CPSVerification

By Jonathan

July 26, 2018

Contents

1 VC_diffKAD

theory *VC-diffKAD-auxiliarities*
imports
Main
afpModified/VC-KAD
Ordinary-Differential-Equations.ODE-Analysis

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* (⌈-⌉)
  and *Archimedean-Field.floor* (⌊-⌋)
  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 lemmata.

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 $(x\ solves\text{-}ode\ f)\ \{0..t\}\ R = ((x\ has\text{-}vderiv\text{-}on\ (\lambda t.\ f\ t\ (x\ t)))\ \{0..t\} \wedge x \in$
$\{0..t\} \rightarrow R)$
  and $f \in A \rightarrow B = (f \in \{f.\ \forall\ x.\ x \in A \longrightarrow (f\ x) \in B\})$
  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)

lemma (**in** *antidomain-kleene-algebra*) *fbox-starI*:
assumes $d\ p \leq d\ i$ **and** $d\ i \leq |x]\ i$ **and** $d\ i \leq d\ q$
shows $d\ p \leq |x^\star]\ q$
proof−
from $\langle d\ i \leq |x]\ i \rangle$ **have** $d\ i \leq |x]\ (d\ i)$
  using *local.fbox-simp* **by** *auto*
hence $|1]\ p \leq |x^\star]\ i$ **using** $\langle d\ p \leq d\ i \rangle$ **by** (*metis* (*no-types*)
  local.dual-order.trans local.fbox-one local.fbox-simp local.fbox-star-induct-var)
thus *?thesis* **using** $\langle d\ i \leq d\ q \rangle$ **by** (*metis* (*full-types*)
  local.fbox-mult local.fbox-one local.fbox-seq-var local.fbox-simp)
qed

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**
(∂ *x*) = ''*d*[''@*x*@'']''

definition *varDiffs* :: *string set* **where**
varDiffs = {*y*. ∃ *x*. *y* = ∂ *x*}

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

proposition *vdiff-noFixPoints*:*x* ≠ (∂ *x*)
by(*simp add*: *vdiff-def*)

lemma *varDiffsI*:*x* = (∂ *z*) ⟹ *x* ∈ *varDiffs*
by(*simp add*: *varDiffs-def vdiff-def*)

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

proposition *vdiff-invarDiffs*:(∂ *x*) ∈ *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 $uInput = [u_1, \ldots, u_n]$, a state s and a time t, and outputs the induced flow $sol\, s[xfList \leftarrow uInput]\, t$.

abbreviation *varDiffs-to-zero* ::*real store* ⇒ *real store* (*sol*) **where**
sol a ≡ (*override-on a* (λ *x*. *0*) *varDiffs*)

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

proposition *varDiffs-to-zero-beginning*[*simp*]: *take 2 x* ≠ ''*d*['' ⟹ (*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 $\textit{vderiv-of}\ f\ S = (\textit{SOME}\ f'.\ (f\ \textit{has-vderiv-on}\ f')\ S)$

primrec $\textit{state-list-upd} :: ((\textit{real} \Rightarrow \textit{real store} \Rightarrow \textit{real}) \times \textit{string} \times (\textit{real store} \Rightarrow \textit{real}))\ \textit{list} \Rightarrow$
$\textit{real} \Rightarrow \textit{real store} \Rightarrow \textit{real store}$ **where**
$\textit{state-list-upd}\ []\ t\ s\ = s|$
$\textit{state-list-upd}\ (\textit{uxf}\ \#\ \textit{tail})\ t\ s = (\textit{state-list-upd tail}\ t\ s)$
$(\quad (\pi_1\ (\pi_2\ \textit{uxf})) := (\pi_1\ \textit{uxf})\ t\ s,$
$\quad \partial\ (\pi_1\ (\pi_2\ \textit{uxf})) := (\textit{if}\ t = 0\ \textit{then}\ (\pi_2\ (\pi_2\ \textit{uxf}))\ s$
$\textit{else vderiv-of}\ (\lambda\ r.\ (\pi_1\ \textit{uxf})\ r\ s)\ \{0<..<\ (2 *_R t)\}\ t))$

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

proposition $\textit{state-list-cross-upd-empty}[\textit{simp}]$: $(s[[]\leftarrow\textit{list}]\ t) = s$
by$(\textit{induction list}, \textit{simp-all})$

lemma *inductive-state-list-cross-upd-its-vars*:
assumes $\textit{distHyp}:\textit{distinct}\ (\textit{map}\ \pi_1\ ((y,\ g)\ \#\ \textit{xftail}))$
and $\textit{varHyp}:\forall\,\textit{xf}\in\textit{set}((y,\ g)\ \#\ \textit{xftail}).\ \pi_1\ \textit{xf} \notin \textit{varDiffs}$
and $\textit{indHyp}:(u,\ x,\ f) \in \textit{set}\ (\textit{utail} \otimes \textit{xftail}) \Longrightarrow (s[\textit{xftail}\leftarrow\textit{utail}]\ t)\ x = u\ t\ s$
and $\textit{disjHyp}:(u,\ x,\ f) = (v,\ y,\ g) \vee (u,\ x,\ f) \in \textit{set}\ (\textit{utail} \otimes \textit{xftail})$
shows $(s[(y,\ g)\ \#\ \textit{xftail}\leftarrow v\ \#\ \textit{utail}]\ t)\ x = u\ t\ s$
using *disjHyp* **proof**
  assume $(u,\ x,\ f) = (v,\ y,\ g)$
  hence $(s[(y,\ g)\ \#\ \textit{xftail}\leftarrow v\ \#\ \textit{utail}]\ t)\ x = ((s[\textit{xftail}\leftarrow\textit{utail}]\ t)(x := u\ t\ s,$
  $\partial\ x := \textit{if}\ t = 0\ \textit{then}\ f\ s\ \textit{else vderiv-of}\ (\lambda\ r.\ u\ r\ s)\ \{0<..<\ (2 *_R t)\}\ t))\ x$ **by** *simp*
  also have ... $= u\ t\ s$ **by** $(\textit{simp add}: \textit{vdiff-def})$
  ultimately show *?thesis* **by** *simp*
next
  assume $\textit{yTailHyp}:(u,\ x,\ f) \in \textit{set}\ (\textit{utail} \otimes \textit{xftail})$
  from *this* **and** *indHyp* **have** $3:(s[\textit{xftail}\leftarrow\textit{utail}]\ t)\ x = u\ t\ s$ **by** *fastforce*
  from *yTailHyp* **and** *distHyp* **have** $2:y \neq x$ **using** *set-zip-left-rightD* **by** *force*
  from *yTailHyp* **and** *varHyp* **have** $1:x \neq \partial\ y$
  using *set-zip-left-rightD vdiff-invarDiffs* **by** *fastforce*
  from *1* **and** *2* **have** $(s[(y,\ g)\ \#\ \textit{xftail}\leftarrow v\ \#\ \textit{utail}]\ t)\ x = (s[\textit{xftail}\leftarrow\textit{utail}]\ t)\ x$ **by** *simp*
  thus *?thesis* **using** *3* **by** *simp*
qed

theorem *state-list-cross-upd-its-vars*:
assumes $\textit{distinctHyp}:\textit{distinct}\ (\textit{map}\ \pi_1\ \textit{xfList})$

