CPSVerification

By Jonathan

May 1, 2018

Contents

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

begin

0.1 VC_KAD Preliminaries

To make our notation less code-like and more mathematical we declare:

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* ($\Delta^c{}_1$)

Their definitions are repeated below.

lemma $\lceil P\rceil = \{(s,\ s)\ |s.\ P\ s\}$ **by** (*simp add*: *p2r-def*)
lemma $\lfloor R\rfloor = (\lambda x.\ x \in \mathit{r2s}\ R)$ **by** (*simp add*: *r2p-def*) — where
lemma $\mathit{r2s}\ R = \{x\ |x.\ \exists\ y.\ (x,y) \in R\}$ **by** *blast* — Moreover

lemma $\pi_1\ (x,y) = x \wedge \pi_2\ (x,y) = y$ **by** *simp*
lemma $\Delta^c{}_1\ R = \{(x,\ x)\ |x.\ \nexists y.\ (x,\ y) \in R\}$ **by** (*simp add*: *rel-ad-def*)
lemma $wp\ R\ Q = \Delta^c{}_1\ (R\ ;\ \Delta^c{}_1\ Q)$ **by** (*simp add*: *rel-antidomain-kleene-algebra.fbox-def*)

Observe also, the following consequences and facts:

proposition $\pi_1(\!|R|\!) = r2s\ R$
by (*simp add*: *fst-eq-Domain*)

proposition $\Delta^c{}_1\ R = Id - \{(s,\ s)\ |s.\ s \in (\pi_1(\!|R|\!))\}$
by(*simp add*: *image-def rel-ad-def*, *fastforce*)

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

proposition *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*)

proposition *rdom-p2r-contents*:$(a,\ b) \in rdom\ \lceil P\rceil = ((a = b) \wedge P\ a)$
proof−
have $(a,\ b) \in rdom\ \lceil P\rceil = ((a = b) \wedge (a,\ a) \in rdom\ \lceil P\rceil)$ **using** *p2r-subid* **by** *fastforce*
also have ... $= ((a = b) \wedge (a,\ a) \in \lceil P\rceil)$ **by** *simp*
also have ... $= ((a = b) \wedge P\ a)$ **by** (*simp add*: *p2r-def*)
ultimately show *?thesis* **by** *simp*
qed

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

proposition *rel-ad-rule2*: $(x,x) \in \Delta^c{}_1\ \lceil P\rceil \Longrightarrow \neg\ P\ x$
by(*metis ComplD VC-KAD.p2r-neg-hom rel-ad-rule1 empty-iff mem-Collect-eq p2s-neg-hom*

rel-antidomain-kleene-algebra.a-one rel-antidomain-kleene-algebra.am1 relcomp.relcompI)

proposition *rel-ad-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*)

proposition *rel-ad-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*)

proposition *boxProgrPred-chrctrztn*:$(x,x) \in wp\ R\ \lceil P\rceil = (\forall\ y.\ (x,y) \in R \longrightarrow P\ y)$
by(*metis boxProgrPred-IsProp rel-ad-rule1 rel-ad-rule2 rel-ad-rule3 rel-ad-rule4 d-p2r wp-simp wp-trafo*)

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

proposition $x \in r2s\ (wp\ R\ P) = (\forall\, y.\ (x,\ y) \in R \longrightarrow \lfloor P \rfloor\ y)$
by(*simp add*: *r2p-def rel-antidomain-kleene-algebra.fbox-def rel-ad-def Domain-iff relcomp.simps*)

0.2 ODEs Preliminaries

Once again, we repeat some definitions.

lemma $\{a..b\} = \{x.\ a \leq x \wedge x \leq b\}$ **by** *fastforce*
lemma $\{a<..<b\} = \{x.\ a < x \wedge x < b\}$ **by** *fastforce*
lemma $(x\ \textit{solves-ode}\ f)\ \{0..t\}\ R = ((x\ \textit{has-vderiv-on}\ (\lambda t.\ f\ t\ (x\ t)))\ \{0..t\} \wedge x \in \{0..t\} \rightarrow R)$
using *solves-ode-def* **by** *simp*
lemma $f \in A \rightarrow B = (f \in \{f.\ \forall\ x.\ x \in A \longrightarrow (f\,x) \in B\})$ **using** *Pi-def* **by** *auto*
lemma $(f\ \textit{has-vderiv-on}\ f')\{0..t\} = (\forall\, x{\in}\{0..t\}.\ (f\ \textit{has-vector-derivative}\ f'\ x)\ (\textit{at}\ x\ \textit{within}\ \{0..t\}))$
using *has-vderiv-on-def* **by** *simp*
lemma $(f\ \textit{has-vector-derivative}\ f')\ (\textit{at}\ x\ \textit{within}\ \{0..t\}) =$
$(f\ \textit{has-derivative}\ (\lambda x.\ x *_R f'))\ (\textit{at}\ x\ \textit{within}\ \{0..t\})$ **using** *has-vector-derivative-def*
by *auto*

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

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

lemma *solves-ivpD*:
assumes $(x\ \textit{solvesTheIVP}\ f\ \textit{withInitCond}\ t0 \mapsto x0)\ A\ B$
shows $(x\ \textit{solves-ode}\ f)\ A\ B$
and $x\ t0 = x0$
using *assms* **by** (*auto simp*: *solves-ivp-def*)

theorem(**in** *unique-on-bounded-closed*) *ivp-unique-solution*:
assumes *xIsSol*:$(x\ \textit{solvesTheIVP}\ f\ \textit{withInitCond}\ t0 \mapsto x0)\ T\ X$
assumes *yIsSol*:$(y\ \textit{solvesTheIVP}\ f\ \textit{withInitCond}\ t0 \mapsto x0)\ T\ X$
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

0.3 VC_diffKAD Preliminaries

In dL, the set of possible program variables is split in two, the set of variables V and their primed counterparts V'. To implement this, we use Isabelle's string-type and define a function that primes a given string. We then define the set of primed-strings based on it.

definition *vdiff* ::*string* $\Rightarrow$ *string* (∂ - [*55*] *70*) **where**
(∂ *x*) = ''*d*[''@*x*@'']''

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

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

proposition *vdiff-noFixPoints*:*str* $\neq$ (∂ *str*)
by(*simp add*: *vdiff-def*)

lemma *varDiffsI*:*x* = (∂ *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*

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

0.3.1 (primed) dSolve preliminaries

The verification components check that a given external input solves the problem at hand. This input is entered as a list. Thus, we introduce a function to combine lists in a component-wise manner.

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*)

— The following lines, test the behavior of our function with other more intuitive functions
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*

— Next, we derive some properties of the cross_list function.

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*:
assumes $\forall$ *xzList*. *xzList* = $(map\ \pi_1\ xyList) \otimes zList$
and $(y, z) \in set\ ((map\ \pi_2\ xyList) \otimes zList)$
shows $(\exists\ x.\ (x,y) \in set\ xyList \wedge (x,z) \in set\ xzList)$
using *assms* **by**(*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*)
prefer *3* **apply**(*rule ballI*, *simp*, *erule disjE*, *simp*) **subgoal by** *blast*
by *simp-all*

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* π_1 *(xList* $\otimes$ *yList) = xList*
by(*induction xList yList rule*: *cross-list.induct*, *simp-all*)

lemma *length-crossListProj2*:
length xList = length yList $\Longrightarrow$ *map* π_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

Now we need to define how to handle the given input in order to return a state at the end of a (differential) evolution. First, we say what the behavior of said state will be on primed-strings.

abbreviation *varDiffs-to-zero* ::*real store* $\Rightarrow$ *real store* (*d2z*) **where**
d2z a $\equiv$ (*override-on a* (λ *str*. *0*) *varDiffs*)

proposition *varDiffs-to-zero-vdiff*[*simp*]: (*d2z a*) (∂ *x*) = *0*
apply(*simp add*: *override-on-def varDiffs-def*)
by *auto*

proposition *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*

— Next, for each entry of the input-list, we update the state using said entry.

definition *vderiv-of f S* = (*SOME f'*. (*f has-vderiv-on f'*) *S*)

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*,
∂ (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*) (-[-←-] - [*64*,*64*,*64*] *63*) **where**
s[*xfList*←*uInput*] *t* $\equiv$ *state-list-upd* (*uInput* $\otimes$ *xfList*) *t s*

proposition *state-list-cross-upd-empty*[*simp*]: (*a*[[]←*list*] *t*) = *a*
by(*induction list*, *simp-all*)

lemma *inductive-state-list-cross-upd-its-vars*:
assumes *distHyp*:*distinct* (*map* π_1 ((*y*, *g*) # *xftail*))
and *varHyp*:$\forall$ *xf*$\in$*set*((*y*, *g*) # *xftail*). π_1 *xf* $\notin$ *varDiffs*
and *indHyp*:(*u*, *x*, *f*) $\in$ *set* (*utail* $\otimes$ *xftail*) $\Longrightarrow$ (*a*[*xftail*←*utail*] *t*) *x* = *u t a*
and *disjHyp*:(*u*, *x*, *f*) = (*v*, *y*, *g*) $\vee$ (*u*, *x*, *f*) $\in$ *set* (*utail* $\otimes$ *xftail*)
shows (*a*[(*y*, *g*) # *xftail*←*v* # *utail*] *t*) *x* = *u t a*
using *disjHyp* **proof**
  assume (*u*, *x*, *f*) = (*v*, *y*, *g*)
  hence (*a*[(*y*, *g*) # *xftail*←*v* # *utail*] *t*) *x* = ((*a*[*xftail*←*utail*] *t*)(*x* := *u t a*,

∂ *x* := *if t* = *0 then f a else vderiv-of* (λ *r. u r a*) {*0*<..< (*2* $*_R$ *t*)} *t*)) *x* **by** *simp*
also have ... = *u t a* **by** (*simp add*: *vdiff-def*)
ultimately show *?thesis* **by** *simp*
next
assume *yTailHyp*:(*u*, *x*, *f*) $\in$ *set* (*utail* $\otimes$ *xftail*)
from *this* **and** *indHyp* **have** *3*:(*a*[*xftail*$\leftarrow$*utail*] *t*) *x* = *u t a* **by** *fastforce*
from *yTailHyp* **and** *distHyp* **have** *2*:*y* $\neq$ *x* **using** *non-empty-crossListElim* **by** *force*
from *yTailHyp* **and** *varHyp* **have** *1*:*x* $\neq$ ∂ *y*
using *non-empty-crossListElim vdiff-invarDiffs* **by** *fastforce*
from *1* **and** *2* **have** (*a*[(*y*, *g*) # *xftail*$\leftarrow$*v* # *utail*] *t*) *x* = (*a*[*xftail*$\leftarrow$*utail*] *t*) *x* **by** *simp*
thus *?thesis* **using** *3* **by** *simp*
qed

theorem *state-list-cross-upd-its-vars*:
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 *its-var*: (*u*,*x*,*f*) $\in$ *set* (*uInput* $\otimes$ *xfList*)
shows (*a*[*xfList*$\leftarrow$*uInput*] *t*) *x* = *u t a*
using *assms* **apply**(*induction xfList uInput rule*: *cross-list.induct*, *simp*, *simp*)
apply(*clarify*, *rule inductive-state-list-cross-upd-its-vars*) **by** *simp-all*

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*)

lemma *inductive-state-list-cross-upd-its-dvars*:
assumes $\exists$ *g*. (*a*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on a g varDiffs*
and $\forall$ *xf*$\in$*set* (*xf* # *xfTail*). π_1 *xf* $\notin$ *varDiffs*
and $\forall$ *uxf*$\in$*set* (*u* # *uTail* $\otimes$ *xf* # *xfTail*). π_1 *uxf 0 a* = *a* (π_1 (π_2 *uxf*))
shows $\exists$ *g*. (*a*[*xf* # *xfTail*$\leftarrow$*u* # *uTail*] *0*) = *override-on a g varDiffs*
proof−
let *?gLHS*=(*a*[(*xf* # *xfTail*)$\leftarrow$(*u* # *uTail*)] *0*)
have *observ*:∂ (π_1 *xf*) $\in$ *varDiffs* **by** (*auto simp*: *varDiffs-def*)
from *assms*(*1*) **obtain** *g* **where** (*a*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on a g varDiffs* **by** *force*
then have *?gLHS* = (*override-on a g varDiffs*)(π_1 *xf* := *u 0 a*, ∂ (π_1 *xf*) := π_2 *xf a*) **by** *simp*
also have . . . = (*override-on a g varDiffs*)(∂ (π_1 *xf*) := π_2 *xf a*)
using *override-on-def varDiffs-def assms* **by** *auto*
also have ... = (*override-on a* (*g*(∂ (π_1 *xf*) := π_2 *xf a*)) *varDiffs*)
using *observ* **and** *override-on-upd* **by** *force*
ultimately show *?thesis* **by** *auto*
qed

proposition *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*) = *override-on a a varDiffs*
unfolding *override-on-def* **by** *simp*
thus *?case* **by** *metis*
next
case (*2 xf xfTail*)
have (*a*[(*xf* # *xfTail*)$\leftarrow$[]] *0*) = *override-on a a varDiffs*
unfolding *override-on-def* **by** *simp*
thus *?case* **by** *metis*
next
case (*3 xf xfTail u uTail*)
then have $\exists$ *g*. (*a*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on a g varDiffs* **by** *simp*
thus *?case* **using** *inductive-state-list-cross-upd-its-dvars 3.prems* **by** *blast*
qed

— Finally, we combine all previous transformations into one single expression.

