Gset)
have *H2-2*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow (\delta^\star)\ i\ ((\varphi^\star)\ g\ w) = (\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w)$
using *init-is-eq-var.eq-var-subset-axioms init-is-eq-var.is-equivar*
by *auto*
have *H2-3*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow (\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w) = \psi\ g\ ((\delta^\star)\ i\ w)$
apply (*rule H1-1*[**where** $s'1 = i$])
apply (*simp add*: *A2-1 in-lists-conv-set H2-0 is-aut.init-state-is-a-state*)+
using *is-aut.init-state-is-a-state labels-a-G-set.element-image*
states-a-G-set.element-image
by *blast*
have *H2-4*: $\psi\ (inv\ _G\ g)\ ((\delta^\star)\ i\ w') = s$
using *A2-0 H2-0*
by (*simp add*: *A1-0 A1-1 states-a-G-set.orbit-sym-aux*)
have *H2-5*: $(\delta^\star)\ i\ ((\varphi^\star)\ (inv\ _G\ g)\ w') = s$
apply (*rule meta-mp*[*of* $w' \in A^\star$])
using *H2-0 H2-1 H2-4 A2-1 H2-2 H2-3*
apply *presburger*
using *A2-1*
by *auto*
have *H2-6*: $(\varphi^\star)\ (inv_G\ g)\ w' \in A^\star$
using *H2-0 A2-1*
by (*metis alt-group-act-is-grp-act group-action.element-image in-listsI*
labels-a-G-set.lists-a-Gset)
thus $\exists\, w{\in}A^\star.\ (\varphi^\star)\ g\ w = w' \wedge (\delta^\star)\ i\ w = s$
using *H2-1 H2-5 H2-6*
by *blast*
next
fix $x\ w$

```
      assume
        A2-0: ∀x∈set w. x ∈ A and
        A2-1: s = (δ⋆) i w
      show (φ⋆) g w ∈ A⋆ ∧ (δ⋆) i ((φ⋆) g w) = ψ g ((δ⋆) i w)
        apply (rule conjI)
         apply (rule meta-mp[of (inv G g) ∈ carrier G])
        using alt-group-act-is-grp-act group-action.element-image in-listsI
          labels-a-G-set.lists-a-Gset
          apply (metis A1-1 A2-0)
       apply (meson A1-1 group.inv-closed group-hom.axioms(1) states-a-G-set.group-hom)
        apply (rule meta-mp[of ψ g i = i])
        using H1-1[where s'1 = i and g1 = g]
         apply (metis A1-1 A2-0 action-on-input in-listsI)
        using init-is-eq-var.eq-var-subset-axioms init-is-eq-var.is-equivar
        by (simp add: A1-1)
    qed
    have H1-3: ∃w. w ∈ A⋆ ∧ (δ⋆) i w = s
      using A1-0 is-reachable
      by auto
    have H1-4: ∃w. w ∈ A⋆ ∧ (δ⋆) i w = ψ g s
      using A1-2 induced-epi-wd1
      by auto
    have H1-5: [(φ⋆) g (SOME w. w ∈ A⋆ ∧ (δ⋆) i w = s)]MN = [SOME w. w ∈
A⋆ ∧ (δ⋆) i w = ψ g s]MN
    proof (rule subset-antisym; clarify)
      fix w'
      assume
        A2-0: w' ∈ [(φ⋆) g (SOME w. w ∈ A⋆ ∧ (δ⋆) i w = s)]MN
      have H2-0: ⋀w. w ∈ A⋆ ∧ (δ⋆) i w = s ⟹ w' ∈ [(φ⋆) g w]MN
        using A2-0 H1-3 H1-2 H1-4 induced-epi-wd2 mem-Collect-eq tfl-some
        by (smt (verit, best))
      obtain w'' where H2-w'': w' ∈ [(φ⋆) g w'']MN ∧ w'' ∈ A⋆ ∧ (δ⋆) i w'' = s
        using A2-0 H1-3 tfl-some
        by (metis (mono-tags, lifting))
      from H1-2 H2-w'' have H2-1: (δ⋆) i ((φ⋆) g w'') = ψ g s
        by blast
      have H2-2: ⋀w. w ∈ A⋆ ⟹ (δ⋆) i w = ψ g s ⟹ w' ∈ [w]MN
      proof −
        fix w''
        assume
          A3-0: w'' ∈ A⋆ and
          A3-1: (δ⋆) i w'' = ψ g s
        have H3-0: (inv Gg) ∈ carrier G
       by (metis A1-1 group.inv-closed group-hom.axioms(1) states-a-G-set.group-hom)
        from A3-0 H3-0 have H3-1: (φ⋆) (inv G g) w'' ∈ A⋆
          by (metis alt-grp-act.axioms group-action.element-image
              labels-a-G-set.lists-a-Gset)
        have H3-2: ⋀g w. g ∈ carrier G ⟹ w ∈ A⋆ ⟹ (δ⋆) i ((φ⋆) g w) = (δ⋆)
(ψ g i) ((φ⋆) g w)
```

using *init-is-eq-var.eq-var-subset-axioms init-is-eq-var.is-equivar*
by *auto*
have *H3-3*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow (\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w) = \psi\ g\ ((\delta^\star)\ i\ w)$
apply (*rule H1-1*[**where** $s'1 = i$])
apply (*simp add*: *A3-1 in-lists-conv-set H2-1 is-aut.init-state-is-a-state*)+
using *is-aut.init-state-is-a-state labels-a-G-set.element-image states-a-G-set.element-image*
by *blast*
have *H3-4*: $s = (\delta^\star)\ i\ ((\varphi^\star)\ (inv\ _G\ g)\ w'')$
using *A3-0 A3-1 H3-0 H3-2 H3-3 A1-0 A1-1 states-a-G-set.orbit-sym-aux*
by *auto*
from *H3-4* **show** $w' \in [w'']_{MN}$
by (*metis* (*mono-tags*, *lifting*) *A1-1 G-set-equiv H2-1 H2-w''* ‹$(\delta^\star)\ i\ w'' = \psi\ g\ s$› *A3-0 eq-var-subset.is-equivar image-eqI induced-epi-wd2 labels-a-G-set.lists-a-Gset*)
qed
from *H2-2* **show** $w' \in [SOME\ w.\ w \in A^\star \land (\delta^\star)\ i\ w = \psi\ g\ s]_{MN}$
by (*smt* (*verit*) *H1-4 some-eq-ex*)
next
fix w'
assume
A2-0: $w' \in [SOME\ w.\ w \in A^\star \land (\delta^\star)\ i\ w = \psi\ g\ s]_{MN}$
obtain w'' **where** *H2-w''*: $w' \in [(\varphi^\star)\ g\ w'']_{MN} \land w'' \in A^\star \land (\delta^\star)\ i\ w'' = s$
using *A2-0 H1-3 tfl-some*
by (*smt* (*verit*) *H1-2 mem-Collect-eq*)
from *H1-2 H2-w''* **have** *H2-0*: $(\delta^\star)\ i\ ((\varphi^\star)\ g\ w'') = \psi\ g\ s$
by *blast*
have *H2-1*: $\bigwedge w.\ w \in A^\star \Longrightarrow (\delta^\star)\ i\ w = s \Longrightarrow w' \in [(\varphi^\star)\ g\ w]_{MN}$
proof −
fix w''
assume
A3-0: $w'' \in A^\star$ **and**
A3-1: $(\delta^\star)\ i\ w'' = s$
have *H3-0*: $(inv\ _G g) \in carrier\ G$
by (*metis A1-1 group.inv-closed group-hom.axioms*(*1*) *states-a-G-set.group-hom*)
have *H3-1*: $(\varphi^\star)\ (inv\ _G\ g)\ w'' \in A^\star$
using *A3-0 H3-0*
by (*metis alt-group-act-is-grp-act group-action.element-image labels-a-G-set.lists-a-Gset*)
have *H3-2*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow (\delta^\star)\ i\ ((\varphi^\star)\ g\ w) = (\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w)$
using *init-is-eq-var.eq-var-subset-axioms init-is-eq-var.is-equivar*
by *auto*
have *H3-3*: $\bigwedge g\ w.\ g \in carrier\ G \Longrightarrow w \in A^\star \Longrightarrow (\delta^\star)\ (\psi\ g\ i)\ ((\varphi^\star)\ g\ w) = \psi\ g\ ((\delta^\star)\ i\ w)$
apply (*rule H1-1*[**where** $s'1 = i$])
apply (*simp add*: *A3-1 in-lists-conv-set H2-0 is-aut.init-state-is-a-state*)+
using *is-aut.init-state-is-a-state labels-a-G-set.element-image*