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*). π_1 *xf* $\notin$ *varDiffs*
and $\forall$ *uxf*$\in$*set* (*u* # *uTail* $\otimes$ *xf* # *xfTail*). π_1 *uxf 0 s* = *s* (π_1 (π_2 *uxf*))
shows $\exists$ *g*. (*s*[*xf* # *xfTail*$\leftarrow$*u* # *uTail*] *0*) = *override-on s g varDiffs*
proof−
let *?gLHS*=(*s*[(*xf* # *xfTail*)$\leftarrow$(*u* # *uTail*)] *0*)
have *observ*:∂ (π_1 *xf*) $\in$ *varDiffs* **by** (*auto simp*: *varDiffs-def*)
from *assms*(*1*) **obtain** *g* **where** (*s*[*xfTail*$\leftarrow$*uTail*] *0*) = *override-on s g varDiffs*
by *force*
then have *?gLHS* = (*override-on s g varDiffs*)(π_1 *xf* := *u 0 s*, ∂ (π_1 *xf*) := π_2 *xf s*) **by** *simp*
also have . . . = (*override-on s g varDiffs*)(∂ (π_1 *xf*) := π_2 *xf s*)
using *override-on-def varDiffs-def assms* **by** *auto*
also have ... = (*override-on s* (*g*(∂ (π_1 *xf*) := π_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*. π_1 *xf* $\notin$ *varDiffs*
and *solHyp1*:$\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0 s* = *s* (π_1 (π_2 *uxf*))
shows $\exists$ *g*. (*s*[*xfList*$\leftarrow$*uInput*] *0*) = (*override-on s g varDiffs*)
using *assms* **proof**(*induct xfList uInput rule*: *list-induct2'*)
case *1*
  have (*s*[[]$\leftarrow$[]] *0*) = *override-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-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-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-vderiv-on* f') $\{0<..<\ t\}$ **and** $t>0$
and (f *has-vderiv-on* f'')$\{0<..<\ t\}$ **and** *tauHyp*:$\tau \in \{0<..<\ t\}$
shows $f'\ \tau = f''\ \tau$
using *assms* **apply**(*simp add*: *has-vderiv-on-def has-vector-derivative-def*)
using *frechet-derivative-unique-within-open-interval* **by** (*metis box-real*(*1*) *scaleR-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*:$\forall\, t{\ge}0.$ (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*: $\forall\ xf \in set\ xfList.\ \pi_1\ xf \notin varDiffs$
and *solHyp2*: $\forall\ t \geq 0.\ ((\lambda\tau.\ (sol\ s[xfList \leftarrow uInput]\ \tau)\ x)$
has-vderiv-on $(\lambda\tau.\ f\ (sol\ s[xfList \leftarrow uInput]\ \tau)))\ \{0..t\}$
and *tHyp*: $t > 0$ **and** *uxfHyp*: $(u,\ x,\ f) \in set\ (uInput \otimes xfList)$
shows *vderiv-of* $(\lambda\tau.\ u\ \tau\ (sol\ s))\ \{0 <..< (2 *_R t)\}\ t = f\ (sol\ s[xfList \leftarrow uInput]$
$t)$
apply(*rule-tac* $f = (\lambda\tau.\ (sol\ s[xfList \leftarrow uInput]\ \tau)\ x)$ **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*: $\forall\ s.\ \forall g.\ \forall xf \in set\ ((x,\ f)\ \#\ xfs).\ \pi_2\ xf\ (override\text{-}on\ s\ g\ varDiffs)$
$= \pi_2\ xf\ s$
and *eqLengths*: $length\ ((x,\ f)\ \#\ xfs) = length\ (u\ \#\ us)$
and *distinct*: $distinct\ (map\ \pi_1\ ((x,\ f)\ \#\ xfs))$
and *vars*: $\forall xf \in set\ ((x,\ f)\ \#\ xfs).\ \pi_1\ xf \notin varDiffs$
and *solHyp1*: $\forall uxf \in set\ ((u\ \#\ us) \otimes ((x,\ f)\ \#\ xfs)).\ \pi_1\ uxf\ 0\ (sol\ s) = sol\ s\ (\pi_1$
$(\pi_2\ uxf))$
and *disjHyp*: $(y,\ g) = (x,\ f) \vee (y,\ g) \in set\ xfs$
and *indHyp*: $(y,\ g) \in set\ xfs \Longrightarrow (sol\ s[xfs \leftarrow us]\ 0)\ (\partial\ y) = g\ (sol\ s[xfs \leftarrow us]\ 0)$
shows $(sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0)\ (\partial\ y) = g\ (sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0)$
proof−
from *assms* **obtain** *h1* **where** *h1Def*: $(sol\ s[((x,\ f)\ \#\ xfs) \leftarrow (u\ \#\ us)]\ 0) =$
$(override\text{-}on\ (sol\ s)\ h1\ varDiffs)$ **using** *state-list-cross-upd-its-dvars* **by** *blast*
from *disjHyp* **show** $(sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0)\ (\partial\ y) = g\ (sol\ s[(x,\ f)\ \#$
$xfs \leftarrow u\ \#\ us]\ 0)$
proof
  assume *eqHeads*: $(y,\ g) = (x,\ f)$
  then have $g\ (sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0) = f\ (sol\ s)$ **using** *h1Def eqFuncs*
by *simp*
  also have $... = (sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0)\ (\partial\ y)$ **using** *eqHeads* **by** *auto*
  ultimately show *?thesis* **by** *linarith*
next
  assume *tailHyp*: $(y,\ g) \in set\ xfs$
  then have $y \neq x$ **using** *distinct set-zip-left-rightD* **by** *force*
  hence $\partial\ x \neq \partial\ y$ **by**(*simp add*: *vdiff-def*)
  have $x \neq \partial\ y$ **using** *vars vdiff-invarDiffs* **by** *auto*
  obtain *h2* **where** *h2Def*: $(sol\ s[xfs \leftarrow us]\ 0) = override\text{-}on\ (sol\ s)\ h2\ varDiffs$
  using *state-list-cross-upd-its-dvars eqLengths distinct vars* **and** *solHyp1* **by** *force*
  have $(sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0)\ (\partial\ y) = g\ (sol\ s[xfs \leftarrow us]\ 0)$
  using *tailHyp indHyp* $\langle x \neq \partial\ y \rangle$ **and** $\langle \partial\ x \neq \partial\ y \rangle$ **by** *simp*
  also have $... = g\ (override\text{-}on\ (sol\ s)\ h2\ varDiffs)$ **using** *h2Def* **by** *simp*
  also have $... = g\ (sol\ s)$ **using** *eqFuncs* **and** *tailHyp* **by** *force*
  also have $... = g\ (sol\ s[(x,\ f)\ \#\ xfs \leftarrow u\ \#\ us]\ 0)$
  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* π_1 ((*y*, *g*) # *xfs*))
and *varHyp*: ∀ *xf*∈*set* ((*y*, *g*) # *xfs*). π_1 *xf* ∉ *varDiffs*
and *indHyp*:(*u*,*x*,*f*) ∈ *set* (*vs* ⊗ *xfs*) ⟹ (*s*[*xfs*←*vs*]*t*)(∂ *x*) = *vderiv-of* (λ*r*. *u r s*) {*0*<..<*2*$*_R$*t*} *t*
and *disjHyp*:(*v*, *y*, *g*) = (*u*, *x*, *f*) ∨ (*u*, *x*, *f*) ∈ *set* (*vs* ⊗ *xfs*) **and** *tHyp*:*t* > *0*
shows (*s*[(*y*, *g*) # *xfs*←*v* # *vs*] *t*) (∂ *x*) = *vderiv-of* (λ*r*. *u r s*) {*0*<..<*2* $*_R$ *t*} *t*
proof−
let *?lhs* = ((*s*[*xfs*←*vs*] *t*)(*y* := *v t s*, ∂ *y* := *vderiv-of* (λ *r*. *v r s*) {*0*<..< (*2* · *t*)} *t*)) (∂ *x*)
let *?rhs* = *vderiv-of* (λ*r*. *u r s*) {*0*<..< (*2* · *t*)} *t*
have (*s*[(*y*, *g*) # *xfs*←*v* # *vs*] *t*) (∂ *x*) = *?lhs* **using** *tHyp* **by** *simp*
also have *vderiv-of* (λ*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-of* (λ *r*. *u r s*) {*0*<..< (*2* · *t*)} *t* **by** *simp*
  also have *vderiv-of* (λ *r*. *u r s*) {*0*<..< (*2* · *t*)} *t* = *?rhs* **using** *uxfEq* **by** *simp*
  ultimately show *?lhs* = *?rhs* **by** *simp*
next
  assume *sygTail*:(*u*, *x*, *f*) ∈ *set* (*vs* ⊗ *xfs*)
  from *this* **have** *y* ≠ *x* **using** *distHyp set-zip-left-rightD* **by** *force*
  hence ∂ *x* ≠ ∂ *y* **by**(*simp add*: *vdiff-def*)
  have *y* ≠ ∂ *x* **using** *varHyp* **using** *vdiff-invarDiffs* **by** *auto*
  then have *?lhs* = (*s*[*xfs*←*vs*] *t*) (∂ *x*) **using** ‹*y* ≠ ∂ *x*› **and** ‹∂ *x* ≠ ∂ *y*› **by** *simp*
  also have (*s*[*xfs*←*vs*] *t*) (∂ *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* π_1 *xfList*)

and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *uxfHyp*:(*u*, *x*, *f*) $\in$ *set* (*uInput* $\otimes$ *xfList*) **and** *tHyp*:*t* > *0*
shows (*s*[*xfList*$\leftarrow$*uInput*] *t*) (∂ *x*) = *vderiv-of* (λ *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*) ($[\![$-$]\!]_t$ [*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*) ($[\![$-$]\!]_P$ [*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* ⇒ *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* × *trms*) *list* ⇒ *trms* ⇒ *trms* (-⟨-⟩ [*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 $trmVars\ \eta \subseteq (UNIV - varDiffs)$
and $set\ (map\ \pi_1\ xtList) \subseteq varDiffs$
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* ⇒ *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* × *trms*) *list* ⇒ *props* ⇒ *props* (-↾-↾ [*54*] *80*) **where**
$xtList\restriction\vartheta \doteq \eta\restriction = ((xtList\langle\vartheta\rangle) \doteq (xtList\langle\eta\rangle))|$
$xtList\restriction\vartheta \prec \eta\restriction = ((xtList\langle\vartheta\rangle) \prec (xtList\langle\eta\rangle))|$
$xtList\restriction\vartheta \preceq \eta\restriction = ((xtList\langle\vartheta\rangle) \preceq (xtList\langle\eta\rangle))|$
$xtList\restriction\varphi \sqcap \psi\restriction = ((xtList\restriction\varphi\restriction) \sqcap (xtList\restriction\psi\restriction))|$
$xtList\restriction\varphi \sqcup \psi\restriction = ((xtList\restriction\varphi\restriction) \sqcup (xtList\restriction\psi\restriction))$

1.2.3 ODE Extras

For exemplification purposes, we compile some concrete derivatives used commonly in classical mechanics. A more general approach should be taken that generates this theorems as instantiations.

