CPSVerification

By Jonathan

April 20, 2018

Contents

1 Differential KAD

theory *VC-diffKAD*
imports
Main
afpModified/VC-KAD
Ordinary-Differential-Equations.IVP/Initial-Value-Problem

begin

— Notation.
no-notation *Archimedean-Field.ceiling* ($\lceil$-$\rceil$)
no-notation *Archimedean-Field.floor* ($\lfloor$-$\rfloor$)
no-notation *Set.image* (‘)
no-notation *Range-Semiring.antirange-semiring-class.ars-r* (r)

notation *p2r* ($\lceil$-$\rceil$)
notation *r2p* ($\lfloor$-$\rfloor$)
notation *Set.image*(-$[\![$-$]\!]$)
notation *Product-Type.prod.fst* (π_1)
notation *Product-Type.prod.snd* (π_2)
notation *rel-ad* (${\Delta^c}_1$)

— Preliminary lemmas and definitions.

thm *p2r-def*
thm *r2p-def*
thm *rel-ad-def*
term *Set.Collect*
term *Domain R*
thm *fbox-def*
thm *rel-antidomain-kleene-algebra.fbox-def*

lemma *rel-ad-proj-chrctrztn*:$\Delta^c{}_1\ R = Id - (\lceil\lambda\ x.\ x \in (\pi_1[\![R]\!])\rceil)$
by(*simp add*: *p2r-def image-def fst-def rel-ad-def*, *fastforce*)

lemma *boxProgrPred-IsProp*: $wp\ R\ \lceil P\rceil \subseteq Id$
by (*simp add*: *rel-antidomain-kleene-algebra.a-subid′ rel-antidomain-kleene-algebra.addual.bbox-def*)

lemma *boxProgrRel-iso*:$P \subseteq Q \Longrightarrow wp\ R\ P \subseteq wp\ R\ Q$
by (*simp add*: *rel-antidomain-kleene-algebra.dka.dom-iso rel-antidomain-kleene-algebra.fbox-iso*)

lemma *rdom-p2r-contents*:$(a,\ b) \in rdom\ \lceil P\rceil = ((a = b) \wedge P\ a)$
proof−
have $(a,\ b) \in rdom\ (p2r\ P) = ((a = b) \wedge (a,\ a) \in rdom\ (p2r\ P))$ **using** *p2r-subid*
by *fastforce*
also have $((a = b) \wedge (a,\ a) \in rdom\ (p2r\ P)) = ((a = b) \wedge (a,\ a) \in (p2r\ P))$ **by**
simp
also have $((a = b) \wedge (a,\ a) \in (p2r\ P)) = ((a = b) \wedge P\ a)$ **by** (*simp add*: *p2r-def*)

ultimately show *?thesis* **by** *simp*
qed

lemma *complement-rule1*: $(x,x) \notin \Delta^c{}_1\ \lceil P\rceil \Longrightarrow P\ x$
by (*auto simp*: *rel-ad-def p2r-subid p2r-def*)

lemma *complement-rule2*: $(x,x) \in \Delta^c{}_1\ \lceil P\rceil \Longrightarrow \neg\ P\ x$
by (*metis ComplD VC-KAD.p2r-neg-hom complement-rule1 empty-iff mem-Collect-eq*
p2s-neg-hom
rel-antidomain-kleene-algebra.a-one rel-antidomain-kleene-algebra.am1 relcomp.relcompI)

lemma *complement-rule3*: $R \subseteq Id \Longrightarrow (x,x) \notin R \Longrightarrow (x,x) \in \Delta^c{}_1\ R$
by (*metis IdI Un-iff d-p2r rel-antidomain-kleene-algebra.addual.ars3*
rel-antidomain-kleene-algebra.addual.ars-r-def rpr)

lemma *complement-rule4*: $(x,x) \in R \Longrightarrow (x,x) \notin \Delta^c{}_1\ R$
by (*metis empty-iff rel-antidomain-kleene-algebra.addual.ars1 relcomp.relcompI*)

lemma *boxProgrPred-chrctrztn*:$(x,x) \in wp\ R\ \lceil P\rceil = (\forall\ y.\ (x,y) \in R \longrightarrow P\ y)$
by (*metis boxProgrPred-IsProp complement-rule1 complement-rule2 complement-rule3*

complement-rule4 d-p2r wp-simp wp-trafo)

lemma *boxProgrRel-chrctrztn1*:$P \subseteq Id \Longrightarrow (x,x) \in wp\ R\ P = (\forall\ y.\ (x,y) \in R$
$\longrightarrow \lfloor P\rfloor\ y)$
by (*metis boxProgrPred-chrctrztn rpr*)

lemma *boxProgrRel-chrctrztn2*:$x \in r2s\ (wp\ R\ P) = (\forall\, y.\ (x,\ y) \in R \longrightarrow \lfloor P\rfloor\ y)$
apply(*auto simp*: *r2p-def rel-antidomain-kleene-algebra.fbox-def rel-ad-def*)
subgoal by *blast*
subgoal by *blast*

done

fun *cross-list* :: $'a$ *list* $\Rightarrow$ $'b$ *list* $\Rightarrow$ $('a \times 'b)$ *list* (**infixl** $\otimes$ *63*) **where**
$[] \otimes$ *list* $= []|$
list $\otimes [] = []|$
$(x \# xtail) \otimes (y \# ytail) = (x,y) \# (xtail \otimes ytail)$

primrec *swap* :: $'a \times 'b \Rightarrow 'b \times 'a$ **where** *swap* $(x,y) = (y,x)$

primrec *listSwap* :: $('a \times 'b)$ *list* $\Rightarrow$ $('b \times 'a)$ *list* **where**
listSwap $[] = []$ |
listSwap $(head \# tail) =$ *swap head* $\#$ (*listSwap tail*)

lemma *listSwap-isMapSwap*:*listSwap l* = *map swap l*
by(*induct-tac l*, *auto*)

lemma *listSwap-crossList*[*simp*]: *listSwap* $(l2 \otimes l1) = l1 \otimes l2$
apply(*induction l1 l2 rule*: *cross-list.induct*)
apply(*metis cross-list.elims cross-list.simps*(*1*) *cross-list.simps*(*2*) *listSwap.simps*(*1*))
apply(*metis cross-list.simps*(*1*) *cross-list.simps*(*2*) *listSwap.simps*(*1*))
by *simp*

lemma *empty-crossListElim*:
$[] = xList \otimes yList \Longrightarrow [] = xList \vee [] = yList$
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *tail-crossListElim*:
$(x,\ y) \# tail = xList \otimes yList \Longrightarrow \exists xTail\ yTail.\ x \# xTail = xList \wedge y \# yTail$
$= yList$
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *non-empty-crossListElim*:
$(x,\ y) \in set\ (xList \otimes yList) \Longrightarrow x \in set\ xList \wedge y \in set\ yList$
by(*induction xList yList rule*: *cross-list.induct*, *auto*)

lemma *crossList-map-projElim*[*simp*]:$(map\ \pi_1\ list) \otimes (map\ \pi_2\ list) = list$
by(*induct-tac list*, *auto*)

lemma *tail-crossList-map-projElim*:
$(x,y)\#list = (map\ \pi_1\ l1) \otimes l2 \Longrightarrow \exists\ z\ tail.\ (x,\ z) \# tail = l1$
proof−
assume *hyp*:$(x,\ y) \# list = (map\ \pi_1\ l1) \otimes l2$
then have *noEmpt*:$(map\ \pi_1\ l1) \neq [] \wedge l2 \neq []$ **by** (*metis cross-list.elims list.discI*)

from *this* **obtain** *hd1 hd2 tl1* **and** *tl2* **where** *hd1Def*:$(map\ \pi_1\ l1) = hd1 \# tl1$
$\wedge\ l2 = hd2 \# tl2$
by (*meson list.exhaust*)
then obtain *z* **and** *tail* **where** *tailDef*:$l1 = (hd1,z) \# tail \wedge (map\ \pi_1\ tail) = tl1$

by *auto*
moreover have $(x,\ y)\ \#\ list = (hd1,hd2)\#(tl1 \otimes tl2)$ **by** (*simp add*: *hd1Def hyp*)
ultimately show *?thesis* **by** *simp*
qed

lemma *non-empty-crossList-map-projEx*:
$\forall\ xzList.\ xzList = (map\ \pi_1\ xyList) \otimes zList \longrightarrow$
$(y,\ z) \in set\ ((map\ \pi_2\ xyList) \otimes zList) \longrightarrow$
$(\exists\ x.\ (x,y) \in set\ xyList \wedge (x,z) \in set\ xzList)$
by(*simp*, *induction xyList zList rule*: *cross-list.induct*, *auto*)

lemma *crossList-length*:
$length\ xList = length\ yList \Longrightarrow length\ (xList \otimes yList) = length\ xList$
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *crossList-lengthEx*:
$length\ xList = length\ yList \Longrightarrow$
$\forall\ x \in set\ xList.\ \exists\ y \in set\ yList.\ (x,y) \in set\ (xList \otimes yList)$
apply(*induction xList yList rule*: *cross-list.induct*)
subgoal by *simp*
subgoal by *simp*
apply(*rule ballI*, *simp*, *erule disjE*, *simp*)
by *blast*

lemma *tail-crossList-length*:
$length\ (xList \otimes yList) = length\ (z\ \#\ zTail) \longrightarrow$
$(\exists\ x\ y\ xTail\ yTail.\ (xList = x\ \#\ xTail) \wedge (yList = y\ \#\ yTail) \wedge$
$length\ (xTail \otimes yTail) = length\ zTail)$
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *length-crossListProj1*:
$length\ xList = length\ yList \Longrightarrow map\ \pi_1\ (xList \otimes yList) = xList$
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *length-crossListProj2*:
$length\ xList = length\ yList \Longrightarrow map\ \pi_2\ (xList \otimes yList) = yList$
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *legnth-crossListEx1*:
$length\ (xList \otimes yList) = length\ yList \Longrightarrow$
$\forall\ y \in set\ yList.\ \exists\ x \in set\ xList.\ (x,\ y) \in set\ (xList \otimes yList)$
apply(*induction xList yList rule*: *cross-list.induct*, *simp*, *simp*)
by(*rule ballI*, *simp*, *erule disjE*, *simp*, *blast*)

lemma *legnth-crossListEx2*:
$length\ ((x\#xTail) \otimes (y\#yTail)) = length\ zList \Longrightarrow$
$\exists\ z\ zTail.\ zList = z\ \#\ zTail \wedge length\ zTail = length\ (xTail \otimes yTail)$
by(*induction zList*, *simp-all*)

lemma *legnth-crossListEx3*:
$\forall$ *zList x y*. *length* (*xList* $\otimes$ *yList*) = *length zList* $\longrightarrow$ (*x*, *y*) $\in$ *set* (*xList* $\otimes$ *yList*) $\longrightarrow$
($\exists$ *z*. (*x*, *z*) $\in$ *set* (*xList* $\otimes$ *zList*) $\wedge$ (*y*, *z*) $\in$ *set* ((*map* π_2 (*xList* $\otimes$ *yList*)) $\otimes$ *zList*))
apply(*induction xList yList rule*: *cross-list.induct*, *simp*, *simp*, *clarify*)
apply(*rename-tac x xTail y yTail zList u v*)
apply(*subgoal-tac* (*u*,*v*) = (*x*,*y*) $\vee$ (*u*,*v*) $\in$ *set* (*xTail* $\otimes$ *yTail*))
apply(*subgoal-tac* $\exists$ *z zTail*. (*zList* = *z* # *zTail*) $\wedge$ (*length*(*xTail* $\otimes$ *yTail*) = *length zTail*))
apply(*erule disjE*)
subgoal by *auto*
subgoal by *fastforce*
subgoal by (*metis cross-list.simps*(*3*) *length-Suc-conv*)
subgoal by *simp*
done

— dL CALCULUS.

term *atLeastAtMost a* (*b*::*real*)
term *greaterThanLessThan a b*
thm *atLeast-def*
term *box a* (*b*::*real*)
thm *box-def*
thm *solves-ode-def*
term *f* $\in$ *A* $\rightarrow$ *B*
thm *Pi-def*
thm *has-vderiv-on-def*
thm *has-vector-derivative-def*
thm *has-field-derivative-def*
term λ *x*. *f has-real-derivative x*
thm *has-derivative-def*

definition *solves-ivp* :: (*real* $\Rightarrow$ *'a*::*banach*) $\Rightarrow$ (*real* $\Rightarrow$ *'a* $\Rightarrow$ *'a*) $\Rightarrow$ *real* $\Rightarrow$ *'a* $\Rightarrow$ *real set* $\Rightarrow$ *'a set* $\Rightarrow$ *bool*
(- *solvesTheIVP* - *withInitCond* - $\mapsto$ - [*70*, *70*, *70*, *70*] *68*) **where**
(*x solvesTheIVP f withInitCond t0* $\mapsto$ *x0*) *Domf Codf* $\equiv$ (*x solves-ode f*) *Domf Codf* $\wedge$ *x t0* = *x0*

lemma *solves-ivpI*:
assumes (*x solves-ode f*) *A B*
assumes *x t0* = *x0*
shows (*x solvesTheIVP f withInitCond t0* $\mapsto$ *x0*) *A B*
using *assms* **by** (*simp add*: *solves-ivp-def*)

lemma *solves-ivpD*:
assumes (*x solvesTheIVP f withInitCond t0* $\mapsto$ *x0*) *A B*

shows $(x \; solves\text{-}ode \; f) \; A \; B$
and $x \; t0 = x0$
using *assms* **by** (*auto simp*: *solves-ivp-def*)

theorem(**in** *unique-on-bounded-closed*) *ivp-unique-solution*:
assumes *xIsSol*:$(x \; solvesTheIVP \; f \; withInitCond \; t0 \mapsto x0) \; T \; X$
assumes *yIsSol*:$(y \; solvesTheIVP \; f \; withInitCond \; t0 \mapsto x0) \; T \; X$
shows $\forall \; t \in T. \; x \; t = y \; t$
proof
fix t **assume** $t \in T$
from *this* **and** *assms* **show** $x \; t = y \; t$
using *unique-solution solves-ivp-def* **by** *blast*
qed

definition *vdiff* ::*string* $\Rightarrow$ *string* **where**
$vdiff \; x = ''d[''@x@'']''$

definition *varDiffs* :: *string set* **where**
$varDiffs = \{str. \; \exists \; x. \; str = vdiff \; x\}$

lemma *vdiff-inj*:$vdiff \; x = vdiff \; y \Longrightarrow x = y$
by(*simp add*: *vdiff-def*)

lemma *vdiff-noFixPoints*:$str \neq vdiff \; str$
by(*simp add*: *vdiff-def*)

lemma *varDiffsI*:$x = vdiff \; z \Longrightarrow x \in varDiffs$
by(*simp add*: *varDiffs-def vdiff-def*)

lemma *varDiffsE*:
assumes $x \in varDiffs$
obtains y **where** $x = ''d[''@y@'']''$
using *assms* **unfolding** *varDiffs-def vdiff-def* **by** *auto*

lemma *vdiff-invarDiffs*:$vdiff \; str \in varDiffs$
by (*simp add*: *varDiffsI*)

definition *solvesStoreIVP* :: $(real \Rightarrow real \; store) \Rightarrow (string \times (real \; store \Rightarrow real))$
$list \Rightarrow$
$real \; store \Rightarrow (real \; store \; pred) \Rightarrow bool$
$((\text{-} \; solvesTheStoreIVP \; \text{-} \; withInitState \; \text{-} \; andGuard \; \text{-}) \; [70, \; 70, \; 70, \; 70] \; 68)$ **where**
$solvesStoreIVP \; F \; xfList \; st \; G \equiv$
$(*$ *F preserves the guard statement and F sends vdiffs−in−list to derivs.* $*)$
$(\forall \; t \geq 0. \; G \; (F \; t) \wedge \; (\forall \; xf \in set \; xfList. \; F \; t \; (vdiff \; (\pi_1 \; xf)) = \pi_2 \; xf \; (F \; t)) \wedge$
$(*$ *F preserves the rest of the variables and F sends derivs of constants to 0.* $*)$
$(\forall \; str. \; (str \notin (\pi_1[\![set \; xfList]\!]) \cup varDiffs \longrightarrow F \; t \; str = st \; str) \wedge$
$(str \notin (\pi_1[\![set \; xfList]\!]) \longrightarrow F \; t \; (vdiff \; str) = 0)) \wedge$