states-a-G-set.element-image
by *blast*
have *H3-4*: ψ (*inv* $_G$ *g*) *s* = ($\delta^\star$) *i* (($\varphi^\star$) (*inv* $_G$ *g*) w'')
using *A3-0 A3-1 H3-0 H3-2 H3-3*
by *auto*
show $w' \in [(\varphi^\star)\ g\ w'']_{MN}$
using *H3-4 H3-1*
by (*smt* (*verit*, *del-insts*) *A1-1 A3-0 A3-1 in-listsI H3-2 H3-3*
‹$\bigwedge$*thesis*. ($\bigwedge w''$. $w' \in [(\varphi^\star)\ g\ w'']_{MN} \wedge w'' \in A^\star \wedge$
($\delta^\star$) *i* w'' = *s* $\Longrightarrow$ *thesis*) $\Longrightarrow$ *thesis*›
alt-group-act-is-grp-act group-action.surj-prop image-eqI induced-epi-wd2
labels-a-G-set.lists-a-Gset)
qed
show $w' \in [(\varphi^\star)\ g\ (SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s)]_{MN}$
using *H2-1 H1-3*
by (*metis* (*mono-tags*, *lifting*) *someI*)
qed
show *words-to-syth-states* (*states-to-words* (ψ *g s*)) =
($[(\varphi^\star)]_{\equiv MN}\ _{A^\star}$) *g* (*words-to-syth-states* (*states-to-words s*))
using *H1-5*
apply (*clarsimp simp del*: *GMN-simps simp add*: *words-to-syth-states-def*
states-to-words-def)
apply (*intro conjI*; *clarify*; *rule conjI*)
using *H1-0*
apply (*auto del*: *subset-antisym simp del*: *GMN-simps simp add*: *words-to-syth-states-def*
states-to-words-def)[*1*]
using *A1-2*
apply *blast*
using *A1-0*
apply *blast*
using *A1-0*
by *blast*
qed
show *?thesis*
apply (*clarsimp del*: *subset-antisym simp del*: *GMN-simps simp add*: *eq-var-func-def*
eq-var-func-axioms-def)
apply (*intro conjI*)
subgoal
using *states-a-G-set.alt-grp-act-axioms*
by *auto*
apply (*metis MN-rel-eq-var MN-rel-equival eq-var-rel.quot-act-is-grp-act*)
apply (*clarsimp simp add*: *FuncSet.extensional-funcset-def Pi-def*)
apply (*rule conjI*)
apply (*clarify*)
subgoal for *s*
using *is-reachable*[**where** *s* = *s*]
apply (*clarsimp simp add*: *induced-epi-def compose-def states-to-words-def*
words-to-syth-states-def)
by (*smt* (*verit*) ‹$s \in S \Longrightarrow \exists\ input \in A^\star.\ (\delta^\star)\ i\ input = s$› *alt-natural-map-MN-def*

lists-eq-set quotientI rel-MN-def singleton-conv someI)
apply (*clarsimp simp del*: *GMN-simps simp add*: *induced-epi-def make-op-def compose-def*)
apply (*clarify*)
apply (*clarsimp simp del*: *GMN-simps simp add*: *induced-epi-def compose-def make-op-def*)
apply (*rule conjI*; *rule impI*)
apply (*simp add*: *H-0*)
using *states-a-G-set.element-image*
by *blast*
qed

The following lemma corresponds to lemma 3.7 from [1]:

lemma *reach-det-G-aut-rec-lang*:
G-aut-epi A S i F δ *MN-equiv MN-init-state MN-fin-states* δ_{MN} $G\ \varphi\ \psi$ $([(\varphi^\star)]_{\equiv MN\ A^\star})$ *induced-epi*
proof−
have *H-0*: $\bigwedge s.\ s \in$ *MN-equiv* $\Longrightarrow \exists$ *input*$\in A^\star$. $(\delta_{MN}{}^\star)$ *MN-init-state input* $= s$
proof−
fix s
assume
A-0: $s \in$ *MN-equiv*
from *A-0* **have** *H-0*: $\exists\, w.\ w \in A^\star \wedge s = [w]_{MN}$
by (*auto simp add*: *quotient-def*)
show $\exists$ *input*$\in A^\star$. $(\delta_{MN}{}^\star)$ *MN-init-state input* $= s$
using *H-0*
by (*metis MN-unique-init-state*)
qed
have *H-1*: $\bigwedge s_0\ a.\ s_0 \in S \Longrightarrow a \in A \Longrightarrow$ *induced-epi* $(\delta\ s_0\ a) = \delta_{MN}$ (*induced-epi* s_0) a
proof−
fix $s_0\ a$
assume
A1-0: $s_0 \in S$ **and**
A1-1: $a \in A$
obtain w **where** *H1-w*: $w \in A^\star \wedge (\delta^\star)\ i\ w = s_0$
using *A1-0 induced-epi-wd1*
by *auto*
have *H1-0*: $[SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s_0]_{MN} = [w]_{MN}$
by (*metis* (*mono-tags*, *lifting*) *H1-w induced-epi-wd2 some-eq-imp*)
have *H1-1*: $(\delta^\star)\ i\ (SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = \delta\ s_0\ a) = (\delta^\star)\ i\ (w\ @\ [a])$
using *A1-0 A1-1 H1-w is-aut.trans-to-charact*[**where** $s = s_0$ **and** $a = a$ **and** $w = w$]
by (*smt* (*verit*, *del-insts*) *induced-epi-wd1 is-aut.trans-func-well-def tfl-some*)
have *H1-2*: $w\ @\ [a] \in A^\star$ **using** *H1-w A1-1* **by** *auto*
have *H1-3*: $[(SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = s_0)\ @\ [a]]_{MN} = [w\ @\ [a]]_{MN}$
by (*metis* (*mono-tags*, *lifting*) *A1-1 H1-0 H1-w MN-trans-func-characterization someI*)
have *H1-4*: ... $= [SOME\ w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = \delta\ s_0\ a]_{MN}$

```
    apply (rule sym)
     apply (rule induced-epi-wd2[where w = SOME w. w ∈ A⋆ ∧ (δ⋆) i w = δ
s0 a
          and w′= w @ [a]])
         apply (metis (mono-tags, lifting) A1-0 A1-1 H1-w some-eq-imp H1-2
is-aut.trans-to-charact)
     apply (rule H1-2)
    using H1-1
    by simp
  show induced-epi (δ s0 a) = δMN (induced-epi s0) a
     apply (clarsimp del: subset-antisym simp del: GMN-simps simp add: in-
duced-epi-def
      words-to-syth-states-def states-to-words-def compose-def is-aut.trans-func-well-def)
    using A1-1 H1-w H1-0 H1-3 H1-4 MN-trans-func-characterization A1-0
     is-aut.trans-func-well-def
    by presburger
qed
have H-2: induced-epi ‘ S = MN-equiv
proof−
  have H1-0: ∀ s ∈ S. ∃ v∈ A⋆. (δ⋆) i v = s ∧ [SOME w. w ∈ A⋆ ∧ (δ⋆) i w =
s]MN = [v]MN
    by (smt (verit) is-reachable tfl-some)
  have H1-1: ⋀v. v∈ A⋆ ⟹ (δ⋆) i v ∈ S
    using is-aut.give-input-closed
    by (auto simp add: is-aut.init-state-is-a-state)
  show ?thesis
   apply (clarsimp simp del: GMN-simps simp add: induced-epi-def words-to-syth-states-def
       states-to-words-def compose-def image-def)
    using H1-0 H1-1
    apply (clarsimp)
    apply (rule subset-antisym; simp del: GMN-simps add: Set.subset-eq)
     apply (metis (no-types, lifting) quotientI)
      by (metis (no-types, lifting) alt-natural-map-MN-def induced-epi-wd2 quo-
tientE)
qed
show ?thesis
  apply (simp del: GMN-simps add: G-aut-epi-def G-aut-epi-axioms-def)
  apply (rule conjI)
  subgoal
    apply (clarsimp simp del: GMN-simps simp add: G-aut-hom-def aut-hom-def
reach-det-G-aut-def
       is-reachable det-G-aut-def reach-det-aut-def reach-det-aut-axioms-def)
    apply (intro conjI)
                 apply (simp add: is-aut.det-aut-axioms)
    using labels-a-G-set.alt-grp-act-axioms
                apply (auto)[1]
    using states-a-G-set.alt-grp-act-axioms
               apply blast
              apply (simp add: accepting-is-eq-var.eq-var-subset-axioms)
```