named-theorems *ubc-definitions definitions used in the locale unique-on-bounded-closed*

declare *unique-on-bounded-closed-def* [*ubc-definitions*]
and *unique-on-bounded-closed-axioms-def* [*ubc-definitions*]
and *unique-on-closed-def* [*ubc-definitions*]
and *compact-interval-def* [*ubc-definitions*]
and *compact-interval-axioms-def* [*ubc-definitions*]
and *self-mapping-def* [*ubc-definitions*]
and *self-mapping-axioms-def* [*ubc-definitions*]
and *continuous-rhs-def* [*ubc-definitions*]
and *closed-domain-def* [*ubc-definitions*]
and *global-lipschitz-def* [*ubc-definitions*]
and *interval-def* [*ubc-definitions*]
and *nonempty-set-def* [*ubc-definitions*]
and *lipschitz-def* [*ubc-definitions*]

named-theorems *poly-deriv temporal compilation of derivatives representing galilean transformations*
named-theorems *galilean-transform temporal compilation of vderivs representing galilean transformations*
named-theorems *galilean-transform-eq the equational version of galilean−transform*

lemma *vector-derivative-line-at-origin*:(*op* · *a has-vector-derivative a*) (*at x within T*)
by (*auto intro*: *derivative-eq-intros*)

lemma [*poly-deriv*]:(*op* · *a has-derivative* ($\lambda x.\ x *_R a$)) (*at x within T*)
using *vector-derivative-line-at-origin* **unfolding** *has-vector-derivative-def* **by** *simp*

lemma *quadratic-monomial-derivative*:
((λt::*real*. $a \cdot t^2$) *has-derivative* ($\lambda t.\ a \cdot (2 \cdot x \cdot t)$)) (*at x within T*)
apply(*rule-tac g'1*=λ *t*. *2* · *x* · *t* **in** *derivative-eq-intros*(*6*))
apply(*rule-tac f'1*=λ *t*. *t* **in** *derivative-eq-intros*(*15*))
by (*auto intro*: *derivative-eq-intros*)

lemma *quadratic-monomial-derivative2*:
((λt::*real*. $a \cdot t^2$ / *2*) *has-derivative* ($\lambda t.\ a \cdot x \cdot t$)) (*at x within T*)

apply(*rule-tac* $f'1 = \lambda t.\ a \cdot (2 \cdot x \cdot t)$ **and** $g'1 = \lambda\ x.\ 0$ **in** *derivative-eq-intros*(*18*))
using *quadratic-monomial-derivative* **by** *auto*

lemma *quadratic-monomial-vderiv*[*poly-deriv*]:$((\lambda t.\ a \cdot t^2\ /\ 2)$ *has-vderiv-on* *op* $\cdot$ $a)\ T$
apply(*simp add*: *has-vderiv-on-def has-vector-derivative-def*, *clarify*)
using *quadratic-monomial-derivative2* **by** (*simp add*: *mult-commute-abs*)

lemma *galilean-position*[*galilean-transform*]:
$((\lambda t.\ a \cdot t^2\ /\ 2 + v \cdot t + x)$ *has-vderiv-on* $(\lambda t.\ a \cdot t + v))\ T$
apply(*rule-tac* $f'1 = \lambda\ x.\ a \cdot x + v$ **and** $g'1 = \lambda\ x.\ 0$ **in** *derivative-intros*(*173*))
apply(*rule-tac* $f'1 = \lambda\ x.\ a \cdot x$ **and** $g'1 = \lambda\ x.\ v$ **in** *derivative-intros*(*173*))
using *poly-deriv*(*2*) **by**(*auto intro*: *derivative-intros*)

lemma [*poly-deriv*]:
$t \in T \Longrightarrow ((\lambda\tau.\ a \cdot \tau^2\ /\ 2 + v \cdot \tau + x)$ *has-derivative* $(\lambda x.\ x *_R (a \cdot t + v)))$ (*at t within T*)
using *galilean-position* **unfolding** *has-vderiv-on-def has-vector-derivative-def* **by** *simp*

lemma [*galilean-transform-eq*]:
$t > 0 \Longrightarrow$ *vderiv-of* $(\lambda t.\ a \cdot t\hat{}2\ /\ 2 + v \cdot t + x)\ \{0<..<2 \cdot t\}\ t = a \cdot t + v$
proof−
let $?f$ = *vderiv-of* $(\lambda t.\ a \cdot t\hat{}2\ /\ 2 + v \cdot t + x)\ \{0<..<2 \cdot t\}$
assume $t > 0$ **hence** $t \in \{0<..<2 \cdot t\}$ **by** *auto*
have $\exists\ f.\ ((\lambda t.\ a \cdot t^2\ /\ 2 + v \cdot t + x)$ *has-vderiv-on* $f)\ \{0<..<2 \cdot t\}$
using *galilean-position* **by** *blast*
hence $((\lambda t.\ a \cdot t\hat{}2\ /\ 2 + v \cdot t + x)$ *has-vderiv-on* $?f)\ \{0<..<2 \cdot t\}$
unfolding *vderiv-of-def* **by** (*metis* (*mono-tags*, *lifting*) *someI-ex*)
also have $((\lambda t.\ a \cdot t^2\ /\ 2 + v \cdot t + x)$ *has-vderiv-on* $(\lambda t.\ a \cdot t + v))\ \{0<..<2 \cdot t\}$
using *galilean-position* **by** *simp*
ultimately show (*vderiv-of* $(\lambda t.\ a \cdot t\hat{}2\ /\ 2 + v \cdot t + x)\ \{0<..<2 \cdot t\})\ t = a \cdot t + v$
apply(*rule-tac* $f' = ?f$ **and** $\tau = t$ **and** $t = 2 \cdot t$ **in** *vderiv-unique-within-open-interval*)
using $\langle t \in \{0<..<2 \cdot t\}\rangle$ **by** *auto*
qed

lemma $t > 0 \Longrightarrow$ *vderiv-of* $(\lambda t.\ a \cdot t\hat{}2\ /\ 2 + v \cdot t + x)\ \{0<..<2 \cdot t\}\ t = a \cdot t + v$
unfolding *vderiv-of-def* **apply**(*subst some1-equality*[*of* - $(\lambda t.\ a \cdot t + v)$])
apply(*rule-tac* $a = \lambda t.\ a \cdot t + v$ **in** *ex1I*)
apply(*simp-all add*: *galilean-position*)
apply(*rule ext*, *rename-tac* $f\ \tau$)
apply(*rule-tac* $f = \lambda t.\ a \cdot t^2\ /\ 2 + v \cdot t + x$ **and** $t = 2 \cdot t$ **and** $f' = f$ **in** *vderiv-unique-within-open-interval*)
apply(*simp-all add*: *galilean-position*)
oops

lemma *galilean-velocity*[*galilean-transform*]:$((\lambda r.\ a \cdot r + v)$ *has-vderiv-on* $(\lambda t.\ a))$
T
apply(*rule-tac f′1*=λ *x. a* **and** *g′1*=λ *x. 0* **in** *derivative-intros*(*173*))
unfolding *has-vderiv-on-def* **by**(*auto intro*: *derivative-eq-intros*)

lemma [*galilean-transform-eq*]:
$t > 0 \Longrightarrow$ *vderiv-of* $(\lambda r.\ a \cdot r + v)\ \{0<..<2 \cdot t\}\ t = a$
proof−
let *?f* = *vderiv-of* $(\lambda r.\ a \cdot r + v)\ \{0<..<2 \cdot t\}$
assume $t > 0$ **hence** $t \in \{0<..<2 \cdot t\}$ **by** *auto*
have $\exists\ f.\ ((\lambda r.\ a \cdot r + v)$ *has-vderiv-on* $f)\ \{0<..<2 \cdot t\}$
using *galilean-velocity* **by** *blast*
hence $((\lambda r.\ a \cdot r + v)$ *has-vderiv-on ?f*$)\ \{0<..<2 \cdot t\}$
unfolding *vderiv-of-def* **by** (*metis* (*mono-tags*, *lifting*) *someI-ex*)
also have $((\lambda r.\ a \cdot r + v)$ *has-vderiv-on* $(\lambda t.\ a))\ \{0<..<2 \cdot t\}$
using *galilean-velocity* **by** *simp*
ultimately show (*vderiv-of* $(\lambda r.\ a \cdot r + v)\ \{0<..<2 \cdot t\})\ t = a$
apply(*rule-tac f′*=*?f* **and** τ=*t* **and** *t*=*2*$\cdot$*t* **in** *vderiv-unique-within-open-interval*)
using ‹$t \in \{0<..<2 \cdot t\}$› **by** *auto*
qed

lemma [*galilean-transform*]:
$((\lambda t.\ v \cdot t - a \cdot t^2 / 2 + x)$ *has-vderiv-on* $(\lambda x.\ v - a \cdot x))\ \{0..t\}$
apply(*subgoal-tac* $((\lambda t.\ - a \cdot t^2 / 2 + v \cdot t\ + x)$ *has-vderiv-on* $(\lambda x.\ - a \cdot x + v))\ \{0..t\}$, *simp*)
by(*rule galilean-transform*)

lemma [*galilean-transform-eq*]:$t > 0 \Longrightarrow$ *vderiv-of* $(\lambda t.\ v \cdot t - a \cdot t\hat{}2 / 2 + x)$ $\{0<..<2 \cdot t\}\ t = v - a \cdot t$
apply(*subgoal-tac vderiv-of* $(\lambda t.\ - a \cdot t^2 / 2 + v \cdot t + x)\ \{0<..<2 \cdot t\}\ t = - a \cdot t + v$, *simp*)
by(*rule galilean-transform-eq*)

lemma [*galilean-transform*]:
$((\lambda t.\ v - a \cdot t)$ *has-vderiv-on* $(\lambda x.\ - a))\ \{0..t\}$
apply(*subgoal-tac* $((\lambda t.\ - a \cdot t + v)$ *has-vderiv-on* $(\lambda x.\ - a))\ \{0..t\}$, *simp*)
by(*rule galilean-transform*)

lemma [*galilean-transform-eq*]:$t > 0 \Longrightarrow$ *vderiv-of* $(\lambda r.\ v - a \cdot r)\ \{0<..<2 \cdot t\}$ $t = - a$
apply(*subgoal-tac vderiv-of* $(\lambda t.\ - a \cdot t + v)\ \{0<..<2 \cdot t\}\ t = - a$, *simp*)
by(*rule galilean-transform-eq*)

lemma [*simp*]:$(\lambda x.$ *case x of* $(t, x) \Rightarrow f\ t) = (\lambda\ x.\ (f \circ \pi_1)\ x)$
by *auto*

