CPSVerification

By Jonathan

May 8, 2018

Contents

1 VC_diffKAD

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

begin

1.1 Stack Theories Preliminaries: VC_KAD and ODEs

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

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

notation *p2r* ($\lceil$-$\rceil$)
  and *r2p* ($\lfloor$-$\rfloor$)

and *Set.image* (-(|-|))
and *Product-Type.prod.fst* (π_1)
and *Product-Type.prod.snd* (π_2)
and *List.zip* (**infixl** $\otimes$ *63*)
and *rel-ad* ($\Delta^c{}_1$)

This and more notation is explained by the following lemma.

lemma shows $\lceil P \rceil = \{(s, s) \mid s.\ P\ s\}$
and $\lfloor R \rfloor = (\lambda x.\ x \in r2s\ R)$
and $r2s\ R = \{x \mid x.\ \exists\ y.\ (x,y) \in R\}$
and $\pi_1\ (x,y) = x \wedge \pi_2\ (x,y) = y$
and $\Delta^c{}_1\ R = \{(x, x) \mid x.\ \nexists y.\ (x, y) \in R\}$
and $wp\ R\ Q = \Delta^c{}_1\ (R\ ;\ \Delta^c{}_1\ Q)$
and $[x1,x2,x3,x4] \otimes [y1,y2] = [(x1,y1),(x2,y2)]$
and $\{a..b\} = \{x.\ a \leq x \wedge x \leq b\}$
and $\{a<..<b\} = \{x.\ a < x \wedge x < b\}$
and $(x\ solves\text{-}ode\ f)\ \{0..t\}\ R = ((x\ has\text{-}vderiv\text{-}on\ (\lambda t.\ f\ t\ (x\ t)))\ \{0..t\} \wedge x \in \{0..t\} \to R)$
and $f \in A \to B = (f \in \{f.\ \forall\ x.\ x \in A \longrightarrow (f\ x) \in B\})$
and $(x\ has\text{-}vderiv\text{-}on\ x')\{0..t\} =$
$(\forall\ r \in \{0..t\}.\ (x\ has\text{-}vector\text{-}derivative\ x'\ r)\ (at\ r\ within\ \{0..t\}))$
and $(x\ has\text{-}vector\text{-}derivative\ x'\ r)\ (at\ r\ within\ \{0..t\}) =$
$(x\ has\text{-}derivative\ (\lambda x.\ x *_R x'\ r))\ (at\ r\ within\ \{0..t\})$
apply(*simp-all add*: *p2r-def r2p-def rel-ad-def rel-antidomain-kleene-algebra.fbox-def*

solves-ode-def has-vderiv-on-def)
apply(*blast*, *fastforce*, *fastforce*)
using *has-vector-derivative-def* **by** *auto*

Observe also, the following consequences and facts:

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

proposition $\Delta^c{}_1\ R = Id - \{(s, s) \mid s.\ s \in (\pi_1(|R|))\}$
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 *cons-eq-zipE*:
$(x,\ y)\ \#\ tail = xList \otimes yList \Longrightarrow \exists xTail\ yTail.\ x\ \#\ xTail = xList \wedge y\ \#\ yTail$
$= yList$
by(*induction xList*, *simp-all*, *induction yList*, *simp-all*)

proposition *set-zip-left-rightD*:
$(x,\ y) \in set\ (xList \otimes yList) \Longrightarrow x \in set\ xList \wedge y \in set\ yList$
apply(*rule conjI*)
apply(*rule-tac y=y* **and** *ys=yList* **in** *set-zip-leftD*, *simp*)
apply(*rule-tac x=x* **and** *xs=xList* **in** *set-zip-rightD*, *simp*)
done

declare *zip-map-fst-snd* [*simp*]

1.2 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* ⇒ *string* (∂ - [*55*] *70*) **where**
$(\partial\ x) = ''d[''@x@'']''$

definition *varDiffs* :: *string set* **where**
$varDiffs = \{y.\ \exists\ x.\ y = \partial\ x\}$

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

proposition *vdiff-noFixPoints*:$x \neq (\partial\ x)$
by(*simp add*: *vdiff-def*)

lemma *varDiffsI*:$x = (\partial\ z) \Longrightarrow x \in$ *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*:$(\partial\ x) \in$ *varDiffs*
by (*simp add*: *varDiffsI*)

1.2.1 (primed) dSolve preliminaries

This subsubsection is to define a function that takes a system of ODEs (expressed as a list $xfList$), a presumed solution $uInput = [u_1, \dots, u_n]$, a state s and a time t, and outputs the induced flow $sol\, s[xfList \leftarrow uInput]\, t$.

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

proposition *varDiffs-to-zero-vdiff*[*simp*]: (*sol s*) $(\partial\ x) = 0$
apply(*simp add*: *override-on-def varDiffs-def*)
by *auto*

proposition *varDiffs-to-zero-beginning*[*simp*]: *take 2 x* $\neq$ $''d[''$ $\Longrightarrow$ (*sol s*) $x = s$ 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 s* = *s*|
state-list-upd (*uxf* # *tail*) *t s* = (*state-list-upd tail t s*)
($(\pi_1\ (\pi_2\ uxf)) := (\pi_1\ uxf)\ t\ s$,
$\partial\ (\pi_1\ (\pi_2\ uxf)) :=$ (*if* $t = 0$ *then* $(\pi_2\ (\pi_2\ uxf))\ s$
else vderiv-of (λ *r*. $(\pi_1\ uxf)$ *r s*) $\{0{<}..{<}\ (2 *_R t)\}$ *t*))

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

proposition *state-list-cross-upd-empty*[*simp*]: $(s[[]\leftarrow list]\ t) = s$
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*) $\Longrightarrow$ (*s*[*xftail*$\leftarrow$*utail*] *t*) *x* = *u t s*
and *disjHyp*:(*u*, *x*, *f*) = (*v*, *y*, *g*) $\vee$ (*u*, *x*, *f*) $\in$ *set* (*utail* $\otimes$ *xftail*)
shows (*s*[(*y*, *g*) # *xftail*$\leftarrow$*v* # *utail*] *t*) *x* = *u t s*
using *disjHyp* **proof**
  assume (*u*, *x*, *f*) = (*v*, *y*, *g*)
  hence (*s*[(*y*, *g*) # *xftail*$\leftarrow$*v* # *utail*] *t*) *x* = ((*s*[*xftail*$\leftarrow$*utail*] *t*)(*x* := *u t s*,
  ∂ *x* := *if t* = *0 then f s else vderiv-of* (λ *r*. *u r s*) {*0*<..< (*2* $*_R$ *t*)} *t*)) *x* **by** *simp*
  also have ... = *u t s* **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*:(*s*[*xftail*$\leftarrow$*utail*] *t*) *x* = *u t s* **by** *fastforce*
  from *yTailHyp* **and** *distHyp* **have** *2*:*y* $\neq$ *x* **using** *set-zip-left-rightD* **by** *force*
  from *yTailHyp* **and** *varHyp* **have** *1*:*x* $\neq$ ∂ *y*
  using *set-zip-left-rightD vdiff-invarDiffs* **by** *fastforce*
  from *1* **and** *2* **have** (*s*[(*y*, *g*) # *xftail*$\leftarrow$*v* # *utail*] *t*) *x* = (*s*[*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 (*s*[*xfList*$\leftarrow$*uInput*] *t*) *x* = *u t s*
using *assms* **apply**(*induct xfList uInput arbitrary*: *x rule*: *list-induct2'*, *simp*, *simp*, *simp*)
by(*clarify*, *rule inductive-state-list-cross-upd-its-vars*, *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*. (*s*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on s g varDiffs*

and $\forall xf \in set\ (xf\ \#\ xfTail).\ \pi_1\ xf \notin varDiffs$
and $\forall uxf \in set\ (u\ \#\ uTail \otimes xf\ \#\ xfTail).\ \pi_1\ uxf\ 0\ s = s\ (\pi_1\ (\pi_2\ uxf))$
shows $\exists g.\ (s[xf\ \#\ xfTail \leftarrow u\ \#\ uTail]\ 0) = override\text{-}on\ s\ g\ varDiffs$
proof−
let $?gLHS{=}(s[(xf\ \#\ xfTail) \leftarrow (u\ \#\ uTail)]\ 0)$
have $observ{:}\partial\ (\pi_1\ xf) \in varDiffs$ **by** $(auto\ simp{:}\ varDiffs\text{-}def)$
from $assms(1)$ **obtain** g **where** $(s[xfTail \leftarrow uTail]\ 0) = override\text{-}on\ s\ g\ varDiffs$
by *force*
then have $?gLHS = (override\text{-}on\ s\ g\ varDiffs)(\pi_1\ xf := u\ 0\ s,\ \partial\ (\pi_1\ xf) := \pi_2\ xf\ s)$ **by** *simp*
also have $\ldots = (override\text{-}on\ s\ g\ varDiffs)(\partial\ (\pi_1\ xf) := \pi_2\ xf\ s)$
using *override-on-def varDiffs-def assms* **by** *auto*
also have ... $= (override\text{-}on\ s\ (g(\partial\ (\pi_1\ xf) := \pi_2\ xf\ s))\ varDiffs)$
using *observ* **and** *override-on-upd* **by** *force*
ultimately show *?thesis* **by** *auto*
qed