using *init-is-eq-var.eq-var-subset-axioms*
apply (*auto*)[*1*]
apply (*simp add*: *trans-is-eq-var.eq-var-func-axioms*)
apply (*simp add*: *is-aut.det-aut-axioms*)
using *syth-aut-is-det-aut*
apply *simp*
using *labels-a-G-set.alt-grp-act-axioms*
apply (*auto*)[*1*]
apply (*metis MN-rel-eq-var MN-rel-equival eq-var-rel.quot-act-is-grp-act*)
using *MN-final-state-equiv*
apply *presburger*
using *MN-init-state-equivar-v2*
apply *presburger*
using *MN-trans-eq-var-func*
apply *blast*
using *syth-aut-is-det-aut*
apply *auto*[*1*]
apply (*clarify*)
apply (*simp add*: *H-0 del*: *GMN-simps*)
apply (*simp add*: *is-aut.det-aut-axioms*)
using *syth-aut-is-det-aut*
apply *blast*
apply (*clarsimp del*: *subset-antisym simp del*: *GMN-simps simp add*: *aut-hom-axioms-def FuncSet.extensional-funcset-def Pi-def extensional-def*)[*1*]
apply (*intro conjI*)
apply (*clarify*)
apply (*simp add*: *induced-epi-def*)
apply (*simp add*: *induced-epi-def words-to-syth-states-def states-to-words-def compose-def*)
apply (*rule meta-mp*[*of* $(\delta^\star)$ *i Nil* = *i*])
using *induced-epi-wd2*[**where** w = *Nil*]
apply (*auto simp add*: *is-aut.init-state-is-a-state del*: *subset-antisym*)[*2*]
subgoal for x
apply (*rule quotientI*)
using *is-reachable*[**where** $s = x$] *someI*[**where** $P = \lambda w.\ w \in A^\star \wedge (\delta^\star)\ i\ w = x$]
by *blast*
apply (*auto simp add*: *induced-epi-def words-to-syth-states-def states-to-words-def compose-def*)[*1*]
apply (*simp add*: *induced-epi-def states-to-words-def compose-def is-aut.init-state-is-a-state*)
apply (*metis* (*mono-tags*, *lifting*) ‹$\bigwedge w'$. ⟦[] $\in A^\star$; $w' \in A^\star$; $(\delta^\star)\ i$ [] = $(\delta^\star)\ i\ w'$⟧ $\Longrightarrow$ *MN-init-state* = $[w']_{MN}$› *alt-natural-map-MN-def give-input.simps*(*1*) *lists.Nil some-eq-imp words-to-syth-states-def*)
apply (*clarify*)
subgoal for s
apply (*rule iffI*)
apply (*smt* (*verit*) *Pi-iff compose-eq in-mono induced-epi-def is-aut.fin-states-are-states*

states-to-words-on-final words-to-syth-states-def)
apply (*clarsimp simp del*: *GMN-simps simp add*: *induced-epi-def words-to-syth-states-def states-to-words-def compose-def*)
apply (*rule meta-mp*[*of* (*SOME w. w* $\in A^\star \wedge (\delta^\star)$ *i w* = *s*) $\in L$])
apply (*smt* (*verit*) *induced-epi-wd1 is-recognised someI*)
using *fin-states-rep-by-lang is-reachable mem-Collect-eq*
by (*metis* (*mono-tags*, *lifting*))
apply (*clarsimp simp del*: *GMN-simps*)
apply (*simp add*: *H-1*)
using *induced-epi-eq-var*
by *blast*
by (*simp add*: *H-2*)
qed

end

lemma (**in** *det-G-aut*) *finite-reachable*:
finite (*orbits G S* ψ) $\Longrightarrow$ *finite* (*orbits G* S_{reach} ψ_{reach})
proof−
assume
A-0: *finite* (*orbits G S* ψ)
have *H-0*: $S_{reach} \subseteq S$
apply (*clarsimp simp add*: *reachable-states-def*)
by (*simp add*: *in-listsI is-aut.give-input-closed is-aut.init-state-is-a-state*)
have *H-1*: $\{\{\psi\ g\ x\ |g.\ g \in carrier\ G\}\ |x.\ x \in S_{reach}\} \subseteq$
$\{\{\psi\ g\ x\ |g.\ g \in carrier\ G\}\ |x.\ x \in S\}$
by (*smt* (*verit*, *best*) *Collect-mono-iff H-0 subsetD*)
have *H-2*: $\bigwedge x.\ x \in S_{reach} \Longrightarrow$
$\{\psi\ g\ x\ |g.\ g \in carrier\ G\} = \{\psi_{reach}\ g\ x\ |g.\ g \in carrier\ G\}$
using *reachable-action-is-restict*
by (*metis*)
hence *H-3*: $\{\{\psi\ g\ x\ |g.\ g \in carrier\ G\}\ |x.\ x \in S_{reach}\} =$
$\{\{\psi_{reach}\ g\ x\ |g.\ g \in carrier\ G\}\ |x.\ x \in S_{reach}\}$
by *blast*
show *finite* (*orbits G* S_{reach} ψ_{reach})
using *A-0* **apply** (*clarsimp simp add*: *orbits-def orbit-def*)
using *Finite-Set.finite-subset H-1 H-3*
by *auto*
qed

lemma (**in** *det-G-aut*)
orbs-pos-card: *finite* (*orbits G S* ψ) $\Longrightarrow$ *card* (*orbits G S* ψ) > *0*
apply (*clarsimp simp add*: *card-gt-0-iff orbits-def*)
using *is-aut.init-state-is-a-state*
by *auto*

lemma (**in** *reach-det-G-aut-rec-lang*) *MN-B2T*:
assumes
Fin: *finite* (*orbits G S* ψ)

shows
finite (*orbits G* (*language.MN-equiv A L*) ((([($\varphi^\star$)]$_{\equiv MN}$ $A^\star$)))
proof−
have *H-0*: *finite* {{ψ *g x* |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*}
using *Fin*
by (*auto simp add*: *orbits-def orbit-def*)
have *H-1*: *induced-epi* ‘ *S* = *MN-equiv*
using *reach-det-G-aut-rec-lang*
by (*auto simp del*: *GMN-simps simp add*: *G-aut-epi-def G-aut-epi-axioms-def*)
have *H-2*: ⋀*B f*. *finite B* ⟹ *finite* {*f b*| *b*. *b* ∈ *B*}
by *auto*
have *H-3*: *finite* {{ψ *g x* |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*} ⟹
finite {*induced-epi* ‘ *b* |*b*. *b* ∈ {{ψ *g x* |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*}}
using *H-2*[**where** *f1* = (λ*x*. *induced-epi* ‘ *x*) **and** *B1* = {{ψ *g x* |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*}]
by *auto*
have *H-4*: ⋀*s*. *s* ∈ *S* ⟹ ∃ *b*. {*induced-epi* (ψ *g s*) |*g*. *g* ∈ *carrier G*}
= {*y*. ∃ *x*∈*b*. *y* = *induced-epi x*} ∧ (∃ *x*. *b* = {ψ *g x* |*g*. *g* ∈ *carrier G*}
∧ *x* ∈ *S*)
proof−
fix *s*
assume
A2-0: *s* ∈ *S*
have *H2-0*: {*induced-epi* (ψ *g s*) |*g*. *g* ∈ *carrier G*} = {*y*. ∃ *x* ∈ {ψ *g s* |*g*. *g* ∈ *carrier G*}. *y* =
induced-epi x}
by *blast*
have *H2-1*: (∃ *x*. {ψ *g s* |*g*. *g* ∈ *carrier G*} = {ψ *g x* |*g*. *g* ∈ *carrier G*} ∧ *x* ∈ *S*)
using *A2-0*
by *auto*
show ∃ *b*. {*induced-epi* (ψ *g s*) |*g*. *g* ∈ *carrier G*} =
{*y*. ∃ *x*∈*b*. *y* = *induced-epi x*} ∧ (∃ *x*. *b* = {ψ *g x* |*g*. *g* ∈ *carrier G*} ∧ *x* ∈ *S*)
using *A2-0 H2-0 H2-1*
by *meson*
qed
have *H-5*: {*induced-epi* ‘ *b* |*b*. *b* ∈ {{ψ *g x* |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*}} =
{{*induced-epi* (ψ *g s*) | *g* . *g* ∈ *carrier G*} |*s*. *s* ∈ *S*}
apply (*clarsimp simp add*: *image-def*)
apply (*rule subset-antisym*; *simp add*: *Set.subset-eq*; *clarify*)
apply *auto*[*1*]
apply (*simp*)
by (*simp add*: *H-4*)
from *H-3 H-5* **have** *H-6*: *finite* {{ψ *g x* |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*} ⟹
finite {{*induced-epi* (ψ *g s*) | *g* . *g* ∈ *carrier G*} |*s*. *s* ∈ *S*}
by *metis*
have *H-7*: *finite* {{*induced-epi* (ψ *g x*) |*g*. *g* ∈ *carrier G*} |*x*. *x* ∈ *S*}
apply (*rule H-6*)
by (*simp add*: *H-0*)