(* *F solves the induced IVP.* *)
($\forall$ *xf* $\in$ *set xfList*. ((λ *t*. *F t* (π_1 *xf*)) *solvesTheIVP* (λ *t*. λ *r*. (π_2 *xf*) (*F t*))
withInitCond 0 $\mapsto$ (*st* (π_1 *xf*))) $\{0..t\}$ *UNIV*))

lemma *solves-store-ivpI*:
assumes $\forall$ *t* $\geq$ *0*. *G* (*F t*)
and $\forall$ *t* $\geq$ *0*.$\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧)$\cup$*varDiffs* $\longrightarrow$ *F t str* = *st str*
and $\forall$ *t* $\geq$ *0*.$\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *F t* (*vdiff str*) = *0*
and $\forall$ *t* $\geq$ *0*.$\forall$ *xf* $\in$ *set xfList*. (*F t* (*vdiff* (π_1 *xf*))) = (π_2 *xf*) (*F t*)
and $\forall$ *t* $\geq$ *0*. $\forall$ *xf* $\in$ *set xfList*. ((λ *t*. *F t* (π_1 *xf*)) *solvesTheIVP* (λ *t*. λ *r*. (π_2 *xf*) (*F t*))
withInitCond 0 $\mapsto$ (*st* (π_1 *xf*))) $\{0..t\}$ *UNIV*
shows *F solvesTheStoreIVP xfList withInitState st andGuard G*
using *assms solvesStoreIVP-def* **by** *auto*

named-theorems *solves-store-ivpE elimination rules for solvesStoreIVP*

lemma [*solves-store-ivpE*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
shows $\forall$ *t* $\geq$ *0*. *G* (*F t*)
and $\forall$ *t* $\geq$ *0*.$\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧)$\cup$*varDiffs* $\longrightarrow$ *F t str* = *st str*
and $\forall$ *t* $\geq$ *0*.$\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *F t* (*vdiff str*) = *0*
and $\forall$ *t* $\geq$ *0*.$\forall$ *xf* $\in$ *set xfList*. (*F t* (*vdiff* (π_1 *xf*))) = (π_2 *xf*) (*F t*)
and $\forall$ *t* $\geq$ *0*. $\forall$ *xf* $\in$ *set xfList*. ((λ *t*. *F t* (π_1 *xf*)) *solvesTheIVP* (λ *t*. λ *r*. (π_2 *xf*) (*F t*))
withInitCond 0 $\mapsto$ (*st* (π_1 *xf*))) $\{0..t\}$ *UNIV*
using *assms solvesStoreIVP-def* **by** *auto*

lemma [*solves-store-ivpE*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
shows $\forall$ *str*. *str* $\notin$ *varDiffs* $\longrightarrow$ *F 0 str* = *st str*
proof(*clarify*, *rename-tac x*)
fix *x* **assume** *x* $\notin$ *varDiffs*
from *assms* **and** *solves-store-ivpE*(*5*)
have $\forall$ *f*. (*x*,*f*)$\in$*set xfList* $\longrightarrow$ ((λ*t*. *F t x*) *solvesTheIVP* (λ*t r*. *f* (*F t*)) *withInitCond*
0 $\mapsto$ *st x*) $\{0..0\}$ *UNIV* **by** *force*
hence *x* $\in$(π_1⟦*set xfList*⟧) $\Longrightarrow$ *F 0 x* = *st x* **using** *solves-ivpD*(*2*) **by** *fastforce*
also have *x* $\notin$(π_1⟦*set xfList*⟧)$\cup$*varDiffs* $\Longrightarrow$ *F 0 x* = *st x*
using *assms* **and** *solves-store-ivpE*(*2*) **by** *simp*
ultimately show *F 0 x* = *st x* **using** ‹*x* $\notin$ *varDiffs*› **by** *auto*
qed

named-theorems *solves-store-ivpD computation rules for solvesStoreIVP*

lemma [*solves-store-ivpD*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
and *t* $\geq$ *0*
shows *G* (*F t*)

using *assms solves-store-ivpE*(*1*) **by** *blast*

lemma [*solves-store-ivpD*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
and $t \geq 0$
and $str \notin (\pi_1[\![set\ xfList]\!]) \cup varDiffs$
shows *F t str* = *st str*
using *assms solves-store-ivpE*(*2*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
and $t \geq 0$
and $str \notin (\pi_1[\![set\ xfList]\!])$
shows *F t* (*vdiff str*) = *0*
using *assms solves-store-ivpE*(*3*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
and $t \geq 0$
and $xf \in set\ xfList$
shows $(F\ t\ (vdiff\ (\pi_1\ xf))) = (\pi_2\ xf)\ (F\ t)$
using *assms solves-store-ivpE*(*4*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
and $t \geq 0$
and $xf \in set\ xfList$
shows $((\lambda\ t.\ F\ t\ (\pi_1\ xf))$ *solvesTheIVP* $(\lambda\ t.\ \lambda\ r.\ (\pi_2\ xf)\ (F\ t))$
withInitCond $0 \mapsto (st\ (\pi_1\ xf)))\ \{0..t\}$ *UNIV*
using *assms solves-store-ivpE*(*5*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes *F solvesTheStoreIVP xfList withInitState st andGuard G*
and $str \notin varDiffs$
shows *F 0 str* = *st str*
using *assms solves-store-ivpE*(*6*) **by** *simp*

thm *solves-store-ivpE*
thm *solves-store-ivpD*

definition *guarDiffEqtn* :: (*string* × (*real store* ⇒ *real*)) *list* ⇒ (*real store pred*) ⇒
real store rel (*ODEsystem - with -* [*70*, *70*] *61*) **where**
ODEsystem xfList with G = $\{(st,F\ t)\ |st\ t\ F.\ t \geq 0 \wedge$ *solvesStoreIVP F xfList st G*}

— Differential Weakening.
lemma *box-evol-guard*:*Id* ⊆ *wp* (*ODEsystem xfList with G*) ⌈*G*⌉
apply(*simp add*: *rel-antidomain-kleene-algebra.fbox-def rel-ad-def guarDiffEqtn-def*

p2r-def)
using *solves-store-ivpD*(*1*) **by** *force*

theorem *dWeakening*:
assumes *guardImpliesPost*: $\lceil G \rceil \subseteq \lceil Q \rceil$
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
using *assms* **and** *box-evol-guard* **by** (*metis* (*no-types*, *hide-lams*) *d-p2r*
order-trans p2r-subid rel-antidomain-kleene-algebra.fbox-iso)

lemma *PRE* (λ *s. s* $''x'' = 0$)
  (*ODEsystem* [($''x''$,(λ *s. s* $''x'' + 1$))] *with* (λ *s. s* $''x'' \geq 0$))
  POST (λ *s. s* $''x'' \geq 0$)
using *dWeakening* **by** *blast*

lemma *PRE* (λ *s. s* $''x'' = 0$)
  (*ODEsystem* [($''x''$,(λ *s. s* $''x'' + 1$))] *with* (λ *s. s* $''x'' \geq 0$))
  POST (λ *s. s* $''x'' \geq 0$)
apply(*clarify*, *simp add*: *p2r-def*)
apply(*simp add*: *rel-ad-def rel-antidomain-kleene-algebra.addual.ars-r-def*)
apply(*simp add*: *rel-antidomain-kleene-algebra.fbox-def*)
apply(*simp add*: *relcomp-def rel-ad-def guarDiffEqtn-def*)
apply(*simp add*: *solvesStoreIVP-def*)
apply(*auto*)
done

— Differential Cut.
lemma *condAfterEvol-remainsAlongEvol*:
assumes *boxDiffC*:(*a*, *a*) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$
assumes *FisSol*:*solvesStoreIVP F xfList a G*
shows *solvesStoreIVP F xfList a* (λ*s. G s* $\wedge$ *C s*)
apply(*rule solves-store-ivpI*)
subgoal proof(*clarify*)
from *boxDiffC* **have** *diffHyp*:$\forall$ *c.* (*a*,*c*) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ *C c*
using *guarDiffEqtn-def wp-trafo* **by** (*metis* (*no-types*, *lifting*) *Domain.intros r2p-def*)
fix *t*::*real* **assume** *tHyp*:$0 \leq t$
then have *odeF*:(*a*,*F t*) $\in$ (*ODEsystem xfList with G*) **using** *FisSol guarDiffEqtn-def*
by *auto*
from *this diffHyp* **and** *tHyp* **show** *G* (*F t*) $\wedge$ *C* (*F t*) **using** *solves-store-ivpD*(*1*)
FisSol **by** *blast*
qed
using *FisSol solvesStoreIVP-def* **by** *auto*

lemma *boxDiffCond-impliesAllTimeInCond*:
assumes *allTime*: (*t*::*real*)≥ 0
assumes *boxDifCond*:(*a*,*a*) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$
assumes *FisSol*:*solvesStoreIVP F xfList a G*
shows (*a*,*F t*) $\in$ (*ODEsystem xfList with* (λ*s. G s* $\wedge$ *C s*))
apply(*simp add*: *guarDiffEqtn-def*)

apply(*rule-tac x=t* **in** *exI*, *rule-tac x=F* **in** *exI*, *simp add*: *allTime*)
apply(*rule condAfterEvol-remainsAlongEvol*)
using *boxDifCond guarDiffEqtn-def FisSol* **by** *safe*

theorem *dCut*:
assumes *pBoxDiffCut*:(*PRE P* (*ODEsystem xfList with G*) *POST C*)
assumes *pBoxCutQ*:(*PRE P* (*ODEsystem xfList with* (λ *s*. *G s* $\wedge$ *C s*)) *POST Q*)
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
proof(*clarify*)
fix *a b*::*real store* **assume** *abHyp*:(*a*,*b*) $\in$ *rdom* $\lceil P \rceil$
from *this* **have** *a* = *b* $\wedge$ *P a* **by** (*metis rdom-p2r-contents*)
from *this abHyp* **and** *pBoxDiffCut* **have** (*a*,*a*) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$ **by** *blast*
moreover
from *pBoxCutQ* **and** *abHyp* **have** $\forall$ *c*. (*a*,*c*) $\in$ (*ODEsystem xfList with* (λ*s*. *G s* $\wedge$ *C s*)) $\longrightarrow$ *Q c*
by (*metis* (*no-types*, *lifting*) $\langle$*a* = *b* $\wedge$ *P a*$\rangle$ *boxProgrPred-chrctrztn set-mp*)
ultimately have $\forall$ *c*. (*a*,*c*) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ *Q c*
using *guarDiffEqtn-def boxDiffCond-impliesAllTimeInCond* **by** *auto*
from *this* **and** $\langle$*a* = *b* $\wedge$ *P a*$\rangle$ **show** (*a*,*b*) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil Q \rceil$
by (*simp add*: *boxProgrPred-chrctrztn*)
qed

— Solve Differential Equation.
definition *vderiv-of f S* = (*SOME f'*. (*f has-vderiv-on f'*) *S*)
abbreviation *varDiffs-to-zero* ::*real store* $\Rightarrow$ *real store* (*d2z*) **where**
d2z a $\equiv$ (*override-on a* (λ *str*. *0*) *varDiffs*)

lemma *varDiffs-to-zero-beginning*[*simp*]: *take 2 x* $\neq$ ''*d*['' $\Longrightarrow$ (*d2z a*) *x* = *a x*
apply(*simp add*: *varDiffs-def override-on-def vdiff-def*)
by(*fastforce*)

lemma *override-on-upd*:*x* $\in$ *X* $\Longrightarrow$ (*override-on f g X*)(*x* := *z*) = (*override-on f* (*g*(*x* := *z*)) *X*)
by(*rule ext*, *simp add*: *override-on-def*)

primrec *state-list-upd* :: ((*real* $\Rightarrow$ *real store* $\Rightarrow$ *real*) $\times$ *string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* $\Rightarrow$
real $\Rightarrow$ *real store* $\Rightarrow$ *real store* **where**
state-list-upd [] *t a* = *a*|
state-list-upd (*uxf* # *tail*) *t a* = (*state-list-upd tail t a*)
((π_1 (π_2 *uxf*)) := (π_1 *uxf*) *t a*,
vdiff (π_1 (π_2 *uxf*)) := (*if t* = *0 then* (π_2 (π_2 *uxf*)) *a*
else vderiv-of (λ *r*. (π_1 *uxf*) *r a*) {*0*<..< (*2* $*_R$ *t*)} *t*))

abbreviation *state-list-cross-upd* ::*real store* $\Rightarrow$ (*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* $\Rightarrow$
(*real* $\Rightarrow$ *real store* $\Rightarrow$ *real*) *list* $\Rightarrow$ *real* $\Rightarrow$ (*char list* $\Rightarrow$ *real*) (-[-$\leftarrow$-] - [*64*,*64*,*64*] *63*) **where**

$s[xfList \leftarrow uInput]\ t \equiv state\text{-}list\text{-}upd\ (uInput \otimes xfList)\ t\ s$

lemma *state-list-cross-upd-empty*[*simp*]: $(a[[] \leftarrow list]\ t) = a$
by(*induction list*, *simp-all*)

lemma *state-list-cross-upd-its-vars*:
$distinct\ (map\ \pi_1\ xfList) \longrightarrow (\forall\ var \in set\ (map\ \pi_1\ xfList).\ var \notin varDiffs) \longrightarrow$
$length\ (xfList) = length\ (uInput) \longrightarrow (\forall\ uxf \in set\ (uInput \otimes xfList).$
$(a[xfList \leftarrow uInput]\ t)\ (\pi_1\ (\pi_2\ uxf)) = (\pi_1\ uxf)\ t\ a)$
apply(*simp*, *induction xfList uInput rule*: *cross-list.induct*, *simp*, *simp*)
proof(*clarify*, *rename-tac x f xfTail u uTail s y g*)
fix $x\ y$::*string* **and** $f\ g$::$real\ store \Rightarrow real$ **and** $uTail$::$(real \Rightarrow real\ store \Rightarrow real)list$
and $xfTail$::$(string \times (real\ store \Rightarrow real))list$ **and** $u\ s$::$real \Rightarrow real\ store \Rightarrow real$
let $?gLHS = (a[((x, f)\ \#\ xfTail) \leftarrow (u\ \#\ uTail)]\ t)\ (\pi_1\ (\pi_2\ (s, y, g)))$
let $?goal = ?gLHS = \pi_1\ (s, y, g)\ t\ a$
assume IH:$distinct\ (map\ \pi_1\ xfTail) \longrightarrow (\forall\ xf \in set\ xfTail.\ \pi_1\ xf \notin varDiffs)$
$\longrightarrow$
$length\ xfTail = length\ uTail \longrightarrow (\forall uxf \in set\ (uTail \otimes xfTail).$
$(a[xfTail \leftarrow uTail]\ t)\ (\pi_1\ (\pi_2\ uxf)) = \pi_1\ uxf\ t\ a)$
and $distHyp$:$distinct\ (map\ \pi_1\ ((x, f)\ \#\ xfTail))$
and $varsHyp$:$\forall xf \in set\ ((x, f)\ \#\ xfTail).\ \pi_1\ xf \notin varDiffs$
and $lengthHyp$:$length\ ((x, f)\ \#\ xfTail) = length\ (u\ \#\ uTail)$
then have $keyHyp$:$\forall uxf \in set\ (uTail \otimes xfTail).$
$(a[xfTail \leftarrow uTail]\ t)\ (\pi_1\ (\pi_2\ uxf)) = \pi_1\ uxf\ t\ a$ **by** *fastforce*
assume $(s, y, g) \in set\ ((u\ \#\ uTail) \otimes ((x, f)\ \#\ xfTail))$
from *this* **have** $(s, y, g) = (u, x, f) \vee (s, y, g) \in set\ (uTail \otimes xfTail)$ **by** *simp*
moreover
{**assume** eq:$(s, y, g) = (u, x, f)$
  then have $?gLHS = ((a[xfTail \leftarrow uTail]\ t)(y := s\ t\ a,\ vdiff\ y := if\ t = 0\ then\ g$
a
  $else\ vderiv\text{-}of\ (\lambda\ r.\ s\ r\ a)\ \{0{<}..{<}\ (2 *_R t)\}\ t))\ y$ **by** *simp*
  also have $((a[xfTail \leftarrow uTail]\ t)(y := s\ t\ a,\ vdiff\ y := if\ t = 0\ then\ g\ a$
  $else\ vderiv\text{-}of\ (\lambda\ r.\ s\ r\ a)\ \{0{<}..{<}\ (2 *_R t)\}\ t))\ y = s\ t\ a$
  using *eq* **by** (*simp add*: *vdiff-def*)
  ultimately have *?goal* **by** *simp*}
moreover
{**assume** $yTailHyp$:$(s, y, g) \in set\ (uTail \otimes xfTail)$
  from *this* **and** *keyHyp* **have** 3:$(a[xfTail \leftarrow uTail]\ t)\ y = s\ t\ a$ **by** *fastforce*
  from *yTailHyp* **and** *distHyp* **have** 2:$y \neq x$ **using** *non-empty-crossListElim* **by**
force
  from *yTailHyp* **and** *varsHyp* **have** 1:$y \neq vdiff\ x$
  using *non-empty-crossListElim vdiff-invarDiffs* **by** *fastforce*
  from *1* **and** *2* **have** $?gLHS = (a[xfTail \leftarrow uTail]\ t)\ y$ **by** (*simp*)
  then have *?goal* **using** *3* **by** *simp*}
ultimately show *?goal* **by** *blast*
qed