theorem *state-list-cross-upd-its-dvars*:
assumes $lengthHyp{:}length\ xfList = length\ uInput$
and $varsHyp{:}\forall\ xf \in set\ xfList.\ \pi_1\ xf \notin varDiffs$
and $solHyp1{:}\forall uxf \in set\ (uInput \otimes xfList).\ (\pi_1\ uxf)\ 0\ s = s\ (\pi_1\ (\pi_2\ uxf))$
shows $\exists\ g.\ (s[xfList \leftarrow uInput]\ 0) = (override\text{-}on\ s\ g\ varDiffs)$
using *assms* **proof**$(induct\ xfList\ uInput\ rule{:}\ list\text{-}induct2')$
case *1*
  have $(s[[] \leftarrow []]\ 0) = override\text{-}on\ s\ s\ varDiffs$
  unfolding *override-on-def* **by** *simp*
  thus *?case* **by** *metis*
next
  case $(2\ xf\ xfTail)$
  have $(s[(xf\ \#\ xfTail) \leftarrow []]\ 0) = override\text{-}on\ s\ s\ varDiffs$
  unfolding *override-on-def* **by** *simp*
  thus *?case* **by** *metis*
next
  case $(3\ u\ utail)$
  have $(s[[] \leftarrow utail]\ 0) = override\text{-}on\ s\ s\ varDiffs$
  unfolding *override-on-def* **by** *simp*
  thus *?case* **by** *force*
next
  case $(4\ xf\ xfTail\ u\ uTail)$
  then have $\exists g.\ (s[xfTail \leftarrow uTail]\ 0) = override\text{-}on\ s\ g\ varDiffs$ **by** *simp*
  thus *?case* **using** *inductive-state-list-cross-upd-its-dvars 4.prems* **by** *blast*
qed

lemma *vderiv-unique-within-open-interval*:
assumes $(f\ has\text{-}vderiv\text{-}on\ f')\ \{0{<}..{<}\ t\}$ **and** $t{>}0$
  and $(f\ has\text{-}vderiv\text{-}on\ f'')\{0{<}..{<}\ t\}$ **and** $tauHyp{:}\tau \in \{0{<}..{<}\ t\}$
shows $f'\ \tau = f''\ \tau$
using *assms* **apply**$(simp\ add{:}\ has\text{-}vderiv\text{-}on\text{-}def\ has\text{-}vector\text{-}derivative\text{-}def)$
using *frechet-derivative-unique-within-open-interval* **by** $(metis\ box\text{-}real(1)\ scaleR\text{-}one$

tauHyp)

lemma *has-vderiv-on-cong-open-interval*:
assumes *gHyp*:$\forall\ \tau > 0.\ f\ \tau = g\ \tau$ **and** *tHyp*: $t{>}0$
and *fHyp*:(f *has-vderiv-on* f') $\{0{<}..{<}t\}$
shows (g *has-vderiv-on* f') $\{0{<}..{<}t\}$
proof−
from *gHyp* **have** $\bigwedge\tau.\ \tau \in \{0{<}..{<}\ t\} \Longrightarrow f\ \tau = g\ \tau$ **using** *tHyp* **by** *force*
hence *eqDs*:(f *has-vderiv-on* f') $\{0{<}..{<}t\}$ = (g *has-vderiv-on* f') $\{0{<}..{<}t\}$
apply(*rule-tac has-vderiv-on-cong*) **by** *auto*
thus (g *has-vderiv-on* f') $\{0{<}..{<}t\}$ **using** *eqDs fHyp* **by** *simp*
qed

lemma *closed-vderiv-on-cong-to-open-vderiv*:
assumes *gHyp*:$\forall\ \tau > 0.\ f\ \tau = g\ \tau$
and *fHyp*:$\forall\, t{\geq}0.$ (f *has-vderiv-on* f') $\{0..t\}$
and *tHyp*: $t{>}0$ **and** *cHyp*: $c > 1$
shows *vderiv-of* g $\{0{<}..{<}\ (c *_R\ t)\}\ t = f'\ t$
proof−
have *ctHyp*:$c \cdot t > 0$ **using** *tHyp* **and** *cHyp* **by** *auto*
from *fHyp* **have** (f *has-vderiv-on* f') $\{0{<}..{<}c \cdot t\}$ **using** *has-vderiv-on-subset*
by (*metis greaterThanLessThan-subseteq-atLeastAtMost-iff less-eq-real-def*)
then have *derivHyp*:(g *has-vderiv-on* f') $\{0{<}..{<}c \cdot t\}$
using *gHyp ctHyp* **and** *has-vderiv-on-cong-open-interval* **by** *blast*
hence f'*Hyp*:$\forall\ f''.$ (g *has-vderiv-on* f'') $\{0{<}..{<}c \cdot t\} \longrightarrow (\forall\ \tau \in \{0{<}..{<}\ c \cdot t\}.$
$f'\ \tau = f''\ \tau)$
using *vderiv-unique-within-open-interval ctHyp* **by** *blast*
also have (g *has-vderiv-on* (*vderiv-of* g $\{0{<}..{<}\ (c *_R\ t)\}$)) $\{0{<}..{<}c \cdot t\}$
by(*simp add*: *vderiv-of-def*, *metis derivHyp someI-ex*)
ultimately show*vderiv-of* g $\{0{<}..{<}c *_R\ t\}\ t = f'\ t$ **using** *tHyp cHyp* **by** *force*
qed

lemma *vderiv-of-to-sol-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 *solHyp2*:$\forall\, t{\geq}0.$ (($\lambda\tau.$ (*sol s*[*xfList*←*uInput*] τ) x)
has-vderiv-on ($\lambda\tau.\ f$ (*sol s*[*xfList*←*uInput*] τ))) $\{0..t\}$
and *tHyp*: $t{>}0$ **and** *uxfHyp*:(u, x, f) $\in$ *set* (*uInput* $\otimes$ *xfList*)
shows *vderiv-of* ($\lambda\tau.\ u\ \tau$ (*sol s*)) $\{0{<}..{<}\ (2 *_R\ t)\}\ t = f$ (*sol s*[*xfList*←*uInput*]
t)
apply(*rule-tac* f=($\lambda\tau.$ (*sol s*[*xfList*←*uInput*] τ) x) **in** *closed-vderiv-on-cong-to-open-vderiv*)
subgoal using *assms* **and** *state-list-cross-upd-its-vars* **by** *metis*
by(*simp-all add*: *solHyp2 tHyp*)

lemma *inductive-to-sol-zero-its-dvars*:
assumes *eqFuncs*:$\forall\ s.\ \forall\, g.\ \forall$ *xf*$\in$*set* ((x, f) # *xfs*). π_2 *xf* (*override-on s g varDiffs*)
= π_2 *xf s*
and *eqLengths*:*length* ((x, f) # *xfs*) = *length* (u # *us*)

and *distinct*:*distinct* (*map* π_1 ((*x*, *f*) # *xfs*))
and *vars*:∀ *xf*∈*set* ((*x*, *f*) # *xfs*). π_1 *xf* ∉ *varDiffs*
and *solHyp1*:∀ *uxf*∈*set* ((*u* # *us*) ⊗ ((*x*, *f*) # *xfs*)). π_1 *uxf 0* (*sol s*) = *sol s* (π_1 (π_2 *uxf*))
and *disjHyp*:(*y*, *g*) = (*x*, *f*) ∨ (*y*, *g*) ∈ *set xfs*
and *indHyp*:(*y*, *g*) ∈ *set xfs* ⟹ (*sol s*[*xfs*←*us*] *0*) (∂ *y*) = *g* (*sol s*[*xfs*←*us*] *0*)
shows (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*) (∂ *y*) = *g* (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*)
proof−
from *assms* **obtain** *h1* **where** *h1Def*:(*sol s*[((*x*, *f*) # *xfs*)←(*u* # *us*)] *0*) = (*override-on* (*sol s*) *h1 varDiffs*) **using** *state-list-cross-upd-its-dvars* **by** *blast*
from *disjHyp* **show** (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*) (∂ *y*) = *g* (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*)
proof
  assume *eqHeads*:(*y*, *g*) = (*x*, *f*)
  then have *g* (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*) = *f* (*sol s*) **using** *h1Def eqFuncs* **by** *simp*
  also have ... = (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*) (∂ *y*) **using** *eqHeads* **by** *auto*
  ultimately show *?thesis* **by** *linarith*
next
  assume *tailHyp*:(*y*, *g*) ∈ *set xfs*
  then have *y* ≠ *x* **using** *distinct set-zip-left-rightD* **by** *force*
  hence ∂ *x* ≠ ∂ *y* **by**(*simp add*: *vdiff-def*)
  have *x* ≠ ∂ *y* **using** *vars vdiff-invarDiffs* **by** *auto*
  obtain *h2* **where** *h2Def*:(*sol s*[*xfs*←*us*] *0*) = *override-on* (*sol s*) *h2 varDiffs*
  using *state-list-cross-upd-its-dvars eqLengths distinct vars* **and** *solHyp1* **by** *force*
  have (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*) (∂ *y*) = *g* (*sol s*[*xfs*←*us*] *0*)
  using *tailHyp indHyp* ‹*x* ≠ ∂ *y*› **and** ‹∂ *x* ≠ ∂ *y*› **by** *simp*
  also have ... = *g* (*override-on* (*sol s*) *h2 varDiffs*) **using** *h2Def* **by** *simp*
  also have ... = *g* (*sol s*) **using** *eqFuncs* **and** *tailHyp* **by** *force*
  also have ... = *g* (*sol s*[(*x*, *f*) # *xfs*←*u* # *us*] *0*)
  using *eqFuncs h1Def tailHyp* **and** *eq-snd-iff* **by** *fastforce*
  ultimately show *?thesis* **by** *simp*
  qed
qed