have *H-8*: $\bigwedge x.\ x \in S \Longrightarrow \{$*induced-epi* $(\psi\ g\ x)\ |g.\ g \in$ *carrier* $G\} =$
$\{([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g$ (*induced-epi* $x)\ |g.\ g \in$ *carrier* $G\}$
using *induced-epi-eq-var*
apply (*simp del*: *GMN-simps add*: *eq-var-func-def eq-var-func-axioms-def make-op-def*)
by *blast*
hence *H-9*: $\{\{$*induced-epi* $(\psi\ g\ x)\ |g.\ g \in$ *carrier* $G\}\ |x.\ x \in S\} =$
$\{\{([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g$ (*induced-epi* $x)\ |g.\ g \in$ *carrier* $G\}\ |x.\ x \in S\}$
by *blast*
have *H-10*: $\bigwedge f\ g\ X\ B\ C.\ g$ ‘ $B = C \Longrightarrow$
$\{\{f\ x\ (g\ b)|x.\ x{\in}X\}|b.\ b \in B\} = \{\{f\ x\ c|x.\ x \in X\}|c.\ c \in C\}$
by *auto*
have *H-11*: $\{\{([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g$ (*induced-epi* $x)\ |g.\ g \in$ *carrier* $G\}\ |x.\ x \in S\} =$
$\{\{([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g\ W\ |g.\ g \in$ *carrier* $G\}\ |W.\ W \in$ *MN-equiv*$\}$
apply (*rule H-10*[**where** *f2* $= ([(\varphi^\star)]_{\equiv MN\ \ A^\star})$ **and** *X2* = *carrier G* **and** *g2* = *induced-epi*
and *B2* = *S* **and** *C2* = *MN-equiv*])
using *H-1*
by *simp*
have *H-12*: $\{\{([(\varphi^\star)]_{\equiv MN\ \ A^\star})\ g\ W\ |g.\ g \in$ *carrier* $G\}\ |W.\ W \in$ *MN-equiv*$\} =$
orbits G (*language.MN-equiv A L*) $(([(\varphi^\star)]_{\equiv MN\ \ A^\star}))$
by (*auto simp add*: *orbits-def orbit-def*)
show *finite* (*orbits G* (*language.MN-equiv A L*) $(([(\varphi^\star)]_{\equiv MN\ \ A^\star})))$
using *H-9 H-11 H-12 H-7*
by *presburger*
qed

context *det-G-aut-rec-lang* **begin**

To avoid duplicate variant of "star":

no-adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*

end

context *det-G-aut-rec-lang* **begin**
adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*
end

lemma (**in** *det-G-aut-rec-lang*) *MN-prep*:
$\exists S'.\ \exists \delta'.\ \exists F'.\ \exists \psi'.$
(*reach-det-G-aut-rec-lang* $A\ S'\ i\ F'\ \delta'\ G\ \varphi\ \psi'\ L\ \wedge$
(*finite* (*orbits* $G\ S\ \psi) \longrightarrow$ *finite* (*orbits* $G\ S'\ \psi')))$
by (*meson G-lang-axioms finite-reachable reach-det-G-aut-rec-lang.intro reach-det-aut-is-det-aut-rec-L*)

lemma (**in** *det-G-aut-rec-lang*) *MN-fin-orbs-imp-fin-states*:
assumes

Fin: *finite* (*orbits G S* ψ)
shows
finite (*orbits G* (*language.MN-equiv A L*) ((([($\varphi^\star$)]$_{\equiv_{MN}}$ $A^\star$)))
using *MN-prep*
by (*metis assms reach-det-G-aut-rec-lang.MN-B2T*)

The following theorem corresponds to theorem 3.8 from [1], i.e. the Myhill-Nerode theorem for G-automata. The left to right direction (see statement below) of the typical Myhill-Nerode theorem would qantify over types (if some condition holds, then there exists some automaton accepting the language). As it is not possible to qantify over types in this way, the equivalence is spit into two directions. In the left to right direction, the explicit type of the syntactic automaton is used. In the right to left direction some type, 's, is fixed. As the two directions are split, the opertunity was taken to strengthen the right to left direction: We do not assume the given automaton is reachable.

This splitting of the directions will be present in all other Myhill-Nerode theorems that will be proved in this document.

theorem (**in** *G-lang*) *G-Myhill-Nerode* :
assumes
finite (*orbits G A* φ)
shows
G-Myhill-Nerode-LR: *finite* (*orbits G MN-equiv* ([($\varphi^\star$)]$_{\equiv_{MN}}$ $A^\star$)) $\Longrightarrow$
($\exists$ *S F* :: *'alpha list set set*. $\exists$ *i* :: *'alpha list set*. $\exists$ δ. $\exists$ ψ.
reach-det-G-aut-rec-lang A S i F δ *G* φ ψ *L* $\wedge$ *finite* (*orbits G S* ψ)) **and**
G-Myhill-Nerode-RL: ($\exists$ *S F* :: *'s set*. $\exists$ *i* :: *'s*. $\exists$ δ. $\exists$ ψ.
det-G-aut-rec-lang A S i F δ *G* φ ψ *L* $\wedge$ *finite* (*orbits G S* ψ))
$\Longrightarrow$ *finite* (*orbits G MN-equiv* ([($\varphi^\star$)]$_{\equiv_{MN}}$ $A^\star$))
subgoal
using *syntact-aut-is-reach-aut-rec-lang*
by *blast*
by (*metis det-G-aut-rec-lang.MN-fin-orbs-imp-fin-states*)

1.6 Proving the standard Myhill-Nerode Theorem

Any automaton is a G-automaton with respect to the trivial group and action, hence the standard Myhill-Nerode theorem is a special case of the G-Myhill-Nerode theorem.

interpretation *triv-act*:
alt-grp-act singleton-group (*undefined*) *X* ($\lambda x \in$ {*undefined*}. *one* (*BijGroup X*))
apply (*simp add*: *group-action-def group-hom-def group-hom-axioms-def*)
apply (*intro conjI*)
apply (*simp add*: *group-BijGroup*)
using *trivial-hom*
by (*smt* (*verit*) *carrier-singleton-group group.hom-restrict group-BijGroup restrict-apply*
singleton-group)

lemma (**in** *det-aut*) *triv-G-aut*:
fixes *triv-G*
assumes *H-triv-G*: *triv-G* = (*singleton-group* (*undefined*))
shows *det-G-aut labels states init-state fin-states δ*
triv-G (λ*x* ∈ {*undefined*}. *one* (*BijGroup labels*)) (λ*x* ∈ {*undefined*}. *one* (*BijGroup states*))
apply (*simp add*: *det-G-aut-def group-hom-def group-hom-axioms-def eq-var-subset-def eq-var-subset-axioms-def eq-var-func-def eq-var-func-axioms-def*)
apply (*intro conjI*)
apply (*rule det-aut-axioms*)
apply (*simp add*: *assms triv-act.group-action-axioms*)+
using *fin-states-are-states*
apply (*auto*)[*1*]
apply (*clarify*; *rule conjI*; *rule impI*)
apply (*simp add*: *H-triv-G BijGroup-def image-def*)
using *fin-states-are-states*
apply *auto*[*1*]
apply (*simp add*: *H-triv-G BijGroup-def image-def*)
apply (*simp add*: *assms triv-act.group-action-axioms*)
apply (*simp add*: *init-state-is-a-state*)
apply (*clarify*; *rule conjI*; *rule impI*)
apply (*simp add*: *H-triv-G BijGroup-def image-def init-state-is-a-state*)+
apply (*clarsimp simp add*: *group-action-def BijGroup-def hom-def group-hom-def group-hom-axioms-def*)
apply (*rule conjI*)
apply (*smt* (*verit*) *BijGroup-def Bij-imp-funcset Id-compose SigmaE case-prod-conv group-BijGroup id-Bij restrict-ext restrict-extensional*)
apply (*rule meta-mp*[*of undefined* $\otimes_{singleton\text{-}group}$ *undefined undefined* = *undefined*])
apply (*auto*)[*1*]
apply (*metis carrier-singleton-group comm-groupE*(*1*) *singletonD singletonI singleton-abelian-group*)
apply (*simp add*: *assms triv-act.group-action-axioms*)
apply (*auto simp add*: *trans-func-well-def*)[*1*]
by (*clarsimp simp add*: *BijGroup-def trans-func-well-def H-triv-G*)

lemma *triv-orbits*:
orbits (*singleton-group* (*undefined*)) *S* (λ*x* ∈ {*undefined*}. *one* (*BijGroup S*)) = {{*s*} |*s*. *s* ∈ *S*}
apply (*simp add*: *BijGroup-def singleton-group-def orbits-def orbit-def*)
by *auto*