lemma *state-list-cross-upd-its-dvars*:
assumes $lengthHyp$:$length\ xfList = length\ uInput$

and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *solHyp3*:$\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0 a* = *a* (π_1 (π_2 *uxf*))
shows $\exists$ *g*. (*a*[*xfList*$\leftarrow$*uInput*] *0*) = (*override-on a g varDiffs*)
using *assms* **proof**(*induction xfList uInput rule*: *cross-list.induct*)
case (*1 list*)
have (*a*[[]$\leftarrow$*list*] *0*) = *a* **by** *simp*
also have *override-on a a varDiffs* = *a*
unfolding *override-on-def* **by** *simp*
ultimately show *?case* **by** *metis*
next
case (*2 xf xfTail*)
have (*a*[(*xf* # *xfTail*)$\leftarrow$[]] *0*) = *a* **by** *simp*
also have *override-on a a varDiffs* = *a*
unfolding *override-on-def* **by** *simp*
ultimately show *?case* **by** *metis*
next
case (*3 xf xfTail u uTail*)
let *?gLHS*=(*a*[(*xf* # *xfTail*)$\leftarrow$(*u* # *uTail*)] *0*)
have *observ*:*vdiff* (π_1 *xf*) $\in$ *varDiffs* **by** (*auto simp*: *varDiffs-def*)
assume *IH*:*length xfTail* = *length uTail* $\Longrightarrow$ $\forall$ *xf*$\in$*set xfTail*. π_1 *xf* $\notin$ *varDiffs* $\Longrightarrow$
$\forall$ *uxf*$\in$*set* (*uTail* $\otimes$ *xfTail*). π_1 *uxf 0 a* = *a* (π_1 (π_2 *uxf*)) $\Longrightarrow$
$\exists$ *g*. (*a*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on a g varDiffs*
assume *length* (*xf* # *xfTail*) = *length* (*u* # *uTail*)
and *solHyp*:$\forall$ *uxf*$\in$*set* ((*u* # *uTail*) $\otimes$ (*xf* # *xfTail*)). π_1 *uxf 0 a* = *a* (π_1 (π_2 *uxf*))
and *no-varDiffs*:$\forall$ *xf*$\in$*set* (*xf* # *xfTail*). π_1 *xf* $\notin$ *varDiffs*
from *this* **and** *IH* **obtain** *g* **where** (*a*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on a g varDiffs*
by *force*
then have *1*:*?gLHS* = (*override-on a g varDiffs*)(π_1 *xf* := *u 0 a*, *vdiff* (π_1 *xf*)
:= π_2 *xf a*) **by** *simp*
also have *2*:(*override-on a g varDiffs*)(π_1 *xf* := *u 0 a*, *vdiff* (π_1 *xf*) := π_2 *xf a*)
=
(*override-on a g varDiffs*)(*vdiff* (π_1 *xf*) := π_2 *xf a*)
using *override-on-def varDiffs-def 3.prems*(*2*) *solHyp* **by** *auto*
also have *3*:(*override-on a g varDiffs*)(*vdiff* (π_1 *xf*) := π_2 *xf a*) =
(*override-on a* (*g*(*vdiff* (π_1 *xf*) := π_2 *xf a*)) *varDiffs*) **using** *observ* **and** *override-on-upd*
by *force*
ultimately show *?case* **by** *auto*
qed

lemma *state-list-cross-upd-uxf-on-x*:
assumes *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *uxfHyp*:(*u*, *x*, *f*) $\in$ *set* (*uInput* $\otimes$ *xfList*)
shows (*a*[*xfList*$\leftarrow$*uInput*] *t*) *x* = *u t a*
using *assms* **and** *state-list-cross-upd-its-vars* **by** *force*

abbreviation *state-to-sol*::*real store* $\Rightarrow$ (*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* $\Rightarrow$
(*real* $\Rightarrow$ *real store* $\Rightarrow$ *real*) *list* $\Rightarrow$ *real* $\Rightarrow$ (*char list* $\Rightarrow$ *real*)

(*sol* -[-←-] - [*64*,*64*,*64*] *63*) **where** *sol s*[*xfList*←*uInput*] *t* ≡ *d2z s*[*xfList*←*uInput*] *t*

lemma *prelim-dSolve*:
assumes *solHyp*:(λ*t*. *sol a*[*xfList*←*uInput*] *t*) *solvesTheStoreIVP xfList withInitState a andGuard G*
and *uniqHyp*:∀ *X*. *solvesStoreIVP X xfList a G* ⟶ (∀ *t* ≥ *0*. (*sol a*[*xfList*←*uInput*] *t*) = *X t*)
and *diffAssgn*: ∀*t*≥*0*. *G* (*sol a*[*xfList*←*uInput*] *t*) ⟶ *Q* (*sol a*[*xfList*←*uInput*] *t*)
shows ∀ *c*. (*a*,*c*) ∈ (*ODEsystem xfList with G*) ⟶ *Q c*
proof(*clarify*)
fix *c* **assume** (*a*,*c*) ∈ (*ODEsystem xfList with G*)
from *this* **obtain** *t*::*real* **and** *F*::*real* ⇒ *real store*
where *FHyp*:*t*≥*0* ∧ *F t* = *c* ∧ *solvesStoreIVP F xfList a G* **using** *guarDiffEqtn-def*
by *auto*
from *this* **and** *uniqHyp* **have** (*sol a*[*xfList*←*uInput*] *t*) = *F t* **by** *blast*
then have *cHyp*:*c* = (*sol a*[*xfList*←*uInput*] *t*) **using** *FHyp* **by** *simp*
from *solHyp* **have** *G* (*sol a*[*xfList*←*uInput*] *t*) **by** (*simp add*: *solvesStoreIVP-def FHyp*)
then show *Q c* **using** *diffAssgn FHyp cHyp* **by** *auto*
qed

theorem *wlp-guard-inv*:
assumes *solHyp*:*solvesStoreIVP* (λ*t*. *sol a*[*xfList*←*uInput*] *t*) *xfList a G*
and *uniqHyp*:∀ *X*. *solvesStoreIVP X xfList a G* ⟶ (∀ *t* ≥ *0*. (*sol a*[*xfList*←*uInput*] *t*) = *X t*)
and *diffAssgn*: ∀*t*≥*0*. *G* (*sol a*[*xfList*←*uInput*] *t*) ⟶ *Q* (*sol a*[*xfList*←*uInput*] *t*)
shows ⌊*wp* (*ODEsystem xfList with G*) ⌈*Q*⌉⌋ *a*
apply(*simp add*: *r2p-def*, *subst boxProgrRel-chrctrztn2*)
apply(*simp-all add*: *p2r-def*, *rule-tac uInput*=*uInput* **in** *prelim-dSolve*)
by (*simp-all add*: *r2p-def Domain-unfold assms*)

theorem *dSolve*:
assumes *solHyp*:∀*st*. *solvesStoreIVP* (λ*t*. *sol st*[*xfList*←*uInput*] *t*) *xfList st G*
and *uniqHyp*:∀ *st*. ∀ *X*. *solvesStoreIVP X xfList st G* ⟶ (∀ *t* ≥ *0*.(*sol st*[*xfList*←*uInput*] *t*) = *X t*)
and *diffAssgn*: ∀*st*. *P st* ⟶ (∀*t*≥*0*. *G* (*sol st*[*xfList*←*uInput*] *t*) ⟶ *Q* (*sol st*[*xfList*←*uInput*] *t*))
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*clarsimp*, *subgoal-tac a*=*b*)
apply(*clarify*, *subst boxProgrPred-chrctrztn*)
apply(*simp-all add*: *p2r-def*)
apply(*rule-tac uInput*=*uInput* **in** *prelim-dSolve*)
apply(*simp add*: *solHyp*, *simp add*: *uniqHyp*)
by (*metis* (*no-types*, *lifting*) *diffAssgn*)

lemma *conds4InitState*:
assumes *initHyp*:$\forall$ *st*. $\forall$ *uxf* $\in$ *set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0 st* = *st* (π_1 (π_2 *uxf*))
shows $\forall$ *str*. *str* $\notin$ *varDiffs* $\longrightarrow$ (*sol a*[*xfList*$\leftarrow$*uInput*] *0*) *str* = *a str*
using *assms* **apply**(*induction uInput xfList rule*: *cross-list.induct*, *simp-all*)
by(*simp add*: *varDiffs-def vdiff-def*)

lemma *conds4InitState2*:
assumes *funcsHyp*:$\forall$ *st*. $\forall$ *g*. $\forall$ *xf* $\in$ *set xfList*.
π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st*
and *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *solHyp3*:$\forall$ *st*. $\forall$ *uxf* $\in$ *set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*) (π_1 (π_2 *uxf*))
shows $\forall$ *st*. $\forall$ *xf* $\in$ *set xfList*.
(*sol st*[*xfList*$\leftarrow$*uInput*] *0*)(*vdiff* (π_1 *xf*)) = π_2 *xf* (*sol st*[*xfList*$\leftarrow$*uInput*] *0*)
using *assms* **apply**(*induction uInput xfList rule*: *cross-list.induct*, *simp*, *simp*)
proof(*clarify*, *rename-tac u uTail x f xfTail a y g*)
fix *x y* ::*string* **and** *f g*::*real store* $\Rightarrow$ *real*
and *u s*::*real* $\Rightarrow$ *real store* $\Rightarrow$ *real* **and** *a*::*real store* **and**
xfTail::(*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* **and** *uTail*::(*real* $\Rightarrow$ *real store* $\Rightarrow$ *real*) *list*
assume *IH*:$\forall$ *st g*. $\forall$ *xf*$\in$*set xfTail*. π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st* $\Longrightarrow$
distinct (*map* π_1 *xfTail*) $\Longrightarrow$ *length xfTail* = *length uTail* $\Longrightarrow$ $\forall$ *xf*$\in$*set xfTail*. π_1 *xf* $\notin$ *varDiffs* $\Longrightarrow$
$\forall$ *st*. $\forall$ *uxf*$\in$*set* (*uTail* $\otimes$ *xfTail*). π_1 *uxf 0* (*d2z st*) = *d2z st* (π_1 (π_2 *uxf*)) $\Longrightarrow$
$\forall$ *st*. $\forall$ *xf*$\in$*set xfTail*. (*sol st*[*xfTail*$\leftarrow$*uTail*] *0*) (*vdiff* (π_1 *xf*)) = π_2 *xf* (*sol st*[*xfTail*$\leftarrow$*uTail*] *0*)
let *?gLHS* = (*sol a*[((*x*, *f*) # *xfTail*)$\leftarrow$(*u* # *uTail*)] *0*) (*vdiff* (π_1 (*y*, *g*)))
let *?gRHS* = π_2 (*y*, *g*) (*sol a*[((*x*, *f*) # *xfTail*)$\leftarrow$(*u* # *uTail*)] *0*)
let *?goal* = *?gLHS* = *?gRHS*
assume *eqFuncs*:$\forall$ *st g*. $\forall$ *xf*$\in$*set* ((*x*, *f*) # *xfTail*). π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st*
and *eqLengths*:*length* ((*x*, *f*) # *xfTail*) = *length* (*u* # *uTail*)
and *distinct*:*distinct* (*map* π_1 ((*x*, *f*) # *xfTail*))
and *vars*:$\forall$ *xf*$\in$*set* ((*x*, *f*) # *xfTail*). π_1 *xf* $\notin$ *varDiffs*
and *solHyp*:$\forall$ *st*.$\forall$ *uxf*$\in$*set* ((*u* # *uTail*) $\otimes$ ((*x*, *f*) # *xfTail*)). π_1 *uxf 0* (*d2z st*) = *d2z st* (π_1 (π_2 *uxf*))
from *this* **obtain** *h1* **where** *h1Def*:(*sol a*[((*x*, *f*) # *xfTail*)$\leftarrow$(*u* # *uTail*)] *0*) = (*override-on* (*d2z a*) *h1 varDiffs*) **using** *state-list-cross-upd-its-dvars* **by** *blast*
from *IH eqFuncs distinct eqLengths vars* **and** *solHyp* **have** *summary*:$\forall$ *xf*$\in$*set xf-Tail*.
(*sol a*[*xfTail*$\leftarrow$*uTail*] *0*) (*vdiff* (π_1 *xf*)) = π_2 *xf* (*sol a*[*xfTail*$\leftarrow$*uTail*] *0*) **by** *simp*
assume (*y*, *g*) $\in$ *set* ((*x*, *f*) # *xfTail*)
then have (*y*, *g*) = (*x*, *f*) $\vee$ (*y*, *g*) $\in$ *set xfTail* **by** *simp*
moreover
{**assume** *eqHeads*:(*y*, *g*) = (*x*, *f*)
then have *1*:*?gRHS* =*f* (*state-list-upd* ((*u*,*x*,*f*) # (*uTail* $\otimes$ *xfTail*)) *0* (*d2z a*)) **by** *simp*

have *2*:*f* (*state-list-upd* ((*u*,*x*,*f*) # (*uTail* ⊗ *xfTail*)) *0* (*d2z a*)) =
f (*override-on* (*d2z a*) *h1 varDiffs*) **using** *h1Def* **by** *simp*
have *3*:*f* (*override-on* (*d2z a*) *h1 varDiffs*) = *f* (*d2z a*) **using** *eqFuncs* **by** *simp*
have *f* (*d2z a*) = *?gLHS* **by** (*simp add*: *eqHeads*)
hence *?goal* **using** *1 2* **and** *3* **by** *simp*}
moreover
{**assume** *tailHyp*:(*y*, *g*) ∈ *set xfTail*
obtain *h2* **where** *h2Def*:(*sol a*[*xfTail*←*uTail*] *0*) = *override-on* (*d2z a*) *h2 varDiffs*
using *state-list-cross-upd-its-dvars eqLengths distinct vars* **and** *solHyp* **by** *force*
from *eqFuncs* **and** *tailHyp* **have** *h2Hyp*:*g* (*override-on* (*d2z a*) *h2 varDiffs*) = *g* (*d2z a*) **by** *force*
from *tailHyp* **have** ∗:*g* (*sol a*[*xfTail*←*uTail*] *0*) = (*sol a*[*xfTail*←*uTail*] *0*) (*vdiff y*)
using *summary* **by** *fastforce*
from *tailHyp* **have** *y* ≠ *x* **using** *distinct non-empty-crossListElim* **by** *force*
hence *dXnotdY*:*vdiff x* ≠ *vdiff y* **by**(*simp add*: *vdiff-def*)
have *xNotdY*:*x* ≠ *vdiff y* **using** *vars vdiff-invarDiffs* **by** *auto*
from ∗ **have** *?gLHS* = *g* (*sol a*[*xfTail*←*uTail*] *0*) **using** *xNotdY* **and** *dXnotdY* **by** *simp*
then have *?gLHS* = *g* (*d2z a*) **using** *h2Hyp* **and** *h2Def* **by** *simp*
also have *?gRHS* = *g* (*d2z a*) **using** *eqFuncs h1Def* **and** *tailHyp* **by** *fastforce*
ultimately have *?goal* **by** *simp*}
ultimately show *?goal* **by** *blast*
qed