lemma *to-sol-zero-its-dvars*:
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 *solHyp1*:∀ *uxf* ∈ *set* (*uInput* ⊗ *xfList*). (π_1 *uxf*) *0* (*sol s*) = (*sol s*) (π_1 (π_2 *uxf*))
and *ygHyp*:(*y*, *g*) ∈ *set xfList*
shows (*sol s*[*xfList*←*uInput*] *0*)(∂ *y*) = *g* (*sol s*[*xfList*←*uInput*] *0*)
using *assms* **apply**(*induct xfList uInput rule*: *list-induct2′*, *simp*, *simp*, *simp*, *clarify*)
by(*rule inductive-to-sol-zero-its-dvars*, *simp-all*)

lemma *inductive-to-sol-greater-than-zero-its-dvars*:
assumes *lengthHyp*:*length* $((y,\ g)\ \#\ xfs) = length\ (v\ \#\ vs)$
and *distHyp*:*distinct* $(map\ \pi_1\ ((y,\ g)\ \#\ xfs))$
and *varHyp*: $\forall xf{\in}set\ ((y,\ g)\ \#\ xfs).\ \pi_1\ xf \notin varDiffs$
and *indHyp*:$(u,x,f) \in set\ (vs \otimes xfs) \Longrightarrow (s[xfs{\leftarrow}vs]t)(\partial\ x) = vderiv\text{-}of\ (\lambda r.\ u\ r\ s)\ \{0{<}..{<}2{*_R}t\}\ t$
and *disjHyp*:$(v,\ y,\ g) = (u,\ x,\ f) \lor (u,\ x,\ f) \in set\ (vs \otimes xfs)$ **and** *tHyp*:$t > 0$
shows $(s[(y,\ g)\ \#\ xfs{\leftarrow}v\ \#\ vs]\ t)\ (\partial\ x) = vderiv\text{-}of\ (\lambda r.\ u\ r\ s)\ \{0{<}..{<}2\ {*_R}\ t\}\ t$
proof−
let $?lhs = ((s[xfs{\leftarrow}vs]\ t)(y := v\ t\ s,\ \partial\ y := vderiv\text{-}of\ (\lambda\ r.\ v\ r\ s)\ \{0{<}..{<}\ (2\ \cdot\ t)\}\ t))\ (\partial\ x)$
let $?rhs = vderiv\text{-}of\ (\lambda r.\ u\ r\ s)\ \{0{<}..{<}\ (2\ \cdot\ t)\}\ t$
have $(s[(y,\ g)\ \#\ xfs{\leftarrow}v\ \#\ vs]\ t)\ (\partial\ x) = ?lhs$ **using** *tHyp* **by** *simp*
also have $vderiv\text{-}of\ (\lambda r.\ u\ r\ s)\ \{0{<}..{<}2\ {*_R}\ t\}\ t = ?rhs$ **by** *simp*
ultimately have *obs*:$?thesis = (?lhs = ?rhs)$ **by** *simp*
from *disjHyp* **have** $?lhs = ?rhs$
proof
  assume *uxfEq*:$(v,\ y,\ g) = (u,\ x,\ f)$
  then have $?lhs = vderiv\text{-}of\ (\lambda\ r.\ u\ r\ s)\ \{0{<}..{<}\ (2\ \cdot\ t)\}\ t$ **by** *simp*
  also have $vderiv\text{-}of\ (\lambda\ r.\ u\ r\ s)\ \{0{<}..{<}\ (2\ \cdot\ t)\}\ t = ?rhs$ **using** *uxfEq* **by** *simp*
  ultimately show $?lhs = ?rhs$ **by** *simp*
next
  assume *sygTail*:$(u,\ x,\ f) \in set\ (vs \otimes xfs)$
  from *this* **have** $y \neq x$ **using** *distHyp set-zip-left-rightD* **by** *force*
  hence $\partial\ x \neq \partial\ y$ **by**(*simp add*: *vdiff-def*)
  have $y \neq \partial\ x$ **using** *varHyp* **using** *vdiff-invarDiffs* **by** *auto*
  then have $?lhs = (s[xfs{\leftarrow}vs]\ t)\ (\partial\ x)$ **using** $\langle y \neq \partial\ x\rangle$ **and** $\langle\partial\ x \neq \partial\ y\rangle$ **by** *simp*
  also have $(s[xfs{\leftarrow}vs]\ t)\ (\partial\ x) = ?rhs$ **using** *indHyp sygTail* **by** *simp*
  ultimately show $?lhs = ?rhs$ **by** *simp*
qed
from *this* **and** *obs* **show** *?thesis* **by** *simp*
qed

lemma *to-sol-greater-than-zero-its-dvars*:
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 *uxfHyp*:$(u,\ x,\ f) \in set\ (uInput \otimes xfList)$ **and** *tHyp*:$t > 0$
shows $(s[xfList{\leftarrow}uInput]\ t)\ (\partial\ x) = vderiv\text{-}of\ (\lambda\ r.\ u\ r\ s)\ \{0{<}..{<}\ (2\ {*_R}\ t)\}\ t$
using *assms* **apply**(*induct xfList uInput rule*: *list-induct2'*, *simp*, *simp*, *simp*, *clarify*)
by(*rule-tac f=f* **in** *inductive-to-sol-greater-than-zero-its-dvars*, *auto*)

1.2.2 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*) ($[\![\text{-}]\!]_t$ [*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* η

fun *substList* ::(*string* $\times$ *trms*) *list* $\Rightarrow$ *trms* $\Rightarrow$ *trms* (-$\langle$-$\rangle$ [*54*] *80*) **where**
xtList$\langle t_C\ r\rangle = t_C\ r|$
$[]\langle t_V\ x\rangle = t_V\ x|$

$((y,\xi)\ \#\ xtTail)\langle Var\ x\rangle = (if\ x = y\ then\ \xi\ else\ xtTail\langle Var\ x\rangle)|$
$xtList\langle\ominus\ \vartheta\rangle = \ominus\ (xtList\langle\vartheta\rangle)|$
$xtList\langle\vartheta \oplus \eta\rangle = (xtList\langle\vartheta\rangle) \oplus (xtList\langle\eta\rangle)|$
$xtList\langle\vartheta \odot \eta\rangle = (xtList\langle\vartheta\rangle) \odot (xtList\langle\eta\rangle)$

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

lemma *substList-help1*:$set\ (map\ \pi_1\ ((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)) \subseteq varDiffs$
apply(*induct xfList uInput rule*: *list-induct2'*, *simp-all add*: *varDiffs-def*)
by *auto*

lemma *substList-help2*:
assumes $trmVars\ \eta \subseteq (UNIV - varDiffs)$
shows $((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\langle\eta\rangle = \eta$
using *assms substList-help1 substList-on-compl-of-varDiffs* **by** *blast*

lemma *substList-cross-vdiff-on-non-ocurring-var*:
assumes $x \notin set\ list1$
shows $((map\ vdiff\ list1) \otimes list2)\langle t_V\ (\partial\ x)\rangle = t_V\ (\partial\ x)$
using *assms* **apply**(*induct list1 list2 rule*: *list-induct2'*, *simp*, *simp*, *clarsimp*)
by(*simp add*: *vdiff-def*)

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

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

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

begin

1.3 Phase Space Relational Semantics

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.* $*$)
$(\forall\ t \geq 0.\ G\ (\varphi_S\ t) \wedge\ (\forall\ xf \in set\ xfList.\ \varphi_S\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (\varphi_S\ t)) \wedge$
($*$ *F preserves the rest of the variables and F sends derivs of constants to 0.* $*$)
$(\forall\ y.\ (y \notin (\pi_1 (|set\ xfList|)) \cup varDiffs \longrightarrow \varphi_S\ t\ y = s\ y) \wedge$
$\quad (y \notin (\pi_1 (|set\ xfList|)) \longrightarrow \varphi_S\ t\ (\partial\ y) = 0)) \wedge$
($*$ *F solves the induced IVP.* $*$)
$(\forall\ xf \in set\ xfList.\ ((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))\ solves\text{-}ode\ (\lambda\ t.\lambda\ r.(\pi_2\ xf)\ (\varphi_S\ t)))\ \{0..t\}$
$UNIV \wedge$
$\varphi_S\ 0\ (\pi_1\ xf) = s(\pi_1\ xf)))$

lemma *solves-store-ivpI*:
assumes $\forall\ t \geq 0.\ G\ (\varphi_S\ t)$
  and $\forall\ t \geq 0.\forall\ xf \in set\ xfList.\ (\varphi_S\ t\ (\partial\ (\pi_1\ xf))) = (\pi_2\ xf)\ (\varphi_S\ t)$
  and $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (|set\ xfList|)) \cup varDiffs \longrightarrow \varphi_S\ t\ y = s\ y$
  and $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (|set\ xfList|)) \longrightarrow \varphi_S\ t\ (\partial\ y) = 0$
  and $\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ ((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))\ solves\text{-}ode\ (\lambda\ t.\lambda\ r.(\pi_2\ xf)$
$(\varphi_S\ t)))\ \{0..t\}\ UNIV$
  and $\forall\ xf \in set\ xfList.\ \varphi_S\ 0\ (\pi_1\ xf) = s(\pi_1\ xf)$
shows φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
apply(*simp add*: *solvesStoreIVP-def*, *safe*)
using *assms* **apply** *simp-all*
by(*force*,*force*,*force*)