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

begin

1.3 Phase Space Relational Semantics

definition *solvesStoreIVP* :: $(real \Rightarrow real\ store) \Rightarrow (string \times (real\ store \Rightarrow real))$ $list \Rightarrow$
$real\ store \Rightarrow bool$
$((-\ solvesTheStoreIVP\ -\ withInitState\ -\ )\ [70,\ 70,\ 70]\ 68)$ **where**
$solvesStoreIVP\ \varphi_S\ xfList\ s \equiv$
$(*\ F\ sends\ vdiffs{-}in{-}list\ to\ derivs.\ *)$
$(\forall\ t \geq 0.\ (\forall\ xf \in set\ xfList.\ \varphi_S\ t\ (\partial\ (\pi_1\ xf)) = \pi_2\ xf\ (\varphi_S\ t))\ \wedge$
$(*\ F\ preserves\ the\ rest\ of\ the\ variables\ and\ F\ sends\ derivs\ of\ constants\ to\ 0.\ *)$
$(\forall\ y.\ (y \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \longrightarrow \varphi_S\ t\ y = s\ y)\ \wedge$
$(y \notin (\pi_1 (\!|set\ xfList|\!)) \longrightarrow \varphi_S\ t\ (\partial\ y) = 0))\ \wedge$
$(*\ F\ solves\ the\ induced\ IVP.\ *)$
$(\forall\ xf \in set\ xfList.\ ((\lambda\ t.\ \varphi_S\ t\ (\pi_1\ xf))\ solves\text{-}ode\ (\lambda\ t.\lambda\ r.(\pi_2\ xf)\ (\varphi_S\ t)))\ \{0..t\}$ $UNIV\ \wedge$
$\varphi_S\ 0\ (\pi_1\ xf) = s(\pi_1\ xf)))$

lemma *solves-store-ivpI*:
assumes $\forall\ t \geq 0.\forall\ xf \in set\ xfList.\ (\varphi_S\ t\ (\partial\ (\pi_1\ xf))) = (\pi_2\ xf)\ (\varphi_S\ t)$
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*
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*
shows $\forall\ t \geq 0.\forall\ y.\ y \notin (\pi_1 (\!|set\ xfList|\!)) \cup varDiffs \longrightarrow \varphi_S\ t\ y = s\ y$
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*
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** $x \in (\pi_1 (\!|set\ xfList|\!)) \Longrightarrow \varphi_S\ 0\ x = s\ x$ **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*(*1*) **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*
  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*(*1*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s*
  and $t \geq 0$
  and $y \notin (\pi_1 (\!|set\ xfList|\!))$
shows $\varphi_S\ t\ (\partial\ y) = 0$
using *assms solves-store-ivpE*(*2*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s*
  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*(*3*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s*
  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*(*4*) **by** *simp*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s*
  and $(x,f) \in set\ xfList$
shows $\varphi_S\ 0\ x = s\ x$
using *assms solves-store-ivpE*(*5*) **by** *fastforce*

lemma [*solves-store-ivpD*]:
assumes φ_S *solvesTheStoreIVP xfList withInitState s*
  and $y \notin varDiffs$
shows $\varphi_S\ 0\ y = s\ y$
using *assms solves-store-ivpE*(*6*) **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 ≥ 0 ∧ (∀ r ∈ {0..t}. G (φ_S r)) ∧ *solvesStoreIVP* φ_S *xfList s*}

1.4 Derivation of Differential Dynamic Logic Rules

1.4.1 "Differential Weakening"

lemma *wlp-evol-guard*:*Id* ⊆ *wp* (*ODEsystem xfList with G*) ⌈*G*⌉
by(*simp add*: *rel-antidomain-kleene-algebra.fbox-def rel-ad-def guarDiffEqtn-def p2r-def*, *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*)

theorem *dW*: *wp* (*ODEsystem xfList with G*) ⌈*Q*⌉ = *wp* (*ODEsystem xfList with G*) ⌈λ*s*. *G s* ⟶ *Q s*⌉
unfolding *rel-antidomain-kleene-algebra.fbox-def rel-ad-def guarDiffEqtn-def*
by(*simp add*: *relcomp.simps p2r-def*, *fastforce*)

1.4.2 "Differential Cut"

lemma *all-interval-guarDiffEqtn*:
assumes *solvesStoreIVP* φ_S *xfList s* ∧ (∀ *r* ∈ {*0..t*}. *G* (φ_S *r*)) ∧ *0* ≤ *t*
shows ∀ *r* ∈ {*0..t*}. (*s*, φ_S *r*) ∈ (*ODEsystem xfList with G*)
unfolding *guarDiffEqtn-def* **using** *atLeastAtMost-iff* **apply** *clarsimp*
apply(*rule-tac x*=*r* **in** *exI*, *rule-tac x*=φ_S **in** *exI*) **using** *assms* **by** *simp*

lemma *condAfterEvol-remainsAlongEvol*:
assumes *boxDiffC*:(*s*, *s*) ∈ *wp* (*ODEsystem xfList with G*) ⌈*C*⌉
and *FisSol*:*solvesStoreIVP* φ_S *xfList s* ∧ (∀ *r* ∈ {*0..t*}. *G* (φ_S *r*)) ∧ *0* ≤ *t*
shows ∀ *r* ∈ {*0..t*}. *G* (φ_S *r*) ∧ *C* (φ_S *r*)
proof−
from *boxDiffC* **have** ∀ *c*. (*s*,*c*) ∈ (*ODEsystem xfList with G*) ⟶ *C c*
  by (*simp add*: *boxProgrPred-chrctrztn*)
also from *FisSol* **have** ∀ *r* ∈ {*0..t*}. (*s*, φ_S *r*) ∈ (*ODEsystem xfList with G*)
  using *all-interval-guarDiffEqtn* **by** *blast*
ultimately show *?thesis*
  using *FisSol atLeastAtMost-iff guarDiffEqtn-def* **by** *fastforce*
qed

theorem *dCut*:
assumes *pBoxDiffCut*:(*PRE P* (*ODEsystem xfList with G*) *POST C*)
assumes *pBoxCutQ*:(*PRE P* (*ODEsystem xfList with* (λ *s*. *G s* ∧ *C s*)) *POST Q*)
shows *PRE P* (*ODEsystem xfList with G*) *POST Q*

apply(*clarify*, *subgoal-tac* $a = b$) **defer**
proof(*metis d-p2r rdom-p2r-contents*, *simp*, *subst boxProgrPred-chrctrztn*, *clarify*)
fix b y **assume** $(b, b) \in \lceil P \rceil$ **and** $(b, y) \in$ *ODEsystem xfList with G*
then obtain φ_S t **where** $*$:*solvesStoreIVP* φ_S *xfList* $b \wedge (\forall\ r \in \{0..t\}.\ G\ (\varphi_S\ r)) \wedge 0 \leq t \wedge \varphi_S\ t = y$
  using *guarDiffEqtn-def* **by** *auto*
hence $\forall\ r \in \{0..t\}.\ (b, \varphi_S\ r) \in$ (*ODEsystem xfList with G*)
  using *all-interval-guarDiffEqtn* **by** *blast*
from *this* **and** *pBoxDiffCut* **have** $\forall\ r \in \{0..t\}.\ C\ (\varphi_S\ r)$
  using *boxProgrPred-chrctrztn* ‹$(b, b) \in \lceil P \rceil$› **by** (*metis* (*no-types*, *lifting*) *d-p2r subsetCE*)
then have $\forall\ r \in \{0..t\}.\ (b, \varphi_S\ r) \in$ (*ODEsystem xfList with* $(\lambda\ s.\ G\ s \wedge C\ s)$)
  using $*$ *all-interval-guarDiffEqtn* **by** (*metis* (*mono-tags*, *lifting*))
from *this* **and** *pBoxCutQ* **have** $\forall\ r \in \{0..t\}.\ Q\ (\varphi_S\ r)$
  using *boxProgrPred-chrctrztn* ‹$(b, b) \in \lceil P \rceil$› **by** (*metis* (*no-types*, *lifting*) *d-p2r subsetCE*)
thus Q y **using** $*$ **by** *auto*
qed

theorem *dC*:
assumes $Id \subseteq$ *wp* (*ODEsystem xfList with G*) $\lceil C \rceil$
shows *wp* (*ODEsystem xfList with G*) $\lceil Q \rceil$ = *wp* (*ODEsystem xfList with* $(\lambda\ s.\ G\ s \wedge C\ s)$) $\lceil Q \rceil$
proof(*rule-tac* $f{=}\lambda\ x.$ *wp* x $\lceil Q \rceil$ **in** *HOL.arg-cong*, *safe*)
  fix a b **assume** $(a, b) \in$ *ODEsystem xfList with G*
  then obtain φ_S t **where** $*$:*solvesStoreIVP* φ_S *xfList* $a \wedge (\forall\ r \in \{0..t\}.\ G\ (\varphi_S\ r)) \wedge 0 \leq t \wedge \varphi_S\ t = b$
    using *guarDiffEqtn-def* **by** *auto*
  hence *1*:$\forall\ r \in \{0..t\}.\ (a, \varphi_S\ r) \in$ *ODEsystem xfList with G*
    by (*meson all-interval-guarDiffEqtn*)
  from *this* **have** $\forall\ r \in \{0..t\}.\ C\ (\varphi_S\ r)$ **using** *assms boxProgrPred-chrctrztn*
    by (*metis IdI boxProgrPred-IsProp subset-antisym*)
  thus $(a, b) \in$ *ODEsystem xfList with* $(\lambda s.\ G\ s \wedge C\ s)$
    using $*$ *guarDiffEqtn-def* **by** *blast*
next
  fix a b **assume** $(a, b) \in$ *ODEsystem xfList with* $(\lambda s.\ G\ s \wedge C\ s)$
  then show $(a, b) \in$ *ODEsystem xfList with G*
  unfolding *guarDiffEqtn-def* **by**(*clarsimp*, *rule-tac* $x{=}t$ **in** *exI*, *rule-tac* $x{=}\varphi_S$ **in** *exI*, *simp*)
qed

