

Ordinary Differential Equations

Fabian Immler

September 23, 2021

Abstract

Session `Ordinary-Differential-Equations` formalizes ordinary differential equations (ODEs) and initial value problems. This work comprises proofs for local and global existence of unique solutions (Picard-Lindelöf theorem). Moreover, it contains a formalization of the (continuous or even differentiable) dependency of the flow on initial conditions as the *flow* of ODEs.

Not in the generated document are the following sessions:

- `HOL-ODE-Numerics`: Rigorous numerical algorithms for computing enclosures of solutions based on Runge-Kutta methods and affine arithmetic. Reachability analysis with splitting and reduction at hyperplanes.
- `HOL-ODE-Examples`: Applications of the numerical algorithms to concrete systems of ODEs (e.g., van der Pol and Lorenz attractor).

Contents

1	Auxiliary Lemmas	3
1.1	there is no inner product for type $'a \Rightarrow_L 'b$	3
1.2	Topology	4
1.3	Vector Spaces	4
1.4	Reals	4
1.5	Balls	4
1.6	Boundedness	4
1.7	Intervals	4
1.8	Extended Real Intervals	5
1.9	Euclidean Components	5
1.10	Operator Norm	5
1.11	Limits	5
1.12	Continuity	5
1.13	Derivatives	5
1.14	Integration	6
1.15	conditionally complete lattice	6

1.16	Lists	6
1.17	Set(sum)	7
1.18	Max	7
1.19	Uniform Limit	7
1.20	Bounded Linear Functions	7
1.21	Order Transitivity Attributes	8
1.22	point reflection	8
1.23	(Counter)Example of Mean Value Theorem in Euclidean Space	9
1.24	Vector derivative on a set	10
1.25	interval integral	14
1.26	Gronwall	18
2	Initial Value Problems	20
2.1	Solutions of IVPs	21
2.1.1	Connecting solutions	23
2.2	unique solution with initial value	23
2.3	ivp on interval	26
2.4	Picard-Lindelof on set of functions into closed set	27
2.4.1	Unique solution	32
2.5	Picard-Lindelof for $X = UNIV$	35
2.6	Picard-Lindelof on cylindric domain	35
2.7	Picard-Lindelof On Open Domains	37
2.7.1	Local Solution with local Lipschitz	37
2.7.2	Global maximal flow with local Lipschitz	40
3	Bounded Linear Operator	44
4	Multivariate Taylor	46
4.1	Symmetric second derivative	47
5	Flow	49
5.1	simp rules for integrability (TODO: move)	49
5.2	Nonautonomous IVP on maximal existence interval	51
5.3	Differentiability of the flow0	63
6	Upper and Lower Solutions	65
6.1	explicit representation of hyperplanes / halfspaces	83
6.2	explicit H representation of polytopes (mind <i>Polytopes.thy</i>)	83
6.3	predicates for reachability analysis	84
6.4	Poincare Map	86
6.5	conditions for continuous return time	88
7	Linear ODE	96

1 Auxiliary Lemmas

theory *ODE-Auxiliarities*

imports

HOL-Analysis.Analysis

HOL-Library.Float

List-Index.List-Index

Affine-Arithmetic.Affine-Arithmetic-Auxiliarities

Affine-Arithmetic.Executable-Euclidean-Space

begin

instantiation *prod* :: (*zero-neq-one*, *zero-neq-one*) *zero-neq-one*

begin

definition *1* = (*1*, *1*)

instance *<proof>*

end

1.1 there is no inner product for type $'a \Rightarrow_L 'b$

lemma (*in real-inner*) *parallelogram-law*: $(\text{norm } (x + y))^2 + (\text{norm } (x - y))^2 = 2 * (\text{norm } x)^2 + 2 * (\text{norm } y)^2$

<proof>

locale *no-real-inner*

begin

lift-definition *fstzero*::(*real*real*) \Rightarrow_L (*real*real*) **is** $\lambda(x, y). (x, 0)$

<proof>

lemma [*simp*]: *fstzero* (*a*, *b*) = (*a*, *0*)

<proof>

lift-definition *zerosnd*::(*real*real*) \Rightarrow_L (*real*real*) **is** $\lambda(x, y). (0, y)$

<proof>

lemma [*simp*]: *zerosnd* (*a*, *b*) = (*0*, *b*)