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 $\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ ((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))\ solves\text{-}ode\ (\lambda\ t.\lambda\ r.(\pi_2\ xf)$
$(\varphi_S\ t)))\ \{0..t\}\ UNIV$
  and $\forall\ xf \in set\ xfList.\ \varphi_S\ 0\ (\pi_1\ xf) = s(\pi_1\ xf)$
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*(*6*) **have** $x \in (\pi_1 (|set\ xfList|)) \Longrightarrow \varphi_S\ 0\ x = s$
x **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** ‹$x \notin varDiffs$› **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 $((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))$ *solves-ode* $(\lambda\ t.\lambda\ r.(\pi_2\ xf)\ (\varphi_S\ t)))\ \{0..t\}\ UNIV$
using *assms solves-store-ivpE(5)* **by** *simp*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s andGuard G*
  and $(x,f) \in set\ xfList$
shows $\varphi_S\ 0\ x = s\ x$
using *assms solves-store-ivpE(6)* **by** *fastforce*

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

and $y \notin$ *varDiffs*
shows φ_S *0* $y = s\ y$
using *assms solves-store-ivpE*(*7*) **by** *simp*

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

1.4 Derivation of Differential Dynamic Logic Rules

1.4.1 "Differential Weakening"

lemma *wlp-evol-guard*:*Id* ⊆ *wp* (*ODEsystem xfList with G*) ⌈*G*⌉
apply(*simp add*: *rel-antidomain-kleene-algebra.fbox-def rel-ad-def guarDiffEqtn-def p2r-def*)
using *solves-store-ivpD*(*1*) **by** *force*

theorem *dWeakening*:
assumes *guardImpliesPost*: ⌈*G*⌉ ⊆ ⌈*Q*⌉
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*)

1.4.2 "Differential Cut"

lemma *condAfterEvol-remainsAlongEvol*:
assumes *boxDiffC*:(*s*, *s*) ∈ *wp* (*ODEsystem xfList with G*) ⌈*C*⌉
and *FisSol*:*solvesStoreIVP* φ_S *xfList s G*
and *tHyp*:$0 \leq t$
shows G (φ_S t) ∧ C (φ_S t)
proof−
from *boxDiffC* **have** ∀ *c*. (*s*,*c*) ∈ (*ODEsystem xfList with G*) ⟶ *C c*
by (*simp add*: *boxProgrPred-chrctrztn*)
also from *tHyp* **have** (*s*, φ_S t) ∈ (*ODEsystem xfList with G*)
using *FisSol guarDiffEqtn-def* **by** *auto*
ultimately show G (φ_S t) ∧ C (φ_S t)
using *solves-store-ivpD*(*1*) *tHyp FisSol* **by** *blast*
qed

lemma *condAfterEvol-isGuard*:
assumes *boxDiffC*:(*s*, *s*) ∈ *wp* (*ODEsystem xfList with G*) ⌈*C*⌉
assumes *FisSol*:*solvesStoreIVP* φ_S *xfList s G*
shows *solvesStoreIVP* φ_S *xfList s* (λ*s*. *G s* ∧ *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* (λ*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

1.4.3 "Solve Differential Equation"

lemma *prelim-dSolve*:
assumes *solHyp*:(λ*t. sol s*[*xfList*$\leftarrow$*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*$\leftarrow$*uInput*] *t*) = *X t*)
and *diffAssgn*: $\forall$ *t*$\geq$*0*. *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) $\longrightarrow$ *Q* (*sol s*[*xfList*$\leftarrow$*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* $\Rightarrow$ *real store*
where *FHyp*:*t*$\geq$*0* $\wedge$ φ_S *t* = *c* $\wedge$ *solvesStoreIVP* φ_S *xfList s G* **using** *guarDiffEqtn-def* **by** *auto*
from *this* **and** *uniqHyp* **have** (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) = φ_S *t* **by** *blast*
then have *cHyp*:*c* = (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) **using** *FHyp* **by** *simp*
from *solHyp* **have** *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) **by** (*simp add*: *solvesStoreIVP-def FHyp*)
then show *Q c* **using** *diffAssgn FHyp cHyp* **by** *auto*
qed

theorem *wlp-guard-inv*:
assumes *solHyp*:*solvesStoreIVP* (λ*t. sol s*[*xfList*$\leftarrow$*uInput*] *t*) *xfList s G*
and *uniqHyp*:$\forall$ *X*. *solvesStoreIVP X xfList s G* $\longrightarrow$ ($\forall$ *t* $\geq$ *0*. (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) = *X t*)
and *diffAssgn*: $\forall$ *t*$\geq$*0*. *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) $\longrightarrow$ *Q* (*sol s*[*xfList*$\leftarrow$*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* (λ*t*. *sol s*[*xfList*$\leftarrow$*uInput*] *t*) *xfList s G*
and *uniqHyp*:$\forall$ *s*. $\forall$ *X*. *solvesStoreIVP X xfList s G* $\longrightarrow$ ($\forall$ *t* $\geq$ *0*.(*sol s*[*xfList*$\leftarrow$*uInput*] *t*) = *X t*)
and *diffAssgn*: $\forall$ *s*. *P s* $\longrightarrow$ ($\forall$ *t*$\geq$*0*. *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) $\longrightarrow$ *Q* (*sol s*[*xfList*$\leftarrow$*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 *conds4vdiffs-prelim*:
assumes *funcsHyp*:$\forall$ *s g*. $\forall$ *xf*$\in$*set xfList*. π_2 *xf* (*override-on s g varDiffs*) = π_2 *xf s*
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 *solHyp1*:$\forall$ *uxf* $\in$ *set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*sol s*) = (*sol s*) (π_1 (π_2 *uxf*))
and *solHyp2*:$\forall$ *t*$\geq$*0*. (($\lambda\tau$. (*sol s*[*xfList*$\leftarrow$*uInput*] τ) *x*) *has-vderiv-on* ($\lambda\tau$. *f* (*sol s*[*xfList*$\leftarrow$*uInput*] τ))) {*0*..*t*}
and *xfHyp*:(*x*, *f*) $\in$ *set xfList* **and** *tHyp*:*t* $\geq$ *0*
shows (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) (∂ *x*) = *f* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)
proof−
from *xfHyp* **obtain** *u* **where** *xfuHyp*: (*u*,*x*,*f*) $\in$ *set* (*uInput* $\otimes$ *xfList*)
by (*metis in-set-impl-in-set-zip2 lengthHyp*)
show (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) (∂ *x*) =*f* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)
  proof(*cases t=0*)
  case *True*
    have (*sol s*[*xfList*$\leftarrow$*uInput*] *0*) (∂ *x*) = *f* (*sol s*[*xfList*$\leftarrow$*uInput*] *0*)
    using *assms* **and** *to-sol-zero-its-dvars* **by** *blast*
    then show *?thesis* **using** *True* **by** *blast*
  next
    case *False*
    from *this* **have** *t* > *0* **using** *tHyp* **by** *simp*
    hence (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) (∂ *x*) = *vderiv-of* (λ *r*. *u r* (*sol s*)) {*0*<..< (*2* $*_R$ *t*)} *t*
    using *xfuHyp assms to-sol-greater-than-zero-its-dvars* **by** *blast*
    also have *vderiv-of* (λ*r*. *u r* (*sol s*)) {*0*<..< (*2* $*_R$ *t*)} *t* = *f* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)
    using *assms xfuHyp* ‹*t* > *0*› **and** *vderiv-of-to-sol-its-vars* **by** *blast*
    ultimately show *?thesis* **by** *simp*
  qed
qed

lemma *conds4vdiffs*:
assumes *funcsHyp*:$\forall s\ g.\ \forall xf{\in}set\ xfList.\ \pi_2\ xf\ (override\text{-}on\ s\ g\ varDiffs) = \pi_2\ xf\ s$
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*:$\forall\ uxf \in set\ (uInput \otimes xfList).\ (\pi_1\ uxf)\ 0\ (sol\ s) = (sol\ s)\ (\pi_1\ (\pi_2\ uxf))$
and *solHyp2*:$\forall t{\geq}0.\ \forall\ xf \in set\ xfList.\ ((\lambda\tau.\ (sol\ s[xfList{\leftarrow}uInput]\ \tau)\ (\pi_1\ xf))\ has\text{-}vderiv\text{-}on\ (\lambda\tau.\ (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]\ \tau)))\ \{0..t\}$
shows $\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ (sol\ s[xfList{\leftarrow}uInput]\ t)\ (\partial\ (\pi_1\ xf)) = (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]\ t)$
apply(*rule allI*, *rule impI*, *rule ballI*, *rule conds4vdiffs-prelim*)
using *assms* **by** *simp-all*