lemma *fin-triv-orbs*:
finite (*orbits* (*singleton-group* (*undefined*)) *S* (λ*x* ∈ {*undefined*}. *one* (*BijGroup S*))) = *finite S*
apply (*subst triv-orbits*)
apply (*rule meta-mp*[*of bij-betw* (λ*s* ∈ *S*. {*s*}) *S* {{*s*} |*s*. *s* ∈ *S*}])
using *bij-betw-finite*

```
  apply (auto)[1]
 by (auto simp add: bij-betw-def image-def)

context language begin

interpretation triv-G-lang:
  G-lang singleton-group (undefined) A (λx ∈ {undefined}. one (BijGroup A)) L
 apply (simp add: G-lang-def G-lang-axioms-def eq-var-subset-def eq-var-subset-axioms-def)
  apply (intro conjI)
      apply (simp add: triv-act.group-action-axioms)
     apply (simp add: language-axioms)
  using triv-act.lists-a-Gset
    apply fastforce
   apply (rule is-lang)
  apply (clarsimp simp add: BijGroup-def image-def)
  apply (rule subset-antisym; simp add: Set.subset-eq; clarify)
  using is-lang
   apply (auto simp add: map-idI)[1]
  using is-lang map-idI
  by (metis in-listsD in-mono inf.absorb-iff1 restrict-apply)

definition triv-G :: 'grp monoid
  where triv-G = (singleton-group (undefined))

definition triv-act :: 'grp ⇒ 'alpha ⇒ 'alpha
  where triv-act = (λx ∈ {undefined}. 1_BijGroup A)

corollary standard-Myhill-Nerode:
  assumes
    H-fin-alph: finite A
  shows
    standard-Myhill-Nerode-LR: finite MN-equiv ⟹
     (∃ S F :: 'alpha list set set. ∃ i :: 'alpha list set. ∃ δ.
     reach-det-aut-rec-lang A S i F δ L ∧ finite S) and
    standard-Myhill-Nerode-RL: (∃ S F :: 's set. ∃ i :: 's. ∃ δ.
     det-aut-rec-lang A S i F δ L ∧ finite S) ⟹ finite MN-equiv
proof−
  assume
    A-0: finite MN-equiv
  have H-0: reach-det-aut-rec-lang A MN-equiv MN-init-state MN-fin-states δ_MN
L
    using triv-G-lang.syntact-aut-is-reach-aut-rec-lang
    apply (clarsimp simp add: reach-det-G-aut-rec-lang-def det-G-aut-rec-lang-def
      reach-det-aut-rec-lang-def reach-det-aut-def reach-det-aut-axioms-def det-G-aut-def)
   by (smt (verit) alt-natural-map-MN-def quotientE triv-G-lang.MN-unique-init-state)
  show ∃ S F:: 'alpha list set set. ∃ i :: 'alpha list set. ∃ δ.
  reach-det-aut-rec-lang A S i F δ L ∧ finite S
    using A-0 H-0
    by auto
```

next
  assume
    A-0: ∃ *S F*:: ′*s set*. ∃ *i* :: ′*s*. ∃ δ. *det-aut-rec-lang A S i F* δ *L* ∧ *finite S*
  obtain *S F* :: ′*s set* **and** *i* :: ′*s* **and** δ
    where *H-MN*: *det-aut-rec-lang A S i F* δ *L* ∧ *finite S*
    using *A-0*
    by *auto*
  have *H-0*: *det-G-aut A S i F* δ *triv-G* (λ*x*∈{*undefined*}. $\mathbf{1}_{BijGroup\ A}$)
   (λ*x*∈{*undefined*}. $\mathbf{1}_{BijGroup\ S}$)
    apply (*rule det-aut.triv-G-aut*[*of A S i F* δ *triv-G*])
    using *H-MN*
     apply (*simp add*: *det-aut-rec-lang-def*)
    by (*rule triv-G-def*)
  have *H-1*: *det-G-aut-rec-lang A S i F* δ *triv-G* (λ*x*∈{*undefined*}. $\mathbf{1}_{BijGroup\ A}$)
   (λ*x*∈{*undefined*}. $\mathbf{1}_{BijGroup\ S}$) *L*
    by (*auto simp add*: *det-G-aut-rec-lang-def H-0 H-MN*)
  have *H-2*: (∃ *S F*:: ′*s set*. ∃ *i* :: ′*s*. ∃ δ ψ.
        det-G-aut-rec-lang A S i F δ (*singleton-group undefined*) (λ*x*∈{*undefined*}.
$\mathbf{1}_{BijGroup\ A}$)
        ψ *L* ∧ *finite* (*orbits* (*singleton-group undefined*) *S* ψ))
    using *H-1*
    by (*metis H-MN fin-triv-orbs triv-G-def*)
  have *H-3*: *finite* (*orbits triv-G A triv-act*)
    apply (*subst triv-G-def*; *subst triv-act-def*; *subst fin-triv-orbs*[*of A*])
    by (*rule H-fin-alph*)
  have *H-4*:*finite* (*orbits triv-G MN-equiv* (*triv-act.induced-quot-map* ($A^\star$)
    (*triv-act.induced-star-map A triv-act*) $\equiv_{MN}$))
    using *H-3*
    apply (*simp add*: *triv-G-def triv-act-def del*: *GMN-simps*)
    using *triv-G-lang.G-Myhill-Nerode H-2*
    by *blast*
 have *H-5*:*triv-act.induced-star-map A triv-act* = (λ*x* ∈ {*undefined*}. $\mathbf{1}_{BijGroup\ (A^\star)}$)
    apply (*simp add*: *BijGroup-def restrict-def fun-eq-iff triv-act-def*)
    by (*clarsimp simp add*: *list.map-ident-strong*)
  have *H-6*: (*triv-act.induced-quot-map* ($A^\star$) (*triv-act.induced-star-map A*
  triv-act) $\equiv_{MN}$) = (λ*x* ∈ {*undefined*}. $\mathbf{1}_{BijGroup\ MN\text{-}equiv}$)
    apply (*subst H-5*)
    apply (*simp add*: *BijGroup-def fun-eq-iff Image-def*)
    apply (*rule allI*; *rule conjI*; *intro impI*)
   apply (*smt* (*verit*) *Collect-cong Collect-mem-eq Eps-cong MN-rel-equival equiv-Eps-in*
      in-quotient-imp-closed quotient-eq-iff)
    using *MN-rel-equival equiv-Eps-preserves*
    by *auto*
  show *finite MN-equiv*
    apply (*subst fin-triv-orbs* [*symmetric*]; *subst H-6* [*symmetric*]; *subst triv-G-def*
[*symmetric*])
    by (*rule H-4*)
qed
end

2 Myhill-Nerode Theorem for Nominal G-Automata

2.1 Data Symmetries, Supports and Nominal Actions

The following locale corresponds to the definition 2.2 from [1]. Note that we let G be an arbitrary group instead of a subgroup of `BijGroup` D, but assume there is a homomoprhism $\pi : G \to$ `BijGroup`D. By `group_hom.img_is_subgroup` this is an equivalent definition:

locale *data-symm* = *group-action* G D π
for
G :: ('*grp*, '*b*) *monoid-scheme* **and**
D :: '*D* *set* (‹$\mathbb{D}$›) **and**
π

The following locales corresponds to definition 4.3 from [1]:

locale *supports* = *data-symm* G D π + *alt-grp-act* G X φ
for
G :: ('*grp*, '*b*) *monoid-scheme* **and**
D :: '*D* *set* (‹$\mathbb{D}$›) **and**
π **and**
X :: '*X* *set* (**structure**) **and**
φ +
fixes
C :: '*D* *set* **and**
x :: '*X*
assumes
is-in-set:
$x \in X$ **and**
is-subset:
$C \subseteq \mathbb{D}$ **and**
supports:
$g \in$ *carrier* $G \Longrightarrow (\forall c.\ c \in C \longrightarrow \pi\ g\ c = c) \Longrightarrow g \odot_\varphi x = x$
begin

The following lemma corresponds to lemma 4.9 from [1]:

lemma *image-supports*:
$\bigwedge g.\ g \in$ *carrier* $G \Longrightarrow$ *supports* $G\ D\ \pi\ X\ \varphi\ (\pi\ g$ ‘ $C)\ (g \odot_\varphi x)$
proof −
fix g
assume
A-0: $g \in$ *carrier* G
have *H-0*: $\bigwedge h.$ *data-symm* $G\ \mathbb{D}\ \pi \Longrightarrow$
group-action $G\ X\ \varphi \Longrightarrow$
$x \in X \Longrightarrow$
$C \subseteq \mathbb{D} \Longrightarrow$
$\forall g.\ g \in$ *carrier* $G \longrightarrow (\forall c.\ c \in C \longrightarrow \pi\ g\ c = c) \longrightarrow \varphi\ g\ x = x \Longrightarrow$
$h \in$ *carrier* $G \Longrightarrow \forall c.\ c \in \pi\ g$ ‘ $C \longrightarrow \pi\ h\ c = c \Longrightarrow$
$\varphi\ h\ (\varphi\ g\ x) = \varphi\ g\ x$
proof −