lemma *state-list-cross-upd-correctInPrimes*:
distinct (*map* π_1 *xfList*) ⟶ (∀ *var* ∈ *set* (*map* π_1 *xfList*). *var* ∉ *varDiffs*) ⟶
length xfList = *length uInput* ⟶ *t* > *0* ⟶ (∀ *uxf* ∈*set* (*uInput* ⊗ *xfList*).
(*a*[*xfList*←*uInput*] *t*) (*vdiff* (π_1 (π_2 *uxf*))) = *vderiv-of* (λ *r*. (π_1 *uxf*) *r a*) {*0*<..< (*2* $*_R$ *t*)} *t*)
apply(*simp*, *induction uInput xfList rule*: *cross-list.induct*, *simp*, *simp*, *clarify*)
proof(*rename-tac u uTail x f xfTail s y g*)
fix *x y* ::*string* **and** *f g*::*real store* ⇒ *real* **and** *u s*::*real* ⇒ *real store* ⇒ *real* **and** *xfTail*::(*string* × (*real store* ⇒ *real*)) *list* **and** *uTail*::(*real* ⇒ *real store* ⇒ *real*) *list*
assume *IH*:*distinct* (*map* π_1 *xfTail*) ⟶ (∀ *var*∈*set xfTail*. π_1 *var* ∉ *varDiffs*) ⟶
length xfTail = *length uTail* ⟶ *0* < *t* ⟶ (∀ *uxf*∈*set* (*uTail* ⊗ *xfTail*).
(*a*[*xfTail*←*uTail*] *t*) (*vdiff* (π_1 (π_2 *uxf*))) = *vderiv-of* (λ*r*. π_1 *uxf r a*) {*0*<..<*2* · *t*} *t*)
assume *lengthHyp*:*length* ((*x*, *f*) # *xfTail*) = *length* (*u* # *uTail*) **and** *tHyp*:*0* < *t*
and *distHyp*:*distinct* (*map* π_1 ((*x*, *f*) # *xfTail*))
and *varHyp*:∀ *xf*∈*set* ((*x*, *f*) # *xfTail*). π_1 *xf* ∉ *varDiffs*
from *this* **and** *IH* **have** *keyFact*:∀ *uxf*∈*set* (*uTail* ⊗ *xfTail*).
(*a*[*xfTail*←*uTail*] *t*) (*vdiff* (π_1 (π_2 *uxf*))) = *vderiv-of* (λ*r*. π_1 *uxf r a*) {*0*<..<*2* · *t*} *t* **by** *simp*
assume *sygHyp*:(*s*, *y*, *g*) ∈ *set* ((*u* # *uTail*) ⊗ ((*x*, *f*) # *xfTail*))
let *?gLHS* = (*a*[(*x*, *f*) # *xfTail*←*u* # *uTail*] *t*) (*vdiff* (π_1 (π_2 (*s*, *y*, *g*))))
let *?gRHS* = *vderiv-of* (λ*r*. π_1 (*s*, *y*, *g*) *r a*) {*0*<..<*2* · *t*} *t*
let *?goal* = *?gLHS* = *?gRHS*

let *?lhs* =
((*a*[*xfTail*←*uTail*] *t*)(*x* := *u t a*, *vdiff x* := *vderiv-of* (λ *r*. *u r a*) {*0*<..< (*2* · *t*)} *t*)) (*vdiff y*)
let *?rhs* = *vderiv-of* (λ*r*. *s r a*) {*0*<..< (*2* · *t*)} *t*
from *sygHyp* **have** (*s*, *y*, *g*) = (*u*, *x*, *f*) ∨ (*s*, *y*, *g*) ∈ *set* (*uTail* ⊗ *xfTail*) **by** *simp*
moreover
{have *?gLHS* = *?lhs* **using** *tHyp* **by** *simp*
also have *?gRHS* =*?rhs* **by** *simp*
ultimately have *?goal* = (*?lhs* = *?rhs*) **by** *simp***}**
moreover
{assume *uxfEq*:(*s*, *y*, *g*) = (*u*, *x*, *f*)
then have *?lhs* = *vderiv-of* (λ *r*. *u r a*) {*0*<..< (*2* · *t*)} *t* **by** *simp*
also have *vderiv-of* (λ *r*. *u r a*) {*0*<..< (*2* · *t*)} *t* = *?rhs* **using** *uxfEq* **by** *simp*
ultimately have *?lhs* = *?rhs* **by** *simp***}**
moreover
{assume *sygTail*:(*s*, *y*, *g*) ∈ *set* (*uTail* ⊗ *xfTail*)
from *this* **have** *y* ≠ *x* **using** *distHyp non-empty-crossListElim* **by** *force*
hence *dXnotdY*:*vdiff x* ≠ *vdiff y* **by**(*simp add*: *vdiff-def*)
have *xNotdY*:*x* ≠ *vdiff y* **using** *varHyp* **using** *vdiff-invarDiffs* **by** *auto*
then have *?lhs* = (*a*[*xfTail*←*uTail*] *t*) (*vdiff y*) **using** *xNotdY* **and** *dXnotdY* **by** *simp*
also have (*a*[*xfTail*←*uTail*] *t*) (*vdiff y*) = *?rhs* **using** *keyFact sygTail* **by** *auto*
ultimately have *?lhs* = *?rhs* **by** *simp***}**
ultimately show *?goal* **by** *blast*
qed

lemma *prelim-conds4vdiffs*:
assumes *funcsHyp*:∀ *st g*. ∀ *xf*∈*set xfList*. π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st*
and *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *varsHyp*:∀ *xf* ∈ *set xfList*. π_1 *xf* ∉ *varDiffs*
and *lengthHyp*:*length xfList* = *length uInput*
and *solHyp3*:∀ *st*. ∀ *uxf* ∈ *set* (*uInput* ⊗ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*) (π_1 (π_2 *uxf*))
and *keyFact*:∀ *st*. ∀ *uxf* ∈ *set* (*uInput* ⊗ *xfList*).∀ *t*>*0*.
vderiv-of (λ*r*. (π_1 *uxf*) *r* (*d2z st*)) {*0*<..< (*2* $*_R$ *t*)} *t* = (π_2 (π_2 *uxf*)) (*sol st*[*xfList*←*uInput*] *t*)
shows ∀ *st*. ∀ *t*≥*0*. ∀ *xf*∈*set xfList*.
(*sol st*[*xfList*←*uInput*] *t*) (*vdiff* (π_1 *xf*)) = π_2 *xf* (*sol st*[*xfList*←*uInput*] *t*)
proof(*clarify*)
fix *t* ::*real* **and** *x*::*string* **and** *f*::*real store* ⇒ *real* **and** *a*::*real store*
assume *tHyp*:*0* ≤ *t* **and** *pairHyp*:(*x*, *f*) ∈ *set xfList*
from *this* **obtain** *u* **where** *xfuHyp*: (*u*,*x*,*f*) ∈ *set* (*uInput* ⊗ *xfList*)
by (*metis crossList-length legnth-crossListEx1 lengthHyp*)
show (*sol a*[*xfList*←*uInput*] *t*) (*vdiff* (π_1 (*x*, *f*))) = π_2 (*x*, *f*) (*sol a*[*xfList*←*uInput*] *t*)
proof(*cases t*=*0*)
case *True*

have $\forall$ *st*. $\forall$ *xf*$\in$*set xfList*.
(*sol st*[*xfList*$\leftarrow$*uInput*] *0*) (*vdiff* (π_1 *xf*)) = π_2 *xf* (*sol st*[*xfList*$\leftarrow$*uInput*] *0*)
using *assms* **and** *conds4InitState2* **by** *blast*
then show *?thesis* **using** *True* **and** *pairHyp* **by** *blast*
next
case *False*
from *this* **have** *t* > *0* **using** *tHyp* **by** *simp*
hence (*sol a*[*xfList*$\leftarrow$*uInput*] *t*) (*vdiff x*) = *vderiv-of* (λ*s*. *u s* (*d2z a*)) {*0*<..< (*2* $*_R$ *t*)} *t*
using *tHyp xfuHyp assms state-list-cross-upd-correctInPrimes* **by** *fastforce*
also have *vderiv-of* (λ*s*. *u s* (*d2z a*)) {*0*<..< (*2* $*_R$ *t*)} *t* = *f* (*sol a*[*xfList*$\leftarrow$*uInput*] *t*)
using *keyFact xfuHyp* **and** ‹*t* > *0*› **by** *force*
ultimately show *?thesis* **by** *simp*
qed
qed

lemma *keyFact-elim*:
assumes *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *solHyp1*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf*$\in$*set xfList*.
((λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λ*t*. π_2 *xf* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))) {*0*..*t*}
shows *keyFact*:$\forall$ *st*. $\forall$ *uxf* $\in$ *set* (*uInput* $\otimes$ *xfList*).$\forall$ *t*>*0*.
vderiv-of (λ*r*. (π_1 *uxf*) *r* (*d2z st*)) {*0*<..< (*2* $*_R$ *t*)} *t* = (π_2 (π_2 *uxf*)) (*sol st*[*xfList*$\leftarrow$*uInput*] *t*)
proof(*clarify*, *rename-tac a u x f t*)
fix *a*::*real store* **and** *t*::*real* **and** *x*::*string*
and *f*::*real store* $\Rightarrow$ *real* **and** *u*::*real* $\Rightarrow$ *real store* $\Rightarrow$ *real*
assume *uxfHyp*:(*u*, *x*, *f*) $\in$ *set* (*uInput* $\otimes$ *xfList*) **and** *tHyp*:*0* < *t*
from *this* **and** *assms* **have** $\forall$ *s* > *0*. (*sol a*[*xfList*$\leftarrow$*uInput*] *s*) *x* = *u s* (*d2z a*)
using *state-list-cross-upd-uxf-on-x* **by** (*metis*)
hence *1*:$\bigwedge$*s*. *s* $\in$ {*0*<..< *2* $\cdot$ *t*} $\Longrightarrow$ (*sol a*[*xfList*$\leftarrow$*uInput*] *s*) *x* = *u s* (*d2z a*)
using *tHyp* **by** *force*
have {*0*<..<*2* $\cdot$ *t*} $\subseteq$ {*0*..*2* $\cdot$ *t*} **by** *auto*
also have $\forall$ *xf*$\in$*set xfList*. ((λ*r*. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) (π_1 *xf*))
has-vderiv-on (λ*r*. π_2 *xf* (*sol a*[*xfList*$\leftarrow$*uInput*] *r*))) {*0*..*2* $\cdot$ *t*} **using** *solHyp1* **and** *tHyp* **by** *simp*
ultimately have $\forall$ *xf*$\in$*set xfList*. ((λ*r*. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) (π_1 *xf*))
has-vderiv-on (λ*r*. π_2 *xf* (*sol a*[*xfList*$\leftarrow$*uInput*] *r*))) {*0*<..<*2* $\cdot$ *t*}
using *ODE-Auxiliarities.has-vderiv-on-subset* **by** *blast*
also from *uxfHyp* **have** *xfHyp*:(*x*,*f*) $\in$ *set xfList* **by** (*meson non-empty-crossListElim*)

ultimately have *2*:((λ*r*. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) *x*)
has-vderiv-on (λ*r*. *f* (*sol a*[*xfList*$\leftarrow$*uInput*] *r*))) {*0*<..<*2* $\cdot$ *t*}
using *has-vderiv-on-subset* **by** *auto*
have ((λ*r*. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) *x*) *has-vderiv-on* (λ*r*. *f* (*sol a*[*xfList*$\leftarrow$*uInput*] *r*))) {*0*<..<*2*$\cdot$*t*} =

((λ r. u r (d2z a)) has-vderiv-on (λr. f (sol a[xfList←uInput] r))) {0<..<2 · t}
apply(rule-tac has-vderiv-on-cong) **using** 1 **by** auto
from this **and** 2 **have** derivHyp:((λ r. u r (d2z a)) has-vderiv-on
(λr. f (sol a[xfList←uInput] r))) {0<..<2 · t} **by** simp
then have ∀ s ∈ {0<..<2 · t}. ((λ r. u r (d2z a)) has-vector-derivative
f (sol a[xfList←uInput] s)) (at s within {0<..<2 · t}) **by** (simp add: has-vderiv-on-def)
{**fix** f′ **assume** ((λs. u s (d2z a)) has-vderiv-on f′) {0<..<2 · t}
then have f′Hyp:∀ rr ∈ {0<..<2 · t}. ((λs. u s (d2z a)) has-derivative (λs. s $*_R$ (f′ rr)))
(at rr within {0<..<2 · t}) **by**(simp add: has-vderiv-on-def has-vector-derivative-def)

{**fix** rr **assume** rrHyp:rr ∈ {0 <..< 2 · t}
have boxDef:box 0 (2 · t) = {0<..<2 · t} ∧ rr ∈ box 0 (2 · t)
using tHyp rrHyp **by** auto
have rr1:((λr. u r (d2z a)) has-derivative (λs. s $*_R$ (f′ rr))) (at rr within box 0 (2 · t))
using tHyp boxDef rrHyp f′Hyp **by** force
from derivHyp **have** ∀ rr ∈ {0<..<2 · t}. ((λ s. u s (d2z a)) has-derivative
(λs. s $*_R$ (f (sol a[xfList←uInput] rr)))) (at rr within {0<..<2 · t})
by(simp add: has-vderiv-on-def has-vector-derivative-def)
hence rr2:((λ s. u s (d2z a)) has-derivative
(λs. s $*_R$ (f (sol a[xfList←uInput] rr)))) (at rr within box 0 (2 · t))**using** rrHyp boxDef **by** force
from boxDef rr1 **and** rr2 **have** (λs. s $*_R$ (f′ rr)) = (λs. s $*_R$ (f (sol a[xfList←uInput] rr)))
using frechet-derivative-unique-within-open-interval **by** blast
hence f′ rr = f (sol a[xfList←uInput] rr) **by** (metis lambda-one real-scaleR-def)}
from this **have** ∀ rr ∈ {0<..< 2 · t}. f′ rr = (f (sol a[xfList←uInput] rr)) **by** force}
then have f′Hyp:∀ f′. ((λs. u s (d2z a)) has-vderiv-on f′) {0<..<2 · t} ⟶
(∀ rr ∈ {0<..< 2 · t}. f′ rr = (f (sol a[xfList←uInput] rr))) **by** force
have ((λs. u s (d2z a)) has-vderiv-on (vderiv-of (λr. u r (d2z a)) {0<..< (2 $*_R$ t)})) {0<..<2 · t}
by(simp add: vderiv-of-def, metis derivHyp someI-ex)
from this **and** f′Hyp **have** ∀ rr ∈ {0<..< 2 · t}.
(vderiv-of (λr. u r (d2z a)) {0<..< (2 $*_R$ t)}) rr = (f (sol a[xfList←uInput] rr))
by blast
thus vderiv-of (λr. π_1 (u, x, f) r (d2z a)) {0<..<2 $*_R$ t} t =
π_2 (π_2 (u, x, f)) (sol a[xfList←uInput] t) **using** tHyp **by** force
qed

lemma conds4vdiffs:
assumes funcsHyp:∀ st g. ∀ xf∈set xfList. π_2 xf (override-on st g varDiffs) = π_2 xf st
and distinctHyp:distinct (map π_1 xfList)
and varsHyp:∀ xf ∈ set xfList. π_1 xf ∉ varDiffs
and lengthHyp:length xfList = length uInput
and solHyp1:∀ st. ∀ t≥0. ∀ xf∈set xfList. ((λt. (sol st[xfList←uInput] t) (π_1 xf))

has-vderiv-on (λ*t*. π_2 *xf* (*sol st*[*xfList*←*uInput*] *t*))) {*0*..*t*}
and *solHyp3*:∀ *st*. ∀ *uxf* ∈ *set* (*uInput* ⊗ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*)
(π_1 (π_2 *uxf*))
shows ∀ *st*. ∀ *t*≥*0*. ∀ *xf*∈*set xfList*.
(*sol st*[*xfList*←*uInput*] *t*) (*vdiff* (π_1 *xf*)) = π_2 *xf* (*sol st*[*xfList*←*uInput*] *t*)
apply(*rule prelim-conds4vdiffs*)
prefer *6* **subgoal using** *assms* **and** *keyFact-elim* **by** *blast*
using *assms* **by** *simp-all*

lemma *conds4Consts*:
assumes *varsHyp*:∀ *xf* ∈ *set xfList*. π_1 *xf* ∉ *varDiffs*
shows ∀ *str*. *str* ∉ (π_1⟦*set xfList*⟧) ⟶ (*sol a*[*xfList*←*uInput*] *t*) (*vdiff str*) = *0*
using *varsHyp* **apply**(*induction xfList uInput rule*: *cross-list.induct*)
apply(*simp-all add*: *override-on-def varDiffs-def vdiff-def*)
by *clarsimp*

lemma *conds4RestOfStrings*:
∀ *str*. *str* ∉ (π_1⟦*set xfList*⟧) ∪ *varDiffs* ⟶ (*sol a*[*xfList*←*uInput*] *t*) *str* = *a str*
apply(*induction xfList uInput rule*: *cross-list.induct*)
by(*auto simp*: *varDiffs-def*)