lemma *conds4Consts*:
assumes *varsHyp*:$\forall\ xf \in set\ xfList.\ \pi_1\ xf \notin varDiffs$
shows $\forall\ x.\ x \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow (sol\ s[xfList{\leftarrow}uInput]\ t)\ (\partial\ x) = 0$
using *varsHyp* **apply**(*induct xfList uInput rule*: *list-induct2'*)
apply(*simp-all add*: *override-on-def varDiffs-def vdiff-def*)
by *clarsimp*

lemma *conds4InitState*:
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*:$\forall uxf{\in}set\ (uInput \otimes xfList).\ (\pi_1\ uxf)\ 0\ (sol\ s) = (sol\ s)\ (\pi_1\ (\pi_2\ uxf))$
and *xfHyp*:$(x, f) \in set\ xfList$
shows $(sol\ s[xfList{\leftarrow}uInput]\ 0)\ x = s\ x$
proof−
from *xfHyp* **obtain** *u* **where** *uxfHyp*:$(u, x, f) \in set\ (uInput \otimes xfList)$
by (*metis in-set-impl-in-set-zip2 lengthHyp*)
from *varsHyp* **have** *toZeroHyp*:$(sol\ s)\ x = s\ x$ **using** *override-on-def xfHyp* **by** *auto*
from *uxfHyp* **and** *solHyp1* **have** $u\ 0\ (sol\ s) = (sol\ s)\ x$ **by** *fastforce*
also have $(sol\ s[xfList{\leftarrow}uInput]\ 0)\ x = u\ 0\ (sol\ s)$
using *state-list-cross-upd-its-vars uxfHyp* **and** *assms* **by** *blast*
ultimately show $(sol\ s[xfList{\leftarrow}uInput]\ 0)\ x = s\ x$ **using** *toZeroHyp* **by** *simp*
qed

lemma *conds4RestOfStrings*:
assumes $x \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs$
shows $(sol\ s[xfList{\leftarrow}uInput]\ t)\ x = s\ x$
using *assms* **apply**(*induct xfList uInput rule*: *list-induct2'*)
by(*auto simp*: *varDiffs-def*)

lemma *conds4storeIVP-on-toSol*:
assumes *funcsHyp*:$\forall s\ g.\ \forall xf{\in}set\ xfList.\ \pi_2\ xf\ (override\text{-}on\ s\ g\ varDiffs) = \pi_2\ xf$

s
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*:$\forall$ $t \geq 0$. *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)
and *solHyp1*:$\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*sol s*) = (*sol s*) (π_1 (π_2 *uxf*))
and *solHyp2*:$\forall$ $t \geq 0$. $\forall$ *xf* $\in$ *set xfList*.
((λt. (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λt. π_2 *xf* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*))) {*0*..*t*}
shows *solvesStoreIVP* (λ *t*. (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)) *xfList s G*
apply(*rule solves-store-ivpI*)
subgoal using *guardHyp* **by** *simp*
subgoal using *conds4vdiffs assms* **by** *blast*
subgoal using *conds4RestOfStrings* **by** *blast*
subgoal using *conds4Consts varsHyp* **by** *blast*
subgoal apply(*rule allI*, *rule impI*, *rule ballI*, *rule solves-odeI*)
  using *solHyp2* **by** *simp-all*
subgoal using *conds4InitState* **and** *assms* **by** *force*
done

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

lemma *conds4UniqSol*:
fixes *f*::*real store* $\Rightarrow$ *real*
assumes *tHyp*:$t \geq 0$
and *contHyp*:*continuous-on* ({*0*..*t*} $\times$ *UNIV*) (λ(*t*, (*r*::*real*)). *f* (φ_s *t*))
shows *unique-on-bounded-closed 0* {*0*..*t*} τ (λt *r*. *f* (φ_s *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 *assms continuous-rhs-def* **by** *fastforce*
done

lemma *solves-store-ivp-at-beginning-overrides*:
assumes *solvesStoreIVP* φ_s *xfList a G*
shows φ_s *0* = *override-on a* (φ_s *0*) *varDiffs*
apply(*rule ext*, *subgoal-tac x* $\notin$ *varDiffs* $\longrightarrow$ φ_s *0 x* = *a x*)
subgoal by (*simp add*: *override-on-def*)
using *assms* **and** *solves-store-ivpD*(*7*) **by** *simp*

lemma *ubcStoreUniqueSol*:
assumes *tHyp*:*t* $\geq$ *0*
assumes *contHyp*:$\forall$ *xf* $\in$ *set xfList*. *continuous-on* ({*0..t*} $\times$ *UNIV*)
(λ(*t*, (*r*::*real*)). (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] *t*))
and *eqDerivs*:$\forall$ *xf* $\in$ *set xfList*. $\forall$ τ $\in$ {*0..t*}. (π_2 *xf*) (φ_s τ) = (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] τ)
and *Fsolves*:*solvesStoreIVP* φ_s *xfList s G*
and *solHyp*:*solvesStoreIVP* (λ τ. (*sol s*[*xfList*$\leftarrow$*uInput*] τ)) *xfList s G*
shows (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) = φ_s *t*
proof
  fix *x*::*string* **show** (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) *x* = φ_s *t x*
  proof(*cases x* $\in$ (π_1(|*set xfList*|)) $\cup$ *varDiffs*)
  case *False*
    then have *notInVars*:*x* $\notin$ (π_1(|*set xfList*|)) $\cup$ *varDiffs* **by** *simp*
    from *solHyp* **have** (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) *x* = *s x*
    using *tHyp notInVars solves-store-ivpD*(*2*) **by** *blast*
   also from *Fsolves* **have** φ_s *t x* = *s x* **using** *tHyp notInVars solves-store-ivpD*(*2*) **by** *blast*
    ultimately show (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) *x* = φ_s *t x* **by** *simp*
  next case *True*
    then have *x* $\in$ (π_1(|*set xfList*|)) $\vee$ *x* $\in$ *varDiffs* **by** *simp*
    from *this* **show** *?thesis*
    proof
      assume *x* $\in$ (π_1(|*set xfList*|))
      from *this* **obtain** *f* **where** *xfHyp*:(*x*, *f*) $\in$ *set xfList* **by** *fastforce*

      then have *expand1*:$\forall$ *xf* $\in$ *set xfList*.(($\lambda\tau$. φ_s τ (π_1 *xf*)) *solves-ode* ($\lambda\tau$ *r*. (π_2 *xf*) (φ_s τ))){*0..t*} *UNIV* $\wedge$ φ_s *0* (π_1 *xf*) = *s* (π_1 *xf*)
      using *Fsolves tHyp* **by** (*simp add*:*solvesStoreIVP-def*)
      hence *expand2*:$\forall$ *xf* $\in$ *set xfList*. $\forall$ τ $\in$ {*0..t*}. ((λr. φ_s *r* (π_1 *xf*))

has-vector-derivative $(\lambda r.\ (\pi_2\ xf)\ (sol\ s[xfList \leftarrow uInput]\ \tau))\ \tau)$ (*at* τ *within* $\{0..t\}$)
using *eqDerivs* **by** (*simp add*: *solves-ode-def has-vderiv-on-def*)

then have $\forall\ xf \in set\ xfList.\ ((\lambda\tau.\ \varphi_s\ \tau\ (\pi_1\ xf))$ *solves-ode*
$(\lambda\tau\ r.\ (\pi_2\ xf)\ (sol\ s[xfList \leftarrow uInput]\ \tau)))\{0..t\}\ UNIV \wedge \varphi_s\ 0\ (\pi_1\ xf) = s\ (\pi_1\ xf)$
by (*simp add*: *has-vderiv-on-def solves-ode-def expand1 expand2*)
then have $1{:}((\lambda\tau.\ \varphi_s\ \tau\ x)$ *solves-ode* $(\lambda\tau\ r.\ f\ (sol\ s[xfList \leftarrow uInput]\ \tau)))\{0..t\}\ UNIV \wedge$
$\varphi_s\ 0\ x = s\ x$ **using** *xfHyp* **by** *fastforce*

from *solHyp* **and** *xfHyp* **have** $2{:}((\lambda\ \tau.\ (sol\ s[xfList \leftarrow uInput]\ \tau)\ x)$ *solves-ode*

$(\lambda\tau\ r.\ f\ (sol\ s[xfList \leftarrow uInput]\ \tau)))\ \{0..t\}\ UNIV \wedge (sol\ s[xfList \leftarrow uInput]\ 0)\ x = s\ x$
using *solvesStoreIVP-def tHyp* **by** *fastforce*