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* -[-$\leftarrow$-] - [*64*,*64*,*64*] *63*) **where** *sol s*[*xfList*$\leftarrow$*uInput*] *t* $\equiv$ *d2z s*[*xfList*$\leftarrow$*uInput*] *t*

0.3.2 dInv preliminaries

Here, we introduce syntactic notation to talk about differential invariants.

no-notation *Antidomain-Semiring.antidomain-left-monoid-class.am-add-op* (**infixl** $\oplus$ *65*)
no-notation *Dioid.times-class.opp-mult* (**infixl** $\odot$ *70*)
no-notation *Lattices.inf-class.inf* (**infixl** $\sqcap$ *70*)
no-notation *Lattices.sup-class.sup* (**infixl** $\sqcup$ *65*)

datatype *trms* = *Const real* (t_C - [*54*] *70*) | *Var string* (t_V - [*54*] *70*) |
Mns trms ($\ominus$ - [*54*] *65*) | *Sum trms trms* (**infixl** $\oplus$ *65*) |
Mult trms trms (**infixl** $\odot$ *68*)

primrec *tval* ::*trms* $\Rightarrow$ (*real store* $\Rightarrow$ *real*) ($[\![$-$]\!]_t$ [*55*] *60*) **where**
$[\![t_C\ r]\!]_t = (\lambda\ s.\ r)|$
$[\![t_V\ x]\!]_t = (\lambda\ s.\ s\ x)|$
$[\![\ominus\ \vartheta]\!]_t = (\lambda\ s.\ -\ ([\![\vartheta]\!]_t)\ s)|$
$[\![\vartheta \oplus \eta]\!]_t = (\lambda\ s.\ ([\![\vartheta]\!]_t)\ s\ +\ ([\![\eta]\!]_t)\ s)|$
$[\![\vartheta \odot \eta]\!]_t = (\lambda\ s.\ ([\![\vartheta]\!]_t)\ s\ \cdot\ ([\![\eta]\!]_t)\ s)$

datatype *props* = *Eq trms trms* (**infixr** $\doteq$ *60*) | *Less trms trms* (**infixr** $\prec$ *62*) |
Leq trms trms (**infixr** $\preceq$ *61*) | *And props props* (**infixl** $\sqcap$ *63*) |

$Or\ props\ props$ (**infixl** $\sqcup$ 64)

primrec $pval$ $::props \Rightarrow (real\ store \Rightarrow bool)$ ($[\![\text{-}]\!]_P$ $[55]$ 60) **where**
$[\![\vartheta \doteq \eta]\!]_P = (\lambda\ s.\ ([\![\vartheta]\!]_t)\ s = ([\![\eta]\!]_t)\ s)|$
$[\![\vartheta \prec \eta]\!]_P = (\lambda\ s.\ ([\![\vartheta]\!]_t)\ s < ([\![\eta]\!]_t)\ s)|$
$[\![\vartheta \preceq \eta]\!]_P = (\lambda\ s.\ ([\![\vartheta]\!]_t)\ s \leq ([\![\eta]\!]_t)\ s)|$
$[\![\varphi \sqcap \psi]\!]_P = (\lambda\ s.\ ([\![\varphi]\!]_P)\ s \wedge ([\![\psi]\!]_P)\ s)|$
$[\![\varphi \sqcup \psi]\!]_P = (\lambda\ s.\ ([\![\varphi]\!]_P)\ s \vee ([\![\psi]\!]_P)\ s)$

primrec $tdiff$ $::trms \Rightarrow trms$ (∂_t - $[54]$ 70) **where**
$(\partial_t\ t_C\ r) = t_C\ 0|$
$(\partial_t\ t_V\ x) = t_V\ (\partial\ x)|$
$(\partial_t \ominus \vartheta) = \ominus\ (\partial_t\ \vartheta)|$
$(\partial_t\ (\vartheta \oplus \eta)) = (\partial_t\ \vartheta) \oplus (\partial_t\ \eta)|$
$(\partial_t\ (\vartheta \odot \eta)) = ((\partial_t\ \vartheta) \odot \eta) \oplus (\vartheta \odot (\partial_t\ \eta))$

primrec $pdiff$ $::props \Rightarrow props$ (∂_P - $[54]$ 70) **where**
$(\partial_P\ (\vartheta \doteq \eta)) = ((\partial_t\ \vartheta) \doteq (\partial_t\ \eta))|$
$(\partial_P\ (\vartheta \prec \eta)) = ((\partial_t\ \vartheta) \preceq (\partial_t\ \eta))|$
$(\partial_P\ (\vartheta \preceq \eta)) = ((\partial_t\ \vartheta) \preceq (\partial_t\ \eta))|$
$(\partial_P\ (\varphi \sqcap \psi)) = (\partial_P\ \varphi) \sqcap (\partial_P\ \psi)|$
$(\partial_P\ (\varphi \sqcup \psi)) = (\partial_P\ \varphi) \sqcap (\partial_P\ \psi)$

primrec $trmVars$ $::\ trms \Rightarrow string\ set$ **where**
$trmVars\ (t_C\ r) = \{\}|$
$trmVars\ (t_V\ x) = \{x\}|$
$trmVars\ (\ominus\ \vartheta) = trmVars\ \vartheta|$
$trmVars\ (\vartheta \oplus \eta) = trmVars\ \vartheta \cup trmVars\ \eta|$
$trmVars\ (\vartheta \odot \eta) = trmVars\ \vartheta \cup trmVars\ \eta$

fun $substList$ $::(string \times trms)\ list \Rightarrow trms \Rightarrow trms$ ($\text{-}\langle\text{-}\rangle$ $[54]$ 80) **where**
$xTrmList\langle t_C\ r\rangle = t_C\ r|$
$[]\langle t_V\ x\rangle = t_V\ x|$
$((y,\xi)\ \#\ xTrmTail)\langle Var\ x\rangle = (if\ x = y\ then\ \xi\ else\ xTrmTail\langle Var\ x\rangle)|$
$xTrmList\langle\ominus\ \vartheta\rangle = \ominus\ (xTrmList\langle\vartheta\rangle)|$
$xTrmList\langle\vartheta \oplus \eta\rangle = (xTrmList\langle\vartheta\rangle) \oplus (xTrmList\langle\eta\rangle)|$
$xTrmList\langle\vartheta \odot \eta\rangle = (xTrmList\langle\vartheta\rangle) \odot (xTrmList\langle\eta\rangle)$

lemma *substList-on-compl-of-varDiffs*:
assumes $trmVars\ \eta \subseteq (UNIV - varDiffs)$
assumes $set\ (map\ \pi_1\ xTrmList) \subseteq varDiffs$
shows $xTrmList\langle\eta\rangle = \eta$
using *assms* **apply**(*induction* η, *simp-all add*: *varDiffs-def*)
by(*induction xTrmList*, *auto*)

lemma *substList-help1*:$set\ (map\ \pi_1\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)) \subseteq varDiffs$
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 ((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*)$\langle\eta\rangle$ = η
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 ((*map vdiff list1*) $\otimes$ *list2*)$\langle t_V\ (\partial\ x)\rangle$ = $t_V\ (\partial\ x)$
using *assms* **apply**(*induction list1 list2 rule*: *cross-list.induct*, *simp*, *simp*, *clarsimp*)
by(*simp add*: *vdiff-inj*)

primrec *propVars* :: *props* $\Rightarrow$ *string set* **where**
propVars $(\vartheta \doteq \eta)$ = *trmVars* ϑ $\cup$ *trmVars* η|
propVars $(\vartheta \prec \eta)$ = *trmVars* ϑ $\cup$ *trmVars* η|
propVars $(\vartheta \preceq \eta)$ = *trmVars* ϑ $\cup$ *trmVars* η|
propVars $(\varphi \sqcap \psi)$ = *propVars* φ $\cup$ *propVars* ψ|
propVars $(\varphi \sqcup \psi)$ = *propVars* φ $\cup$ *propVars* ψ

primrec *subspList* :: (*string* $\times$ *trms*) *list* $\Rightarrow$ *props* $\Rightarrow$ *props* (-$\upharpoonleft$-$\upharpoonleft$ [*54*] *80*) **where**
xTrmList$\upharpoonleft\vartheta \doteq \eta\upharpoonleft$ = ((*xTrmList*$\langle\vartheta\rangle$) $\doteq$ (*xTrmList*$\langle\eta\rangle$))|
xTrmList$\upharpoonleft\vartheta \prec \eta\upharpoonleft$ = ((*xTrmList*$\langle\vartheta\rangle$) $\prec$ (*xTrmList*$\langle\eta\rangle$))|
xTrmList$\upharpoonleft\vartheta \preceq \eta\upharpoonleft$ = ((*xTrmList*$\langle\vartheta\rangle$) $\preceq$ (*xTrmList*$\langle\eta\rangle$))|
xTrmList$\upharpoonleft\varphi \sqcap \psi\upharpoonleft$ = ((*xTrmList*$\upharpoonleft\varphi\upharpoonleft$) $\sqcap$ (*xTrmList*$\upharpoonleft\psi\upharpoonleft$))|
xTrmList$\upharpoonleft\varphi \sqcup \psi\upharpoonleft$ = ((*xTrmList*$\upharpoonleft\varphi\upharpoonleft$) $\sqcup$ (*xTrmList*$\upharpoonleft\psi\upharpoonleft$))

end

1 VC_diffKAD

theory *VC-diffKAD*
imports *VC-diffKAD-auxiliarities*

begin

definition *solvesStoreIVP* :: (*real* $\Rightarrow$ *real store*) $\Rightarrow$ (*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* $\Rightarrow$
real store $\Rightarrow$ (*real store pred*) $\Rightarrow$ *bool*
((- *solvesTheStoreIVP* - *withInitState* - *andGuard* -) [*70*, *70*, *70*, *70*] *68*) **where**
solvesStoreIVP φ_S *xfList s G* $\equiv$
($*$ *F preserves the guard statement and F sends vdiffs*$-$*in*$-$*list to derivs*. $*$)
($\forall$ $t \geq 0$. *G* (φ_S *t*) $\wedge$ ($\forall$ *xf* $\in$ *set xfList*. φ_S *t* (∂ (π_1 *xf*)) = π_2 *xf* (φ_S *t*)) $\wedge$
($*$ *F preserves the rest of the variables and F sends derivs of constants to 0*. $*$)
($\forall$ *y*. (*y* $\notin$ (π_1(|*set xfList*|)) $\cup$ *varDiffs* $\longrightarrow$ φ_S *t y* = *s y*) $\wedge$
(*y* $\notin$ (π_1(|*set xfList*|)) $\longrightarrow$ φ_S *t* (∂ *y*) = *0*)) $\wedge$
($*$ *F solves the induced IVP*. $*$)
($\forall$ *xf* $\in$ *set xfList*. ((λ *t*. φ_S *t* (π_1 *xf*)) *solvesTheIVP* (λ *t*.λ *r*.(π_2 *xf*) (φ_S *t*))
withInitCond *0* $\mapsto$ *s*(π_1 *xf*)) {*0*..*t*} *UNIV*))

lemma *solves-store-ivpI*:
assumes $\forall\ t \geq 0.\ G\ (\varphi_S\ t)$
  and $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \longrightarrow \varphi_S\ t\ y = s\ y$
  and $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow \varphi_S\ t\ (\partial\ y) = 0$
  and $\forall\ t \geq 0.\forall\ xf \in set\ xfList.\ (\varphi_S\ t\ (\partial\ (\pi_1\ xf))) = (\pi_2\ xf)\ (\varphi_S\ t)$
  and $\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ ((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))\ solvesTheIVP\ (\lambda\ t.\ \lambda\ r.$ $(\pi_2\ xf)\ (\varphi_S\ t))$
$withInitCond\ 0 \mapsto (s\ (\pi_1\ xf)))\ \{0..t\}\ UNIV$
shows φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
using *assms solvesStoreIVP-def* **by** *auto*