lemma *conds4solvesIVP*:
assumes *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:∀ *xf* ∈ *set xfList*. π_1 *xf* ∉ *varDiffs*
and *solHyp1*:∀ *st*. ∀ *t*≥*0*. ∀ *xf* ∈ *set xfList*.
((λ*t*. (*sol st*[*xfList*←*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λ*t*. π_2 *xf* (*sol st*[*xfList*←*uInput*]
t))) {*0*..*t*}
and *solHyp2*:∀ *st*. ∀ *t*≥*0*. ∀ *xf*∈*set xfList*. (λ*t*. (*sol st*[*xfList*←*uInput*] *t*) (π_1 *xf*))
∈ {*0*..*t*} → *UNIV*
and *solHyp3*:∀ *st*. ∀ *uxf*∈*set* (*uInput* ⊗ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*) (π_1
(π_2 *uxf*))
shows ∀ *st*. ∀ *t*≥*0*.∀ *xf*∈*set xfList*. ((λ*t*. (*sol st*[*xfList*←*uInput*] *t*) (π_1 *xf*))
solvesTheIVP (λ*t r*. π_2 *xf* (*sol st*[*xfList*←*uInput*] *t*)) *withInitCond* *0* ↦ *st* (π_1
xf)) {*0*..*t*} *UNIV*
apply(*rule allI*, *rule allI*, *rule impI*, *rule ballI*, *rule solves-ivpI*, *rule solves-odeI*)
subgoal using *solHyp1* **by** *simp*
subgoal using *solHyp2* **by** *simp*
proof(*clarify*, *rename-tac a t x f*)
fix *t*::*real* **and** *x*::*string* **and** *f*::*real store* ⇒ *real* **and** *a*::*real store*
assume *tHyp*:*0* ≤ *t* **and** *xfHyp*:(*x*, *f*) ∈ *set xfList*
then obtain *u* **where** *uxfHyp*:(*u*, *x*, *f*) ∈ *set* (*uInput* ⊗ *xfList*)
by (*metis crossList-map-projElim in-set-impl-in-set-zip2 lengthHyp map-fst-zip map-snd-zip*)
from *varsHyp* **have** *toZeroHyp*:(*d2z a*) *x* = *a x* **using** *override-on-def xfHyp* **by**
auto
from *uxfHyp* **and** *solHyp3* **have** *u 0* (*d2z a*) = (*d2z a*) *x* **by** *fastforce*
also have (*sol a*[*xfList*←*uInput*] *0*) (π_1 (*x*, *f*)) = *u 0* (*d2z a*)
using *state-list-cross-upd-uxf-on-x uxfHyp* **and** *assms* **by** *fastforce*
ultimately show (*sol a*[*xfList*←*uInput*] *0*) (π_1 (*x*, *f*)) = *a* (π_1 (*x*, *f*)) **using**

toZeroHyp **by** *simp*
qed

lemma *conds4storeIVP-on-toSol*:
assumes *funcsHyp*:$\forall$ *st*. $\forall$ *g*. $\forall$ *xf* $\in$ *set xfList*. π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st*
and *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *guardHyp*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. *G* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*)
and *solHyp1*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf* $\in$ *set xfList*.
((λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λ*t*. π_2 *xf* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))) {*0..t*}
and *solHyp2*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf*$\in$*set xfList*. (λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) $\in$ {*0..t*} $\rightarrow$ *UNIV*
and *solHyp3*:$\forall$ *st*. $\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*) (π_1 (π_2 *uxf*))
shows $\forall$ *st*. *solvesStoreIVP* (λ *t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*)) *xfList st G*
apply(*rule allI*, *rule solves-store-ivpI*)
subgoal using *guardHyp* **by** *simp*
subgoal using *conds4RestOfStrings* **by** *blast*
subgoal using *conds4Consts varsHyp* **by** *blast*
subgoal using *conds4vdiffs* **and** *assms* **by** *blast*
subgoal using *conds4solvesIVP* **and** *assms* **by** *blast*
done

theorem *dSolve-toSolve*:
assumes *funcsHyp*:$\forall$ *st*. $\forall$ *g*. $\forall$ *xf* $\in$ *set xfList*. π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st*
and *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *guardHyp*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. *G* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*)
and *solHyp1*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf* $\in$ *set xfList*.
((λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λ*t*. π_2 *xf* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))) {*0..t*}
and *solHyp2*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf*$\in$*set xfList*. (λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) $\in$ {*0..t*} $\rightarrow$ *UNIV*
and *solHyp3*:$\forall$ *st*. $\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*) (π_1 (π_2 *uxf*))
and *uniqHyp*:$\forall$ *st*.$\forall$ *X*. *solvesStoreIVP X xfList st G* $\longrightarrow$ ($\forall$ *t* $\geq$ *0*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) = *X t*)
and *postCondHyp*:$\forall$ *st*. *P st* $\longrightarrow$ ($\forall$ *t*$\geq$*0*. *Q* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac uInput*=*uInput* **in** *dSolve*)
subgoal using *assms* **and** *conds4storeIVP-on-toSol* **by** *simp*
subgoal by (*simp add*: *uniqHyp*)
using *postCondHyp guardHyp postCondHyp* **by** *simp*

term *unique-on-bounded-closed t0 T x0 f X L*
thm *unique-on-bounded-closed-def*
thm *unique-on-bounded-closed-axioms-def*
thm *unique-on-closed-def*

lemma *conds4UniqSol*:
assumes *sHyp*:$t \geq 0$
assumes *contHyp*:$\forall$ *xf* $\in$ *set xfList*. *continuous-on* $(\{0..t\} \times UNIV)$
$(\lambda(t, (r{::}real)).\ (\pi_2\ xf)\ (sol\ a[xfList{\leftarrow}uInput]\ t))$
shows $\forall$ *xf* $\in$ *set xfList*. *unique-on-bounded-closed* 0 $\{0..t\}$ $(a\ (\pi_1\ xf))$
$(\lambda t\ r.\ (\pi_2\ xf)\ (sol\ a[xfList{\leftarrow}uInput]\ t))$ *UNIV* $(if\ t = 0\ then\ 1\ else\ 1/(t+1))$
apply(*simp add*: *unique-on-bounded-closed-def unique-on-bounded-closed-axioms-def*

unique-on-closed-def compact-interval-def compact-interval-axioms-def nonempty-set-def

interval-def self-mapping-def self-mapping-axioms-def closed-domain-def global-lipschitz-def

lipschitz-def, *rule conjI*)
subgoal using *contHyp continuous-rhs-def* **by** *fastforce*
subgoal
  using *contHyp continuous-rhs-def sHyp* **by** *fastforce*
done

lemma *solves-store-ivp-at-beginning-overrides*:
assumes *Fsolves*:*solvesStoreIVP F xfList a G*
shows $F\ 0 = override\text{-}on\ a\ (F\ 0)\ varDiffs$
apply(*rule ext*, *subgoal-tac* $x \notin varDiffs \longrightarrow F\ 0\ x = a\ x$)
subgoal by (*simp add*: *override-on-def*)
using *assms* **and** *solves-store-ivpD*(*6*) **by** *simp*

lemma *ubcStoreUniqueSol*:
assumes *sHyp*:$s \geq 0$
assumes *contHyp*:$\forall$ *xf* $\in$ *set xfList*. *continuous-on* $(\{0..s\} \times UNIV)$
$(\lambda(t, (r{::}real)).\ (\pi_2\ xf)\ (sol\ a[xfList{\leftarrow}uInput]\ t))$
and *eqDerivs*:$\forall$ *xf* $\in$ *set xfList*. $\forall$ $t \in \{0..s\}$. $(\pi_2\ xf)\ (F\ t) = (\pi_2\ xf)\ (sol$
$a[xfList{\leftarrow}uInput]\ t)$
and *Fsolves*:*solvesStoreIVP F xfList a G*
and *solHyp*:*solvesStoreIVP* $(\lambda\ t.\ (sol\ a[xfList{\leftarrow}uInput]\ t))$ *xfList a G*
shows $(sol\ a[xfList{\leftarrow}uInput]\ s) = F\ s$
proof
  fix *str*::*string* **show** $(sol\ a[xfList{\leftarrow}uInput]\ s)\ str = F\ s\ str$
  proof(*cases* $str \in (\pi_1[\![set\ xfList]\!]) \cup varDiffs$)
  case *False*
    then have *notInVars*:$str \notin (\pi_1[\![set\ xfList]\!]) \cup varDiffs$ **by** *simp*
    from *solHyp* **have** $\forall t{\geq}0.\ \forall str.\ str \notin (\pi_1[\![set\ xfList]\!]) \cup varDiffs \longrightarrow$
    $(sol\ a[xfList{\leftarrow}uInput]\ t)\ str = a\ str$ **by** (*simp add*: *solvesStoreIVP-def*)

hence *1*:(*sol a*[*xfList*←*uInput*] *s*) *str* = *a str* **using** *sHyp notInVars* **by** *blast*
from *Fsolves* **have** ∀ *t*≥*0*. ∀ *str*. *str* ∉ (π_1⟦*set xfList*⟧) ∪ *varDiffs* ⟶ *F t str* = *a str*
by (*simp add*: *solvesStoreIVP-def*)
then have *2*:*F s str* = *a str* **using** *sHyp notInVars* **by** *blast*
thus (*sol a*[*xfList*←*uInput*] *s*) *str* = *F s str* **using** *1* **and** *2* **by** *simp*
next case *True*
then have *str* ∈ (π_1⟦*set xfList*⟧) ∨ *str* ∈ *varDiffs* **by** *simp*
moreover
{assume *str* ∈ (π_1⟦*set xfList*⟧) **from** *this* **obtain** *f*::((*char list* ⇒ *real*) ⇒ *real*) **where**
strfHyp:(*str*, *f*) ∈ *set xfList* **by** *fastforce*

from *Fsolves* **and** *sHyp* **have** (∀ *xf* ∈ *set xfList*. ((λ*t*. *F t* (π_1 *xf*)) *solvesTheIVP*
(λ*t r*. π_2 *xf* (*F t*)) *withInitCond* *0* ↦ *a* (π_1 *xf*)) {*0*..*s*} *UNIV*)
by (*simp add*: *solvesStoreIVP-def*)
then have *expand1*:∀ *xf* ∈ *set xfList*.((λ*t*. *F t* (π_1 *xf*)) *solves-ode*
(λ*t r*. (π_2 *xf*) (*F t*))){*0*..*s*} *UNIV* ∧ *F 0* (π_1 *xf*) = *a* (π_1 *xf*) **by** (*simp add*: *solves-ivp-def*)
hence *expand2*:∀ *xf* ∈ *set xfList*. ∀ *t* ∈ {*0*..*s*}. ((λ*r*. *F r* (π_1 *xf*))
has-vector-derivative (λ*r*. (π_2 *xf*) (*sol a*[*xfList*←*uInput*] *t*)) *t*) (*at t within* {*0*..*s*})
using *eqDerivs* **by** (*simp add*: *solves-ode-def has-vderiv-on-def*)

then have ∀ *xf* ∈ *set xfList*. ((λ*t*. *F t* (π_1 *xf*)) *solves-ode*
(λ*t r*. (π_2 *xf*) (*sol a*[*xfList*←*uInput*] *t*))){*0*..*s*} *UNIV* ∧ *F 0* (π_1 *xf*) = *a* (π_1 *xf*)
by (*simp add*: *has-vderiv-on-def solves-ode-def expand1 expand2*)
then have *1*:((λ*t*. *F t str*) *solvesTheIVP* (λ*t r*. *f* (*sol a*[*xfList*←*uInput*] *t*))
withInitCond *0* ↦ *a str*) {*0*..*s*} *UNIV* **using** *strfHyp solves-ivp-def* **by** *fastforce*

from *solHyp* **and** *strfHyp* **have** *2*:((λ *t*. (*sol a*[*xfList*←*uInput*] *t*) *str*)
solvesTheIVP (λ*t r*. *f* (*sol a*[*xfList*←*uInput*] *t*)) *withInitCond* *0* ↦ *a str*) {*0*..*s*} *UNIV*
using *solvesStoreIVP-def sHyp* **by** *fastforce*

from *sHyp* **and** *contHyp* **have** ∀ *xf* ∈ *set xfList*. *unique-on-bounded-closed 0* {*0*..*s*} (*a* (π_1 *xf*))
(λ*t r*. (π_2 *xf*) (*sol a*[*xfList*←*uInput*] *t*)) *UNIV* (*if s* = *0 then 1 else 1*/(*s*+*1*))

using *conds4UniqSol* **by** *simp*
from *this* **have** *3*:*unique-on-bounded-closed 0* {*0*..*s*} (*a str*) (λ*t r*. *f* (*sol a*[*xfList*←*uInput*] *t*))
UNIV (*if s* = *0 then 1 else 1*/(*s*+*1*)) **using** *strfHyp* **by** *fastforce*
from *1 2* **and** *3* **have** (*sol a*[*xfList*←*uInput*] *s*) *str* = *F s str*
using *unique-on-bounded-closed.ivp-unique-solution* **using** *real-Icc-closed-segment sHyp* **by** *blast***}**

moreover
{assume *str* ∈ *varDiffs*
then obtain *x* **where** *xDef*:*str* = *vdiff x* **by** (*auto simp*: *varDiffs-def*)
have (*sol a*[*xfList*←*uInput*] *s*) *str* = *F s str*
proof(*cases x* ∈ *set* (*map* π_1 *xfList*))
case *True*
then obtain *f* **where** *strFhyp*:(*x*, *f*) ∈ *set xfList* **by** *fastforce*
from *sHyp* **and** *Fsolves* **have** *F s str* = *f* (*F s*)
using *solves-store-ivpD*(*4*) *strFhyp xDef* **by** *force*
moreover from *solHyp* **and** *sHyp* **have** (*sol a*[*xfList*←*uInput*] *s*) *str* =
f (*sol a*[*xfList*←*uInput*] *s*) **using** *solves-store-ivpD*(*4*) *strFhyp xDef* **by**
force
ultimately show *?thesis* **using** *eqDerivs strFhyp sHyp* **by** *auto*
next
case *False*
from *this Fsolves* **and** *sHyp* **have** *F s str* = *0* **using** *xDef solves-store-ivpD*(*3*)
by *simp*
also have (*sol a*[*xfList*←*uInput*] *s*) *str* = *0*
using *False solHyp sHyp solves-store-ivpD*(*3*) *xDef* **by** *fastforce*
ultimately show *?thesis* **by** *simp*
qed}
ultimately show (*sol a*[*xfList*←*uInput*] *s*) *str* = *F s str* **by** *blast*
qed
qed

theorem *dSolveUBC*:
assumes *contHyp*:∀ *st*. ∀ *t*≥*0*. ∀ *xf* ∈ *set xfList*. *continuous-on* ({*0*..*t*} × *UNIV*)