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

apply(*clarify*) **apply**(*rule conds4UniqSol*) **by**(*auto*)
from *this* **have** 3:*unique-on-bounded-closed* $0\ \{0..t\}\ (s\ x)\ (\lambda\tau\ r.\ f\ (sol\ s[xfList \leftarrow uInput]\ \tau))$
$UNIV$ (*if* $t = 0$ *then* 1 *else* $1/(t+1)$) **using** *xfHyp* **by** *fastforce*
from $1\ 2$ **and** 3 **show** $(sol\ s[xfList \leftarrow uInput]\ t)\ x = \varphi_s\ t\ x$
using *unique-on-bounded-closed.unique-solution* **using** *real-Icc-closed-segment tHyp* **by** *blast*
next
assume $x \in varDiffs$
then obtain y **where** $xDef{:}x = \partial\ y$ **by** (*auto simp*: *varDiffs-def*)
show $(sol\ s[xfList \leftarrow uInput]\ t)\ x = \varphi_s\ t\ x$
proof(*cases* $y \in set\ (map\ \pi_1\ xfList)$)
case *True*
then obtain f **where** $xfHyp{:}(y, f) \in set\ xfList$ **by** *fastforce*
from *tHyp* **and** *Fsolves* **have** $\varphi_s\ t\ x = f\ (\varphi_s\ t)$
using *solves-store-ivpD(4) xfHyp xDef* **by** *force*
also have $(sol\ s[xfList \leftarrow uInput]\ t)\ x = f\ (sol\ s[xfList \leftarrow uInput]\ t)$
using *solves-store-ivpD(4) xfHyp xDef solHyp tHyp* **by** *force*
ultimately show *?thesis* **using** *eqDerivs xfHyp tHyp* **by** *auto*
next case *False*
then have $\varphi_s\ t\ x = 0$
using *xDef solves-store-ivpD(3) Fsolves tHyp* **by** *simp*
also have $(sol\ s[xfList \leftarrow uInput]\ t)\ x = 0$
using *False solHyp tHyp solves-store-ivpD(3) xDef* **by** *fastforce*
ultimately show *?thesis* **by** *simp*
qed
qed

qed
qed

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

$(\lambda(t,\ (r{::}real)).\ (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]\ t))$
and *solHyp*:$\forall\ s.\ solvesStoreIVP\ (\lambda\ t.\ (sol\ s[xfList{\leftarrow}uInput]\ t))\ xfList\ s\ G$
and *uniqHyp*:$\forall\ s.\ \forall\ \varphi_s.\ \varphi_s\ solvesTheStoreIVP\ xfList\ withInitState\ s\ andGuard\ G$
$\longrightarrow$
$(\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ \forall\ r \in \{0..t\}.\ (\pi_2\ xf)\ (\varphi_s\ r) = (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]$
$r))$
and *diffAssgn*: $\forall\ s.\ P\ s \longrightarrow (\forall\ t{\geq}0.\ G\ (sol\ s[xfList{\leftarrow}uInput]\ t) \longrightarrow Q\ (sol\ s[xfList{\leftarrow}uInput]$
$t))$
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac uInput=uInput* **in** *dSolve*)
prefer *2* **subgoal proof**(*clarify*)
fix $s{::}real\ store$ **and** $\varphi_s{::}real \Rightarrow real\ store$ **and** $t{::}real$
assume *isSol*:$solvesStoreIVP\ \varphi_s\ xfList\ s\ G$ **and** *sHyp*:$0 \leq t$
from *this* **and** *uniqHyp* **have** $\forall\ xf \in set\ xfList.\ \forall\ t \in \{0..t\}.$
$(\pi_2\ xf)\ (\varphi_s\ t) = (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]\ t)$ **by** *auto*
also have $\forall\ xf \in set\ xfList.\ continuous\text{-}on\ (\{0..t\} \times UNIV)$
$(\lambda(t,\ (r{::}real)).\ (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]\ t))$ **using** *contHyp sHyp* **by** *blast*
ultimately show $(sol\ s[xfList{\leftarrow}uInput]\ t) = \varphi_s\ t$
using *sHyp isSol ubcStoreUniqueSol solHyp* **by** *simp*
qed using *assms* **by** *simp-all*

theorem *dSolve-toSolveUBC*:
assumes *funcsHyp*:$\forall\ s\ g.\ \forall\ xf{\in}set\ xfList.\ \pi_2\ xf\ (override\text{-}on\ s\ g\ varDiffs) = \pi_2\ xf$
s
and *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 *guardHyp*:$\forall\ s.\ \forall\ t{\geq}0.\ G\ (sol\ s[xfList{\leftarrow}uInput]\ t)$
and *solHyp1*:$\forall\ s.\ \forall\ uxf{\in}set\ (uInput \otimes xfList).\ \pi_1\ uxf\ 0\ (sol\ s) = sol\ s\ (\pi_1\ (\pi_2$
$uxf))$
and *solHyp2*:$\forall\ s.\ \forall\ t{\geq}0.\ \forall\ xf{\in}set\ xfList.\ ((\lambda t.\ (sol\ s[xfList{\leftarrow}uInput]\ t)\ (\pi_1\ xf))$
$has\text{-}vderiv\text{-}on$
$(\lambda t.\ \pi_2\ xf\ (sol\ s[xfList{\leftarrow}uInput]\ t)))\ \{0..t\}$
and *contHyp*:$\forall\ s.\ \forall\ t{\geq}0.\ \forall\ xf \in set\ xfList.\ continuous\text{-}on\ (\{0..t\} \times UNIV)$
$(\lambda(t,\ (r{::}real)).\ (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]\ t))$
and *uniqHyp*:$\forall\ s.\ \forall\ \varphi_s.\ \varphi_s\ solvesTheStoreIVP\ xfList\ withInitState\ s\ andGuard\ G$
$\longrightarrow$
$(\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ \forall\ r \in \{0..t\}.\ (\pi_2\ xf)\ (\varphi_s\ r) = (\pi_2\ xf)\ (sol\ s[xfList{\leftarrow}uInput]$
$r))$
and *postCondHyp*:$\forall\ s.\ P\ s \longrightarrow (\forall\ t{\geq}0.\ Q\ (sol\ s[xfList{\leftarrow}uInput]\ t))$
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*
apply(*rule-tac uInput=uInput* **in** *dSolveUBC*)
using *contHyp* **apply** *simp*

apply(*rule allI*, *rule-tac uInput=uInput* **in** *conds4storeIVP-on-toSol*)
using *assms* **by** *auto*

1.4.4 "Differential Invariant."

lemma *solvesStoreIVP-couldBeModified*:
fixes F::*real* $\Rightarrow$ *real store*
assumes *vars*:$\forall\, t \geq 0.\ \forall\, xf \in set\ xfList.\ ((\lambda t.\ F\ t\ (\pi_1\ xf))\ solves\text{-}ode\ (\lambda t\ r.\ \pi_2\ xf\ (F\ t)))\ \{0..t\}\ UNIV$
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** $((\lambda t.\ F\ t\ x)\ solves\text{-}ode\ (\lambda t\ r.\ f\ (F\ t)))\ \{0..t\}\ UNIV$
using *tHyp* **by** *fastforce*
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*: *solves-ode-def 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 $\forall\ x \in (UNIV\ -\ varDiffs).\ \forall\ t \geq 0.\ \forall\, r \in \{0..t\}.$
$((\lambda\ t.\ F\ t\ x)\ has\text{-}vector\text{-}derivative\ F\ r\ (\partial\ x))\ (at\ r\ within\ \{0..t\})$
proof
fix x
assume $x \in UNIV\ -\ varDiffs$
then have *notVarDiff*:$\forall\ z.\ x \neq \partial\ z$ **using** *varDiffs-def* **by** *fastforce*
  show $\forall\, t \geq 0.\ \forall\, r \in \{0..t\}.\ ((\lambda t.\ F\ t\ x)\ has\text{-}vector\text{-}derivative\ F\ r\ (\partial\ x))\ (at\ r\ within\ \{0..t\})$
  proof(*cases* $x \in set\ (map\ \pi_1\ xfList)$)
    case *True*
    from *this* **and** *solves* **have** $\forall\ t \geq 0.\ \forall\, r \in \{0..t\}.\ \forall\ xf \in set\ xfList.$
    $((\lambda\ t.\ F\ t\ (\pi_1\ xf))\ has\text{-}vector\text{-}derivative\ F\ r\ (\partial\ (\pi_1\ xf)))\ (at\ r\ within\ \{0..t\})$
    apply(*rule-tac 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:∀ t ≥ 0. ∀r∈{0..t}. ((λ r. a x) has-vector-derivative 0) (at r
within {0..t})
    by (auto intro: derivative-eq-intros)
    {fix t r::real
      assume t≥0 and r ∈ {0..t}
      hence ((λ s. a x) has-vector-derivative 0) (at r within {0..t}) by (simp add:
constD)
      moreover have ⋀s. s ∈ {0..t} ⟹ (λ r. F r x) s = (λ r. a x) s
      using const by (simp add: ‹0 ≤ t›)
      ultimately have ((λ s. F s x) has-vector-derivative 0) (at r within {0..t})
      using has-vector-derivative-imp by (metis ‹r ∈ {0..t}›)}
    hence isZero:∀t≥0.∀r∈{0..t}.((λ t. F t x)has-vector-derivative 0)(at r within
{0..t})by blast
    from False solves and notVarDiff have ∀ t ≥ 0. F t (∂ 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 ≥ 0
and termVarsHyp:∀ x ∈ trmVars η. x ∈ (UNIV − varDiffs)
shows ∀r∈{0..t}. ((λ s. (⟦η⟧t) (F s))has-vector-derivative (⟦∂t η⟧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 ∈ 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 η1 η2)
  then show ?case
  apply(clarsimp)
  by(rule derivative-intros, simp-all)
next
  case (Mult η1 η2)
  then show ?case
  apply(clarsimp)
  apply(subgoal-tac ((λs. (⟦η1⟧t) (F s) *R (⟦η2⟧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**(*induct xfList uInput rule*: *list-induct2'*, *simp*, *simp*, *simp*)
proof(*clarify*, *rename-tac y g xfTail* ϑ *trmTail x*)
fix $x\ y{::}string$ **and** $\vartheta{::}trms$ **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* $x \in set\ (map\ \pi_1\ ((y,\ g)\ \#\ xfTail))$)
case *True*
then have $x = y \vee (x \neq y \wedge x \in set\ (map\ \pi_1\ xfTail))$ **by** *auto*
moreover
{assume $x = y$
from *this* **have** $((map\ (vdiff \circ \pi_1)\ ((y,\ g)\ \#\ xfTail)) \otimes (\vartheta\ \#\ trmTail))\langle t_V\ (\partial\ x)\rangle = \vartheta$ **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 $x \neq y \wedge x \in set\ (map\ \pi_1\ xfTail)$
then have $\partial\ x \neq \partial\ y$ **using** *vdiff-inj* **by** *auto*
from *this* **have** $((map\ (vdiff \circ \pi_1)\ ((y,\ g)\ \#\ xfTail)) \otimes (\vartheta\ \#\ trmTail))\ \langle t_V\ (\partial\ x)\rangle =$
$((map\ (vdiff \circ \pi_1)\ xfTail) \otimes trmTail)\ \langle t_V\ (\partial\ x)\rangle$ **by** *simp*
hence *?thesis* **using** $*$ **by** *simp***}**
ultimately show *?thesis* **by** *blast*
next
case *False*
then have $((map\ (vdiff \circ \pi_1)\ ((y,\ g)\ \#\ xfTail)) \otimes (\vartheta\ \#\ trmTail))\ \langle t_V\ (\partial\ x)\rangle = t_V\ (\partial\ x)$