1.4.3 "Solve Differential Equation"

lemma *prelim-dSolve*:
assumes *solHyp*:(λt. *sol* s[*xfList*$\leftarrow$*uInput*] t) *solvesTheStoreIVP xfList withInitState s*
and *uniqHyp*:$\forall\ X.$ *solvesStoreIVP* X *xfList* $s \longrightarrow (\forall\ t \geq 0.$ (*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*:$t \geq 0 \wedge \varphi_S\ t = c \wedge$ *solvesStoreIVP* φ_S *xfList s* $\wedge$ ($\forall$ $r \in \{0..t\}$. *G* (φ_S *r*))
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 *this* **have** *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) **using** *FHyp* **by** *force*
then show *Q c* **using** *diffAssgn FHyp cHyp* **by** *auto*
qed

theorem *dS*:
assumes *solHyp*:$\forall$ *s*. *solvesStoreIVP* (λt. *sol s*[*xfList*$\leftarrow$*uInput*] *t*) *xfList s*
and *uniqHyp*:$\forall$ *s X*. *solvesStoreIVP X xfList s* $\longrightarrow$ ($\forall$ $t \geq 0$. (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) = *X t*)
shows *wp* (*ODEsystem xfList with G*) $\lceil Q \rceil$ =
$\lceil\lambda$ *s*. $\forall\, t \geq 0$. ($\forall\, r \in \{0..t\}$. *G* (*sol s*[*xfList*$\leftarrow$*uInput*] *r*)) $\longrightarrow$ *Q* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)$\rceil$
apply(*simp add*: *p2r-def*, *rule subset-antisym*)
unfolding *guarDiffEqtn-def rel-antidomain-kleene-algebra.fbox-def rel-ad-def*
using *solHyp* **apply**(*simp add*: *relcomp.simps*) **apply** *clarify*
apply(*rule-tac x=x* **in** *exI*, *clarsimp*)
apply(*erule-tac x=sol x*[*xfList*$\leftarrow$*uInput*] *t* **in** *allE*, *erule disjE*)
apply(*erule-tac x=x* **in** *allE*, *erule-tac x=t* **in** *allE*)
apply(*erule impE*, *simp*, *erule-tac x=*λt. *sol x*[*xfList*$\leftarrow$*uInput*] *t* **in** *allE*)
apply(*simp-all*, *clarify*, *rule-tac x=s* **in** *exI*, *simp add*: *relcomp.simps*)
using *uniqHyp* **by** *fastforce*

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

— We proceed to refine the previous rule by finding the necessary restrictions on varFunList and uInput so that the solution to the store-IVP is guaranteed.
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* $(map\ \pi_1\ xfList)$
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*:$\forall\ t \geq 0.\ ((\lambda\tau.\ (sol\ s[xfList \leftarrow uInput]\ \tau)\ x)$
has-vderiv-on $(\lambda\tau.\ f\ (sol\ s[xfList \leftarrow uInput]\ \tau)))\ \{0..t\}$
and *xfHyp*:$(x, f) \in set\ xfList$ **and** *tHyp*:$t \geq 0$
shows $(sol\ s[xfList \leftarrow uInput]\ t)\ (\partial\ x) = f\ (sol\ s[xfList \leftarrow uInput]\ t)$
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 $(sol\ s[xfList \leftarrow uInput]\ t)\ (\partial\ x) = f\ (sol\ s[xfList \leftarrow uInput]\ t)$
  proof(*cases t=0*)
  case *True*
    have $(sol\ s[xfList \leftarrow uInput]\ 0)\ (\partial\ x) = f\ (sol\ s[xfList \leftarrow uInput]\ 0)$
    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 $(sol\ s[xfList \leftarrow uInput]\ t)\ (\partial\ x) = vderiv\text{-}of\ (\lambda\ r.\ u\ r\ (sol\ s))\ \{0<..<(2 *_R t)\}\ t$
    using *xfuHyp assms to-sol-greater-than-zero-its-dvars* **by** *blast*
    also have $vderiv\text{-}of\ (\lambda r.\ u\ r\ (sol\ s))\ \{0<..<(2 *_R t)\}\ t = f\ (sol\ s[xfList \leftarrow uInput]\ t)$
    using *assms xfuHyp* $\langle t > 0 \rangle$ **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-vderiv-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* π_1 *xfList*)
and *lengthHyp*:*length xfList* = *length uInput*
and *varsHyp*:$\forall$ *xf* $\in$ *set xfList*. π_1 *xf* $\notin$ *varDiffs*
and *solHyp1*:$\forall$ *uxf*$\in$*set* (*uInput* $\otimes$ *xfList*). (π_1 *uxf*) *0* (*sol s*) = (*sol s*) (π_1 (π_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* $\notin$ (π_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*. 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 *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*.
((λ*t*. (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) (π_1 *xf*)) *has-vderiv-on* (λ*t*. π_2 *xf* (*sol s*[*xfList*$\leftarrow$*uInput*] *t*))) {*0*..*t*}
shows *solvesStoreIVP* (λ *t*. (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)) *xfList s*
apply(*rule solves-store-ivpI*)
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 *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* $\longrightarrow$ ($\forall$ *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 postCondHyp* **by** *simp*

— As before, we keep refining the rule dSolve. This time we find the necessary restrictions to attain uniqueness.

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 (λ*t 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*
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*(*6*) **by** *simp*

lemma *ubcStoreUniqueSol*:

assumes *tHyp*:$t \geq 0$
assumes *contHyp*:$\forall$ *xf* $\in$ *set xfList*. *continuous-on* ($\{0..t\}$ $\times$ *UNIV*) ($\lambda(t, (r::real))$. (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] *t*))
and *eqDerivs*:$\forall$ *xf* $\in$ *set xfList*. $\forall$ $\tau \in \{0..t\}$. (π_2 *xf*) (φ_s τ) = (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] τ)
and *Fsolves*:*solvesStoreIVP* φ_s *xfList s*
and *solHyp*:*solvesStoreIVP* (λ τ. (*sol s*[*xfList*$\leftarrow$*uInput*] τ)) *xfList s*
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*(*1*) **by** *blast*
   also from *Fsolves* **have** φ_s *t x* = *s x* **using** *tHyp notInVars solves-store-ivpD*(*1*) **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*. $\forall$ $\tau \in \{0..t\}$. ((λr. φ_s *r* (π_1 *xf*)) *has-vector-derivative* (λr. (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] τ)) τ) (*at* τ *within* $\{0..t\}$)
      using *eqDerivs* **by** (*simp add*: *solves-ode-def has-vderiv-on-def*)

      then have $\forall$ *xf* $\in$ *set xfList*. (($\lambda\tau$. φ_s τ (π_1 *xf*)) *solves-ode* ($\lambda\tau$ *r*. (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] τ))){*0..t*} *UNIV* $\wedge$ φ_s *0* (π_1 *xf*) = *s* (π_1 *xf*)
      by (*simp add*: *has-vderiv-on-def solves-ode-def expand1 expand2*)
     then have *1*:(($\lambda\tau$. φ_s τ *x*) *solves-ode* ($\lambda\tau$ *r*. *f* (*sol s*[*xfList*$\leftarrow$*uInput*] τ))){*0..t*} *UNIV* $\wedge$
      φ_s *0 x* = *s x* **using** *xfHyp* **by** *fastforce*