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

lemma [*solves-store-ivpE*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
shows $\forall\ t \geq 0.\ G\ (\varphi_S\ t)$
  and $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \longrightarrow \varphi_S\ t\ y = s\ y$
  and $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow \varphi_S\ t\ (\partial\ y) = 0$
  and $\forall\ t \geq 0.\forall\ xf \in set\ xfList.\ (\varphi_S\ t\ (\partial\ (\pi_1\ xf))) = (\pi_2\ xf)\ (\varphi_S\ t)$
  and $\forall\ t \geq 0.\forall\ xf \in set\ xfList.\ ((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))\ solvesTheIVP\ (\lambda\ t.\ \lambda\ r.\ (\pi_2$ $xf)\ (\varphi_S\ t))$
$withInitCond\ 0 \mapsto (s\ (\pi_1\ xf)))\ \{0..t\}\ UNIV$
using *assms solvesStoreIVP-def* **by** *auto*

lemma [*solves-store-ivpE*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
shows $\forall\ y.\ y \notin varDiffs \longrightarrow \varphi_S\ 0\ y = s\ y$
proof(*clarify*, *rename-tac x*)
fix *x* **assume** $x \notin varDiffs$
from *assms* **and** *solves-store-ivpE*(*5*)
have $\forall\ f.\ (x,f) \in set\ xfList \longrightarrow ((\lambda t.\ \varphi_S\ t\ x)\ solvesTheIVP\ (\lambda t\ r.\ f\ (\varphi_S\ t))$
$withInitCond\ 0 \mapsto s\ x)\ \{0..0\}\ UNIV$ **by** *force*
hence $x \in (\pi_1 (\!|set\ xfList|\!)) \Longrightarrow \varphi_S\ 0\ x = s\ x$ **using** *solves-ivpD*(*2*) **by** *fastforce*
also have $x \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \Longrightarrow \varphi_S\ 0\ x = s\ x$
using *assms* **and** *solves-store-ivpE*(*2*) **by** *simp*
ultimately show $\varphi_S\ 0\ x = s\ x$ **using** $\langle x \notin varDiffs\rangle$ **by** *auto*
qed

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

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
  and $t \geq 0$
shows $G\ (\varphi_S\ t)$
using *assms solves-store-ivpE*(*1*) **by** *blast*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*

and $t \geq 0$
and $y \notin (\pi_1 (\!| set\ xfList |\!)) \cup varDiffs$
shows $\varphi_S\ t\ y = s\ y$
using *assms solves-store-ivpE*(*2*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
and $t \geq 0$
and $y \notin (\pi_1 (\!| set\ xfList |\!))$
shows $\varphi_S\ t\ (\partial\ y) = 0$
using *assms solves-store-ivpE*(*3*) **by** *simp*

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

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

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

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

definition *guarDiffEqtn* :: (*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* $\Rightarrow$ (*real store pred*) $\Rightarrow$
real store rel (*ODEsystem - with -* [*70*, *70*] *61*) **where**
ODEsystem xfList with G = $\{(s,\varphi_S\ t)\ |s\ t\ \varphi_S.\ t \geq 0 \land$ *solvesStoreIVP* φ_S *xfList s G*$\}$

— Differential Weakening.
lemma *wlp-evol-guard*:*Id* $\subseteq$ *wp* (*ODEsystem xfList with G*) $\lceil G \rceil$
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** *wlp-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*:(s, s) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$
and *FisSol*:*solvesStoreIVP* φ_S *xfList s G*
and *tHyp*:$0 \leq t$
shows G (φ_S t) $\wedge$ C (φ_S t)
proof−
from *boxDiffC* **have** $\forall$ *c*. (s,c) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ *C c*
by (*simp add*: *boxProgrPred-chrctrztn*)
also from *tHyp* **have** (s, φ_S t) $\in$ (*ODEsystem xfList with G*)
using *FisSol guarDiffEqtn-def* **by** *auto*
ultimately show G (φ_S t) $\wedge$ C (φ_S t)
using *solves-store-ivpD*(*1*) *tHyp FisSol* **by** *blast*
qed

lemma *condAfterEvol-isGuard*:
assumes *boxDiffC*:(s, s) $\in$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$
assumes *FisSol*:*solvesStoreIVP* φ_S *xfList s G*
shows *solvesStoreIVP* φ_S *xfList s* (λs. *G s* $\wedge$ *C s*)
apply(*rule solves-store-ivpI*)
using *assms condAfterEvol-remainsAlongEvol* **apply**(*fastforce*)
using *FisSol solvesStoreIVP-def* **by** *auto*

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* $\lceil P \rceil$ **{hence** $a = b$ **by** (*metis rdom-p2r-contents*)**}**
then have $(a,a) \in$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$ **using** *abHyp* **and** *pBoxDiffCut* **by** *blast*
moreover have $\forall$ *c*. $(a,c) \in$ (*ODEsystem xfList with* ($\lambda s.\ G\ s \wedge C\ s$)) $\longrightarrow Q\ c$
using *pBoxCutQ* **by** (*metis* (*no-types*, *lifting*) ‹$a = b$› *abHyp boxProgrPred-chrctrztn subsetCE*)
ultimately have $\forall$ *c*. $(a,c) \in$ (*ODEsystem xfList with G*) $\longrightarrow Q\ c$
using *guarDiffEqtn-def condAfterEvol-isGuard* **by** *fastforce*
thus $(a,b) \in$ *wp* (*ODEsystem xfList with G*) $\lceil Q \rceil$
using ‹$a = b$› **by** (*simp add*: *boxProgrPred-chrctrztn*)
qed

— Solve Differential Equation.
lemma *prelim-dSolve*:
assumes *solHyp*:(λt. *sol s*[*xfList*←*uInput*] *t*) *solvesTheStoreIVP xfList withInitState s andGuard G*
and *uniqHyp*:$\forall$ *X*. *solvesStoreIVP X xfList s G* $\longrightarrow$ ($\forall$ $t \geq 0$. (*sol s*[*xfList*←*uInput*] *t*) = *X t*)
and *diffAssgn*: $\forall t \geq 0$. *G* (*sol s*[*xfList*←*uInput*] *t*) $\longrightarrow$ *Q* (*sol s*[*xfList*←*uInput*] *t*)
shows $\forall$ *c*. $(s,c) \in$ (*ODEsystem xfList with G*) $\longrightarrow Q\ c$
proof(*clarify*)
fix *c* **assume** $(s,c) \in$ (*ODEsystem xfList with G*)
from *this* **obtain** *t*::*real* **and** φ_S::*real* ⇒ *real store*
where *FHyp*:$t \geq 0 \wedge \varphi_S\ t = c \wedge$ *solvesStoreIVP* φ_S *xfList s G* **using** *guarDiffEqtn-def* **by** *auto*
from *this* **and** *uniqHyp* **have** (*sol s*[*xfList*←*uInput*] *t*) = $\varphi_S\ t$ **by** *blast*
then have *cHyp*:*c* = (*sol s*[*xfList*←*uInput*] *t*) **using** *FHyp* **by** *simp*
from *solHyp* **have** *G* (*sol s*[*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* (λt. *sol s*[*xfList*←*uInput*] *t*) *xfList s G*
and *uniqHyp*:$\forall$ *X*. *solvesStoreIVP X xfList s G* $\longrightarrow$ ($\forall$ $t \geq 0$. (*sol s*[*xfList*←*uInput*] *t*) = *X t*)
and *diffAssgn*: $\forall t \geq 0$. *G* (*sol s*[*xfList*←*uInput*] *t*) $\longrightarrow$ *Q* (*sol s*[*xfList*←*uInput*] *t*)
shows $\lfloor$*wp* (*ODEsystem xfList with G*) $\lceil Q \rceil \rfloor$ *s*
apply(*simp add*: *r2p-def Domain-iff*)
apply(*rule exI*, *subst boxProgrPred-chrctrztn*)
apply(*rule-tac uInput*=*uInput* **in** *prelim-dSolve*)
by (*simp-all add*: *r2p-def Domain-unfold assms*)

theorem *dSolve*:
assumes *solHyp*:$\forall$ *s*. *solvesStoreIVP* (λt. *sol s*[*xfList*←*uInput*] *t*) *xfList s G*
and *uniqHyp*:$\forall$ *s*. $\forall$ *X*. *solvesStoreIVP X xfList s G* $\longrightarrow$ ($\forall$ $t \geq 0$.(*sol s*[*xfList*←*uInput*] *t*) = *X t*)

and *diffAssgn*: ∀ *s*. *P s* ⟶ (∀ *t*≥*0*. *G* (*sol s*[*xfList*←*uInput*] *t*) ⟶ *Q* (*sol s*[*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*:∀ *s*. ∀ *uxf* ∈ *set* (*uInput* ⊗ *xfList*). (π_1 *uxf*) *0 s* = *s* (π_1 (π_2 *uxf*))
shows ∀ *y*. *y* ∉ *varDiffs* ⟶ (*sol s*[*xfList*←*uInput*] *0*) *y* = *s y*
using *assms* **apply**(*induction uInput xfList rule*: *cross-list.induct*, *simp-all*)
by(*simp add*: *varDiffs-def vdiff-def*)

lemma *conds4InitState2*:
assumes *funcsHyp*:∀ *s*. ∀ *g*. ∀ *xf* ∈ *set xfList*. π_2 *xf* (*override-on s g varDiffs*) = π_2 *xf s*
and *distinctHyp*:*distinct* (*map* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:∀ *xf* ∈ *set xfList*. π_1 *xf* ∉ *varDiffs*
and *solHyp3*:∀ *s*.∀ *uxf* ∈ *set* (*uInput* ⊗ *xfList*). (π_1 *uxf*) *0* (*d2z s*) = (*d2z s*) (π_1 (π_2 *uxf*))
shows ∀ *s*.∀ *xf* ∈ *set xfList*.(*sol s*[*xfList*←*uInput*] *0*)(∂ (π_1 *xf*)) = π_2 *xf* (*sol s*[*xfList*←*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* ⇒ *real*
and *u s*::*real* ⇒ *real store* ⇒ *real* **and** *a*::*real store* **and**
xfTail::(*string* × (*real store* ⇒ *real*)) *list* **and** *uTail*::(*real* ⇒ *real store* ⇒ *real*) *list*
assume *IH*:∀ *st g*. ∀ *xf*∈*set xfTail*. π_2 *xf* (*override-on st g varDiffs*) = π_2 *xf st* ⟹ *distinct* (*map* π_1 *xfTail*) ⟹ *length xfTail* = *length uTail* ⟹ ∀ *xf*∈*set xfTail*. π_1 *xf* ∉ *varDiffs* ⟹
∀ *st*. ∀ *uxf*∈*set* (*uTail* ⊗ *xfTail*). π_1 *uxf 0* (*d2z st*) = *d2z st* (π_1 (π_2 *uxf*)) ⟹
∀ *st*. ∀ *xf*∈*set xfTail*. (*sol st*[*xfTail*←*uTail*] *0*) (∂ (π_1 *xf*)) = π_2 *xf* (*sol st*[*xfTail*←*uTail*] *0*)
let *?gLHS* = (*sol a*[((*x*, *f*) # *xfTail*)←(*u* # *uTail*)] *0*) (∂ (π_1 (*y*, *g*)))
let *?gRHS* = π_2 (*y*, *g*) (*sol a*[((*x*, *f*) # *xfTail*)←(*u* # *uTail*)] *0*)
let *?goal* = *?gLHS* = *?gRHS*
assume *eqFuncs*:∀ *st g*. ∀ *xf*∈*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*:∀ *xf*∈*set* ((*x*, *f*) # *xfTail*). π_1 *xf* ∉ *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*) (∂ (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*) $\lor$ (*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* $\otimes$ *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*) $\in$ *set xfTail*
obtain *h2* **where** *h2Def*:(*sol a*[*xfTail*$\leftarrow$*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*$\leftarrow$*uTail*] *0*) = (*sol a*[*xfTail*$\leftarrow$*uTail*] *0*) (∂ *y*)