fix *h*
assume
A1-0: *data-symm G* $\mathbb{D}$ π **and**
A1-1: *group-action G X* φ **and**
A1-2: $\forall$ *g. g* $\in$ *carrier G* $\longrightarrow$ ($\forall$ *c. c* $\in$ *C* $\longrightarrow$ π *g c* = *c*) $\longrightarrow$ φ *g x* = *x* **and**
A1-3: *h* $\in$ *carrier G* **and**
A1-4: $\forall$ *c. c* $\in$ π *g* ' *C* $\longrightarrow$ π *h c* = *c*
have *H1-0*: $\bigwedge$*g. g* $\in$ *carrier G* $\Longrightarrow$ ($\forall$ *c. c* $\in$ *C* $\longrightarrow$ π *g c* = *c*) $\Longrightarrow$ φ *g x* = *x*
using *A1-2*
by *auto*
have *H1-1*: $\forall$ *c. c* $\in$ *C* $\longrightarrow$ π ((*inv*$_G$ *g*) $\otimes_G$ *h* $\otimes_G$ *g*) *c* = *c* $\Longrightarrow$
φ ((*inv*$_G$ *g*) $\otimes_G$ *h* $\otimes_G$ *g*) *x* = *x*
apply (*rule H1-0*[*of* ((*inv*$_G$ *g*) $\otimes_G$ *h* $\otimes_G$ *g*)])
apply (*meson A-0 A1-3 group.subgroupE*(*3*) *group.subgroup-self group-hom group-hom.axioms*(*1*)
subgroup.m-closed)
by *simp*
have *H2*: π (((*inv*$_G$ *g*) $\otimes_G$ *h*) $\otimes_G$ *g*) = *compose* $\mathbb{D}$ (π ((*inv*$_G$ *g*) $\otimes_G$ *h*)) (π *g*)
using *A1-0*
apply (*clarsimp simp add*: *data-symm-def group-action-def BijGroup-def group-hom-def*
group-hom-axioms-def hom-def restrict-def)
apply (*rule meta-mp*[*of* π *g* $\in$ *Bij* $\mathbb{D}$ $\wedge$ π ((*inv*$_G$ *g*) $\otimes_G$ *h*) $\in$ *Bij* $\mathbb{D}$])
apply (*smt* (*verit*) *A-0 A1-3 data-symm.axioms data-symm-axioms group.inv-closed*
group.surj-const-mult group-action.bij-prop0 image-eqI)
apply (*rule conjI*)
using *A-0*
apply *blast*
by (*meson A-0 A1-3 data-symm.axioms data-symm-axioms group.subgroupE*(*3*) *group.subgroupE*(*4*)
group.subgroup-self group-action.bij-prop0)
also have *H1-3*: ... = *compose* $\mathbb{D}$ (*compose* $\mathbb{D}$ (π (*inv*$_G$ *g*)) (π *h*)) (π *g*)
using *A1-0*
apply (*clarsimp simp add*: *data-symm-def group-action-def BijGroup-def comp-def*
group-hom-def group-hom-axioms-def hom-def restrict-def)
apply (*rule meta-mp*[*of* π (*inv*$_G$ *g*) $\in$ *Bij* $\mathbb{D}$ $\wedge$ π *h* $\in$ *Bij* $\mathbb{D}$])
apply (*simp add*: *A-0 A1-3*)
apply (*rule conjI*)
apply (*simp add*: *A-0 Pi-iff*)
using *A1-3*
by *blast*
also have *H1-4*: ... = *compose* $\mathbb{D}$ ((π (*inv*$_G$ *g*)) $\circ$ (π *h*)) (π *g*)
using *A1-0*
apply (*clarsimp simp add*: *data-symm-def group-action-def BijGroup-def comp-def group-hom-def*
group-hom-axioms-def hom-def restrict-def compose-def)
using *A-0 A1-3*
by (*meson data-symm.axioms data-symm-axioms group.inv-closed group-action.element-image*)

also have *H1-5*: ... = $(\lambda d \in \mathbb{D}.\ ((\pi\ (inv_G\ g)) \circ (\pi\ h) \circ (\pi\ g))\ d)$
by (*simp add*: *compose-def*)
have *H1-6*: $\bigwedge c.\ c \in C \Longrightarrow ((\pi\ h) \circ (\pi\ g))\ c = (\pi\ g)\ c$
using *A1-4*
by *auto*
have *H1-7*: $\bigwedge c.\ c \in C \Longrightarrow ((\pi\ (inv_G\ g)) \circ (\pi\ h) \circ (\pi\ g))\ c = c$
using *H1-6 A1-0*
apply (*simp add*: *data-symm-def group-action-def BijGroup-def compose-def group-hom-def*
group-hom-axioms-def hom-def)
by (*meson A-0 data-symm.axioms data-symm-axioms group-action.orbit-sym-aux is-subset subsetD*)
have *H1-8*: $\forall c.\ c \in C \longrightarrow \pi\ ((inv_G\ g) \otimes_G h \otimes_G g)\ c = c$
using *H1-7 H1-5*
by (*metis calculation is-subset restrict-apply′ subsetD*)
have *H1-9*: $\varphi\ ((inv_G\ g) \otimes_G h \otimes_G g)\ x = x$
using *H1-8*
by (*simp add*: *H1-1*)
hence *H1-10*: $\varphi\ ((inv_G\ g) \otimes_G h \otimes_G g)\ x = \varphi\ ((inv_G\ g) \otimes_G (h \otimes_G g))\ x$
by (*smt* (*verit*, *ccfv-SIG*) *A-0 A1-3 group.inv-closed group.subgroupE*(*4*) *group.subgroup-self*
group-action.composition-rule group-action.element-image group-action-axioms group-hom
group-hom.axioms(*1*) *is-in-set*)
have *H1-11*: ... = $((\varphi\ (inv_G\ g)) \circ (\varphi\ (h \otimes_G g)))\ x$
using *A-0 A1-3 group.subgroupE*(*4*) *group.subgroup-self group-action.composition-rule*
group-action-axioms group-hom group-hom.axioms(*1*) *is-in-set*
by *fastforce*
have *H1-12*: ... = $((the\text{-}inv\text{-}into\ X\ (\varphi\ g)) \circ (\varphi\ (h \otimes_G g)))\ x$
using *A1-1*
apply (*simp add*: *group-action-def*)
by (*smt* (*verit*) *A-0 A1-3 group.inv-closed group.subgroupE*(*4*) *group.subgroup-self*
group-action.element-image group-action.inj-prop group-action.orbit-sym-aux
group-action-axioms group-hom.axioms(*1*) *is-in-set the-inv-into-f-f*)
have *H1-13*: $((the\text{-}inv\text{-}into\ X\ (\varphi\ g)) \circ (\varphi\ (h \otimes_G g)))\ x = x$
using *H1-9 H1-10 H1-11 H1-12*
by *auto*
hence *H1-14*: $(\varphi\ (h \otimes_G g))\ x = \varphi\ g\ x$
using *H1-13*
by (*metis A-0 A1-3 comp-apply composition-rule element-image f-the-inv-into-f inj-prop is-in-set*
surj-prop)
show $\varphi\ h\ (\varphi\ g\ x) = \varphi\ g\ x$
using *A1-3 A1-2 A-0 H1-14 composition-rule*
by (*simp add*: *is-in-set*)
qed
show *supports* $G\ D\ \pi\ X\ \varphi\ (\pi\ g\ `\ C)\ (g \odot_\varphi x)$
using *supports-axioms*
apply (*clarsimp simp add*: *supports-def supports-axioms-def*)

apply (*intro conjI*)
using *element-image is-in-set A-0*
apply *blast*
apply (*metis A-0 data-symm-def group-action.surj-prop image-mono is-subset*)
apply (*rule allI*; *intro impI*)
apply (*rename-tac h*)
by (*simp add*: *H-0*)
qed
end

locale *nominal* = *data-symm G D π* + *alt-grp-act G X φ*
for
G :: (*'grp*, *'b*) *monoid-scheme* **and**
D :: *'D set* (‹𝔻›) **and**
π **and**
X :: *'X set* (**structure**) **and**
φ +
assumes
is-nominal:
$\bigwedge g\ x.\ g \in carrier\ G \Longrightarrow x \in X \Longrightarrow \exists C.\ C \subseteq \mathbb{D} \wedge finite\ C \wedge supports\ G\ \mathbb{D}\ \pi\ X\ \varphi\ C\ x$

locale *nominal-det-G-aut* = *det-G-aut* +
nominal G D π A φ + *nominal G D π S ψ*
for
D :: *'D set* (‹𝔻›) **and**
π