<proof>

lemma *fstzero-add-zerosnd*: *fstzero* + *zerosnd* = *id-blinfun*

<proof>

lemma *norm-fstzero-zerosnd*: *norm* *fstzero* = *1* *norm* *zerosnd* = *1* *norm* (*fstzero* - *zerosnd*) = *1*

<proof>

compare with $(\text{norm } (?x + ?y))^2 + (\text{norm } (?x - ?y))^2 = 2 * (\text{norm } ?x)^2 + 2 * (\text{norm } ?y)^2$

lemma $(\text{norm } (\text{fstzero} + \text{zerosnd}))^2 + (\text{norm } (\text{fstzero} - \text{zerosnd}))^2 \neq 2 * (\text{norm } \text{fstzero})^2 + 2 * (\text{norm } \text{zerosnd})^2$
 ⟨proof⟩

end

1.2 Topology

1.3 Vector Spaces

lemma *ex-norm-eq-1*: $\exists x. \text{norm } (x::'a::\{\text{real-normed-vector}, \text{perfect-space}\}) = 1$
 ⟨proof⟩

1.4 Reals

1.5 Balls

sometimes $(?y \in \text{ball } ?x ?e) = (\text{dist } ?x ?y < ?e)$ etc. are not good [*simp*] rules (although they are often useful): not sure that inequalities are “simpler” than set membership (distorts automatic reasoning when only sets are involved)

lemmas [*simp del*] = *mem-ball mem-cball mem-sphere mem-ball-0 mem-cball-0*

1.6 Boundedness

lemma *bounded-subset-cboxE*:
assumes $\bigwedge i. i \in \text{Basis} \implies \text{bounded } ((\lambda x. x \cdot i) ' X)$
obtains *a b where* $X \subseteq \text{cbox } a b$
 ⟨proof⟩

lemma *bounded-euclideanI*:
assumes $\bigwedge i. i \in \text{Basis} \implies \text{bounded } ((\lambda x. x \cdot i) ' X)$
shows *bounded* X
 ⟨proof⟩

1.7 Intervals

notation *closed-segment* $((1\{---\}))$
notation *open-segment* $((1\{<--<-\}))$

lemma *min-zero-mult-nonneg-le*: $0 \leq h' \implies h' \leq h \implies \text{min } 0 (h * k::\text{real}) \leq h' * k$
 ⟨proof⟩

lemma *max-zero-mult-nonneg-le*: $0 \leq h' \implies h' \leq h \implies h' * k \leq \text{max } 0 (h * k::\text{real})$
 ⟨proof⟩

lemmas *closed-segment-eq-real-ivl = closed-segment-eq-real-ivl*

lemma *bdd-above-is-interval**I*: *bdd-above I if is-interval I a ≤ b a ∈ I b ∉ I for I::real set*
 ⟨proof⟩

lemma *bdd-below-is-interval**I*: *bdd-below I if is-interval I a ≤ b a ∉ I b ∈ I for I::real set*
 ⟨proof⟩

1.8 Extended Real Intervals

1.9 Euclidean Components

1.10 Operator Norm

1.11 Limits

lemma *eventually-open-cball*:
 assumes *open X*
 assumes *x ∈ X*
 shows *eventually (λe. cball x e ⊆ X) (at-right 0)*
 ⟨proof⟩

1.12 Continuity

1.13 Derivatives

lemma
if-eventually-has-derivative:
 assumes *(f has-derivative F') (at x within S)*
 assumes $\forall_F x$ *in at x within S. P x P x x ∈ S*
 shows *((λx. if P x then f x else g x) has-derivative F') (at x within S)*
 ⟨proof⟩

lemma *norm-le-in-cube**I*: *norm x ≤ norm y*
 if $\bigwedge i. i \in \text{Basis} \implies \text{abs } (x \cdot i) \leq \text{abs } (y \cdot i)$ **for** *x y*
 ⟨proof⟩