using *summary* **by** *fastforce*
from *tailHyp* **have** *y* $\neq$ *x* **using** *distinct non-empty-crossListElim* **by** *force*
hence *dXnotdY*:∂ *x* $\neq$ ∂ *y* **by**(*simp add*: *vdiff-def*)
have *xNotdY*:*x* $\neq$ ∂ *y* **using** *vars vdiff-invarDiffs* **by** *auto*
from $*$ **have** *?gLHS* = *g* (*sol a*[*xfTail*$\leftarrow$*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*) $\longrightarrow$ ($\forall$ *var* $\in$ *set* (*map* π_1 *xfList*). *var* $\notin$ *varDiffs*) $\longrightarrow$ *length xfList* = *length uInput* $\longrightarrow$ *t* > *0* $\longrightarrow$ ($\forall$ *uxf* $\in$*set* (*uInput* $\otimes$ *xfList*).
(*a*[*xfList*$\leftarrow$*uInput*] *t*) (∂ (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* $\Rightarrow$ *real* **and** *u s*::*real* $\Rightarrow$ *real store* $\Rightarrow$ *real* **and** *xfTail*::(*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list* **and** *uTail*::(*real* $\Rightarrow$ *real store* $\Rightarrow$ *real*) *list*
assume *IH*:*distinct* (*map* π_1 *xfTail*) $\longrightarrow$ ($\forall$ *var*$\in$*set xfTail*. π_1 *var* $\notin$ *varDiffs*) $\longrightarrow$

length xfTail = *length uTail* $\longrightarrow$ *0* < *t* $\longrightarrow$ ($\forall$ *uxf*$\in$*set* (*uTail* $\otimes$ *xfTail*).
 (*a*[*xfTail*$\leftarrow$*uTail*] *t*) (∂ (π_1 (π_2 *uxf*))) = *vderiv-of* (λ*r*. π_1 *uxf r a*) {*0*<..<*2* $\cdot$ *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*:$\forall$ *xf*$\in$*set* ((*x*, *f*) # *xfTail*). π_1 *xf* $\notin$ *varDiffs*
from *this* **and** *IH* **have** *keyFact*:$\forall$ *uxf*$\in$*set* (*uTail* $\otimes$ *xfTail*).
 (*a*[*xfTail*$\leftarrow$*uTail*] *t*) (∂ (π_1 (π_2 *uxf*))) = *vderiv-of* (λ*r*. π_1 *uxf r a*) {*0*<..<*2* $\cdot$ *t*} *t* **by** *simp*
assume *sygHyp*:(*s*, *y*, *g*) $\in$ *set* ((*u* # *uTail*) $\otimes$ ((*x*, *f*) # *xfTail*))
let *?gLHS* = (*a*[(*x*, *f*) # *xfTail*$\leftarrow$*u* # *uTail*] *t*) (∂ (π_1 (π_2 (*s*, *y*, *g*))))
let *?gRHS* = *vderiv-of* (λ*r*. π_1 (*s*, *y*, *g*) *r a*) {*0*<..<*2* $\cdot$ *t*} *t*
let *?goal* = *?gLHS* = *?gRHS*
let *?lhs* =
((*a*[*xfTail*$\leftarrow$*uTail*] *t*)(*x* := *u t a*, ∂ *x* := *vderiv-of* (λ *r*. *u r a*) {*0*<..< (*2* $\cdot$ *t*)} *t*)) (∂ *y*)
let *?rhs* = *vderiv-of* (λ*r*. *s r a*) {*0*<..< (*2* $\cdot$ *t*)} *t*
from *sygHyp* **have** (*s*, *y*, *g*) = (*u*, *x*, *f*) $\vee$ (*s*, *y*, *g*) $\in$ *set* (*uTail* $\otimes$ *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* $\cdot$ *t*)} *t* **by** *simp*
 also have *vderiv-of* (λ *r*. *u r a*) {*0*<..< (*2* $\cdot$ *t*)} *t* = *?rhs* **using** *uxfEq* **by** *simp*
 ultimately have *?lhs* = *?rhs* **by** *simp*}
moreover
{**assume** *sygTail*:(*s*, *y*, *g*) $\in$ *set* (*uTail* $\otimes$ *xfTail*)
 from *this* **have** *y* $\neq$ *x* **using** *distHyp non-empty-crossListElim* **by** *force*
 hence *dXnotdY*:∂ *x* $\neq$ ∂ *y* **by**(*simp add*: *vdiff-def*)
 have *xNotdY*:*x* $\neq$ ∂ *y* **using** *varHyp* **using** *vdiff-invarDiffs* **by** *auto*
 then have *?lhs* = (*a*[*xfTail*$\leftarrow$*uTail*] *t*) (∂ *y*) **using** *xNotdY* **and** *dXnotdY* **by** *simp*
 also have (*a*[*xfTail*$\leftarrow$*uTail*] *t*) (∂ *y*) = *?rhs* **using** *keyFact sygTail* **by** *auto*
 ultimately have *?lhs* = *?rhs* **by** *simp*}
ultimately show *?goal* **by** *blast*
qed

lemma *prelim-conds4vdiffs*:
assumes *funcsHyp*:$\forall$ *st 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 *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *lengthHyp*:*length xfList* = *length uInput*
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 *keyFact*:$\forall$ *st*. $\forall$ *uxf* $\in$ *set* (*uInput* $\otimes$ *xfList*).$\forall$ *t*>*0*.

vderiv-of $(\lambda r.\ (\pi_1\ uxf)\ r\ (d2z\ st))\ \{0{<}..{<}\ (2\ *_R\ t)\}\ t\ =\ (\pi_2\ (\pi_2\ uxf))\ (sol\ st[xfList{\leftarrow}uInput]\ t)$
shows $\forall\ st.\ \forall\ t{\geq}0.\ \forall\ xf{\in}set\ xfList.$
$(sol\ st[xfList{\leftarrow}uInput]\ t)\ (\partial\ (\pi_1\ xf))\ =\ \pi_2\ xf\ \ (sol\ st[xfList{\leftarrow}uInput]\ t)$
proof(*clarify*)
fix t ::*real* **and** x::*string* **and** f::*real store* $\Rightarrow$ *real* **and** a::*real store*
assume *tHyp*:$0\ \leq\ t$ **and** *pairHyp*:$(x,\ f)\ \in\ set\ xfList$
from *this* **obtain** u **where** *xfuHyp*: $(u,x,f)\ \in\ set\ (uInput\ \otimes\ xfList)$
by (*metis crossList-length legnth-crossListEx1 lengthHyp*)
show $(sol\ a[xfList{\leftarrow}uInput]\ t)\ (\partial\ (\pi_1\ (x,\ f)))\ =\ \pi_2\ (x,\ f)\ (sol\ a[xfList{\leftarrow}uInput]\ t)$
proof(*cases* $t{=}0$)
case *True*
have $\forall\ st.\ \forall\ xf{\in}set\ xfList.$
$(sol\ st[xfList{\leftarrow}uInput]\ 0)\ (\partial\ (\pi_1\ xf))\ =\ \pi_2\ xf\ \ (sol\ st[xfList{\leftarrow}uInput]\ 0)$
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 $(sol\ a[xfList{\leftarrow}uInput]\ t)\ (\partial\ x)\ =$ *vderiv-of* $(\lambda s.\ u\ s\ (d2z\ a))\ \{0{<}..{<}\ (2\ *_R\ t)\}\ t$
using *tHyp xfuHyp assms state-list-cross-upd-correctInPrimes* **by** *fastforce*
also have *vderiv-of* $(\lambda s.\ u\ s\ (d2z\ a))\ \{0{<}..{<}\ (2\ *_R\ t)\}\ t\ =\ f\ (sol\ a[xfList{\leftarrow}uInput]\ t)$
using *keyFact xfuHyp* **and** $\langle t\ >\ 0\rangle$ **by** *force*
ultimately show *?thesis* **by** *simp*
qed
qed

lemma *keyFact-elim*:
assumes *distinctHyp*:*distinct* $(map\ \pi_1\ xfList)$
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall\ \ xf\ \in\ set\ xfList.\ \pi_1\ xf\ \notin\ varDiffs$
and *solHyp1*:$\forall\ \ st.\ \forall\ t{\geq}0.\ \forall\ xf{\in}set\ xfList.$
$((\lambda t.\ (sol\ st[xfList{\leftarrow}uInput]\ t)\ (\pi_1\ xf))$ *has-vderiv-on* $(\lambda t.\ \pi_2\ xf\ (sol\ st[xfList{\leftarrow}uInput]\ t)))\ \{0..t\}$
shows *keyFact*:$\forall\ \ st.\ \forall\ \ uxf\ \in\ set\ (uInput\ \otimes\ xfList).\forall\ t{>}0.$
vderiv-of $(\lambda r.\ (\pi_1\ uxf)\ r\ (d2z\ st))\ \{0{<}..{<}\ (2\ *_R\ t)\}\ t\ =\ (\pi_2\ (\pi_2\ uxf))\ (sol\ st[xfList{\leftarrow}uInput]\ t)$
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-its-vars* **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*. ((λr. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) (π_1 *xf*))
has-vderiv-on (λr. 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*. ((λr. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) (π_1 *xf*))
has-vderiv-on (λr. 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*:((λr. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) *x*)
has-vderiv-on (λr. *f* (*sol a*[*xfList*$\leftarrow$*uInput*] *r*))) {*0*<..<*2* $\cdot$ *t*}
using *has-vderiv-on-subset* **by** *auto*
have ((λr. (*sol a*[*xfList*$\leftarrow$*uInput*] *r*) *x*) *has-vderiv-on* (λr. *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*$\leftarrow$*uInput*] *r*))) {*0*<..<*2* $\cdot$ *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*$\leftarrow$*uInput*] *r*))) {*0*<..<*2* $\cdot$ *t*} **by** *simp*
then have $\forall$ *s* $\in$ {*0*<..<*2* $\cdot$ *t*}. ((λ *r*. *u r* (*d2z a*)) *has-vector-derivative*
f (*sol a*[*xfList*$\leftarrow$*uInput*] *s*)) (*at s within* {*0*<..<*2* $\cdot$ *t*}) **by** (*simp add*: *has-vderiv-on-def*)
{fix *f'* **assume** ((λs. *u s* (*d2z a*)) *has-vderiv-on f'*) {*0*<..<*2* $\cdot$ *t*}
then have *f'Hyp*:$\forall$ *rr* $\in$ {*0*<..<*2* $\cdot$ *t*}. ((λs. *u s* (*d2z a*)) *has-derivative* (λs. *s* $*_R$ (*f' rr*)))
(*at rr within* {*0*<..<*2* $\cdot$ *t*}) **by**(*simp add*: *has-vderiv-on-def has-vector-derivative-def*)