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

      ($\lambda\tau$ *r*. *f* (*sol s*[*xfList*$\leftarrow$*uInput*] τ))) {*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* (π_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(3) xfHyp xDef* **by** *force*
also have $(sol\ s[xfList \leftarrow uInput]\ t)\ x = f\ (sol\ s[xfList \leftarrow uInput]\ t)$
using *solves-store-ivpD(3) 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(2) Fsolves tHyp* **by** *simp*
also have $(sol\ s[xfList \leftarrow uInput]\ t)\ x = 0$
using *False solHyp tHyp solves-store-ivpD(2) 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*:$\forall\ s.\ solvesStoreIVP\ (\lambda\ t.\ (sol\ s[xfList \leftarrow uInput]\ t))\ xfList\ s$
and *uniqHyp*:$\forall\ s.\ \forall\ \varphi_s.\ \varphi_s\ solvesTheStoreIVP\ xfList\ withInitState\ s \longrightarrow$
$(\forall\ t \geq 0.\ \forall\ xf \in set\ xfList.\ \forall\ r \in \{0..t\}.\ (\pi_2\ xf)\ (\varphi_s\ r) = (\pi_2\ xf)\ (sol\ s[xfList \leftarrow uInput]$
$r))$
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** φ_s::*real* $\Rightarrow$ *real store* **and** t::*real*
assume *isSol*:*solvesStoreIVP* φ_s *xfList s* **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-on* ($\{0..t\}$ $\times$ *UNIV*)
($\lambda(t, (r{::}real))$. (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] *t*)) **using** *contHyp sHyp* **by** *blast*
ultimately show (*sol s*[*xfList*$\leftarrow$*uInput*] *t*) = φ_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*. 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 *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 *contHyp*:$\forall$ *s*. $\forall$ *t*$\geq$*0*. $\forall$ *xf* $\in$ *set xfList*. *continuous-on* ($\{0..t\}$ $\times$ *UNIV*)
($\lambda(t, (r{::}real))$. (π_2 *xf*) (*sol s*[*xfList*$\leftarrow$*uInput*] *t*))
and *uniqHyp*:$\forall$ *s*. $\forall$ φ_s. φ_s *solvesTheStoreIVP xfList withInitState s* $\longrightarrow$
($\forall$ *t* $\geq$ *0*. $\forall$ *xf* $\in$ *set xfList*. $\forall$ *r* $\in$ $\{0..t\}$. (π_2 *xf*) (φ_s *r*) = (π_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*. ((λt. *F t* (π_1 *xf*)) *solves-ode* (λt *r*. π_2 *xf* (*F t*))) $\{0..t\}$ *UNIV*
and *dvars*:$\forall$ *t* $\geq$ *0*. $\forall$ *xf*$\in$*set xfList*. (*F t* (∂ (π_1 *xf*))) = (π_2 *xf*) (*F t*)
shows $\forall$ *t* $\geq$ *0*. $\forall$ *r*$\in\{0..t\}$. $\forall$ *xf* $\in$ *set xfList*.
((λ *t*. *F t* (π_1 *xf*)) *has-vector-derivative F r* (∂ (π_1 *xf*))) (*at r within* $\{0..t\}$)
proof(*clarify*, *rename-tac t r x f*)
fix *x f* **and** *t r*::*real*
assume *tHyp*:*0* $\leq$ *t* **and** *xfHyp*:(*x*, *f*) $\in$ *set xfList* **and** *rHyp*:*r* $\in$ $\{0..t\}$
from *this* **and** *vars* **have** ((λt. *F t x*) *solves-ode* (λt *r*. *f* (*F t*))) $\{0..t\}$ *UNIV*
using *tHyp* **by** *fastforce*
hence $*$:$\forall$ *r*$\in\{0..t\}$. ((λ *t*. *F t x*) *has-vector-derivative* (λ *t*. *f* (*F t*)) *r*) (*at r within* $\{0..t\}$)
by (*simp add*: *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* (∂ (π_1 *xf*))) = (π_2 *xf*) (*F r*)
using *assms* **by** *auto*
from *this rHyp* **and** *xfHyp* **have** (*F r* (∂ *x*)) = *f* (*F r*) **by** *force*

then show $((\lambda t.\ F\ t\ (\pi_1\ (x, f)))$ *has-vector-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*
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* π_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 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*(*1*) **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*: $\langle 0 \leq t \rangle$)
ultimately have $((\lambda\ s.\ F\ s\ x)$ *has-vector-derivative* $0)$ (*at r within* $\{0..t\}$)
using *has-vector-derivative-imp* **by** (*metis* $\langle r \in \{0..t\} \rangle$)**}**
hence *isZero*:$\forall t\geq 0.\forall r\in\{0..t\}.((\lambda\ t.\ F\ t\ x)$*has-vector-derivative* $0)$(*at r within* $\{0..t\}$)**by** *blast*
from *False solves* **and** *notVarDiff* **have** $\forall\ t \geq 0.\ F\ t\ (\partial\ x) = 0$
using *solves-store-ivpD*(*2*) **by** *simp*
then show *?thesis* **using** *isZero* **by** *simp*
qed
qed

lemma *derivationLemma*:
assumes *solvesStoreIVP F xfList a*
and *tHyp*:$t \geq 0$

and *termVarsHyp*:$\forall$ $x \in$ *trmVars* η. $x \in$ (*UNIV* $-$ *varDiffs*)
shows $\forall r{\in}\{0..t\}$. ((λ s. ($[\![\eta]\!]_t$) (F s))*has-vector-derivative* ($[\![\partial_t\ \eta]\!]_t$) ($F$ 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* ((λs. ($[\![\eta 1]\!]_t$) (F s) $*_R$ ($[\![\eta 2]\!]_t$) (F s)) *has-vector-derivative* ($[\![\partial_t\ \eta 1]\!]_t$) ($F$ r) $\cdot$ ($[\![\eta 2]\!]_t$) (F r) + ($[\![\eta 1]\!]_t$) (F r) $\cdot$ ($[\![\partial_t\ \eta 2]\!]_t$) ($F$ r)) (*at* r *within* $\{0..t\}$),*simp*)
  apply(*rule-tac* $f'1$=($[\![\partial_t\ \eta 1]\!]_t$) ($F$ r) **and** $g'1$=($[\![\partial_t\ \eta 2]\!]_t$) ($F$ 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 (∂ (π_1 *xf*)) = π_2 *xf* (F t)
and *tHyp*:(t::*real*) $\geq$ *0*
and *listsHyp*:*map* π_2 *xfList* = *map tval uInput*
and *termVarsHyp*:*trmVars* $\eta \subseteq$ (*UNIV* $-$ *varDiffs*)
shows ($[\![\partial_t\ \eta]\!]_t$) ($F$ t) = ($[\![$((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*)$\langle\partial_t\ \eta\rangle]\!]_t$) ($F$ 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** ϑ::*trms* **and** g **and** *xfTail*::((*string* $\times$ (*real store* $\Rightarrow$ *real*)) *list*) **and** *trmTail*
assume *IH*:$\bigwedge x$. $x \notin$ *varDiffs* $\Longrightarrow$ *map* π_2 *xfTail* = *map tval trmTail* $\Longrightarrow$
$\forall$ *xf*$\in$*set xfTail*. F t (∂ (π_1 *xf*)) = π_2 *xf* (F t) $\Longrightarrow$
F t (∂ x) = ($[\![$(*map* (*vdiff* $\circ$ π_1) *xfTail* $\otimes$ *trmTail*)$\langle t_V$ (∂ x)$\rangle]\!]_t$) (F t)
and *1*:$x \notin$ *varDiffs* **and** *2*:*map* π_2 ((y, g) # *xfTail*) = *map tval* (ϑ # *trmTail*)
and *3*:$\forall$ *xf*$\in$*set* ((y, g) # *xfTail*). F t (∂ (π_1 *xf*)) = π_2 *xf* (F t)

hence $*$:($[\![(map\ (vdiff\ \circ\ \pi_1)\ xfTail\ \otimes\ trmTail)\langle Var\ (\partial\ x)\rangle]\!]_t$) ($F\ t$) = $F\ t\ (\partial\ x)$
using *tHyp* **by** *auto*
show $F\ t\ (\partial\ x)$ = ($[\![((map\ (vdiff\ \circ\ \pi_1)\ ((y,\ g)\ \#\ xfTail))\ \otimes\ (\vartheta\ \#\ trmTail))\ \langle t_V$
$(\partial\ x)\rangle]\!]_t$) ($F\ t$)
  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** ($[\![\vartheta]\!]_t$) ($F\ t$) = $g\ (F\ t)$ **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* $\wedge$ ($\forall$ *r*$\in${*0..t*}. *G* (*F r*))

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*}. ((λ*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* (∂ (π_1 *xf*)) = π_2 *xf* (*F r*)
using *tcHyp solves-store-ivpD*(*3*) **by** *fastforce*
hence $\forall$ *r*$\in${*0..t*}. ($[\![\partial_t\ \eta]\!]_t$) (*F r*) = ($[\![$((*map* (*vdiff* $\circ$ π_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* $\circ$ π_1) *xfList*) $\otimes$ *uInput*)$\langle\partial_t$
$\eta\rangle]\!]_t$) (*F r*) = *0*
using *solves-store-ivpD*(*2*) *tcHyp* **by** *fastforce*
ultimately have $\forall$ *r*$\in${*0..t*}. ((λ*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*}. ((λ*x*. ($[\![\eta]\!]_t$) (*F x*)) *has-derivative* (λ*x*. *x*
$*_R$ *0*))
(*at s within* {*0..t*}) **by** (*metis has-vector-derivative-def*)
hence ($[\![\eta]\!]_t$) (*F t*) − ($[\![\eta]\!]_t$) (*F 0*) = (λ*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* $\notin$ (π_1(|*set xfList*|)) $\longrightarrow$ *st* (∂ *str*) = *0*) $\longrightarrow$
($[\![$((*map* (*vdiff* $\circ$ π_1) *xfList*) $\otimes$ *uInput*) $\langle\partial_t\ \eta\rangle]\!]_t$) *st* = *0*
and *termVarsHyp*:*trmVars* η $\subseteq$ (*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 *diff-subst-prprty-4props*:
assumes *solves*:$\forall$ *xf* $\in$ *set xfList*. *F t* (∂ (π_1 *xf*)) = π_2 *xf* (*F t*)
and *tHyp*:$t \geq 0$
and *listsHyp*:*map* π_2 *xfList* = *map tval uInput*
and *propVarsHyp*:*propVars* $\varphi \subseteq$ (*UNIV* $-$ *varDiffs*)
shows ($[\![\partial_P\ \varphi]\!]_P$) (*F t*) = ($[\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\restriction\partial_P\ \varphi\restriction]\!]_P$) (*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* $\notin$ (π_1(|*set xfList*|)) $\longrightarrow$ *st* (∂ *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*
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* $\wedge$ ($\forall r \in \{0..t\}$. *G* (*F r*))