(λ(*t*, (*r*::*real*)). (π_2 *xf*) (*sol st*[*xfList*←*uInput*] *t*))
and *solHyp*:∀ *st*. *solvesStoreIVP* (λ *t*. (*sol st*[*xfList*←*uInput*] *t*)) *xfList st G*
and *uniqHyp*:∀ *st*. ∀ *X*. *X solvesTheStoreIVP xfList withInitState st andGuard G*
⟶
(∀ *t* ≥ *0*. ∀ *xf* ∈ *set xfList*. ∀ *r* ∈ {*0*..*t*}. (π_2 *xf*) (*X r*) =
(π_2 *xf*) (*sol st*[*xfList*←*uInput*] *r*))
and *diffAssgn*: ∀ *st*. *P st* ⟶ (∀ *t*≥*0*. *G* (*sol st*[*xfList*←*uInput*] *t*) ⟶ *Q* (*sol*
st[*xfList*←*uInput*] *t*))
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac uInput*=*uInput* **in** *dSolve*)
subgoal using *solHyp* **by** *simp*
subgoal proof(*clarify*)
fix *a*::*real store* **and** *X*::*real* ⇒ *real store* **and** *s*::*real*
assume *XisSol*:*solvesStoreIVP X xfList a G* **and** *sHyp*:*0* ≤ *s*
from *this* **and** *uniqHyp* **have** ∀ *xf* ∈ *set xfList*. ∀ *t* ∈ {*0*..*s*}.
(π_2 *xf*) (*X t*) = (π_2 *xf*) (*sol a*[*xfList*←*uInput*] *t*) **by** *auto*
moreover have ∀ *xf* ∈ *set xfList*. *continuous-on* ({*0*..*s*} × *UNIV*)
(λ(*t*, (*r*::*real*)). (π_2 *xf*) (*sol a*[*xfList*←*uInput*] *t*)) **using** *contHyp sHyp* **by** *blast*
ultimately show (*sol a*[*xfList*←*uInput*] *s*) = *X s*
using *sHyp XisSol ubcStoreUniqueSol solHyp* **by** *simp*
qed

subgoal using *diffAssgn* **by** *simp*
done

theorem *dSolve-toSolveUBC*:
assumes *funcsHyp*:$\forall$ *st*. $\forall$ *g*. $\forall$ *xf* $\in$ *set xfList*. π_2 *xf* (*override-on st g varDiffs*) $= \pi_2$ *xf st*
and *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *guardHyp*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. *G* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*)
and *solHyp1*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf* $\in$ *set xfList*.
((λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λ*t*. π_2 *xf* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))) {*0*..*t*}
and *solHyp2*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf*$\in$*set xfList*. (λ*t*. (*sol st*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) $\in$ {*0*−−*t*} $\rightarrow$ *UNIV*
and *solHyp3*:$\forall$ *st*. $\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*d2z st*) = (*d2z st*) (π_1 (π_2 *uxf*))
and *contHyp*:$\forall$ *st*. $\forall$ *t*$\geq$*0*. $\forall$ *xf* $\in$ *set xfList*. *continuous-on* ({*0*..*t*} $\times$ *UNIV*) (λ(*t*, (*r*::*real*)). (π_2 *xf*) (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))
and *uniqHyp*:$\forall$ *st*. $\forall$ *X*. *solvesStoreIVP X xfList st G* $\longrightarrow$
($\forall$ *t* $\geq$ *0*. $\forall$ *xf* $\in$ *set xfList*. $\forall$ *r* $\in$ {*0*..*t*}. (π_2 *xf*) (*X r*) = (π_2 *xf*) (*sol st*[*xfList*$\leftarrow$*uInput*] *r*))
and *postCondHyp*:$\forall$ *st*. *P st* $\longrightarrow$ ($\forall$ *t*$\geq$*0*. *Q* (*sol st*[*xfList*$\leftarrow$*uInput*] *t*))
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac uInput*=*uInput* **in** *dSolveUBC*)
subgoal using *contHyp* **by** *simp*
subgoal
  apply(*rule-tac uInput*=*uInput* **in** *conds4storeIVP-on-toSol*)
  using *assms* **by** *auto*
subgoal using *uniqHyp* **by** *simp*
using *postCondHyp* **by** *simp*

thm *derivative-intros*(*173*)
thm *derivative-intros*
thm *derivative-intros*(*176*)
thm *derivative-eq-intros*(*8*)
thm *derivative-eq-intros*(*17*)
thm *derivative-eq-intros*(*6*)
thm *derivative-eq-intros*(*15*)
thm *derivative-eq-intros*
thm *continuous-intros*

lemma *PRE* (λ *s*. *s* ''*station*'' < *s* ''*pos*'' $\wedge$ *s* ''*vel*'' > *0*)
  (*ODEsystem* [(''*pos*'',(λ *s*. *s* ''*vel*''))] *with* (λ *s*. *True*))
  POST (λ *s*. (*s* ''*station*'' < *s* ''*pos*''))
apply(*rule-tac uInput*=[λ *t s*. *s* ''*vel*'' $\cdot$ *t* + *s* ''*pos*''] **in** *dSolve-toSolveUBC*)
prefer *11* **subgoal by**(*simp add*: *wp-trafo vdiff-def add-strict-increasing2*)

apply(*simp-all add*: *vdiff-def varDiffs-def*)
subgoal
  apply(*clarify*)
  apply(*rule-tac* $f'1=\lambda$ *x. st ''vel''* **and** $g'1=\lambda$ *x. 0* **in** *derivative-intros*(*173*))
  apply(*rule-tac* $f'1=\lambda$ *x.0* **and** $g'1=\lambda$ *x.1* **in** *derivative-intros*(*176*))
  by(*auto intro*: *derivative-intros*)
subgoal by(*clarify*, *rule continuous-intros*)
subgoal by(*simp add*: *solvesStoreIVP-def vdiff-def varDiffs-def*)
done

— Differential Invariant.

datatype *trms* = *Const real* | *Var string* | *Mns trms* | *Sum trms trms* | *Mult trms trms*

primrec *rval* ::*trms* ⇒ (*real store* ⇒ *real*) **where**
rval (*Const r*) = (λ *s. r*)|
rval (*Var x*) = (λ *s. s x*)|
rval (*Mns* ϑ) = (λ *s.* − (*rval* ϑ *s*))|
rval (*Sum* ϑ η) = (λ *s. rval* ϑ *s* + *rval* η *s*)|
rval (*Mult* ϑ η) = (λ *s. rval* ϑ *s* · *rval* η *s*)

datatype *props* = *Eq trms trms* | *Less trms trms* | *Leq trms trms* | *And props props* | *Or props props*

primrec *pval* ::*props* ⇒ (*real store* ⇒ *bool*) **where**
pval (*Eq* ϑ η) = (λ *s.* (*rval* ϑ) *s* = (*rval* η) *s*)|
pval (*Less* ϑ η) = (λ *s.* (*rval* ϑ) *s* < (*rval* η) *s*)|
pval (*Leq* ϑ η) = (λ *s.* (*rval* ϑ) *s* ≤ (*rval* η) *s*)|
pval (*And* φ ψ) = (λ *s.* (*pval* φ) *s* ∧ (*pval* ψ) *s*)|
pval (*Or* φ ψ) = (λ *s.* (*pval* φ) *s* ∨ (*pval* ψ) *s*)

primrec *rdiff* ::*trms* ⇒ *trms* **where**
rdiff (*Const r*) = *Const 0*|
rdiff (*Var x*) = *Var* (*vdiff x*)|
rdiff (*Mns* ϑ) = *Mns* (*rdiff* ϑ)|
rdiff (*Sum* ϑ η) = *Sum* (*rdiff* ϑ) (*rdiff* η)|
rdiff (*Mult* ϑ η) = *Sum* (*Mult* (*rdiff* ϑ) η) (*Mult* ϑ (*rdiff* η))

primrec *pdiff* ::*props* ⇒ *props* **where**
pdiff (*Eq* ϑ η) = *Eq* (*rdiff* ϑ) (*rdiff* η)|
pdiff (*Less* ϑ η) = *Leq* (*rdiff* ϑ) (*rdiff* η)|
pdiff (*Leq* ϑ η) = *Leq* (*rdiff* ϑ) (*rdiff* η)|
pdiff (*And* φ ψ) = *And* (*pdiff* φ) (*pdiff* ψ)|
pdiff (*Or* φ ψ) = *And* (*pdiff* φ) (*pdiff* ψ)

lemma *solvesStoreIVP-couldBeModified*:
fixes *F*::*real* ⇒ *real store*
assumes *storeIVP-vars*:∀ *t* ≥ *0*. ∀ *xf* ∈ *set xfList*. ((λ *t. F t* (π_1 *xf*))

solvesTheIVP (λ *t*. λ *r*. (π_2 *xf*) (*F t*)) *withInitCond 0* $\mapsto$ (*a* (π_1 *xf*))) {*0*..*t*} *UNIV*
and *storeIVP-dvars*:$\forall$ *t* $\geq$ *0*. $\forall$ *xf*$\in$*set xfList*. (*F t* (*vdiff* (π_1 *xf*))) = (π_2 *xf*) (*F t*)
shows $\forall$ *t* $\geq$ *0*. $\forall$ *r*$\in${*0*..*t*}. $\forall$ *xf* $\in$ *set xfList*.
((λ *t*. *F t* (π_1 *xf*)) *has-vector-derivative F r* (*vdiff* (π_1 *xf*))) (*at r within* {*0*..*t*})
proof(*clarify*, *rename-tac t r x f*)
fix *x f* **and** *t r*::*real*
assume *tHyp*:*0* $\leq$ *t* **and** *xfHyp*:(*x*, *f*) $\in$ *set xfList* **and** *rHyp*:*r* $\in$ {*0*..*t*}
from *this* **and** *storeIVP-vars* **have** ((λ *t*. *F t x*) *solvesTheIVP* (λ *t*. λ *r*. *f* (*F t*)) *withInitCond 0* $\mapsto$ (*a x*)) {*0*..*t*} *UNIV* **using** *tHyp* **by** *fastforce*
then have ((λ *t*. *F t x*) *has-vderiv-on* (λ *t*. *f* (*F t*))) {*0*..*t*}
by (*simp add*: *solves-ode-def solves-ivp-def*)
thm *has-vderiv-on-def*
hence $*$:$\forall$ *r*$\in${*0*..*t*}. ((λ *t*. *F t x*) *has-vector-derivative* (λ *t*. *f* (*F t*)) *r*) (*at r within* {*0*..*t*})
by (*simp add*: *has-vderiv-on-def tHyp*)
have $\forall$ *t* $\geq$ *0*. $\forall$ *r*$\in${*0*..*t*}. $\forall$ *xf* $\in$ *set xfList*. (*F r* (*vdiff* (π_1 *xf*))) = (π_2 *xf*) (*F r*)
using *assms* **by** *auto*
from *this rHyp* **and** *xfHyp* **have** (*F r* (*vdiff x*)) = *f* (*F r*) **by** *force*
then show ((λ*t*. *F t* (π_1 (*x*, *f*))) *has-vector-derivative*
F r (*vdiff* (π_1 (*x*, *f*)))) (*at r within* {*0*..*t*}) **using** $*$ *rHyp* **by** *auto*
qed

lemma *derivationLemma-baseCase*:
fixes *F*::*real* $\Rightarrow$ *real store*
assumes *solves*:*solvesStoreIVP F xfList a G*
shows $\forall$ *x* $\in$ (*UNIV* $-$ *varDiffs*). $\forall$ *t* $\geq$ *0*. $\forall$ *r*$\in${*0*..*t*}.
((λ *t*. *F t x*) *has-vector-derivative F r* (*vdiff x*)) (*at r within* {*0*..*t*})
proof
fix *x*
assume *x* $\in$ *UNIV* $-$ *varDiffs*
then have *notVarDiff*:$\forall$ *z*. *x* $\neq$ *vdiff z* **using** *varDiffs-def* **by** *fastforce*
  show $\forall$ *t*$\geq$*0*. $\forall$ *r*$\in${*0*..*t*}. ((λ*t*. *F t x*) *has-vector-derivative F r* (*vdiff x*)) (*at r within* {*0*..*t*})
  proof(*cases x* $\in$ *set* (*map* π_1 *xfList*))
    case *True*
    from *this* **and** *solves* **have** $\forall$ *t* $\geq$ *0*. $\forall$ *r*$\in${*0*..*t*}. $\forall$ *xf* $\in$ *set xfList*.
    ((λ *t*. *F t* (π_1 *xf*)) *has-vector-derivative F r* (*vdiff* (π_1 *xf*))) (*at r within* {*0*..*t*})
    apply(*rule-tac a*=*a* **in** *solvesStoreIVP-couldBeModified*) **using** *solves solves-store-ivpD* **by** *auto*
    from *this* **show** *?thesis* **using** *True* **by** *auto*
  next
    case *False*
    from *this notVarDiff* **and** *solves* **have** *const*:$\forall$ *t* $\geq$ *0*. *F t x* = *a x*
    using *solves-store-ivpD*(*2*) **by** (*simp add*: *varDiffs-def*)
    have *constD*:$\forall$ *t* $\geq$ *0*. $\forall$ *r*$\in${*0*..*t*}. ((λ *r*. *a x*) *has-vector-derivative 0*) (*at r within* {*0*..*t*})
    by (*auto intro*: *derivative-eq-intros*)
    {**fix** *t r*::*real*

assume $t{\geq}0$ **and** $r \in \{0..t\}$
hence $((\lambda\ s.\ a\ x)$ *has-vector-derivative* $0)$ $(at\ r\ within\ \{0..t\})$ **by** $(simp\ add$: $constD)$
moreover have $\bigwedge s.\ s \in \{0..t\} \Longrightarrow (\lambda\ r.\ F\ r\ x)\ s = (\lambda\ r.\ a\ x)\ s$
using *const* **by** $(simp\ add$: $\langle 0 \leq t\rangle)$
ultimately have $((\lambda\ s.\ F\ s\ x)$ *has-vector-derivative* $0)$ $(at\ r\ within\ \{0..t\})$
using *has-vector-derivative-imp* **by** $(metis\ \langle r \in \{0..t\}\rangle)$**}**
hence $isZero$:$\forall\, t{\geq}0.\ \forall\, r{\in}\{0..t\}.((\lambda\ t.\ F\ t\ x)$ *has-vector-derivative* $0)$ $(at\ r\ within\ \{0..t\})$ **by** *blast*
from *False solves* **and** *notVarDiff* **have** $\forall\ t \geq 0.\ F\ t\ (vdiff\ x) = 0$
using *solves-store-ivpD*(3) **by** *simp*
then show *?thesis* **using** *isZero* **by** *simp*
qed
qed

primrec *trmVars* :: *trms* $\Rightarrow$ *string set* **where**
$trmVars\ (Const\ r) = \{\}|$
$trmVars\ (Var\ x) = \{x\}|$
$trmVars\ (Mns\ \vartheta) = trmVars\ \vartheta|$
$trmVars\ (Sum\ \vartheta\ \eta) = trmVars\ \vartheta \cup trmVars\ \eta|$
$trmVars\ (Mult\ \vartheta\ \eta) = trmVars\ \vartheta \cup trmVars\ \eta$

lemma *derivationLemma*:
assumes *solvesStoreIVP F xfList a G*
and $tHyp$:$t \geq 0$
and $termVarsHyp$:$\forall\ x \in trmVars\ \eta.\ x \in (UNIV - varDiffs)$
shows $\forall\, r{\in}\{0..t\}.\ ((\lambda\ s.\ (rval\ \eta)\ (F\ s))$
has-vector-derivative $(rval\ (rdiff\ \eta))\ (F\ r))\ (at\ r\ within\ \{0..t\})$
using *termVarsHyp* **proof**$(induction\ \eta)$
case $(Const\ r)$
then show *?case* **by** *simp*
next
case $(Var\ y)$
then have $yHyp$:$y \in UNIV - varDiffs$ **by** *auto*
from *this tHyp* **and** *assms*(1) **show** *?case*
using *derivationLemma-baseCase* **by** *auto*
next
case $(Mns\ \eta)$
then show *?case*
apply$(clarsimp)$
by$(rule\ derivative\text{-}intros,\ simp)$
next
case $(Sum\ \eta 1\ \eta 2)$
then show *?case*
apply$(clarsimp)$
by$(rule\ derivative\text{-}intros,\ simp\text{-}all)$
next
case $(Mult\ \eta 1\ \eta 2)$
then show *?case*

apply(*clarsimp*)
apply(*subgoal-tac* (($\lambda s.$ *rval* $\eta 1$ (*F s*) $*_R$ *rval* $\eta 2$ (*F s*)) *has-vector-derivative* *rval* (*rdiff* $\eta 1$) (*F r*) $\cdot$ *rval* $\eta 2$ (*F r*) + *rval* $\eta 1$ (*F r*) $\cdot$ *rval* (*rdiff* $\eta 2$) (*F r*)) (*at r within* {*0..t*}), *simp*)
apply(*rule-tac* $f'1$=*rval* (*rdiff* $\eta 1$) (*F r*) **and** $g'1$=*rval* (*rdiff* $\eta 2$) (*F r*) **in** *derivative-eq-intros*(*25*))
by (*simp-all add*: *has-field-derivative-iff-has-vector-derivative*)
qed

fun *substList* ::(*string* $\times$ *trms*) *list* $\Rightarrow$ *trms* $\Rightarrow$ *trms* **where**
substList xTrmList (*Const r*) = *Const r*|
substList [] (*Var x*) = *Var x*|
substList ((*y*,ξ) # *xTrmTail*) (*Var x*) = (*if x = y then* ξ *else substList xTrmTail* (*Var x*))|
substList xTrmList (*Mns* ϑ) = *Mns* (*substList xTrmList* ϑ)|
substList xTrmList (*Sum* ϑ η) = (*Sum* (*substList xTrmList* ϑ) (*substList xTrmList* η))|
substList xTrmList (*Mult* ϑ η) = (*Mult* (*substList xTrmList* ϑ) (*substList xTrmList* η))

lemma *substList-on-compl-of-varDiffs*:
assumes *trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
assumes *set* (*map* π_1 *xTrmList*) $\subseteq$ *varDiffs*
shows *substList xTrmList* $\eta = \eta$
using *assms* **apply**(*induction* η, *simp-all add*: *varDiffs-def*)
by(*induction xTrmList*, *auto*)

lemma *substList-help1*:*set* (*map* π_1 ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*)) $\subseteq$ *varDiffs*
apply(*induction xfList uInput rule*: *cross-list.induct*, *simp-all add*: *varDiffs-def*)
by *auto*

lemma *substList-help2*:
assumes *trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
shows *substList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) $\eta = \eta$
using *assms substList-help1 substList-on-compl-of-varDiffs* **by** *blast*

lemma *substList-cross-vdiff-on-non-ocurring-var*:
assumes $x \notin$ *set list1*
shows *substList* ((*map vdiff list1*) $\otimes$ *list2*) (*Var* (*vdiff x*))
= *Var* (*vdiff x*)
using *assms* **apply**(*induction list1 list2 rule*: *cross-list.induct*, *simp*, *simp*, *clarsimp*)
by(*simp add*: *vdiff-inj*)