{fix *rr* **assume** *rrHyp*:*rr* $\in$ {*0* <..< *2* $\cdot$ *t*}
have *boxDef*:*box 0* (*2* $\cdot$ *t*) = {*0*<..<*2* $\cdot$ *t*} $\wedge$ *rr* $\in$ *box 0* (*2* $\cdot$ *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* $\cdot$ *t*))
using *tHyp boxDef rrHyp f'Hyp* **by** *force*
from *derivHyp* **have** $\forall$ *rr* $\in$ {*0*<..<*2* $\cdot$ *t*}. ((λ *s*. *u s* (*d2z a*)) *has-derivative*
(λs. *s* $*_R$ (*f* (*sol a*[*xfList*$\leftarrow$*uInput*] *rr*)))) (*at rr within* {*0*<..<*2* $\cdot$ *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*$\leftarrow$*uInput*] *rr*)))) (*at rr within box 0* (*2* $\cdot$ *t*))**using** *rrHyp boxDef* **by** *force*
from *boxDef rr1* **and** *rr2* **have** (λs. *s* $*_R$ (*f' rr*)) = (λs. *s* $*_R$ (*f* (*sol a*[*xfList*$\leftarrow$*uInput*] *rr*)))
using *frechet-derivative-unique-within-open-interval* **by** *blast*
hence *f' rr* = *f* (*sol a*[*xfList*$\leftarrow$*uInput*] *rr*) **by** (*metis lambda-one real-scaleR-def*)**}**
from *this* **have** $\forall$ *rr* $\in$ {*0*<..< *2* $\cdot$ *t*}. *f' rr* = (*f* (*sol a*[*xfList*$\leftarrow$*uInput*] *rr*)) **by** *force***}**
then have *f'Hyp*:$\forall$ *f'*. ((λs. *u s* (*d2z a*)) *has-vderiv-on f'*) {*0*<..<*2* $\cdot$ *t*} $\longrightarrow$
($\forall$ *rr* $\in$ {*0*<..< *2* $\cdot$ *t*}. *f' rr* = (*f* (*sol a*[*xfList*$\leftarrow$*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* $\cdot$ *t*}
by(*simp add*: *vderiv-of-def*, *metis derivHyp someI-ex*)
from *this* **and** *f'Hyp* **have** $\forall$ *rr* $\in$ {*0*<..< *2* $\cdot$ *t*}.
(*vderiv-of* (λr. *u r* (*d2z a*)) {*0*<..< (*2* $*_R$ *t*)}) *rr* = (*f* (*sol a*[*xfList*$\leftarrow$*uInput*] *rr*))

by *blast*
thus *vderiv-of* $(\lambda r.\ \pi_1\ (u,\ x,\ f)\ r\ (d2z\ a))\ \{0{<}..{<}2 *_R t\}\ t =$
$\pi_2\ (\pi_2\ (u,\ x,\ f))\ (sol\ a[xfList{\leftarrow}uInput]\ t)$ **using** *tHyp* **by** *force*
qed

lemma *conds4vdiffs*:
assumes *funcsHyp*:$\forall st\ g.\ \forall xf{\in}set\ xfList.\ \pi_2\ xf\ (override\text{-}on\ st\ g\ varDiffs) = \pi_2\ xf\ st$
and *distinctHyp*:$distinct\ (map\ \pi_1\ xfList)$
and *varsHyp*:$\forall\ xf \in set\ xfList.\ \pi_1\ xf \notin varDiffs$
and *lengthHyp*:$length\ xfList = length\ uInput$
and *solHyp1*:$\forall\ st.\ \forall t{\geq}0.\ \forall xf{\in}set\ xfList.\ ((\lambda t.\ (sol\ st[xfList{\leftarrow}uInput]\ t)\ (\pi_1\ xf))$

$has\text{-}vderiv\text{-}on\ (\lambda t.\ \pi_2\ xf\ (sol\ st[xfList{\leftarrow}uInput]\ t)))\ \{0..t\}$
and *solHyp3*:$\forall\ st.\ \forall\ uxf \in set\ (uInput \otimes xfList).\ (\pi_1\ uxf)\ 0\ (d2z\ st) = (d2z\ st)\ (\pi_1\ (\pi_2\ uxf))$
shows $\forall\ st.\ \forall\ t{\geq}0.\ \forall\ xf{\in}set\ xfList.$
$(sol\ st[xfList{\leftarrow}uInput]\ t)\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (sol\ st[xfList{\leftarrow}uInput]\ t)$
apply(*rule prelim-conds4vdiffs*)
prefer *6* **subgoal using** *assms* **and** *keyFact-elim* **by** *blast*
using *assms* **by** *simp-all*

lemma *conds4Consts*:
assumes *varsHyp*:$\forall\ xf \in set\ xfList.\ \pi_1\ xf \notin varDiffs$
shows $\forall\ str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow (sol\ a[xfList{\leftarrow}uInput]\ t)\ (\partial\ str) = 0$
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*:
$\forall str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \longrightarrow (sol\ a[xfList{\leftarrow}uInput]\ t)\ str = a\ str$
apply(*induction xfList uInput rule*: *cross-list.induct*)
by(*auto simp*: *varDiffs-def*)

lemma *conds4solvesIVP*:
assumes *distinctHyp*:$distinct\ (map\ \pi_1\ xfList)$
and *lengthHyp*:$length\ xfList = length\ uInput$
and *varsHyp*:$\forall\ xf \in set\ xfList.\ \pi_1\ xf \notin varDiffs$
and *solHyp1*:$\forall\ st.\ \forall t{\geq}0.\ \forall\ xf \in set\ xfList.$
$((\lambda t.\ (sol\ st[xfList{\leftarrow}uInput]\ t)\ (\pi_1\ xf))\ has\text{-}vderiv\text{-}on\ (\lambda t.\ \pi_2\ xf\ (sol\ st[xfList{\leftarrow}uInput]\ t)))\ \{0..t\}$
and *solHyp2*:$\forall\ st.\ \forall t{\geq}0.\ \forall xf{\in}set\ xfList.\ (\lambda t.\ (sol\ st[xfList{\leftarrow}uInput]\ t)\ (\pi_1\ xf)) \in \{0..t\} \to UNIV$
and *solHyp3*:$\forall\ st.\ \forall uxf{\in}set\ (uInput \otimes xfList).\ (\pi_1\ uxf)\ 0\ (d2z\ st) = (d2z\ st)\ (\pi_1\ (\pi_2\ uxf))$
shows $\forall\ st.\ \forall t{\geq}0.\forall xf{\in}set\ xfList.\ ((\lambda t.\ (sol\ st[xfList{\leftarrow}uInput]\ t)\ (\pi_1\ xf))$
$solvesTheIVP\ (\lambda t\ r.\ \pi_2\ xf\ (sol\ st[xfList{\leftarrow}uInput]\ t))\ withInitCond\ \ 0 \mapsto st\ (\pi_1\ xf))\ \{0..t\}\ UNIV$
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*:$0 \leq t$ **and** *xfHyp*:$(x, f) \in$ *set xfList*
then obtain *u* **where** *uxfHyp*:$(u, x, f) \in$ *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-its-vars 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*:∀ *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 *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:∀ *xf* ∈ *set xfList*. π_1 *xf* ∉ *varDiffs*
and *guardHyp*:∀ *st*. ∀ *t*≥*0*. *G* (*sol st*[*xfList*←*uInput*] *t*)
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*. *solvesStoreIVP* (λ *t*. (*sol st*[*xfList*←*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*:∀ *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 *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:∀ *xf* ∈ *set xfList*. π_1 *xf* ∉ *varDiffs*
and *guardHyp*:∀ *st*. ∀ *t*≥*0*. *G* (*sol st*[*xfList*←*uInput*] *t*)
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*. ∀ $t \geq 0$. ∀ *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*))
and *uniqHyp*:∀ *st*.∀ *X*. *solvesStoreIVP X xfList st G* ⟶ (∀ *t* ≥ *0*. (*sol st*[*xfList*←*uInput*]
t) = *X t*)
and *postCondHyp*:∀ *st*. *P st* ⟶ (∀ $t \geq 0$. *Q* (*sol st*[*xfList*←*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* ≥ *0*
assumes *contHyp*:∀ *xf* ∈ *set xfList*. *continuous-on* ({*0*..*t*} × *UNIV*)
(λ(*t*, (*r*::*real*)). (π_2 *xf*) (*sol a*[*xfList*←*uInput*] *t*))
shows ∀ *xf* ∈ *set xfList*. *unique-on-bounded-closed 0* {*0*..*t*} (*a* (π_1 *xf*))
(λ*t r*. (π_2 *xf*) (*sol a*[*xfList*←*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-on a* (*F 0*) *varDiffs*
apply(*rule ext*, *subgoal-tac x* ∉ *varDiffs* ⟶ *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* ≥ *0*
assumes *contHyp*:∀ *xf* ∈ *set xfList*. *continuous-on* ({*0*..*s*} × *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*:$(sol\ a[xfList \leftarrow uInput]\ s)\ str = a\ str$ **using** *sHyp notInVars* **by** *blast*
from *Fsolves* **have** $\forall\ t \geq 0.\ \forall\ str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \longrightarrow F\ t\ str = a\ str$
by (*simp add*: *solvesStoreIVP-def*)
then have *2*:$F\ s\ str = a\ str$ **using** *sHyp notInVars* **by** *blast*
thus $(sol\ a[xfList \leftarrow uInput]\ s)\ str = F\ s\ str$ **using** *1* **and** *2* **by** *simp*
next case *True*
then have $str \in (\pi_1 (\!|set\ xfList|\!)) \vee str \in varDiffs$ **by** *simp*
moreover
{assume $str \in (\pi_1 (\!|set\ xfList|\!))$ **from** *this* **obtain** $f::((char\ list \Rightarrow real) \Rightarrow real)$ **where**
strfHyp:$(str, f) \in set\ xfList$ **by** *fastforce*

from *Fsolves* **and** *sHyp* **have** $(\forall\ xf \in set\ xfList.\ ((\lambda t.\ F\ t\ (\pi_1\ xf))\ solvesTheIVP$
$(\lambda t\ r.\ \pi_2\ xf\ (F\ t))\ withInitCond\ \ 0 \mapsto a\ (\pi_1\ xf))\ \{0..s\}\ UNIV)$
by (*simp add*: *solvesStoreIVP-def*)
then have *expand1*:$\forall\ xf \in set\ xfList.((\lambda t.\ F\ t\ (\pi_1\ xf))\ solves\text{-}ode$
$(\lambda t\ r.\ (\pi_2\ xf)\ (F\ t)))\{0..s\}\ UNIV \wedge F\ 0\ (\pi_1\ xf) = a\ (\pi_1\ xf)$ **by** (*simp add*: *solves-ivp-def*)
hence *expand2*:$\forall\ xf \in set\ xfList.\ \forall\ t \in \{0..s\}.\ ((\lambda r.\ F\ r\ (\pi_1\ xf))$
$has\text{-}vector\text{-}derivative\ (\lambda r.\ (\pi_2\ xf)\ (sol\ a[xfList \leftarrow uInput]\ t))\ t)\ (at\ t\ within\ \{0..s\})$
using *eqDerivs* **by** (*simp add*: *solves-ode-def has-vderiv-on-def*)

then have $\forall\ xf \in set\ xfList.\ ((\lambda t.\ F\ t\ (\pi_1\ xf))\ solves\text{-}ode$
$(\lambda t\ r.\ (\pi_2\ xf)\ (sol\ a[xfList \leftarrow uInput]\ t)))\{0..s\}\ UNIV \wedge F\ 0\ (\pi_1\ xf) = a\ (\pi_1\ xf)$
by (*simp add*: *has-vderiv-on-def solves-ode-def expand1 expand2*)
then have *1*:$((\lambda t.\ F\ t\ str)\ solvesTheIVP\ (\lambda t\ r.\ f\ (sol\ a[xfList \leftarrow uInput]\ t))$
$withInitCond\ \ 0 \mapsto a\ str)\ \{0..s\}\ UNIV$ **using** *strfHyp solves-ivp-def* **by** *fastforce*

from *solHyp* **and** *strfHyp* **have** *2*:$((\lambda\ t.\ (sol\ a[xfList \leftarrow uInput]\ t)\ str)$
$solvesTheIVP\ (\lambda t\ r.\ f\ (sol\ a[xfList \leftarrow uInput]\ t))\ withInitCond\ \ 0 \mapsto a\ str)\ \{0..s\}\ UNIV$

using *solvesStoreIVP-def sHyp* **by** *fastforce*

from *sHyp* **and** *contHyp* **have** $\forall$ *xf* $\in$ *set xfList. unique-on-bounded-closed 0* $\{0..s\}$ $(a\ (\pi_1\ xf))$
$(\lambda t\ r.\ (\pi_2\ xf)\ (sol\ a[xfList \leftarrow uInput]\ t))$ *UNIV* (*if s = 0 then 1 else 1/(s+1)*)

using *conds4UniqSol* **by** *simp*
from *this* **have** *3*:*unique-on-bounded-closed 0* $\{0..s\}$ $(a\ str)$ $(\lambda t\ r.\ f\ (sol\ a[xfList \leftarrow uInput]\ t))$
UNIV (*if s = 0 then 1 else 1/(s+1)*) **using** *strfHyp* **by** *fastforce*
from *1 2* **and** *3* **have** $(sol\ a[xfList \leftarrow uInput]\ s)\ str = F\ s\ str$
using *unique-on-bounded-closed.ivp-unique-solution* **using** *real-Icc-closed-segment sHyp* **by** *blast*}
moreover
{**assume** *str* $\in$ *varDiffs*
then obtain *x* **where** *xDef*:*str* $= \partial\ x$ **by** (*auto simp*: *varDiffs-def*)
have $(sol\ a[xfList \leftarrow uInput]\ s)\ str = F\ s\ str$
proof(*cases x* $\in$ *set* (*map* π_1 *xfList*))
case *True*
then obtain *f* **where** *strFhyp*:$(x, f) \in$ *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** $(sol\ a[xfList \leftarrow uInput]\ s)\ str =$
$f\ (sol\ a[xfList \leftarrow uInput]\ s)$ **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 $(sol\ a[xfList \leftarrow uInput]\ s)\ str = 0$
using *False solHyp sHyp solves-store-ivpD(3) xDef* **by** *fastforce*
ultimately show *?thesis* **by** *simp*
qed}
ultimately show $(sol\ a[xfList \leftarrow uInput]\ s)\ str = F\ s\ str$ **by** *blast*
qed
qed

theorem *dSolveUBC*:
assumes *contHyp*:$\forall$ *st.* $\forall\ t \geq 0.\ \forall$ *xf* $\in$ *set xfList. continuous-on* $(\{0..t\} \times UNIV)$