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 $([\![\eta]\!]_t)\ a = ([\![\eta]\!]_t)\ b$
using *assms* **by**(*induction* η, *simp-all*)

lemma *non-empty-funList-implies-non-empty-trmList*:
shows $\forall\ list.(x,f) \in set\ list \wedge map\ \pi_2\ list = map\ tval\ tList \longrightarrow (\exists\ \vartheta.([\![\vartheta]\!]_t) = f \wedge \vartheta \in set\ tList)$
by(*induction tList*, *auto*)

lemma *dInvForTrms-prelim*:
assumes *substHyp*:
$\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle]\!]_t)\ st = 0$
and *termVarsHyp*:$trmVars\ \eta \subseteq (UNIV\ -\ varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList = map\ tval\ uInput$
shows $([\![\eta]\!]_t)\ a = 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow ([\![\eta]\!]_t)\ c = 0)$
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-store-ivpD*(*7*) **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 $\forall r\in\{0..t\}.\ \forall\ xf \in set\ xfList.\ F\ r\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ r)$
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 $\forall r\in\{0..t\}.\ ((\lambda s.\ ([\![\eta]\!]_t)\ (F\ s))\ has\text{-}vector\text{-}derivative\ 0)\ (at\ r\ within\ \{0..t\})$
using *obs2* **by** *auto*
from *this* **and** *tcHyp* **have** $\forall s\in\{0..t\}.\ ((\lambda x.\ ([\![\eta]\!]_t)\ (F\ x))\ has\text{-}derivative\ (\lambda x.\ x *_R 0))$
$(at\ s\ within\ \{0..t\})$ **by** (*metis has-vector-derivative-def*)

hence $(\llbracket\eta\rrbracket_t)\ (F\ t) - (\llbracket\eta\rrbracket_t)\ (F\ 0) = (\lambda x.\ x *_R\ 0)\ (t - 0)$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then show $(\llbracket\eta\rrbracket_t)\ c = 0$ **using** *obs1 tcHyp* **by** *auto*
qed

theorem *dInvForTrms*:
assumes $\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$(\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle\rrbracket_t)\ st = 0$
and *termVarsHyp*:$trmVars\ \eta \subseteq (UNIV - varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList = map\ tval\ uInput$
and *eta-f*:$f = (\llbracket\eta\rrbracket_t)$
shows $PRE\ (\lambda\ s.\ f\ s = 0)\ (ODEsystem\ xfList\ with\ G)\ POST\ (\lambda\ s.\ f\ s = 0)$
using *eta-f* **proof**(*clarsimp*)
fix *a b*
assume $(a,\ b) \in \lceil\lambda s.\ (\llbracket\eta\rrbracket_t)\ s = 0\rceil$ **and** $f = (\llbracket\eta\rrbracket_t)$
from *this* **have** *aHyp*:$a = b \wedge (\llbracket\eta\rrbracket_t)\ a = 0$ **by** (*metis* (*full-types*) *d-p2r rdom-p2r-contents*)
have $(\llbracket\eta\rrbracket_t)\ a = 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow (\llbracket\eta\rrbracket_t)\ c =$
$0)$
using *assms dInvForTrms-prelim* **by** *metis*
from *this* **and** *aHyp* **have** $\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow (\llbracket\eta\rrbracket_t)\ c$
$= 0$ **by** *blast*
thus $(a,\ b) \in wp\ (ODEsystem\ xfList\ with\ G\ )\ \lceil\lambda s.\ (\llbracket\eta\rrbracket_t)\ s = 0\rceil$
using *aHyp* **by** (*simp add*: *boxProgrPred-chrctrztn*)
qed

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 $(\llbracket\partial_P\ \varphi\rrbracket_P)\ (F\ t) = (\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\restriction\partial_P\ \varphi\restriction\rrbracket_P)\ (F\ t)$
using *propVarsHyp* **apply**(*induction* φ, *simp-all*)
using *assms diff-subst-prprty-4terms* **apply** *fastforce*
using *assms diff-subst-prprty-4terms* **apply** *fastforce*
using *assms diff-subst-prprty-4terms* **by** *fastforce*

lemma *dInvForProps-prelim*:
assumes *substHyp*:
$\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$(\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle\rrbracket_t)\ st \geq 0$
and *termVarsHyp*:$trmVars\ \eta \subseteq (UNIV - varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList = map\ tval\ uInput$
shows $(\llbracket\eta\rrbracket_t)\ a > 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow (\llbracket\eta\rrbracket_t)\ c >$
$0)$
and $(\llbracket\eta\rrbracket_t)\ a \geq 0 \longrightarrow (\forall\ c.\ (a,c) \in (ODEsystem\ xfList\ with\ G) \longrightarrow (\llbracket\eta\rrbracket_t)\ c \geq 0)$
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(7)* **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 $(\forall t{\geq}0.\ \forall\ xf \in set\ xfList.\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t))$
using *tcHyp solves-store-ivpD(4)* **by** *blast*
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-store-ivpD(1) solves-store-ivpD(3) tcHyp* **by** (*metis atLeastAtMost-iff*)
ultimately have $*$:$\forall r{\in}\{0..t\}.\ ([\![\partial_t\ \eta]\!]_t)\ (F\ r) \geq 0$ **by** (*simp*)
from *obs2* **and** *tcHyp* **have** $\forall r{\in}\{0..t\}.\ ((\lambda s.\ ([\![\eta]\!]_t)\ (F\ s))$ *has-derivative*
$(\lambda x.\ x *_R (([\![\partial_t\ \eta]\!]_t)\ (F\ r))))$ (*at r within* $\{0..t\}$) **by** (*simp add*: *has-vector-derivative-def*)

hence $\exists r{\in}\{0..t\}.\ ([\![\eta]\!]_t)\ (F\ t) - ([\![\eta]\!]_t)\ (F\ 0) = t \cdot ([\![(\partial_t\ \eta)]\!]_t)\ (F\ r)$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then obtain r **where** $([\![\partial_t\ \eta]\!]_t)\ (F\ r) \geq 0 \wedge 0 \leq r \wedge r \leq t \wedge ([\![\partial_t\ \eta]\!]_t)\ (F\ t) \geq 0$
$\wedge\ ([\![\eta]\!]_t)\ (F\ t) - ([\![\eta]\!]_t)\ (F\ 0) = t \cdot (([\![\partial_t\ \eta]\!]_t)\ (F\ r))$ **using** $*$ *tcHyp* **by** *fastforce*
thus $([\![\eta]\!]_t)\ c > 0$
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 $0 \leq ([\![\eta]\!]_t)\ a \longrightarrow (\forall c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow 0 \leq ([\![\eta]\!]_t)\ c)$
proof(*clarify*)
fix c **assume** *aHyp*:$([\![\eta]\!]_t)\ a \geq 0$ **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(7)* **by** *blast*
from *this* **have** $([\![\eta]\!]_t)\ a = ([\![\eta]\!]_t)\ (F\ 0)$ **using** *termVarsHyp eqInVars-impl-eqInTrms*
by *blast*
hence *obs1*:$([\![\eta]\!]_t)\ (F\ 0) \geq 0$ **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 $(\forall t{\geq}0.\ \forall\ xf \in set\ xfList.\ F\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (F\ t))$
using *tcHyp solves-store-ivpD(4)* **by** *blast*
from *this* **and** *tcHyp* **have** $\forall r{\in}\{0..t\}.\ ([\![\partial_t\ \eta]\!]_t)\ (F\ r) =$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\ \langle\partial_t\ \eta\rangle]\!]_t)\ (F\ r)$
using *diff-subst-prprty-4terms termVarsHyp 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-store-ivpD(1) solves-store-ivpD(3) tcHyp* **by** (*metis atLeastAtMost-iff*)