lemma *diff-subst-prprty-4terms*:
assumes *solves*:$\forall$ *xf* $\in$ *set xfList*. *F t* (*vdiff* (π_1 *xf*)) = π_2 *xf* (*F t*)
and *tHyp*:(*t*::*real*) $\geq$ *0*
and *listsHyp*:*map* π_2 *xfList* = *map rval uInput*

and *termVarsHyp*:*trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
shows *rval* (*rdiff* η) (*F t*) =
rval (*substList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*rdiff* η)) (*F t*)
using *termVarsHyp* **apply**(*induction* η) **apply**(*simp-all add*: *substList-help2*)
using *listsHyp* **and** *solves* **apply**(*induction xfList uInput rule*: *cross-list.induct*, *simp*, *simp*)
proof(*clarify*, *rename-tac y g xfTail* ϑ *trmTail x*)
fix *x y*::*string* **and** ϑ::*trms* **and** *g* **and** *xfTail*::((*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list*)
and *trmTail*
assume *IH*:$\bigwedge x$. $x \notin$ *varDiffs* $\Longrightarrow$ *map* π_2 *xfTail* = *map rval trmTail* $\Longrightarrow$
$\forall$ *xf*$\in$*set xfTail*. *F t* (*vdiff* (π_1 *xf*)) = π_2 *xf* (*F t*) $\Longrightarrow$
F t (*vdiff x*) = *rval* (*substList* (*map* (*vdiff* $\circ$ π_1) *xfTail* $\otimes$ *trmTail*) (*Var* (*vdiff x*))) (*F t*)
and *1*:$x \notin$ *varDiffs* **and** *2*:*map* π_2 ((*y*, *g*) # *xfTail*) = *map rval* (ϑ # *trmTail*)
and *3*:$\forall$ *xf*$\in$*set* ((*y*, *g*) # *xfTail*). *F t* (*vdiff* (π_1 *xf*)) = π_2 *xf* (*F t*)
hence $*$:*rval* (*substList* ((*map* (*vdiff* $\circ$ π_1) *xfTail*) $\otimes$ *trmTail*) (*Var* (*vdiff x*))) (*F t*) =
F t (*vdiff x*) **using** *tHyp* **by** *auto*
show *F t* (*vdiff x*) =
rval (*substList* ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*)) (*Var* (*vdiff x*))) (*F t*)
  proof(*cases* $x \in$ *set* (*map* π_1 ((*y*, *g*) # *xfTail*)))
    case *True*
    then have $x = y \vee (x \neq y \wedge x \in$ *set* (*map* π_1 *xfTail*)) **by** *auto*
    moreover
    {**assume** $x = y$
      from *this* **have** *substList* ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*))
      (*Var* (*vdiff x*)) = ϑ **by** *simp*
      also from *3 tHyp* **have** *F t* (*vdiff y*) = *g* (*F t*) **by** *simp*
      moreover from *2* **have** *rval* ϑ (*F t*) = *g* (*F t*) **by** *simp*
      ultimately have *?thesis* **by** (*simp add*: ‹$x = y$›)}
    moreover
    {**assume** $x \neq y \wedge x \in$ *set* (*map* π_1 *xfTail*)
      then have *vdiff x* $\neq$ *vdiff y* **using** *vdiff-inj* **by** *auto*
      from *this* **have** *substList* ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*))
      (*Var* (*vdiff x*)) = *substList* ((*map* (*vdiff* $\circ$ π_1) *xfTail*) $\otimes$ *trmTail*) (*Var* (*vdiff x*))
      by *simp*
      hence *?thesis* **using** $*$ **by** *simp*}
    ultimately show *?thesis* **by** *blast*
  next
    case *False*
    then have *substList* ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*)) (*Var* (*vdiff x*))
    = *Var* (*vdiff x*) **using** *substList-cross-vdiff-on-non-ocurring-var*
    by (*metis* (*no-types*, *lifting*) *List.map.compositionality*)
    thus *?thesis* **by** *simp*

qed
qed

lemma *eqInVars-impl-eqInTrms*:
assumes *termVarsHyp*:*trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
and *initHyp*:$\forall x.\ x \notin$ *varDiffs* $\longrightarrow b\ x = a\ x$
shows (*rval* η) a = (*rval* η) b
using *assms* **by**(*induction* η, *simp-all*)

lemma *non-empty-funList-implies-non-empty-trmList*:
shows $\forall$ *list*. $(x,f) \in$ *set list* $\wedge$ *map* π_2 *list* = *map rval tList* $\longrightarrow$
($\exists\ \vartheta$. *rval* $\vartheta = f \wedge \vartheta \in$ *set tList*)
by(*induction tList*, *auto*)

lemma *dInvForTrms-prelim*:
assumes *substHyp*:
$\forall$ *st*. *G st* $\longrightarrow$ ($\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *st* (*vdiff str*) = *0*) $\longrightarrow$
rval (*substList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*rdiff* η)) *st* = *0*
and *termVarsHyp*:*trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
and *listsHyp*:*map* π_2 *xfList* = *map rval uInput*
shows (*rval* η) a = *0* $\longrightarrow$ ($\forall$ c. $(a,c) \in$ (*ODEsystem xfList with G*) $\longrightarrow$ (*rval* η)
c = *0*)
proof(*clarify*)
fix c **assume** *aHyp*:(*rval* η) a = *0* **and** *cHyp*:$(a, c) \in$ *ODEsystem xfList with G*
from *this* **obtain** *t*::*real* **and** *F*::*real* $\Rightarrow$ *real store*
where *tcHyp*:$t \geq 0 \wedge F\ t = c \wedge$ *solvesStoreIVP F xfList a G* **using** *guarDiffEqtn-def*
by *auto*
then have $\forall x.\ x \notin$ *varDiffs* $\longrightarrow F\ 0\ x = a\ x$ **using** *solves-store-ivpD*(*6*) **by** *blast*
from *this* **have** *rval* η a = *rval* η (*F 0*) **using** *termVarsHyp eqInVars-impl-eqInTrms*
by *blast*
hence *obs1*:*rval* η (*F 0*) = *0* **using** *aHyp tcHyp* **by** *simp*
from *tcHyp* **have** *obs2*:$\forall r \in \{0..t\}$. ((λs. *rval* η (*F s*)) *has-vector-derivative*
rval (*rdiff* η) (*F r*)) (*at r within* $\{0..t\}$) **using** *derivationLemma termVarsHyp* **by**
blast
have $\forall r \in \{0..t\}$. $\forall$ *xf* $\in$ *set xfList*. *F r* (*vdiff* (π_1 *xf*)) = π_2 *xf* (*F r*)
using *tcHyp solves-store-ivpD*(*4*) **by** *fastforce*
from *this* **and** *tcHyp* **have** $\forall r \in \{0..t\}$. *rval* (*rdiff* η) (*F r*) =
rval (*substList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*rdiff* η)) (*F r*)
using *diff-subst-prprty-4terms termVarsHyp listsHyp* **by** *fastforce*
also from *substHyp* **have** $\forall r \in \{0..t\}$.
rval (*substList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*rdiff* η)) (*F r*) = *0*
using *solves-store-ivpD*(*1*) *solves-store-ivpD*(*3*) *tcHyp* **by** *fastforce*
ultimately have $\forall r \in \{0..t\}$. ((λs. *rval* η (*F s*)) *has-vector-derivative 0*) (*at r*
within $\{0..t\}$)
using *obs2* **by** *auto*
from *this* **and** *tcHyp* **have** $\forall s \in \{0..t\}$. ((λx. *rval* η (*F x*)) *has-derivative* (λx. x
$*_R$ *0*))
(*at s within* $\{0..t\}$) **by** (*metis has-vector-derivative-def*)
hence *rval* η (*F t*) $-$ *rval* η (*F 0*) = (λx. $x *_R$ *0*) ($t - 0$)

using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then show *rval* η *c* = *0* **using** *obs1 tcHyp* **by** *auto*
qed

theorem *dInvForTrms*:
assumes $\forall$ *st*. *G st* $\longrightarrow$ ($\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *st* (*vdiff str*) = *0*) $\longrightarrow$
rval (*substList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*rdiff* η)) *st* = *0*
and *termVarsHyp*:*trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
and *listsHyp*:*map* π_2 *xfList* = *map rval uInput*
and *eta-f*:*f* = *rval* η
shows *PRE* (λ *s*. *f s* = *0*) (*ODEsystem xfList with G*) *POST* (λ *s*. *f s* = *0*)
using *eta-f* **proof**(*clarsimp*)
fix *a b*
assume (*a*, *b*) $\in$ $\lceil\lambda s.$ *rval* η *s* = *0*$\rceil$ **and** *f* = *rval* η
from *this* **have** *aHyp*:*a* = *b* $\wedge$ *rval* η *a* = *0* **by** (*metis* (*full-types*) *d-p2r rdom-p2r-contents*)
have (*rval* η) *a* = *0* $\longrightarrow$ ($\forall$ *c*. (*a*,*c*) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ (*rval* η) *c*
= *0*)
using *assms dInvForTrms-prelim* **by** *metis*
from *this* **and** *aHyp* **have** $\forall$ *c*. (*a*,*c*) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ (*rval* η)
c = *0* **by** *blast*
thus (*a*, *b*) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil\lambda s.$ *rval* η *s* = *0*$\rceil$
using *aHyp* **by** (*simp add*: *boxProgrPred-chrctrztn*)
qed

lemma *circular-motion*:
PRE (λ *s*. (*s* ''*x*'') $\cdot$ (*s* ''*x*'') + (*s* ''*y*'') $\cdot$ (*s* ''*y*'') $-$ (*s* ''*r*'') $\cdot$ (*s* ''*r*'') = *0*)
(*ODEsystem* [(''*x*'',(λ *s*. *s* ''*y*'')),(''*y*'',(λ *s*. $-$ *s* ''*x*''))] *with G*)
POST (λ *s*. (*s* ''*x*'') $\cdot$ (*s* ''*x*'') + (*s* ''*y*'') $\cdot$ (*s* ''*y*'') $-$ (*s* ''*r*'') $\cdot$ (*s* ''*r*'') = *0*)
apply(*rule-tac* η=*Sum* (*Sum* (*Mult* (*Var* ''*x*'') (*Var* ''*x*'')) (*Mult* (*Var* ''*y*'') (*Var*
''*y*'')))
(*Mns* (*Mult* (*Var* ''*r*'') (*Var* ''*r*''))) **and** *uInput*=[*Var* ''*y*'', *Mns* (*Var* ''*x*'')]**in**
dInvForTrms)
apply(*simp-all add*: *vdiff-def varDiffs-def*)
apply(*clarsimp*, *erule-tac x*=''*r*'' **in** *allE*)
by *simp*

primrec *propVars* :: *props* $\Rightarrow$ *string set* **where**
propVars (*Eq* ϑ η) = *trmVars* ϑ $\cup$ *trmVars* η|
propVars (*Less* ϑ η) = *trmVars* ϑ $\cup$ *trmVars* η|
propVars (*Leq* ϑ η) = *trmVars* ϑ $\cup$ *trmVars* η|
propVars (*And* φ ψ) = *propVars* φ $\cup$ *propVars* ψ|
propVars (*Or* φ ψ) = *propVars* φ $\cup$ *propVars* ψ

primrec *subspList* :: (*string* $\times$ *trms*) *list* $\Rightarrow$ *props* $\Rightarrow$ *props* **where**
subspList xTrmList (*Eq* ϑ η) = (*Eq* (*substList xTrmList* ϑ) (*substList xTrmList*
η))|
subspList xTrmList (*Less* ϑ η) = (*Less* (*substList xTrmList* ϑ) (*substList xTrmList*
η))|
subspList xTrmList (*Leq* ϑ η) = (*Leq* (*substList xTrmList* ϑ) (*substList xTrmList*

$\eta))|$
$subspList\ xTrmList\ (And\ \varphi\ \psi) = (And\ (subspList\ xTrmList\ \varphi)\ (subspList\ xTrmList\ \psi))|$
$subspList\ xTrmList\ (Or\ \varphi\ \psi) = (Or\ (subspList\ xTrmList\ \varphi)\ (subspList\ xTrmList\ \psi))$

lemma *diff-subst-prprty-4props*:
assumes $solves{:}\forall\ xf \in set\ xfList.\ F\ t\ (vdiff\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t)$
and $tHyp{:}t \geq 0$
and $listsHyp{:}map\ \pi_2\ xfList = map\ rval\ uInput$
and $propVarsHyp{:}propVars\ \varphi \subseteq (UNIV - varDiffs)$
shows $pval\ (pdiff\ \varphi)\ (F\ t) =$
$pval\ (subspList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (pdiff\ \varphi))\ (F\ t)$
using *propVarsHyp* **apply**(*induction* φ, *simp-all*)
using *assms diff-subst-prprty-4terms* **apply** *fastforce*
using *assms diff-subst-prprty-4terms* **apply** *fastforce*
using *assms diff-subst-prprty-4terms* **by** *fastforce*

lemma *dInvForProps-prelim*:
assumes *substHyp*:
$\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1[\![set\ xfList]\!]) \longrightarrow st\ (vdiff\ str) = 0) \longrightarrow$
$rval\ (substList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (rdiff\ \eta))\ st \geq 0$
and $termVarsHyp{:}trmVars\ \eta \subseteq (UNIV - varDiffs)$
and $listsHyp{:}map\ \pi_2\ xfList = map\ rval\ uInput$
shows $(rval\ \eta)\ a > 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow (rval\ \eta)\ c > 0)$
and $(rval\ \eta)\ a \geq 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow (rval\ \eta)\ c \geq 0)$
proof(*clarify*)
fix c **assume** $aHyp{:}(rval\ \eta)\ a > 0$ **and** $cHyp{:}(a,\ c) \in ODEsystem\ xfList\ with\ G$
from *this* **obtain** $t{::}real$ **and** $F{::}real \Rightarrow real\ store$
where $tcHyp{:}t{\geq}0 \wedge F\ t = c \wedge solvesStoreIVP\ F\ xfList\ a\ G$ **using** *guarDiffEqtn-def* **by** *auto*
then have $\forall x.\ x \notin varDiffs \longrightarrow F\ 0\ x = a\ x$ **using** *solves-store-ivpD*(*6*) **by** *blast*
from *this* **have** $rval\ \eta\ a = rval\ \eta\ (F\ 0)$ **using** *termVarsHyp eqInVars-impl-eqInTrms* **by** *blast*
hence $obs1{:}rval\ \eta\ (F\ 0) > 0$ **using** *aHyp tcHyp* **by** *simp*
from *tcHyp* **have** $obs2{:}\forall r{\in}\{0..t\}.\ ((\lambda s.\ rval\ \eta\ (F\ s))\ has\text{-}vector\text{-}derivative$
$rval\ (rdiff\ \eta)\ (F\ r))\ (at\ r\ within\ \{0..t\})$ **using** *derivationLemma termVarsHyp* **by** *blast*
have $(\forall t{\geq}0.\ \forall\ xf \in set\ xfList.\ F\ t\ (vdiff\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t))$
using *tcHyp solves-store-ivpD*(*4*) **by** *blast*
from *this* **and** *tcHyp* **have** $\forall r{\in}\{0..t\}.\ rval\ (rdiff\ \eta)\ (F\ r) =$
$rval\ (substList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (rdiff\ \eta))\ (F\ r)$
using *diff-subst-prprty-4terms termVarsHyp listsHyp* **by** *fastforce*
also from *substHyp* **have** $\forall r{\in}\{0..t\}.$
$rval\ (substList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (rdiff\ \eta))\ (F\ r) \geq 0$
using *solves-store-ivpD*(*1*) *solves-store-ivpD*(*3*) *tcHyp* **by** (*metis atLeastAtMost-iff*)
ultimately have $*{:}\forall r{\in}\{0..t\}.\ rval\ (rdiff\ \eta)\ (F\ r) \geq 0$ **by** (*simp*)

from *obs2* **and** *tcHyp* **have** $\forall\, r{\in}\{0..t\}.\ ((\lambda s.\ rval\ \eta\ (F\ s))$ *has-derivative* $(\lambda x.\ x *_R (rval\ (rdiff\ \eta)\ (F\ r))))\ (at\ r\ within\ \{0..t\})$ **by** (*simp add*: *has-vector-derivative-def*)