$(\lambda(t, (r{::}real)).\ (\pi_2\ xf)\ (sol\ st[xfList \leftarrow uInput]\ t))$
and *solHyp*:$\forall$ *st. solvesStoreIVP* $(\lambda\ t.\ (sol\ st[xfList \leftarrow uInput]\ t))$ *xfList st G*
and *uniqHyp*:$\forall$ *st.* $\forall$ *X. X solvesTheStoreIVP xfList withInitState st andGuard G*
$\longrightarrow$
$(\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ \forall\ r \in \{0..t\}.\ (\pi_2\ xf)\ (X\ r) =$
$(\pi_2\ xf)\ (sol\ st[xfList \leftarrow uInput]\ r))$
and *diffAssgn*: $\forall st.\ P\ st \longrightarrow (\forall t \geq 0.\ G\ (sol\ st[xfList \leftarrow uInput]\ t) \longrightarrow Q\ (sol\ st[xfList \leftarrow uInput]\ t))$

shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac uInput=uInput* **in** *dSolve*)
subgoal using *solHyp* **by** *simp*
subgoal proof(*clarify*)
fix *a*::*real store* **and** *X*::*real* $\Rightarrow$ *real store* **and** *s*::*real*
assume *XisSol*:*solvesStoreIVP X xfList a G* **and** *sHyp*:$0 \leq s$
from *this* **and** *uniqHyp* **have** $\forall$ *xf* $\in$ *set xfList*. $\forall$ *t* $\in$ {*0..s*}.
(π_2 *xf*) (*X t*) = (π_2 *xf*) (*sol a*[*xfList*$\leftarrow$*uInput*] *t*) **by** *auto*
moreover have $\forall$ *xf* $\in$ *set xfList*. *continuous-on* ({*0..s*} $\times$ *UNIV*)
(λ(*t*, (*r*::*real*)). (π_2 *xf*) (*sol a*[*xfList*$\leftarrow$*uInput*] *t*)) **using** *contHyp sHyp* **by** *blast*
ultimately show (*sol a*[*xfList*$\leftarrow$*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*)
= 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 *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 $PRE\ (\lambda\ s.\ s\ ''station'' < s\ ''pos''\ \wedge s\ ''vel'' > 0)$
$(ODEsystem\ [(''pos'',(\lambda\ s.\ s\ ''vel''))]\ with\ (\lambda\ s.\ True))$
$POST\ (\lambda\ s.\ (s\ ''station'' < s\ ''pos''))$
apply(*rule-tac uInput*=$[\lambda\ t\ s.\ s\ ''vel'' \cdot t + s\ ''pos'']$ **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.

lemma *solvesStoreIVP-couldBeModified*:
fixes F::*real* $\Rightarrow$ *real store*
assumes *vars*:$\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ ((\lambda\ t.\ F\ t\ (\pi_1\ xf))$
$solvesTheIVP\ (\lambda\ t.\ \lambda\ r.\ (\pi_2\ xf)\ (F\ t))\ withInitCond\ 0 \mapsto (a\ (\pi_1\ xf)))\ \{0..t\}\ UNIV$
and *dvars*:$\forall\ t \geq 0.\ \forall xf \in set\ xfList.\ (F\ t\ (\partial\ (\pi_1\ xf))) = (\pi_2\ xf)\ (F\ t)$
shows $\forall\ t \geq 0.\ \forall r \in \{0..t\}.\ \forall\ xf \in set\ xfList.$
$((\lambda\ t.\ F\ t\ (\pi_1\ xf))\ has\text{-}vector\text{-}derivative\ F\ r\ (\partial\ (\pi_1\ xf)))\ (at\ r\ within\ \{0..t\})$
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** *vars* **have** $((\lambda\ t.\ F\ t\ x)\ solvesTheIVP\ (\lambda\ t.\ \lambda\ r.\ f\ (F\ t))$
$withInitCond\ 0 \mapsto (a\ x))\ \{0..t\}\ UNIV$ **using** *tHyp* **by** *fastforce*
then have $((\lambda\ t.\ F\ t\ x)\ has\text{-}vderiv\text{-}on\ (\lambda\ t.\ f\ (F\ t)))\ \{0..t\}$
by (*simp add*: *solves-ode-def solves-ivp-def*)
hence $*$:$\forall r \in \{0..t\}.\ ((\lambda\ t.\ F\ t\ x)\ has\text{-}vector\text{-}derivative\ (\lambda\ t.\ f\ (F\ t))\ r)\ (at\ r\ within$
$\{0..t\})$
by (*simp add*: *has-vderiv-on-def tHyp*)
have $\forall\ t \geq 0.\ \forall r \in \{0..t\}.\ \forall\ xf \in set\ xfList.\ (F\ r\ (\partial\ (\pi_1\ xf))) = (\pi_2\ xf)\ (F\ r)$
using *assms* **by** *auto*
from *this rHyp* **and** *xfHyp* **have** $(F\ r\ (\partial\ x)) = f\ (F\ r)$ **by** *force*
then show $((\lambda t.\ F\ t\ (\pi_1\ (x, f)))\ has\text{-}vector\text{-}derivative\ F\ r\ (\partial\ (\pi_1\ (x, f))))\ (at\ r$
$within\ \{0..t\})$
using $*$ *rHyp* **by** *auto*
qed

lemma *derivationLemma-baseCase*:
fixes F::*real* $\Rightarrow$ *real store*
assumes *solves*:*solvesStoreIVP F xfList a G*
shows $\forall\ x \in (UNIV - varDiffs).\ \forall\ t \geq 0.\ \forall r{\in}\{0..t\}.$
$((\lambda\ t.\ F\ t\ x)$ *has-vector-derivative* $F\ r\ (\partial\ x))$ $(at\ r\ within\ \{0..t\})$
proof
fix x
assume $x \in UNIV - varDiffs$
then have *notVarDiff*:$\forall\ z.\ x \neq \partial\ z$ **using** *varDiffs-def* **by** *fastforce*
show $\forall t{\geq}0.\ \forall r{\in}\{0..t\}.\ ((\lambda t.\ F\ t\ x)$ *has-vector-derivative* $F\ r\ (\partial\ x))$ $(at\ r\ within\ \{0..t\})$
proof(*cases* $x \in set\ (map\ \pi_1\ xfList)$)
case *True*
from *this* **and** *solves* **have** $\forall\ t \geq 0.\ \forall r{\in}\{0..t\}.\ \forall\ xf \in set\ xfList.$
$((\lambda\ t.\ F\ t\ (\pi_1\ xf))$ *has-vector-derivative* $F\ r\ (\partial\ (\pi_1\ xf)))$ $(at\ r\ within\ \{0..t\})$
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*:$\forall\ t \geq 0.\ \forall r{\in}\{0..t\}.\ ((\lambda\ r.\ a\ x)$ *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*: ‹$0 \leq t$›)
ultimately have $((\lambda\ s.\ F\ s\ x)$ *has-vector-derivative* $0)$ $(at\ r\ within\ \{0..t\})$
using *has-vector-derivative-imp* **by** (*metis* ‹$r \in \{0..t\}$›)}
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** $\forall\ t \geq 0.\ F\ t\ (\partial\ x) = 0$
using *solves-store-ivpD*(*3*) **by** *simp*
then show *?thesis* **using** *isZero* **by** *simp*
qed
qed

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 $\forall r{\in}\{0..t\}.\ ((\lambda\ s.\ (\llbracket\eta\rrbracket_t)\ (F\ s))$*has-vector-derivative* $(\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r))$ $(at\ r\ within\ \{0..t\})$
using *termVarsHyp* **proof**(*induction* η)

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* η)
then show *?case*
apply(*clarsimp*)
by(*rule derivative-intros*, *simp*)
next
case (*Sum* $\eta 1$ $\eta 2$)
then show *?case*
apply(*clarsimp*)
by(*rule derivative-intros*, *simp-all*)
next
case (*Mult* $\eta 1$ $\eta 2$)
then show *?case*
apply(*clarsimp*)
apply(*subgoal-tac* $((\lambda s.\ (\llbracket\eta 1\rrbracket_t)\ (F\ s) *_R (\llbracket\eta 2\rrbracket_t)\ (F\ s))$ *has-vector-derivative*
$(\llbracket\partial_t\ \eta 1\rrbracket_t)\ (F\ r) \cdot (\llbracket\eta 2\rrbracket_t)\ (F\ r) + (\llbracket\eta 1\rrbracket_t)\ (F\ r) \cdot (\llbracket\partial_t\ \eta 2\rrbracket_t)\ (F\ r))$ (*at r within*
$\{0..t\}$),*simp*)
apply(*rule-tac* $f'1=(\llbracket\partial_t\ \eta 1\rrbracket_t)\ (F\ r)$ **and** $g'1=(\llbracket\partial_t\ \eta 2\rrbracket_t)\ (F\ r)$ **in** *derivative-eq-intros*(*25*))
by (*simp-all add*: *has-field-derivative-iff-has-vector-derivative*)
qed

lemma *diff-subst-prprty-4terms*:
assumes *solves*:$\forall\ xf \in set\ xfList.\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t)$
and *tHyp*:$(t{::}real) \geq 0$
and *listsHyp*:$map\ \pi_2\ xfList = map\ tval\ uInput$
and *termVarsHyp*:$trmVars\ \eta \subseteq (UNIV - varDiffs)$
shows $(\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ t) = (\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\langle\partial_t\ \eta\rangle\rrbracket_t)\ (F\ t)$
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*:$\bigwedge x.\ x \notin varDiffs \Longrightarrow map\ \pi_2\ xfTail = map\ tval\ trmTail \Longrightarrow$
$\forall xf{\in}set\ xfTail.\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t) \Longrightarrow$
$F\ t\ (\partial\ x) = (\llbracket(map\ (vdiff \circ \pi_1)\ xfTail \otimes trmTail)\langle t_V\ (\partial\ x)\rangle\rrbracket_t)\ (F\ t)$
and *1*:$x \notin varDiffs$ **and** *2*:$map\ \pi_2\ ((y,\ g)\ \#\ xfTail) = map\ tval\ (\vartheta\ \#\ trmTail)$
and *3*:$\forall xf{\in}set\ ((y,\ g)\ \#\ xfTail).\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t)$
hence $*$:$(\llbracket(map\ (vdiff \circ \pi_1)\ xfTail \otimes trmTail)\langle Var\ (\partial\ x)\rangle\rrbracket_t)\ (F\ t) = F\ t\ (\partial\ x)$
using *tHyp* **by** *auto*
show $F\ t\ (\partial\ x) = (\llbracket((map\ (vdiff \circ \pi_1)\ ((y,\ g)\ \#\ xfTail)) \otimes (\vartheta\ \#\ trmTail))\ \langle t_V$
$(\partial\ x)\rangle\rrbracket_t)\ (F\ t)$