The following lemma corresponds to lemma 4.8 from [1]:

lemma (**in** *eq-var-func*) *supp-el-pres*:
$supports\ G\ D\ \pi\ X\ \varphi\ C\ x \Longrightarrow supports\ G\ D\ \pi\ Y\ \psi\ C\ (f\ x)$
apply (*clarsimp simp add*: *supports-def supports-axioms-def*)
apply (*rule conjI*)
using *eq-var-func-axioms*
apply (*simp add*: *eq-var-func-def eq-var-func-axioms-def*)
apply (*intro conjI*)
using *is-ext-func-bet*
apply *blast*
apply *clarify*
by (*metis is-eq-var-func'*)

lemma (**in** *nominal*) *support-union-lem*:
fixes *f sup-C col*
assumes *H-f*: $f = (\lambda x.\ (SOME\ C.\ C \subseteq \mathbb{D} \wedge finite\ C \wedge supports\ G\ \mathbb{D}\ \pi\ X\ \varphi\ C\ x))$
and *H-col*: $col \subseteq X \wedge finite\ col$
and *H-sup-C*: $sup\text{-}C = \bigcup\{Cx.\ Cx \in f\ `\ col\}$
shows $\bigwedge x.\ x \in col \Longrightarrow sup\text{-}C \subseteq \mathbb{D} \wedge finite\ sup\text{-}C \wedge supports\ G\ \mathbb{D}\ \pi\ X\ \varphi\ sup\text{-}C\ x$

proof −
  fix *x*
  assume *A-0*: $x \in col$
  have *H-0*: $\bigwedge x.\ x \in X \Longrightarrow \exists\, C.\ C \subseteq \mathbb{D} \wedge$ *finite* $C \wedge$ *supports* $G\ \mathbb{D}\ \pi\ X\ \varphi\ C\ x$
    using *nominal-axioms*
    apply (*clarsimp simp add*: *nominal-def nominal-axioms-def*)
    using *stabilizer-one-closed stabilizer-subset*
    by *blast*
  have *H-1*: $\bigwedge x.\ x \in col \Longrightarrow f\,x \subseteq \mathbb{D} \wedge$ *finite* $(f\,x) \wedge$ *supports* $G\ \mathbb{D}\ \pi\ X\ \varphi\ (f\,x)\ x$
    apply (*subst H-f*)
    using *someI-ex H-col H-f H-0*
    by (*metis* (*no-types*, *lifting*) *in-mono*)
  have *H-2*: *sup-C* $\subseteq \mathbb{D}$
    using *H-1*
    by (*simp add*: *H-sup-C UN-least*)
  have *H-3*: *finite sup-C*
    using *H-1 H-col H-sup-C*
    by *simp*
  have *H-4*: $f\,x \subseteq$ *sup-C*
    using *H-1 H-sup-C A-0*
    by *blast*
  have *H-5*: $\bigwedge g\ c.\ [\![g \in$ *carrier* G; $(c \in$ *sup-C* $\longrightarrow \pi\ g\ c = c)$; $c \in (f\,x)]\!] \Longrightarrow \pi\ g$
$c = c$
    using *H-4 H-1 A-0*
    by (*auto simp add*: *image-def supports-def supports-axioms-def*)
  have *H-6*: *supports* $G\ \mathbb{D}\ \pi\ X\ \varphi$ *sup-C* x
    apply (*simp add*: *supports-def supports-axioms-def*)
    apply (*intro conjI*)
        apply (*simp add*: *data-symm-axioms*)
    using *A-0 H-1 supports.axioms*(*2*)
       apply *fastforce*
    using *H-col A-0*
      apply *blast*
     apply (*rule H-2*)
    apply (*clarify*)
    using *supports-axioms-def*[*of* $G\ D\ \pi\ X\ \varphi$ *sup-C*]
    apply (*clarsimp*)
    using *H-1 A-0*
    apply (*clarsimp simp add*: *supports-def supports-axioms-def*)
    using *A-0 H-5*
    by *presburger*
  show *sup-C* $\subseteq \mathbb{D} \wedge$ *finite sup-C* $\wedge$ *supports* $G\ \mathbb{D}\ \pi\ X\ \varphi$ *sup-C* x
    using *H-2 H-3 H-6* **by** *auto*
qed

lemma (**in** *nominal*) *set-of-list-nom*:
  nominal $G\ D\ \pi\ (X^\star)\ (\varphi^\star)$
proof−
  have *H-0*: $\bigwedge g\ x.\ g \in$ *carrier* $G \Longrightarrow \forall\, x{\in}set\ x.\ x \in X \Longrightarrow$

$\exists\, C \subseteq \mathbb{D}.$ *finite* $C \wedge$ *supports* $G\ \mathbb{D}\ \pi\ (X^\star)\ (\varphi^\star)\ C\ x$
proof −
fix *g w*
assume
A1-0: $g \in$ *carrier* G **and**
A1-1: $\forall\, x \in$ *set* $w.\ x \in X$
have *H1-0*: $\bigwedge x.\ x \in X \Longrightarrow \exists\, C \subseteq \mathbb{D}.$ *finite* $C \wedge$ *supports* $G\ \mathbb{D}\ \pi\ X\ \varphi\ C\ x$
using *A1-0 is-nominal* **by** *force*
define *f* **where** *H1-f*: $f = (\lambda x.\ (SOME\ C.\ C \subseteq \mathbb{D} \wedge$ *finite* $C \wedge$ *supports* G $\mathbb{D}\ \pi\ X\ \varphi\ C\ x))$
define *sup-C* :: *'D set* **where** *H1-sup-C*: $sup\text{-}C = \bigcup \{Cx.\ Cx \in f\ `\ set\ w\}$
have *H1-1*: $\bigwedge x.\ x \in set\ w \Longrightarrow sup\text{-}C \subseteq \mathbb{D} \wedge$ *finite sup-C* $\wedge$ *supports* $G\ \mathbb{D}\ \pi\ X$ φ *sup-C x*
apply (*rule support-union-lem*[**where** $f = f$ **and** $col = set\ w$])
apply (*rule H1-f*)
using *A1-0*
apply (*simp add*: *A1-1 subset-code*(*1*))
apply (*rule H1-sup-C*)
by *simp*
have *H1-2*: *supports* $G\ \mathbb{D}\ \pi\ (X^\star)\ (\varphi^\star)$ *sup-C w*
apply (*clarsimp simp add*: *supports-def supports-axioms-def simp del*: *GMN-simps*)
apply (*intro conjI*)
apply (*simp add*: *data-symm-axioms*)
using *lists-a-Gset*
apply (*auto*)[*1*]
apply (*simp add*: *A1-1 in-listsI*)
using *H1-1 H1-sup-C*
apply *blast*
apply (*rule allI*; *intro impI*)
apply *clarsimp*
apply (*rule conjI*)
using *H1-1*
by (*auto simp add*: *supports-def supports-axioms-def map-idI*)
show $\exists\, C \subseteq \mathbb{D}.$ *finite* $C \wedge$ *supports* $G\ \mathbb{D}\ \pi\ (X^\star)\ (\varphi^\star)\ C\ w$
using *nominal-axioms-def*
apply (*clarsimp simp add*: *nominal-def simp del*: *GMN-simps*)
using *H1-1 H1-2*
by (*metis Collect-empty-eq H1-sup-C Union-empty empty-iff image-empty infinite-imp-nonempty*
subset-empty subset-emptyI supports.is-subset)
qed
show *?thesis*
apply (*clarsimp simp add*: *nominal-def nominal-axioms-def simp del*: *GMN-simps*)
apply (*intro conjI*)
using *group.subgroupE*(*2*) *group.subgroup-self group-hom group-hom.axioms*(*1*)
apply (*simp add*: *data-symm-axioms*)
apply (*rule lists-a-Gset*)
apply (*clarify*)
by (*simp add*: *H-0 del*: *GMN-simps*)

qed

2.2 Proving the Myhill-Nerode Theorem for Nominal G-Automata

context *det-G-aut* **begin**
adhoc-overloading
  star ⇌ *labels-a-G-set.induced-star-map*
end

lemma (**in** *det-G-aut*) *input-to-init-eqvar*:
  eq-var-func G $(A^\star)$ $(\varphi^\star)$ S ψ $(\lambda w{\in}A^\star.\ (\delta^\star)\ i\ w)$
proof−
  have *H-0*: $\bigwedge a\ g.\ [\![\forall x{\in}set\ a.\ x \in A;\ map\ (\varphi\ g)\ a \in A^\star;\ g \in carrier\ G]\!] \Longrightarrow$
      $(\delta^\star)\ i\ (map\ (\varphi\ g)\ a) = \psi\ g\ ((\delta^\star)\ i\ a)$
  proof−
    fix *w g*
    assume
      A1-0: $\forall x{\in}set\ w.\ x \in A$ **and**
      A1-1: $map\ (\varphi\ g)\ w \in A^\star$ **and**
      A1-2: $g \in carrier\ G$
    have *H1-0*: $(\delta^\star)\ (\psi\ g\ i)\ (map\ (\varphi\ g)\ w) = \psi\ g\ ((\delta^\star)\ i\ w)$
      using *give-input-eq-var*
      apply (*clarsimp simp add*: *eq-var-func-axioms-def eq-var-func-def*)
     using *A1-0 A1-1 A1-2 in-listsI is-aut.init-state-is-a-state states-a-G-set.element-image*