lemma *has-derivative-partials-euclidean-convex**I*:
 fixes *f::'a::euclidean-space ⇒ 'b::real-normed-vector*
 assumes *f'*: $\bigwedge i x xi. i \in \text{Basis} \implies (\forall j \in \text{Basis}. x \cdot j \in X j) \implies xi = x \cdot i \implies$
 $((\lambda p. f (x + (p - x \cdot i) *_R i)) \text{ has-vector-derivative } f' i x)$ *(at xi within X i)*
 assumes *df-cont*: $\bigwedge i. i \in \text{Basis} \implies (f' i \longrightarrow (f' i x))$ *(at x within {x. ∀ j ∈ Basis. x · j ∈ X j})*
 assumes $\bigwedge i. i \in \text{Basis} \implies x \cdot i \in X i$
 assumes $\bigwedge i. i \in \text{Basis} \implies \text{convex } (X i)$
 shows *(f has-derivative (λh. ∑ j ∈ Basis. (h · j) *_R f' j x)) (at x within {x. ∀ j ∈ Basis. x · j ∈ X j})*
 (is - *(at x within ?S)*)
 ⟨proof⟩

lemma*frechet-derivative-equals-partial-derivative:***fixes** $f :: 'a :: euclidean-space \Rightarrow 'a$ **assumes** $Df: \bigwedge x. (f \text{ has-derivative } Df \ x) \text{ (at } x)$ **assumes** $f': ((\lambda p. f \ (x + (p - x \cdot i) *_{\mathbb{R}} i) \cdot b) \text{ has-real-derivative } f' \ x \ i \ b) \text{ (at } (x \cdot i))$ **shows** $Df \ x \ i \cdot b = f' \ x \ i \ b$ *<proof>*

1.14 Integration

lemmas *content-real[simp]***lemmas** *integrable-continuous[intro, simp]***and** *integrable-continuous-real[intro, simp]***lemma** *integral-eucl-le:***fixes** $f \ g :: 'a :: euclidean-space \Rightarrow 'b :: ordered-euclidean-space$ **assumes** $f \text{ integrable-on } s$ **and** $g \text{ integrable-on } s$ **and** $\bigwedge x. x \in s \implies f \ x \leq g \ x$ **shows** $\text{integral } s \ f \leq \text{integral } s \ g$ *<proof>***lemma***integral-ivl-bound:***fixes** $l \ u :: 'a :: ordered-euclidean-space$ **assumes** $\bigwedge x \ h'. h' \in \{t0 \ .. \ h\} \implies x \in \{t0 \ .. \ h\} \implies (h' - t0) *_{\mathbb{R}} f \ x \in \{l \ .. \ u\}$ **assumes** $t0 \leq h$ **assumes** $f\text{-int}: f \text{ integrable-on } \{t0 \ .. \ h\}$ **shows** $\text{integral } \{t0 \ .. \ h\} \ f \in \{l \ .. \ u\}$ *<proof>***lemma***add-integral-ivl-bound:***fixes** $l \ u :: 'a :: ordered-euclidean-space$ **assumes** $\bigwedge x \ h'. h' \in \{t0 \ .. \ h\} \implies x \in \{t0 \ .. \ h\} \implies (h' - t0) *_{\mathbb{R}} f \ x \in \{l - x0 \ .. \ u - x0\}$ **assumes** $t0 \leq h$ **assumes** $f\text{-int}: f \text{ integrable-on } \{t0 \ .. \ h\}$ **shows** $x0 + \text{integral } \{t0 \ .. \ h\} \ f \in \{l \ .. \ u\}$ *<proof>*

1.15 conditionally complete lattice

1.16 Lists

lemma*Ball-set-Cons[simp]:* $(\forall a \in \text{set-Cons } x \ y. P \ a) \iff (\forall a \in x. \forall b \in y. P \ (a \# b))$

<proof>

lemma *set-cons-eq-empty*[iff]: *set-Cons* $a\ b = \{\}$ $\longleftrightarrow a = \{\} \vee b = \{\}$
<proof>

lemma *listset-eq-empty-iff*[iff]: *listset* $XS = \{\}$ $\longleftrightarrow \{\} \in \text{set } XS$
<proof>

lemma *sing-in-sings*[simp]: $[x] \in (\lambda x. [x])\ 'xd \longleftrightarrow x \in xd$
<proof>

lemma *those-eq-None-set-iff*: *those* $xs = \text{None}$ $\longleftrightarrow \text{None} \in \text{set } xs$
<proof>

lemma *those-eq-Some-lengthD*: *those* $xs = \text{Some } ys \implies \text{length } xs = \text{length } ys$
<proof>

lemma *those-eq-Some-map-Some-iff*: *those* $xs = \text{Some } ys \longleftrightarrow (xs = \text{map } \text{Some } ys)$ (is ?l \longleftrightarrow ?r)
<proof>