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** ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*))$\langle t_V$ $(\partial\ x)\rangle = \vartheta$ **by** *simp*
also from *3 tHyp* **have** $F\ t\ (\partial\ y) = g\ (F\ t)$ **by** *simp*
moreover from *2* **have** $(\llbracket\vartheta\rrbracket_t)\ (F\ t) = g\ (F\ t)$ **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 $\partial\ x \neq \partial\ y$ **using** *vdiff-inj* **by** *auto*
from *this* **have** ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*)) $\langle t_V$ $(\partial\ x)\rangle =$
((*map* (*vdiff* $\circ$ π_1) *xfTail*) $\otimes$ *trmTail*) $\langle t_V\ (\partial\ x)\rangle$ **by** *simp*
hence *?thesis* **using** $*$ **by** *simp*}
ultimately show *?thesis* **by** *blast*
next
case *False*
then have ((*map* (*vdiff* $\circ$ π_1) ((*y*, *g*) # *xfTail*)) $\otimes$ (ϑ # *trmTail*)) $\langle t_V\ (\partial\ x)\rangle$ $= t_V\ (\partial\ x)$
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 $(\llbracket\eta\rrbracket_t)\ a = (\llbracket\eta\rrbracket_t)\ b$
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 tval tList* $\longrightarrow$ ($\exists$ ϑ.($\llbracket\vartheta\rrbracket_t$) = *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* (∂ *str*) = *0*) $\longrightarrow$
($\llbracket$((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) $\langle\partial_t\ \eta\rangle\rrbracket_t$) *st* = *0*
and *termVarsHyp*:*trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
and *listsHyp*:*map* π_2 *xfList* = *map tval uInput*
shows $(\llbracket\eta\rrbracket_t)$ *a* = *0* $\longrightarrow$ ($\forall$ *c*. (*a*,*c*) $\in$ (*ODEsystem xfList with G*) $\longrightarrow$ $(\llbracket\eta\rrbracket_t)$ *c* = *0*)
proof(*clarify*)
fix *c* **assume** *aHyp*:$(\llbracket\eta\rrbracket_t)$ *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** $([\![\eta]\!]_t)\ a = ([\![\eta]\!]_t)\ (F\ 0)$ **using** *termVarsHyp eqInVars-impl-eqInTrms* **by** *blast*
hence *obs1*:$([\![\eta]\!]_t)\ (F\ 0) = 0$ **using** *aHyp tcHyp* **by** *simp*
from *tcHyp* **have** *obs2*:$\forall r \in \{0..t\}.\ ((\lambda s.\ ([\![\eta]\!]_t)\ (F\ s))$ *has-vector-derivative* $([\![\partial_t\ \eta]\!]_t)\ (F\ r))$ (*at r within* $\{0..t\}$) **using** *derivationLemma termVarsHyp* **by** *blast*
have $\forall r \in \{0..t\}.\ \forall\ xf \in$ *set xfList*. $F\ r\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ r)$
using *tcHyp solves-store-ivpD(4)* **by** *fastforce*
hence $\forall r \in \{0..t\}.\ ([\![\partial_t\ \eta]\!]_t)\ (F\ r) = ([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle \partial_t\ \eta \rangle]\!]_t)\ (F\ r)$
using *tcHyp diff-subst-prprty-4terms termVarsHyp listsHyp* **by** *fastforce*
also from *substHyp* **have** $\forall r \in \{0..t\}.\ ([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\langle \partial_t\ \eta \rangle]\!]_t)\ (F\ r) = 0$
using *solves-store-ivpD(1) solves-store-ivpD(3) tcHyp* **by** *fastforce*
ultimately have $\forall r \in \{0..t\}.\ ((\lambda s.\ ([\![\eta]\!]_t)\ (F\ s))$ *has-vector-derivative 0*) (*at r within* $\{0..t\}$)
using *obs2* **by** *auto*
from *this* **and** *tcHyp* **have** $\forall s \in \{0..t\}.\ ((\lambda x.\ ([\![\eta]\!]_t)\ (F\ x))$ *has-derivative* $(\lambda x.\ x *_R 0))$
(*at s within* $\{0..t\}$) **by** (*metis has-vector-derivative-def*)
hence $([\![\eta]\!]_t)\ (F\ t) - ([\![\eta]\!]_t)\ (F\ 0) = (\lambda x.\ x *_R 0)\ (t - 0)$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then show $([\![\eta]\!]_t)\ c = 0$ **using** *obs1 tcHyp* **by** *auto*
qed

theorem *dInvForTrms*:
assumes $\forall\ st.\ G\ st \longrightarrow (\forall\ str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow ([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle \partial_t\ \eta \rangle]\!]_t)\ st = 0$
and *termVarsHyp*:*trmVars* $\eta \subseteq (UNIV - varDiffs)$
and *listsHyp*:*map* π_2 *xfList = map tval uInput*
and *eta-f*:$f = ([\![\eta]\!]_t)$
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 $(a,\ b) \in \lceil \lambda s.\ ([\![\eta]\!]_t)\ s = 0 \rceil$ **and** $f = ([\![\eta]\!]_t)$
from *this* **have** *aHyp*:$a = b \wedge ([\![\eta]\!]_t)\ a = 0$ **by** (*metis (full-types) d-p2r rdom-p2r-contents*)
have $([\![\eta]\!]_t)\ a = 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow ([\![\eta]\!]_t)\ c = 0)$
using *assms dInvForTrms-prelim* **by** *metis*
from *this* **and** *aHyp* **have** $\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow ([\![\eta]\!]_t)\ c = 0$ **by** *blast*
thus $(a,\ b) \in wp\ (ODEsystem\ xfList\ with\ G\ )\ \lceil \lambda s.\ ([\![\eta]\!]_t)\ s = 0 \rceil$
using *aHyp* **by** (*simp add*: *boxProgrPred-chrctrztn*)
qed

lemma *circular-motion*:
PRE $(\lambda\ s.\ (s\ ''x'') \cdot (s\ ''x'') + (s\ ''y'') \cdot (s\ ''y'') - (s\ ''r'') \cdot (s\ ''r'') = 0)$

$(ODEsystem\ [(''x'',(\lambda\ s.\ s\ ''y'')),(''y'',(\lambda\ s.\ -\ s\ ''x''))]\ with\ G)$
$POST\ (\lambda\ s.\ (s\ ''x'')\cdot(s\ ''x'') + (s\ ''y'')\cdot(s\ ''y'') - (s\ ''r'')\cdot(s\ ''r'') = 0)$
apply$(rule\text{-}tac\ \eta{=}(t_V\ ''x'')\odot(t_V\ ''x'') \oplus (t_V\ ''y'')\odot(t_V\ ''y'') \oplus (\ominus(t_V\ ''r'')\odot(t_V\ ''r''))$
and $uInput{=}[t_V\ ''y'',\ \ominus\ (t_V\ ''x'')]$ **in** $dInvForTrms)$
apply$(simp\text{-}all\ add{:}\ vdiff\text{-}def\ varDiffs\text{-}def)$
apply$(clarsimp,\ erule\text{-}tac\ x{=}''r''$ **in** $allE)$
by *simp*

lemma *diff-subst-prprty-4props*:
assumes $solves{:}\forall\ xf \in set\ xfList.\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t)$
and $tHyp{:}t \geq 0$
and $listsHyp{:}map\ \pi_2\ xfList = map\ tval\ uInput$
and $propVarsHyp{:}propVars\ \varphi \subseteq (UNIV - varDiffs)$
shows $([\![\partial_P\ \varphi]\!]_P)\ (F\ t) = ([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput){\upharpoonleft}\partial_P\ \varphi{\upharpoonright}]\!]_P)\ (F\ t)$
using *propVarsHyp* **apply**$(induction\ \varphi,\ simp\text{-}all)$
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*:
$\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle]\!]_t)\ st \geq 0$
and $termVarsHyp{:}trmVars\ \eta \subseteq (UNIV - varDiffs)$
and $listsHyp{:}map\ \pi_2\ xfList = map\ tval\ uInput$
shows $([\![\eta]\!]_t)\ a > 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow ([\![\eta]\!]_t)\ c > 0)$
and $([\![\eta]\!]_t)\ a \geq 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow ([\![\eta]\!]_t)\ c \geq 0)$
proof$(clarify)$
fix c **assume** $aHyp{:}([\![\eta]\!]_t)\ a > 0$ **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** $solves\text{-}store\text{-}ivpD(6)$ **by** *blast*
from *this* **have** $([\![\eta]\!]_t)\ a = ([\![\eta]\!]_t)\ (F\ 0)$ **using** *termVarsHyp eqInVars-impl-eqInTrms*
by *blast*
hence $obs1{:}([\![\eta]\!]_t)\ (F\ 0) > 0$ **using** *aHyp tcHyp* **by** *simp*
from *tcHyp* **have** $obs2{:}\forall r{\in}\{0..t\}.\ ((\lambda s.\ ([\![\eta]\!]_t)\ (F\ s))\ has\text{-}vector\text{-}derivative$
$([\![\partial_t\ \eta]\!]_t)\ (F\ r))\ (at\ r\ within\ \{0..t\})$ **using** *derivationLemma termVarsHyp* **by** *blast*
have $(\forall t{\geq}0.\ \forall\ xf \in set\ xfList.\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t))$
using *tcHyp* $solves\text{-}store\text{-}ivpD(4)$ **by** *blast*
hence $\forall r{\in}\{0..t\}.\ ([\![\partial_t\ \eta]\!]_t)\ (F\ r) = ([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle]\!]_t)\ (F\ r)$
using *diff-subst-prprty-4terms termVarsHyp tcHyp listsHyp* **by** *fastforce*
also from *substHyp* **have** $\forall r{\in}\{0..t\}.\ ([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle]\!]_t)\ (F\ r) \geq 0$
using $solves\text{-}store\text{-}ivpD(1)\ solves\text{-}store\text{-}ivpD(3)\ tcHyp$ **by** $(metis\ atLeastAtMost\text{-}iff)$
ultimately have $*{:}\forall r{\in}\{0..t\}.\ ([\![\partial_t\ \eta]\!]_t)\ (F\ r) \geq 0$ **by** $(simp)$

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

hence $\exists\, r \in \{0..t\}.\ (\llbracket\eta\rrbracket_t)\ (F\ t) - (\llbracket\eta\rrbracket_t)\ (F\ 0) = t \cdot (\llbracket(\partial_t\ \eta)\rrbracket_t)\ (F\ r)$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then obtain r **where** $(\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r) \geq 0 \wedge 0 \leq r \wedge r \leq t \wedge (\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ t) \geq 0$
$\wedge\ (\llbracket\eta\rrbracket_t)\ (F\ t) - (\llbracket\eta\rrbracket_t)\ (F\ 0) = t \cdot ((\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r))$ **using** $*$ *tcHyp* **by** *fastforce*
thus $(\llbracket\eta\rrbracket_t)\ c > 0$
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 $0 \leq (\llbracket\eta\rrbracket_t)\ a \longrightarrow (\forall\, c.\ (a,\ c) \in$ *ODEsystem xfList with G* $\longrightarrow 0 \leq (\llbracket\eta\rrbracket_t)\ c)$
proof(*clarify*)
fix c **assume** *aHyp*:$(\llbracket\eta\rrbracket_t)\ a \geq 0$ **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** $(\llbracket\eta\rrbracket_t)\ a = (\llbracket\eta\rrbracket_t)\ (F\ 0)$ **using** *termVarsHyp eqInVars-impl-eqInTrms*
by *blast*
hence *obs1*:$(\llbracket\eta\rrbracket_t)\ (F\ 0) \geq 0$ **using** *aHyp tcHyp* **by** *simp*
from *tcHyp* **have** *obs2*:$\forall\, r \in \{0..t\}.\ ((\lambda s.\ (\llbracket\eta\rrbracket_t)\ (F\ s))$ *has-vector-derivative*
$(\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r))$ $(at\ r\ within\ \{0..t\})$ **using** *derivationLemma termVarsHyp* **by** *blast*
have $(\forall\, t \geq 0.\ \forall\ xf \in$ *set xfList*. $F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t))$
using *tcHyp solves-store-ivpD*(*4*) **by** *blast*
from *this* **and** *tcHyp* **have** $\forall\, r \in \{0..t\}.\ (\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r) =$
$(\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle\rrbracket_t)\ (F\ r)$
using *diff-subst-prprty-4terms termVarsHyp listsHyp* **by** *fastforce*
also from *substHyp* **have** $\forall\, r \in \{0..t\}.\ (\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t$
$\eta\rangle\rrbracket_t)\ (F\ r) \geq 0$
using *solves-store-ivpD*(*1*) *solves-store-ivpD*(*3*) *tcHyp* **by** (*metis atLeastAtMost-iff*)
ultimately have $*$:$\forall\, r \in \{0..t\}.\ (\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r) \geq 0$ **by** (*simp*)
from *obs2* **and** *tcHyp* **have** $\forall\, r \in \{0..t\}.\ ((\lambda s.\ (\llbracket\eta\rrbracket_t)\ (F\ s))$ *has-derivative*
$(\lambda x.\ x *_R ((\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r))))$ $(at\ r\ within\ \{0..t\})$ **by** (*simp add*: *has-vector-derivative-def*)