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*:$\forall r \in \{0..t\}$. ((λs. ($[\![\eta]\!]_t$) (*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* (∂ (π_1 *xf*)) = π_2 *xf* (*F t*))
using *tcHyp solves-store-ivpD*(*3*) **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*) ≥ 0
using *solves-store-ivpD*(*2*) *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** $\forall r \in \{0..t\}$. ((λs. ($[\![\eta]\!]_t$) (*F s*)) *has-derivative*
(λx. $x *_R$ (($[\![\partial_t\ \eta]\!]_t$) (*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** (*meson atLeastAtMost-iff order-refl*)
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*:$t \geq 0 \wedge F\ t = c \wedge solvesStoreIVP\ F\ xfList\ a \wedge (\forall r \in \{0..t\}.\ G\ (F\ r))$

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*:$([\![\eta]\!]_t)\ (F\ 0) \geq 0$ **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*(*3*) **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*(*2*) *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** (*meson atLeastAtMost-iff order-refl*)
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 $([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput)\upharpoonleft\partial_P\ (\vartheta \prec \eta)\upharpoonleft]\!]_P)\ st$

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* ⇒ *real store*
where *tcHyp*:$t \geq 0 \wedge F\ t = c \wedge solvesStoreIVP\ F\ xfList\ a$ **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 $\forall\ st.\ G\ st \longrightarrow (\forall\, str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput) \upharpoonleft \partial_P\ \varphi \upharpoonleft]\!]_P)\ st$
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 $\forall\ st.\ G\ st \longrightarrow (\forall\, str.\ str \notin (\pi_1 (\!| set\ xfList |\!)) \longrightarrow st\ (\partial\ str) = 0) \longrightarrow$
$([\![((map\ (vdiff \circ \pi_1)\ xfList) \otimes uInput) \upharpoonleft \partial_P\ \varphi \upharpoonleft]\!]_P)\ st$
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* φ=φ **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 and a single differential equation: $x' = v$.
lemma *motion-with-constant-velocity*:
PRE (λ *s. s ''y'' < s ''x''* $\wedge$ *s ''v'' > 0*)
(*ODEsystem* [(''x'',(λ *s. s ''v''*))] *with* (λ *s. True*))
POST (λ *s.* (*s ''y'' < s ''x''*))
apply(*rule-tac uInput*=[λ *t s. s ''v''* $\cdot$ *t* + *s ''x''*] **in** *dSolve-toSolveUBC*)
prefer *9* **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*=λ *x. s ''v''* **and** *g'1*=λ *x. 0* **in** *derivative-intros*(*173*))
apply(*rule-tac f'1*=λ *x.0* **and** *g'1*=λ *x.1* **in** *derivative-intros*(*176*))
by(*auto intro*: *derivative-intros*)

Same hybrid program verified with dSolve and the system of ODEs: $x' = v, v' = a$. The uniqueness part of the proof requires a preliminary lemma.

lemma *flow-vel-is-galilean-vel*:
assumes *solHyp*:φ_s *solvesTheStoreIVP* [(*x*, $\lambda s.$ *s v*), (*v*, $\lambda s.$ *s a*)] *withInitState s*
and *tHyp*:$r \leq t$ **and** *rHyp*:$0 \leq r$ **and** *distinct*:$x \neq v \wedge v \neq a \wedge x \neq a \wedge a \notin$ *varDiffs*
shows $\varphi_s\ r\ v = s\ a \cdot r + s\ v$
proof−
from *assms* **have** *1*:(($\lambda t.\ \varphi_s\ t\ v$) *solves-ode* ($\lambda t\ r.\ \varphi_s\ t\ a$)) {*0..t*} *UNIV* $\wedge\ \varphi_s\ 0\ v = s\ v$

by (*simp add*: *solvesStoreIVP-def*)
from *assms* **have** *obs*:$\forall$ $r \in \{0..t\}$. φ_s r a $=$ s a
by(*auto simp*: *solvesStoreIVP-def varDiffs-def*)
have *2*:(($\lambda t.$ s $a \cdot t + s$ v) *solves-ode* (λt $r.$ φ_s t a)) $\{0..t\}$ *UNIV*
unfolding *solves-ode-def* **apply**(*subgoal-tac* (($\lambda x.$ s $a \cdot x + s$ v) *has-vderiv-on* ($\lambda x.$ s a)) $\{0..t\}$)
using *obs* **apply** (*simp add*: *has-vderiv-on-def*) **by**(*rule galilean-transform*)
have *3*:*unique-on-bounded-closed 0* $\{0..t\}$ (s v) (λt $r.$ φ_s t a) *UNIV* (*if* $t = 0$ *then 1 else* $1/(t+1)$)
apply(*simp add*: *ubc-definitions del*: *comp-apply*, *rule conjI*)
using *rHyp tHyp obs* **apply**(*simp-all del*: *comp-apply*)
apply(*clarify*, *rule continuous-intros*) **prefer** *3* **apply** *safe*
apply(*rule continuous-intros*)
apply(*auto intro*: *continuous-intros*)
by (*metis continuous-on-const continuous-on-eq*)
thus φ_s r $v = s$ $a \cdot r + s$ v
apply(*rule-tac unique-on-bounded-closed.unique-solution*[*of 0* $\{0..t\}$ s v (λt $r.$ φ_s t a) *UNIV* (*if* $t = 0$ *then 1 else* 1 / ($t + 1$)) ($\lambda t.$ φ_s t v)])
using *rHyp tHyp 1 2* **and** *3* **by** *auto*
qed

lemma *motion-with-constant-acceleration*:
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 uInput*=[λ t $s.$ s $''a'' \cdot t$ ^ $2/2$ + s $''v'' \cdot t + s$ $''x''$,
λ t $s.$ s $''a'' \cdot t + s$ $''v''$] **in** *dSolve-toSolveUBC*)
prefer *9* **subgoal by**(*simp add*: *wp-trafo vdiff-def add-strict-increasing2*)
prefer *6* **subgoal**
apply(*simp add*: *vdiff-def*, *clarify*, *rule conjI*)
by(*rule galilean-transform*)+
prefer *6* **subgoal**
apply(*simp add*: *vdiff-def*, *safe*)
apply(*rule continuous-intros*)
by(*auto intro*: *continuous-intros*)
prefer *6* **subgoal**
apply(*simp add*: *vdiff-def*, *safe*)
subgoal for s φ_s t r **apply**(*rule flow-vel-is-galilean-vel*[*of* φ_s $''x''$ - - - - t])
by(*simp-all add*: *varDiffs-def vdiff-def*)
apply(*simp add*: *solvesStoreIVP-def vdiff-def varDiffs-def*) **done**
by(*auto simp*: *varDiffs-def vdiff-def*)

Example of a hybrid system with two modes verified with the equality dS. We also need to provide a previous (similar) lemma.

lemma *flow-vel-is-galilean-vel2*:
assumes *solHyp*:φ_s *solvesTheStoreIVP* [(x, $\lambda s.$ s v), (v, $\lambda s.$ $-$ s a)] *withInitState* s
and *tHyp*:$r \leq t$ **and** *rHyp*:$0 \leq r$ **and** *distinct*:$x \neq v \wedge v \neq a \wedge x \neq a \wedge a \notin$ *varDiffs*

shows φ_s *r v* = *s v* − *s a* · *r*
proof−
from *assms* **have** *1*:((λ*t*. φ_s *t v*) *solves-ode* (λ*t r*. − φ_s *t a*)) {*0*..*t*} *UNIV* ∧ φ_s *0 v* = *s v*
  by (*simp add*: *solvesStoreIVP-def*)
from *assms* **have** *obs*:∀ *r* ∈ {*0*..*t*}. φ_s *r a* = *s a*
  by(*auto simp*: *solvesStoreIVP-def varDiffs-def*)
have *2*:((λ*t*. − *s a* · *t* + *s v*) *solves-ode* (λ*t r*. − φ_s *t a*)) {*0*..*t*} *UNIV*
  unfolding *solves-ode-def* **apply**(*subgoal-tac* ((λ*x*. − *s a* · *x* + *s v*) *has-vderiv-on* (λ*x*. − *s a*)) {*0*..*t*})
  using *obs* **apply** (*simp add*: *has-vderiv-on-def*) **by**(*rule galilean-transform*)
have *3*:*unique-on-bounded-closed 0* {*0*..*t*} (*s v*) (λ*t r*. − φ_s *t a*) *UNIV* (*if t* = *0 then 1 else 1*/(*t*+*1*))
  apply(*simp add*: *ubc-definitions del*: *comp-apply*, *rule conjI*)
  using *rHyp tHyp obs* **apply**(*simp-all del*: *comp-apply*)
  apply(*clarify*, *rule continuous-intros*) **prefer** *3* **apply** *safe*
  apply(*rule continuous-intros*)
  apply(*auto intro*: *continuous-intros*)
  by (*metis continuous-on-const continuous-on-eq*)
thus φ_s *r v* = *s v* − *s a* · *r*
  apply(*rule-tac unique-on-bounded-closed.unique-solution*[*of 0* {*0*..*t*} *s v* (λ*t r*. − φ_s *t a*) *UNIV* (*if t* = *0 then 1 else 1* / (*t* + *1*)) (λ*t*. φ_s *t v*)])
  using *rHyp tHyp 1 2* **and** *3* **by** *auto*
qed