      by (*smt* (*verit*, *del-insts*))
    have *H1-1*: $(\psi\ g\ i) = i$
      using *A1-2 is-aut.init-state-is-a-state init-is-eq-var.is-equivar*
      by *force*
    show $(\delta^\star)\ i\ (map\ (\varphi\ g)\ w) = \psi\ g\ ((\delta^\star)\ i\ w)$
      using *H1-0 H1-1*
      by *simp*
  qed
  show *?thesis*
    apply (*clarsimp simp add*: *eq-var-func-def eq-var-func-axioms-def*)
    apply (*intro conjI*)
    using *labels-a-G-set.lists-a-Gset*
       apply *fastforce*
      apply (*simp add*: *states-a-G-set.group-action-axioms del*: *GMN-simps*)
     apply (*simp add*: *in-listsI is-aut.give-input-closed is-aut.init-state-is-a-state*)
    apply *clarify*
    apply (*rule conjI*; *intro impI*)
     apply (*simp add*: *H-0*)
    using *labels-a-G-set.surj-prop*
    by *fastforce*
qed

lemma (**in** *reach-det-G-aut*) *input-to-init-surj*:
  $(\lambda w{\in}A^\star.\ (\delta^\star)\ i\ w)$ ‘ $(A^\star) = S$

using *reach-det-G-aut-axioms*
apply (*clarsimp simp add*: *image-def reach-det-G-aut-def reach-det-aut-def reach-det-aut-axioms-def*)
using *is-aut.give-input-closed is-aut.init-state-is-a-state*
by *blast*

context *reach-det-G-aut* **begin**
adhoc-overloading
star ⇌ *labels-a-G-set.induced-star-map*
end

The following lemma corresponds to proposition 5.1 from [1]:

proposition (**in** *reach-det-G-aut*) *alpha-nom-imp-states-nom*:
$nominal\ G\ D\ \pi\ A\ \varphi \Longrightarrow nominal\ G\ D\ \pi\ S\ \psi$
proof−
assume
A-0: $nominal\ G\ D\ \pi\ A\ \varphi$
have *H-0*: $\bigwedge g\ x.\ [\![g \in carrier\ G;\ data\text{-}symm\ G\ D\ \pi;\ group\text{-}action\ G\ A\ \varphi;\ \forall x.\ x \in A \longrightarrow (\exists C{\subseteq}D.\ finite\ C \wedge supports\ G\ D\ \pi\ A\ \varphi\ C\ x);\ x \in S]\!] \Longrightarrow \exists C{\subseteq}D.\ finite\ C \wedge supports\ G\ D\ \pi\ S\ \psi\ C\ x$
proof −
fix *g s*
assume
A1-0: $g \in carrier\ G$ **and**
A1-1: $data\text{-}symm\ G\ D\ \pi$ **and**
A1-2: $group\text{-}action\ G\ A\ \varphi$ **and**
A1-3: $\forall x.\ x \in A \longrightarrow (\exists C{\subseteq}D.\ finite\ C \wedge supports\ G\ D\ \pi\ A\ \varphi\ C\ x)$ **and**
A1-4: $s \in S$
have *H1-0*: $\bigwedge x.\ x \in (A^\star) \Longrightarrow \exists C{\subseteq}D.\ finite\ C \wedge supports\ G\ D\ \pi\ (A^\star)\ (\varphi^\star)\ C\ x$
using *nominal.set-of-list-nom*[*of* $G\ D\ \pi\ A\ \varphi$] *A1-2*
apply (*clarsimp simp add*: *nominal-def*)
by (*metis A1-0 A1-1 A1-3 in-listsI labels-a-G-set.induced-star-map-def nominal-axioms-def*)
define *f* **where** *H1-f*: $f = (\lambda w{\in}A^\star.\ (\delta^\star)\ i\ w)$
obtain *w* **where** *H1-w0*: $s = f\ w$ **and** *H1-w1*: $w \in (A^\star)$
using *input-to-init-surj A1-4*
apply (*simp add*: *H1-f image-def*)
by (*metis is-reachable*)
obtain *C* **where** *H1-C*: $finite\ C \wedge supports\ G\ D\ \pi\ (A^\star)\ (labels\text{-}a\text{-}G\text{-}set.induced\text{-}star\text{-}map\ \varphi)\ C\ w$
by (*meson H1-0 H1-w0 H1-w1*)
have *H1-2*: $supports\ G\ D\ \pi\ S\ \psi\ C\ s$
apply (*subst H1-w0*)
apply (*rule eq-var-func.supp-el-pres*[**where** $X = A^\star$ **and** $\varphi = \varphi^\star$])
apply (*subst H1-f*)
apply (*rule det-G-aut.input-to-init-eqvar*[*of* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi$])
using *reach-det-G-aut-axioms*
apply (*simp add*: *reach-det-G-aut-def*)

using *H1-C*
by *simp*
show $\exists\, C \subseteq D.$ *finite* $C \wedge$ *supports* $G\ D\ \pi\ S\ \psi\ C\ s$
using *H1-2 H1-C*
by (*meson supports.is-subset*)
qed
show *?thesis*
apply (*rule meta-mp*[*of* ($\exists\, g.\ g \in$ *carrier* G)])
subgoal
using *A-0* **apply** (*clarsimp simp add*: *nominal-def nominal-axioms-def*)
apply (*rule conjI*)
subgoal for *g*
by (*clarsimp simp add*: *states-a-G-set.group-action-axioms*)
apply *clarify*
by (*simp add*: *H-0*)
by (*metis bot.extremum-unique ex-in-conv is-aut.init-state-is-a-state states-a-G-set.stabilizer-one-closed states-a-G-set.stabilizer-subset*)
qed

The following theorem corresponds to theorem 5.2 from [1]:

theorem (**in** *G-lang*) *Nom-G-Myhill-Nerode*:
assumes
orb-fin: *finite* (*orbits* $G\ A\ \varphi$) **and**
nom: *nominal* $G\ D\ \pi\ A\ \varphi$
shows
Nom-G-Myhill-Nerode-LR: *finite* (*orbits* G *MN-equiv* $([(\varphi^\star)]_{\equiv MN}\ \ A^\star)) \Longrightarrow$
($\exists\, S\ F$ :: *'alpha list set set*. $\exists\, i$ :: *'alpha list set*. $\exists\,\delta.\ \exists\,\psi.$
reach-det-G-aut-rec-lang $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ L \wedge$ *finite* (*orbits* $G\ S\ \psi$)
$\wedge$ *nominal-det-G-aut* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ D\ \pi$) **and**
Nom-G-Myhill-Nerode-RL: ($\exists\, S\ F$ :: *'s set*. $\exists\, i$ :: *'s*. $\exists\,\delta.\ \exists\,\psi.$
det-G-aut-rec-lang $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ L \wedge$ *finite* (*orbits* $G\ S\ \psi$)
$\wedge$ *nominal-det-G-aut* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ D\ \pi$)
$\Longrightarrow$ *finite* (*orbits* G *MN-equiv* $([(\varphi^\star)]_{\equiv MN}\ \ A^\star)$)
proof −
assume
A-0: *finite* (*orbits* G *MN-equiv* $([\varphi^\star]_{\equiv MN} A^\star)$)
obtain $S\ F$ :: *'alpha list set set* **and** i :: *'alpha list set* **and** $\delta\ \psi$
where *H-MN*: *reach-det-G-aut-rec-lang* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ L \wedge$ *finite* (*orbits* G
$S\ \psi$)
using *A-0 orb-fin G-Myhill-Nerode-LR*
by *blast*
have *H-0*: *nominal* $G\ D\ \pi\ S\ \psi$
using *H-MN*
apply (*clarsimp simp del*: *GMN-simps simp add*: *reach-det-G-aut-rec-lang-def*)
using *nom reach-det-G-aut.alpha-nom-imp-states-nom*
by *metis*
show $\exists\, S\ F$ :: *'alpha list set set*. $\exists\, i$ :: *'alpha list set*. $\exists\,\delta.\ \exists\,\psi.$
reach-det-G-aut-rec-lang $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ L \wedge$
finite (*orbits* $G\ S\ \psi$) $\wedge$ *nominal-det-G-aut* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ D\ \pi$

apply (*simp add*: *nominal-det-G-aut-def reach-det-G-aut-rec-lang-def*)
using *nom H-MN H-0*
apply (*clarsimp simp add*: *reach-det-G-aut-rec-lang-def reach-det-G-aut-def reach-det-aut-axioms-def*)
by *blast*
next
assume *A0*: $\exists S\ F\ i\ \delta\ \psi.$ *det-G-aut-rec-lang* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ L \wedge$ *finite* (*orbits* $G\ S\ \psi$)
$\wedge$ *nominal-det-G-aut* $A\ S\ i\ F\ \delta\ G\ \varphi\ \psi\ D\ \pi$
show *finite* (*orbits* G *MN-equiv* $([\varphi^\star]_{\equiv MN\, A^\star})$)
using *A0 orb-fin*
by (*meson G-Myhill-Nerode-RL*)
qed
end

References

[1] M. Bojańczyk, B. Klin, and S. Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10, 2014.