hence $\exists\, r \in \{0..t\}.\ (\llbracket\eta\rrbracket_t)\ (F\ t) - (\llbracket\eta\rrbracket_t)\ (F\ 0) = t \cdot ((\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r))$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then obtain r **where** $(\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r) \geq 0 \wedge 0 \leq r \wedge r \leq t \wedge (\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ t) \geq 0$
$\wedge\ (\llbracket\eta\rrbracket_t)\ (F\ t) - (\llbracket\eta\rrbracket_t)\ (F\ 0) = t \cdot ((\llbracket\partial_t\ \eta\rrbracket_t)\ (F\ r))$ **using** $*$ *tcHyp* **by** *fastforce*
thus $(\llbracket\eta\rrbracket_t)\ c \geq 0$
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-tval*:
assumes $([\![((map\ (vdiff\ \circ\ \pi_1)\ xfList)\ \otimes\ uInput)\restriction\partial_P\ (\vartheta\ \prec\ \eta)\restriction]\!]_P)\ st$
shows $([\![((map\ (vdiff{\circ}\pi_1)\ xfList)\ \otimes\ uInput)\langle\partial_t\ (\eta\ \oplus\ (\ominus\ \vartheta))\rangle]\!]_t)\ st\ \geq\ 0$
using *assms* **by**(*auto*)

lemma *leq-pval-to-tval*:
assumes $([\![((map\ (vdiff\ \circ\ \pi_1)\ xfList)\ \otimes\ uInput)\restriction\partial_P\ (\vartheta\ \preceq\ \eta)\restriction]\!]_P)\ st$
shows $([\![((map\ (vdiff{\circ}\pi_1)\ xfList)\ \otimes\ uInput)\langle\partial_t\ (\eta\ \oplus\ (\ominus\ \vartheta))\rangle]\!]_t)\ st\ \geq\ 0$
using *assms* **by**(*auto*)

lemma *dInv-prelim*:
assumes *substHyp*:$\forall\ st.\ G\ st\ \longrightarrow\ (\forall\,str.\ str\ \notin\ (\pi_1(\!|set\ xfList|\!))\ \longrightarrow\ st\ (\partial\ str)\ =$
$0)\ \longrightarrow$
$([\![((map\ (vdiff\ \circ\ \pi_1)\ xfList)\ \otimes\ uInput)\restriction\partial_P\ \varphi\restriction]\!]_P)\ st$
and *propVarsHyp*:$propVars\ \varphi\ \subseteq\ (UNIV\ -\ varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList\ =\ map\ tval\ uInput$
shows $([\![\varphi]\!]_P)\ a\ \longrightarrow\ (\forall\ c.\ (a,c)\ \in\ (ODEsystem\ xfList\ with\ G)\ \longrightarrow\ ([\![\varphi]\!]_P)\ c)$
proof(*clarify*)
fix c **assume** *aHyp*:$([\![\varphi]\!]_P)\ a$ **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** $([\![\varphi]\!]_P)\ c$
proof(*induction* φ)
case $(Eq\ \vartheta\ \eta)$
hence *hyp*:$\forall\,st.\ G\ st\ \longrightarrow\ (\forall\,str.\ str\ \notin\ (\pi_1(\!|set\ xfList|\!))\ \longrightarrow\ st\ (\partial\ str)\ =\ 0)\ \longrightarrow$
$([\![((map\ (vdiff\ \circ\ \pi_1)\ xfList)\ \otimes\ uInput)\restriction\partial_P\ (\vartheta\ \doteq\ \eta)\restriction]\!]_P)\ st$ **by** *blast*
then have $\forall\,st.\ G\ st\ \longrightarrow\ (\forall\,str.\ str\ \notin\ (\pi_1(\!|set\ xfList|\!))\ \longrightarrow\ st\ (\partial\ str)\ =\ 0)\ \longrightarrow$
$([\![((map\ (vdiff\ \circ\ \pi_1)\ xfList)\ \otimes\ uInput)\langle\partial_t\ (\vartheta\ \oplus\ (\ominus\ \eta))\rangle]\!]_t)\ st\ =\ 0$ **by** *simp*
also have $trmVars\ (\vartheta\ \oplus\ (\ominus\ \eta))\ \subseteq\ UNIV\ -\ varDiffs$ **using** *Eq.prems*(*2*) **by** *simp*
moreover have $([\![\vartheta\ \oplus\ (\ominus\ \eta)]\!]_t)\ a\ =\ 0$ **using** *Eq.prems*(*1*) **by** *simp*
ultimately have $(\forall\,c.\ (a,\ c)\ \in\ ODEsystem\ xfList\ with\ G\ \ \longrightarrow\ ([\![\vartheta\ \oplus\ (\ominus\ \eta)]\!]_t)\ c$
$=\ 0)$
using *dInvForTrms-prelim listsHyp* **by** *blast*
hence $([\![\vartheta\ \oplus\ (\ominus\ \eta)]\!]_t)\ (F\ t)\ =\ 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $([\![\vartheta]\!]_t)\ (F\ t)\ =\ ([\![\eta]\!]_t)\ (F\ t)$ **by** *simp*
also have $([\![\vartheta\ \doteq\ \eta]\!]_P)\ c\ =\ (([\![\vartheta]\!]_t)\ (F\ t)\ =\ ([\![\eta]\!]_t)\ (F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case $(Less\ \vartheta\ \eta)$
hence $\forall\,st.\ G\ st\ \longrightarrow\ (\forall\,str.\ str\ \notin\ (\pi_1(\!|set\ xfList|\!))\ \longrightarrow\ st\ (\partial\ str)\ =\ 0)\ \longrightarrow$
$0\ \leq\ ([\![(map\ (vdiff\ \circ\ \pi_1)\ xfList\ \otimes\ uInput)\langle\partial_t\ (\eta\ \oplus\ (\ominus\ \vartheta))\rangle]\!]_t)\ st$
using *less-pval-to-tval* **by** *metis*
also from *Less.prems*(*2*)**have** $trmVars\ (\eta\ \oplus\ (\ominus\ \vartheta))\ \subseteq\ UNIV\ -\ varDiffs$ **by** *simp*
moreover have $([\![\eta\ \oplus\ (\ominus\ \vartheta)]\!]_t)\ a\ >\ 0$ **using** *Less.prems*(*1*) **by** *simp*
ultimately have $(\forall\,c.\ (a,\ c)\ \in\ ODEsystem\ xfList\ with\ G\ \ \longrightarrow\ ([\![\eta\ \oplus\ (\ominus\ \vartheta)]\!]_t)\ c$
$>\ 0)$
using *dInvForProps-prelim*(*1*) *listsHyp* **by** *blast*
hence $([\![\eta\ \oplus\ (\ominus\ \vartheta)]\!]_t)\ (F\ t)\ >\ 0$ **using** *tcHyp cHyp* **by** *simp*

from *this* **have** $([\![\eta]\!]_t)$ $(F\ t) > ([\![\vartheta]\!]_t)$ $(F\ t)$ **by** *simp*
also have $([\![\vartheta \prec \eta]\!]_P)$ $c = (([\![\vartheta]\!]_t)$ $(F\ t) < ([\![\eta]\!]_t)$ $(F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case $(Leq\ \vartheta\ \eta)$
hence $\forall st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$0 \leq ([\![(map\ (vdiff \circ \pi_1)\ xfList \otimes uInput)\langle\partial_t\ (\eta \oplus (\ominus\ \vartheta))\rangle]\!]_t)$ *st* **using** *leq-pval-to-tval*
by *metis*
also from *Leq.prems(2)***have** $trmVars\ (\eta \oplus (\ominus\ \vartheta)) \subseteq UNIV - varDiffs$ **by** *simp*
moreover have $([\![\eta \oplus (\ominus\ \vartheta)]\!]_t)$ $a \geq 0$ **using** *Leq.prems(1)* **by** *simp*
ultimately have $(\forall c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow ([\![\eta \oplus (\ominus\ \vartheta)]\!]_t)\ c$
$\geq 0)$
using *dInvForProps-prelim(2) listsHyp* **by** *blast*
hence $([\![\eta \oplus (\ominus\ \vartheta)]\!]_t)$ $(F\ t) \geq 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $(([\![\eta]\!]_t)$ $(F\ t) \geq ([\![\vartheta]\!]_t)$ $(F\ t))$ **by** *simp*
also have $([\![\vartheta \preceq \eta]\!]_P)$ $c = (([\![\vartheta]\!]_t)$ $(F\ t) \leq ([\![\eta]\!]_t)$ $(F\ t))$ **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 $\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\restriction\partial_P\ \varphi\restriction]\!]_P)$ *st*
and *termVarsHyp*:$propVars\ \varphi \subseteq (UNIV - varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList = map\ tval\ uInput$
and *phi-p*:$P = ([\![\varphi]\!]_P)$
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 $\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\restriction\partial_P\ \varphi\restriction]\!]_P)$ *st*
and *termVarsHyp*:$propVars\ \varphi \subseteq (UNIV - varDiffs)$

and *listsHyp*:*map* π_2 *xfList* = *map tval uInput*
and *impls*:$\lceil P \rceil \subseteq \lceil F \rceil \wedge \lceil F \rceil \subseteq \lceil Q \rceil$
and *phi-f*:$F = ([\![\varphi]\!]_P)$
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac* $C=([\![\varphi]\!]_P)$ **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* $\varphi=\varphi$ **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* $C = \lambda$ *s. s* ''*v*'' > *0* **in** *dCut*)
apply(*rule-tac* $\varphi=(t_C\ 0) \prec (t_V$ ''*v*''$)$ **and** *uInput*=[t_V ''*v*'']**in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*, *clarify*, *erule-tac x*=''*v*'' **in** *allE*, *simp*)
apply(*rule-tac* $C = \lambda$ *s. s* ''*x*'' > *0* **in** *dCut*)
apply(*rule-tac* $\varphi=(t_C\ 0) \prec (t_V$ ''*x*''$)$ **and** *uInput*=[t_V ''*v*'']
  and $F=\lambda$ *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* $C = \lambda$ *s. s* ''*a*'' > *0* **in** *dCut*)
apply(*rule-tac* $\varphi = (t_C\ 0) \prec (t_V$ ''*a*''$)$ **and** *uInput*=[t_V ''*v*'', t_V ''*a*'']**in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*, *clarify*, *erule-tac x*=''*a*'' **in** *allE*, *simp*)
apply(*rule-tac* $C = \lambda$ *s. s* ''*v*'' $\geq$ *0* **in** *dCut*)
apply(*rule-tac* $\varphi = (t_C\ 0) \preceq (t_V$ ''*v*''$)$ **and** *uInput*=[t_V ''*v*'', t_V ''*a*''] **in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*)
apply(*rule-tac* $C = \lambda$ *s. s* ''*x*'' > *s* ''*y*'' **in** *dCut*)
apply(*rule-tac* $\varphi = (t_V$ ''*y*''$) \prec (t_V$ ''*x*''$)$ **and** *uInput*=[t_V ''*v*'', t_V ''*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