lemma *single-hop-ball*:
  PRE (λ *s*. *0* ≤ *s* ''*x*'' ∧ *s* ''*x*'' = *H* ∧ *s* ''*v*'' = *0* ∧ *s* ''*g*'' > *0* ∧ *1* ≥ *c* ∧ *c* ≥ *0*)
  (((*ODEsystem* [(''*x*'', λ *s*. *s* ''*v*''),(''*v*'',λ *s*. − *s* ''*g*'')] *with* (λ *s*. *0* ≤ *s* ''*x*''))));
  (*IF* (λ *s*. *s* ''*x*'' = *0*) *THEN* (''*v*'' ::= (λ *s*. − *c* · *s* ''*v*'')) *ELSE* (''*v*'' ::= (λ *s*. *s* ''*v*'')) *FI*))
  POST (λ *s*. *0* ≤ *s* ''*x*'' ∧ *s* ''*x*'' ≤ *H*)
  apply(*simp*, *subst dS*[*of* [λ *t s*. − *s* ''*g*'' · *t* ˆ *2*/*2* + *s* ''*v*'' · *t* + *s* ''*x*'', λ *t s*. − *s* ''*g*'' · *t* + *s* ''*v*'']])
  — Given solution is actually a solution.
  apply(*simp add*: *vdiff-def varDiffs-def solvesStoreIVP-def solves-ode-def has-vderiv-on-singleton*, *safe*)
  apply(*rule galilean-transform-eq*, *simp*)+
  apply(*rule galilean-transform*)+
  — Uniqueness of the flow.
  apply(*rule ubcStoreUniqueSol*, *simp*)
  apply(*simp add*: *vdiff-def del*: *comp-apply*)
  apply(*auto intro*: *continuous-intros del*: *comp-apply*)[*1*]
  apply(*rule continuous-intros*)+
  apply(*simp add*: *vdiff-def*, *safe*)
  apply(*clarsimp*) **subgoal for** *s X t τ*
  apply(*rule flow-vel-is-galilean-vel2*[*of X* ''*x*''])
  by(*simp-all add*: *varDiffs-def vdiff-def*)
  apply(*simp add*: *vdiff-def varDiffs-def solvesStoreIVP-def*)

apply(*simp add*: *vdiff-def varDiffs-def solvesStoreIVP-def solves-ode-def has-vderiv-on-singleton galilean-transform-eq galilean-transform*)
— Relation Between the guard and the postcondition.
by(*auto simp*: *vdiff-def p2r-def*)

— Example of hybrid program verified with differential weakening.
lemma *system-where-the-guard-implies-the-postcondition*:
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 *system-where-the-guard-implies-the-postcondition2*:
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 solvesStoreIVP-def*)
by *auto*

— Example of system proved with a differential invariant.
lemma *circular-motion*:
PRE (λ *s.* (*s* ''*x*'') $\cdot$ (*s* ''*x*'') + (*s* ''*y*'') $\cdot$ (*s* ''*y*'') − (*s* ''*r*'') $\cdot$ (*s* ''*r*'') = *0*)
(*ODEsystem* [(''*x*'',(λ *s. s* ''*y*'')),(''*y*'',(λ *s.* − *s* ''*x*''))] *with G*)
POST (λ *s.* (*s* ''*x*'') $\cdot$ (*s* ''*x*'') + (*s* ''*y*'') $\cdot$ (*s* ''*y*'') − (*s* ''*r*'') $\cdot$ (*s* ''*r*'') = *0*)
apply(*rule-tac* η=(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 (λ *s. s* ''*x*'' > *s* ''*y*'' $\wedge$ *s* ''*v*'' > *0*)
(*ODEsystem* [(''*x*'', λ *s. s* ''*v*'')] *with* (λ *s. True*))
POST (λ *s. s* ''*x*''> *s* ''*y*'')
apply(*rule-tac C* = λ *s. s* ''*v*'' > *0* **in** *dCut*)
apply(*rule-tac* φ = (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* = λ *s. s* ''*x*'' > *s* ''*y*'' **in** *dCut*)
apply(*rule-tac* φ=(t_V ''*y*'') $\prec$ (t_V ''*x*'') **and** *uInput*=[t_V ''*v*''] **and** *F*=λ *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$(rule\text{-}tac\ C = \lambda\ s.\ \ s\ ''a'' > 0$ **in** $dCut)$
apply$(rule\text{-}tac\ \varphi = (t_C\ 0) \prec (t_V\ ''a'')$ **and** $uInput=[t_V\ ''v'', t_V\ ''a'']$**in** $dInvFinal)$
apply$(simp\text{-}all\ add\text{: } vdiff\text{-}def\ varDiffs\text{-}def, clarify, erule\text{-}tac\ x=''a''$ **in** $allE, simp)$
apply$(rule\text{-}tac\ C = \lambda\ s.\ \ s\ ''v'' \geq 0$ **in** $dCut)$
apply$(rule\text{-}tac\ \varphi = (t_C\ 0) \preceq (t_V\ ''v'')$ **and** $uInput=[t_V\ ''v'', t_V\ ''a'']$ **in** $dInvFi\text{-}$
$nal)$
apply$(simp\text{-}all\ add\text{: } vdiff\text{-}def\ varDiffs\text{-}def)$
apply$(rule\text{-}tac\ C = \lambda\ s.\ \ s\ ''x'' >\ \ s\ ''y''$ **in** $dCut)$
apply$(rule\text{-}tac\ \varphi = (t_V\ ''y'') \prec (t_V\ ''x'')$ **and** $uInput=[t_V\ ''v'', t_V\ ''a'']$**in** $dInv\text{-}$
$Final)$
apply$(simp\text{-}all\ add\text{: } varDiffs\text{-}def\ vdiff\text{-}def, clarify, erule\text{-}tac\ x=''y''$ **in** $allE, simp)$
using *dWeakening* **by** *simp*

— We revisit the two modes example from before, and prove it with invariants.
lemma *single-hop-ball-and-invariants*:
$PRE\ (\lambda\ s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' = H \wedge s\ ''v'' = 0 \wedge s\ ''g'' > 0 \wedge 1 \geq c \wedge c$
$\geq 0)$
$(((ODEsystem\ [(''x'', \lambda\ s.\ s\ ''v''),(''v'',\lambda\ s.\ -\ s\ ''g'')]\ with\ (\lambda\ s.\ 0 \leq s\ ''x'')));$
$(IF\ (\lambda\ s.\ s\ ''x'' = 0)\ THEN\ (''v'' ::= (\lambda\ s.\ -\ c \cdot s\ ''v''))\ ELSE\ (''v'' ::= (\lambda$
$s.\ s\ ''v''))\ FI))$
$POST\ (\lambda\ s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' \leq H)$
apply$(simp\ add\text{: } d\text{-}p2r,\ subgoal\text{-}tac\ rdom\ \lceil\lambda s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' = H \wedge s$
$''v'' = 0 \wedge 0 < s\ ''g'' \wedge c \leq 1 \wedge 0 \leq c\rceil$
$\subseteq wp\ (ODEsystem\ [(''x'', \lambda s.\ s\ ''v''), (''v'', \lambda s.\ -\ s\ ''g'')]\ with\ (\lambda s.\ 0 \leq s\ ''x'')$
$)$
$\lceil inf\ (sup\ (-\ (\lambda s.\ s\ ''x'' = 0))\ (\lambda s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' \leq H))\ (sup\ (\lambda s.\ s$
$''x'' = 0)\ (\lambda s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' \leq H))\rceil)$
apply$(simp\ add\text{: } d\text{-}p2r,\ rule\text{-}tac\ C = \lambda\ s.\ \ s\ ''g'' > 0$ **in** $dCut)$
apply$(rule\text{-}tac\ \varphi = (t_C\ 0) \prec (t_V\ ''g'')$ **and** $uInput=[t_V\ ''v'', \ominus\ t_V\ ''g'']$**in**
$dInvFinal)$
apply$(simp\text{-}all\ add\text{: } vdiff\text{-}def\ varDiffs\text{-}def, clarify, erule\text{-}tac\ x=''g''$ **in** $allE,$
$simp)$
apply$(rule\text{-}tac\ C =\lambda\ s.\ \ s\ ''v'' \leq 0$ **in** $dCut)$
apply$(rule\text{-}tac\ \varphi = (t_V\ ''v'') \preceq (t_C\ 0)$ **and** $uInput=[t_V\ ''v'', \ominus\ t_V\ ''g'']$ **in**
$dInvFinal)$
apply$(simp\text{-}all\ add\text{: } vdiff\text{-}def\ varDiffs\text{-}def)$
apply$(rule\text{-}tac\ C = \lambda\ s.\ \ s\ ''x'' \leq\ \ H$ **in** $dCut)$
apply$(rule\text{-}tac\ \varphi = (t_V\ ''x'') \preceq (t_C\ H)$ **and** $uInput=[t_V\ ''v'', \ominus\ t_V\ ''g'']$**in**
$dInvFinal)$
apply$(simp\text{-}all\ add\text{: } varDiffs\text{-}def\ vdiff\text{-}def)$
using *dWeakening* **by** *simp*
declare *d-p2r* [*simp*]

end