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

hence $\exists\, r{\in}\{0..t\}.$ $([\![\eta]\!]_t)\ (F\ t) - ([\![\eta]\!]_t)\ (F\ 0) = t \cdot (([\![\partial_t\ \eta]\!]_t)\ (F\ r))$
using *mvt-very-simple* **and** *tcHyp* **by** *fastforce*
then obtain r **where** $([\![\partial_t\ \eta]\!]_t)\ (F\ r) \geq 0 \wedge 0 \leq r \wedge r \leq t \wedge ([\![\partial_t\ \eta]\!]_t)\ (F\ t) \geq 0$
$\wedge\ ([\![\eta]\!]_t)\ (F\ t) - ([\![\eta]\!]_t)\ (F\ 0) = t \cdot (([\![\partial_t\ \eta]\!]_t)\ (F\ r))$ **using** $*$ *tcHyp* **by** *fastforce*
thus $([\![\eta]\!]_t)\ c \geq 0$
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 $(\llbracket\vartheta \oplus (\ominus\ \eta)\rrbracket_t)\ (F\ t) = 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $(\llbracket\vartheta\rrbracket_t)\ (F\ t) = (\llbracket\eta\rrbracket_t)\ (F\ t)$ **by** *simp*
also have $(\llbracket\vartheta \doteq \eta\rrbracket_P)\ c = ((\llbracket\vartheta\rrbracket_t)\ (F\ t) = (\llbracket\eta\rrbracket_t)\ (F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case (*Less* ϑ η)
hence $\forall\, st.\ G\ st \longrightarrow (\forall\, str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$0 \leq (\llbracket(map\ (vdiff \circ \pi_1)\ xfList \otimes uInput)\langle\partial_t\ (\eta \oplus (\ominus\ \vartheta))\rangle\rrbracket_t)\ st$
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 $(\llbracket\eta \oplus (\ominus\ \vartheta)\rrbracket_t)\ a > 0$ **using** *Less.prems(1)* **by** *simp*
ultimately have $(\forall\, c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow (\llbracket\eta \oplus (\ominus\ \vartheta)\rrbracket_t)\ c$
$> 0)$
using *dInvForProps-prelim(1) listsHyp* **by** *blast*
hence $(\llbracket\eta \oplus (\ominus\ \vartheta)\rrbracket_t)\ (F\ t) > 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $(\llbracket\eta\rrbracket_t)\ (F\ t) > (\llbracket\vartheta\rrbracket_t)\ (F\ t)$ **by** *simp*
also have $(\llbracket\vartheta \prec \eta\rrbracket_P)\ c = ((\llbracket\vartheta\rrbracket_t)\ (F\ t) < (\llbracket\eta\rrbracket_t)\ (F\ t))$ **using** *tcHyp* **by** *simp*
ultimately show *?case* **by** *simp*
next
case (*Leq* ϑ η)
hence $\forall\, st.\ G\ st \longrightarrow (\forall\, str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$0 \leq (\llbracket(map\ (vdiff \circ \pi_1)\ xfList \otimes uInput)\langle\partial_t\ (\eta \oplus (\ominus\ \vartheta))\rangle\rrbracket_t)\ st$ **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 $(\llbracket\eta \oplus (\ominus\ \vartheta)\rrbracket_t)\ a \geq 0$ **using** *Leq.prems(1)* **by** *simp*
ultimately have $(\forall\, c.\ (a,\ c) \in ODEsystem\ xfList\ with\ G\ \longrightarrow (\llbracket\eta \oplus (\ominus\ \vartheta)\rrbracket_t)\ c$
$\geq 0)$
using *dInvForProps-prelim(2) listsHyp* **by** *blast*
hence $(\llbracket\eta \oplus (\ominus\ \vartheta)\rrbracket_t)\ (F\ t) \geq 0$ **using** *tcHyp cHyp* **by** *simp*
from *this* **have** $((\llbracket\eta\rrbracket_t)\ (F\ t) \geq (\llbracket\vartheta\rrbracket_t)\ (F\ t))$ **by** *simp*
also have $(\llbracket\vartheta \preceq \eta\rrbracket_P)\ c = ((\llbracket\vartheta\rrbracket_t)\ (F\ t) \leq (\llbracket\eta\rrbracket_t)\ (F\ t))$ **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 $\forall\ st.\ G\ st \longrightarrow (\forall\, str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$(\llbracket((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\restriction\partial_P\ \varphi\restriction\rrbracket_P)\ st$
and *termVarsHyp*:*propVars* $\varphi \subseteq (UNIV - varDiffs)$
and *listsHyp*:*map* π_2 *xfList* = *map tval uInput*
and *phi-p*:$P = (\llbracket\varphi\rrbracket_P)$
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 $\forall\ st.\ G\ st \longrightarrow (\forall str.\ str \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$ $([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput) \upharpoonright \partial_P\ \varphi \upharpoonright]\!]_P)\ st$
and *termVarsHyp*:$propVars\ \varphi \subseteq (UNIV - varDiffs)$
and *listsHyp*:$map\ \pi_2\ xfList = map\ tval\ uInput$
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*)
prefer *5* **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* **apply** *simp*
using *assms* **by** *simp-all*

end
theory *VC-diffKAD-examples*
imports *VC-diffKAD*

begin

1.5 Rules Testing

In this section we test the recently developed rules with simple dynamical systems.

— Example of hybrid program verified with the rule dSolve.
lemma *motion-with-constant-velocity*:
    $PRE\ (\lambda\ s.\ s\ ''y'' < s\ ''x''\ \wedge s\ ''v'' > 0)$
    $(ODEsystem\ [(''x'',(\lambda\ s.\ s\ ''v''))]\ with\ (\lambda\ s.\ True))$
    $POST\ (\lambda\ s.\ (s\ ''y'' < s\ ''x''))$
apply(*rule-tac* $uInput{=}[\lambda\ t\ s.\ s\ ''v'' \cdot t + s\ ''x'']$ **in** *dSolve-toSolveUBC*)
prefer *10* **subgoal by**(*simp add*: *wp-trafo vdiff-def add-strict-increasing2*)
apply(*simp-all add*: *vdiff-def varDiffs-def*)

prefer *2* **apply**(*clarify*, *rule continuous-intros*)
prefer *2* **apply**(*simp add*: *solvesStoreIVP-def vdiff-def varDiffs-def*)
apply(*clarify*, *rule-tac* $f'1=\lambda\ x.\ s\ ''v''$ **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*)

— Example of hybrid program verified with differential weakening.
lemma *system-where-the-guard-implies-the-postcondition*:
$PRE\ (\lambda\ s.\ s\ ''x'' = 0)$
$(ODEsystem\ [(''x'',(\lambda\ s.\ s\ ''x'' + 1))]\ with\ (\lambda\ s.\ s\ ''x'' \geq 0))$
$POST\ (\lambda\ s.\ s\ ''x'' \geq 0)$
using *dWeakening* **by** *blast*

lemma *system-where-the-guard-implies-the-postcondition2*:
$PRE\ (\lambda\ s.\ s\ ''x'' = 0)$
$(ODEsystem\ [(''x'',(\lambda\ s.\ s\ ''x'' + 1))]\ with\ (\lambda\ s.\ s\ ''x'' \geq 0))$
$POST\ (\lambda\ s.\ s\ ''x'' \geq 0)$
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 solvesStoreIVP-def*)
by *auto*

— Example of system proved with a differential invariant.
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-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-all add*: *vdiff-def varDiffs-def*)
apply(*clarsimp*, *erule-tac* $x=''r''$ **in** *allE*)
by *simp*

— Example of systems proved with differential invariants, cuts and weakenings.
declare *d-p2r* [*simp del*]
lemma *motion-with-constant-velocity-and-invariants*:
$PRE\ (\lambda\ s.\ s\ ''x'' > s\ ''y'' \wedge s\ ''v'' > 0)$
$(ODEsystem\ [(''x'',\ \lambda\ s.\ s\ ''v'')]\ with\ (\lambda\ s.\ True))$
$POST\ (\lambda\ s.\ s\ ''x''>\ s\ ''y'')$
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* $C = \lambda\ s.\ \ s\ ''x'' > s\ ''y''$ **in** *dCut*)
apply(*rule-tac* $\varphi=(t_V\ ''y'') \prec (t_V\ ''x'')$ **and** $uInput=[t_V\ ''v'']$ **and**
$F=\lambda\ s.\ \ s\ ''x'' > s\ ''y''$ **in** *dInvFinal*)
apply(*simp-all add*: *vdiff-def varDiffs-def*, *clarify*, *erule-tac* $x=''y''$ **in** *allE*, *simp*)
using *dWeakening* **by** *simp*

lemma *motion-with-constant-acceleration-and-invariants*:

$PRE\ (\lambda\ s.\ s\ ''y'' < s\ ''x''\ \wedge s\ ''v'' \geq 0 \wedge s\ ''a'' > 0)$
$(ODEsystem\ [(''x'',(\lambda\ s.\ s\ ''v'')),(''v'',(\lambda\ s.\ s\ ''a''))]\ with\ (\lambda\ s.\ True))$
$POST\ (\lambda\ s.\ (s\ ''y'' < s\ ''x''))$

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