hence $\exists\, r{\in}\{0..t\}.\ rval\ \eta\ (F\ t) - rval\ \eta\ (F\ 0) = t \cdot (rval\ (rdiff\ \eta)\ (F\ r))$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then obtain r **where** $rval\ (rdiff\ \eta)\ (F\ r) \geq 0 \wedge 0 \leq r \wedge r \leq t \wedge rval\ (rdiff\ \eta)$
$(F\ t) \geq 0$
$\wedge\ rval\ \eta\ (F\ t) - rval\ \eta\ (F\ 0) = t \cdot (rval\ (rdiff\ \eta)\ (F\ r))$
using $*$ *tcHyp* **by** *fastforce*
thus $rval\ \eta\ c > 0$
using *obs1 tcHyp* **by** (*metis cancel-comm-monoid-add-class.diff-cancel diff-ge-0-iff-ge*

diff-strict-mono linorder-neqE-linordered-idom linordered-field-class.sign-simps(*45*)
not-le)
next
show $0 \leq rval\ \eta\ a \longrightarrow (\forall\, c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow 0 \leq rval\ \eta$
$c)$
proof(*clarify*)
fix c **assume** *aHyp*:$(rval\ \eta)\ a \geq 0$ **and** *cHyp*:$(a,\ c) \in ODEsystem\ xfList\ with\ G$
from *this* **obtain** t::*real* **and** F::*real* $\Rightarrow$ *real store*
where *tcHyp*:$t{\geq}0 \wedge F\ t = c \wedge solvesStoreIVP\ F\ xfList\ a\ G$ **using** *guarDiffEqtn-def*
by *auto*
then have $\forall\, x.\ x \notin varDiffs \longrightarrow F\ 0\ x = a\ x$ **using** *solves-store-ivpD*(*6*) **by** *blast*
from *this* **have** $rval\ \eta\ a = rval\ \eta\ (F\ 0)$ **using** *termVarsHyp eqInVars-impl-eqInTrms*
by *blast*
hence *obs1*:$rval\ \eta\ (F\ 0) \geq 0$ **using** *aHyp tcHyp* **by** *simp*
from *tcHyp* **have** *obs2*:$\forall\, r{\in}\{0..t\}.\ ((\lambda s.\ rval\ \eta\ (F\ s))$ *has-vector-derivative*
$rval\ (rdiff\ \eta)\ (F\ r))\ (at\ r\ within\ \{0..t\})$ **using** *derivationLemma termVarsHyp* **by**
blast
have $(\forall\, t{\geq}0.\ \forall\ xf \in set\ xfList.\ F\ t\ (vdiff\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t))$
using *tcHyp solves-store-ivpD*(*4*) **by** *blast*
from *this* **and** *tcHyp* **have** $\forall\, r{\in}\{0..t\}.\ rval\ (rdiff\ \eta)\ (F\ r) =$
$rval\ (substList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (rdiff\ \eta))\ (F\ r)$
using *diff-subst-prprty-4terms termVarsHyp listsHyp* **by** *fastforce*
also from *substHyp* **have** $\forall\, r{\in}\{0..t\}.$
$rval\ (substList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (rdiff\ \eta))\ (F\ r) \geq 0$
using *solves-store-ivpD*(*1*) *solves-store-ivpD*(*3*) *tcHyp* **by** (*metis atLeastAtMost-iff*)
ultimately have $*$:$\forall\, r{\in}\{0..t\}.\ rval\ (rdiff\ \eta)\ (F\ r) \geq 0$ **by** (*simp*)
from *obs2* **and** *tcHyp* **have** $\forall\, r{\in}\{0..t\}.\ ((\lambda s.\ rval\ \eta\ (F\ s))$ *has-derivative* $(\lambda x.\ x *_R (rval\ (rdiff\ \eta)\ (F\ r))))\ (at\ r\ within\ \{0..t\})$ **by** (*simp add*: *has-vector-derivative-def*)

hence $\exists\, r{\in}\{0..t\}.\ rval\ \eta\ (F\ t) - rval\ \eta\ (F\ 0) = t \cdot (rval\ (rdiff\ \eta)\ (F\ r))$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then obtain r **where** $rval\ (rdiff\ \eta)\ (F\ r) \geq 0 \wedge 0 \leq r \wedge r \leq t \wedge rval\ (rdiff\ \eta)$
$(F\ t) \geq 0$
$\wedge\ rval\ \eta\ (F\ t) - rval\ \eta\ (F\ 0) = t \cdot (rval\ (rdiff\ \eta)\ (F\ r))$
using $*$ *tcHyp* **by** *fastforce*
thus $rval\ \eta\ c \geq 0$
using *obs1 tcHyp* **by** (*metis cancel-comm-monoid-add-class.diff-cancel diff-ge-0-iff-ge*

diff-strict-mono linorder-neqE-linordered-idom linordered-field-class.sign-simps(45) not-le)
qed
qed

lemma *less-pval-to-rval*:
assumes *pval* (*subspList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*pdiff* (*Less* ϑ η))) *st*
shows *rval* (*substList* ((*map* (*vdiff*$\circ\pi_1$) *xfList*) $\otimes$ *uInput*) (*rdiff* (*Sum* η (*Mns* ϑ)))) *st* $\geq$ *0*
using *assms* **by**(*auto*)

lemma *leq-pval-to-rval*:
assumes *pval* (*subspList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*pdiff* (*Leq* ϑ η))) *st*
shows *rval* (*substList* ((*map* (*vdiff*$\circ\pi_1$) *xfList*) $\otimes$ *uInput*) (*rdiff* (*Sum* η (*Mns* ϑ)))) *st* $\geq$ *0*
using *assms* **by**(*auto*)

lemma *dInv-prelim*:
assumes *substHyp*:$\forall$ *st*. *G st* $\longrightarrow$ ($\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *st* (*vdiff str*) = *0*) $\longrightarrow$
pval (*subspList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*pdiff* φ)) *st*
and *propVarsHyp*:*propVars* φ $\subseteq$ (*UNIV* $-$ *varDiffs*)
and *listsHyp*:*map* π_2 *xfList* = *map rval uInput*
shows (*pval* φ) *a* $\longrightarrow$ ($\forall$ *c*. (*a*,*c*) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ (*pval* φ) *c*)
proof(*clarify*)
fix *c* **assume** *aHyp*:*pval* φ *a* **and** *cHyp*:(*a*, *c*) $\in$ *ODEsystem xfList with G*
from *this* **obtain** *t*::*real* **and** *F*::*real* $\Rightarrow$ *real store*
where *tcHyp*:*t*$\geq$*0* $\wedge$ *F t* = *c* $\wedge$ *solvesStoreIVP F xfList a G* **using** *guarDiffEqtn-def*
by *auto*
from *aHyp propVarsHyp* **and** *substHyp* **show** *pval* φ *c*
proof(*induction* φ)
case (*Eq* ϑ η)
hence *hyp*:$\forall$ *st*. *G st* $\longrightarrow$ ($\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *st* (*vdiff str*) = *0*) $\longrightarrow$

pval (*subspList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*pdiff* (*Eq* ϑ η))) *st* **by** *blast*
then have $\forall$ *st*. *G st* $\longrightarrow$ ($\forall$ *str*. *str* $\notin$ (π_1⟦*set xfList*⟧) $\longrightarrow$ *st* (*vdiff str*) = *0*) $\longrightarrow$

rval (*substList* ((*map* (*vdiff*$\circ\pi_1$) *xfList*) $\otimes$ *uInput*) (*rdiff* (*Sum* ϑ (*Mns* η)))) *st* = *0* **by** *simp*
also have *trmVars* (*Sum* ϑ (*Mns* η)) $\subseteq$ *UNIV* $-$ *varDiffs* **using** *Eq.prems*(*2*) **by** *simp*
moreover have *rval* (*Sum* ϑ (*Mns* η)) *a* = *0* **using** *Eq.prems*(*1*) **by** *simp*
ultimately have ($\forall$ *c*. (*a*, *c*) $\in$ *ODEsystem xfList with G* $\longrightarrow$ *rval* (*Sum* ϑ (*Mns* η)) *c* = *0*)
using *dInvForTrms-prelim listsHyp* **by** *blast*
hence *rval* (*Sum* ϑ (*Mns* η)) (*F t*) = *0* **using** *tcHyp cHyp* **by** *simp*

from *this* **have** $(rval\ \vartheta\ (F\ t) = rval\ \eta\ (F\ t))$ **by** *simp*
also have $pval\ (Eq\ \vartheta\ \eta)\ c = (rval\ \vartheta\ (F\ t) = rval\ \eta\ (F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case $(Less\ \vartheta\ \eta)$
hence $\forall st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1[\![set\ xfList]\!]) \longrightarrow st\ (vdiff\ str) = 0) \longrightarrow$
$0 \leq rval\ (substList\ (map\ (vdiff \circ \pi_1)\ xfList \otimes uInput)\ (rdiff\ (Sum\ \eta\ (Mns\ \vartheta))))$
st
using *less-pval-to-rval* **by** *metis*
also from *Less.prems(2)***have** $trmVars\ (Sum\ \eta\ (Mns\ \vartheta)) \subseteq UNIV - varDiffs$ **by**
simp
moreover have $rval\ (Sum\ \eta\ (Mns\ \vartheta))\ a > 0$ **using** *Less.prems(1)* **by** *simp*
ultimately have $(\forall c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow rval\ (Sum\ \eta\ (Mns$
$\vartheta))\ c > 0)$
using *dInvForProps-prelim(1) listsHyp* **by** *blast*
hence $rval\ (Sum\ \eta\ (Mns\ \vartheta))\ (F\ t) > 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $(rval\ \eta\ (F\ t) > rval\ \vartheta\ (F\ t))$ **by** *simp*
also have $pval\ (Less\ \vartheta\ \eta)\ c = (rval\ \vartheta\ (F\ t) < rval\ \eta\ (F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case $(Leq\ \vartheta\ \eta)$
hence $\forall st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1[\![set\ xfList]\!]) \longrightarrow st\ (vdiff\ str) = 0) \longrightarrow$
$0 \leq rval\ (substList\ (map\ (vdiff \circ \pi_1)\ xfList \otimes uInput)\ (rdiff\ (Sum\ \eta\ (Mns\ \vartheta))))$
st
using *leq-pval-to-rval* **by** *metis*
also from *Leq.prems(2)***have** $trmVars\ (Sum\ \eta\ (Mns\ \vartheta)) \subseteq UNIV - varDiffs$ **by**
simp
moreover have $rval\ (Sum\ \eta\ (Mns\ \vartheta))\ a \geq 0$ **using** *Leq.prems(1)* **by** *simp*
ultimately have $(\forall c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow rval\ (Sum\ \eta\ (Mns$
$\vartheta))\ c \geq 0)$
using *dInvForProps-prelim(2) listsHyp* **by** *blast*
hence $rval\ (Sum\ \eta\ (Mns\ \vartheta))\ (F\ t) \geq 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $(rval\ \eta\ (F\ t) \geq rval\ \vartheta\ (F\ t))$ **by** *simp*
also have $pval\ (Leq\ \vartheta\ \eta)\ c = (rval\ \vartheta\ (F\ t) \leq rval\ \eta\ (F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case $(And\ \varphi 1\ \varphi 2)$
then show *?case* **by**(*simp*)
next
case $(Or\ \varphi 1\ \varphi 2)$
from *this* **show** *?case* **by** *auto*
qed
qed

theorem *dInv*:
assumes $\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1[\![set\ xfList]\!]) \longrightarrow st\ (vdiff\ str) = 0) \longrightarrow$
$pval\ (subspList\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ (pdiff\ \varphi))\ st$
and *termVarsHyp*:$propVars\ \varphi \subseteq (UNIV - varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList = map\ rval\ uInput$

and *phi-p*:P = *pval* φ
shows *PRE P* (*ODEsystem xfList with G*) *POST P*
proof(*clarsimp*)
fix *a b*
assume $(a,\ b) \in \lceil P \rceil$
from *this* **have** *aHyp*:$a = b \wedge P\ a$ **by** (*metis* (*full-types*) *d-p2r rdom-p2r-contents*)
have $P\ a \longrightarrow (\forall\ c.\ (a,c) \in$ (*ODEsystem xfList with G*) $\longrightarrow P\ c)$
using *assms dInv-prelim* **by** *metis*
from *this* **and** *aHyp* **have** $\forall\ c.\ (a,c) \in$ (*ODEsystem xfList with G*) $\longrightarrow P\ c$ **by** *blast*
thus $(a,\ b) \in$ *wp* (*ODEsystem xfList with G*) $\lceil P \rceil$
using *aHyp* **by** (*simp add*: *boxProgrPred-chrctrztn*)
qed

theorem *dInvFinal*:
assumes $\forall\ st.\ G\ st \longrightarrow (\forall\, str.\ str \notin (\pi_1 [\![set\ xfList]\!]) \longrightarrow st\ (vdiff\ str) = 0) \longrightarrow$ *pval* (*subspList* ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) (*pdiff* φ)) *st*
and *termVarsHyp*:*propVars* $\varphi \subseteq$ (*UNIV* $-$ *varDiffs*)
and *listsHyp*:*map* π_2 *xfList* = *map rval uInput*
and *impls*:$\lceil P \rceil \subseteq \lceil F \rceil \wedge \lceil F \rceil \subseteq \lceil Q \rceil$
and *phi-f*:F = *pval* φ
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac C*=*pval* φ **in** *dCut*)
apply(*subgoal-tac* $\lceil F \rceil \subseteq$ *wp* (*ODEsystem xfList with G*) $\lceil F \rceil$, *simp*)
using *impls* **and** *phi-f* **apply** *blast*
apply(*subgoal-tac PRE F* (*ODEsystem xfList with G*) *POST F*, *simp*)
apply(*rule-tac* φ=φ **and** *uInput*=*uInput* **in** *dInv*)
  subgoal using *assms*(*1*) **by** *simp*
  subgoal using *termVarsHyp* **by** *simp*
  subgoal using *listsHyp* **by** *simp*
  subgoal using *phi-f* **by** *simp*
apply(*subgoal-tac PRE P* (*ODEsystem xfList with* ($\lambda s.\ G\ s \wedge F\ s$)) *POST Q*, *simp add*: *phi-f*)
apply(*rule dWeakening*)
using *impls* **by** *simp*

declare *d-p2r* [*simp del*]
lemma *motion-with-constant-velocity-and-invariants*:
    PRE (λ *s. s* $''x''$ $>0 \wedge$ *s* $''v''$ > 0)
    (*ODEsystem* [($''x''$, λ *s. s* $''v''$)] *with* (λ *s. True*))
    POST (λ *s. s* $''x''$$> 0$)
apply(*rule-tac C* = λ *s.* *s* $''v''$ > 0 **in** *dCut*)
apply(*rule-tac* φ=*Less* (*Const 0*) (*Var* $''v''$) **and** *uInput*=[*Var* $''v''$]**in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*, *clarify*, *erule-tac x*=$''v''$ **in** *allE*, *simp*)
apply(*rule-tac C* = λ *s.* *s* $''x''$ > 0 **in** *dCut*)
apply(*rule-tac* φ=(*Less* (*Const 0*) (*Var* $''x''$)) **and** *uInput*=[*Var* $''v''$]
  and *F*=λ *s.* *s* $''x''$ > 0 **in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*)
using *dWeakening* **by** *simp*

lemma *motion-with-constant-acceleration-and-invariants*:
PRE (λ *s. s ''y'' < s ''x''* $\wedge$ *s ''v''* $\geq$ *0* $\wedge$ *s ''a'' > 0*)
(*ODEsystem* [(''x'',(λ *s. s ''v''*)),(''v'',(λ *s. s ''a''*))] *with* (λ *s. True*))
POST (λ *s.* (*s ''y'' < s ''x''*))
apply(*rule-tac C* = λ *s. s ''a'' > 0* **in** *dCut*)
apply(*rule-tac* φ = *Less* (*Const 0*) (*Var ''a''*) **and** *uInput*=[*Var ''v''*, *Var ''a''*]**in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*, *clarify*, *erule-tac x*=*''a''* **in** *allE*, *simp*)
apply(*rule-tac C* = λ *s. s ''v''* $\geq$ *0* **in** *dCut*)
apply(*rule-tac* φ = *Leq* (*Const 0*) (*Var ''v''*) **and** *uInput*=[*Var ''v''*, *Var ''a''*] **in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*)
apply(*rule-tac C* = λ *s. s ''x'' > s ''y''* **in** *dCut*)
apply(*rule-tac* φ = *Less* (*Var ''y''*) (*Var ''x''*) **and** *uInput*=[*Var ''v''*, *Var ''a''*]**in** *dInvFinal*)
apply(*simp-all add*: *varDiffs-def vdiff-def*, *clarify*, *erule-tac x*=*''y''* **in** *allE*, *simp*)
using *dWeakening* **by** *simp*
declare *d-p2r* [*simp*]

end