1.17 Set(sum)

1.18 Max

1.19 Uniform Limit

1.20 Bounded Linear Functions

lift-definition *comp3*::— TODO: name?
 $('c::\text{real-normed-vector} \Rightarrow_L 'd::\text{real-normed-vector}) \Rightarrow ('b::\text{real-normed-vector} \Rightarrow_L 'c) \Rightarrow_L 'b \Rightarrow_L 'd$ is
 $\lambda(cd::('c \Rightarrow_L 'd)) (bc::'b \Rightarrow_L 'c). (cd\ o_L\ bc)$
<proof>

lemma *blinfun-apply-comp3*[simp]: *blinfun-apply* (*comp3* a) $b = (a\ o_L\ b)$
<proof>

lemma *bounded-linear-comp3*[bounded-linear]: *bounded-linear* *comp3*
<proof>

lift-definition *comp12*::— TODO: name?
 $('a::\text{real-normed-vector} \Rightarrow_L 'c::\text{real-normed-vector}) \Rightarrow ('b::\text{real-normed-vector} \Rightarrow_L 'c) \Rightarrow ('a \times 'b) \Rightarrow_L 'c$
is $\lambda f\ g (a, b). f\ a + g\ b$
<proof>

lemma *blinfun-apply-comp12*[simp]: *blinfun-apply* (*comp12* $f\ g$) $b = f (fst\ b) + g (snd\ b)$
<proof>

1.21 Order Transitivity Attributes

$\langle ML \rangle$

1.22 point reflection

definition $\text{preflect}::'a::\text{real-vector} \Rightarrow 'a \Rightarrow 'a$ **where** $\text{preflect} \equiv \lambda t0 t. 2 *_{\mathbb{R}} t0 - t$

lemma $\text{preflect-preflect}[simp]: \text{preflect } t0 (\text{preflect } t0 t) = t$
 $\langle proof \rangle$

lemma $\text{preflect-preflect-image}[simp]: \text{preflect } t0 ` \text{preflect } t0 ` S = S$
 $\langle proof \rangle$

lemma $\text{is-interval-preflect}[simp]: \text{is-interval } (\text{preflect } t0 ` S) \longleftrightarrow \text{is-interval } S$
 $\langle proof \rangle$

lemma $\text{iv-in-preflect-image}[intro, simp]: t0 \in T \Longrightarrow t0 \in \text{preflect } t0 ` T$
 $\langle proof \rangle$

lemma $\text{preflect-tendsto}[tendsto-intros]:$
fixes $l::'a::\text{real-normed-vector}$
shows $(g \longrightarrow l) F \Longrightarrow (h \longrightarrow m) F \Longrightarrow ((\lambda x. \text{preflect } (g x) (h x)) \longrightarrow \text{preflect } l m) F$
 $\langle proof \rangle$

lemma $\text{continuous-preflect}[continuous-intros]:$
fixes $a::'a::\text{real-normed-vector}$
shows $\text{continuous } (\text{at } a \text{ within } A) (\text{preflect } t0)$
 $\langle proof \rangle$

lemma
fixes $t0::'a::\text{ordered-real-vector}$
shows $\text{preflect-le}[simp]: t0 \leq \text{preflect } t0 b \longleftrightarrow b \leq t0$
and $\text{le-preflect}[simp]: \text{preflect } t0 b \leq t0 \longleftrightarrow t0 \leq b$
and $\text{antimono-preflect}: \text{antimono } (\text{preflect } t0)$
and $\text{preflect-le-preflect}[simp]: \text{preflect } t0 a \leq \text{preflect } t0 b \longleftrightarrow b \leq a$
and $\text{preflect-eq-cancel}[simp]: \text{preflect } t0 a = \text{preflect } t0 b \longleftrightarrow a = b$
 $\langle proof \rangle$

lemma $\text{preflect-eq-point-iff}[simp]: t0 = \text{preflect } t0 s \longleftrightarrow t0 = s \text{ preflect } t0 s = t0$
 $\longleftrightarrow t0 = s$
 $\langle proof \rangle$

lemma $\text{preflect-minus-self}[simp]: \text{preflect } t0 s - t0 = t0 - s$
 $\langle proof \rangle$

end

theory $MVT-Ex$

imports

HOL–Analysis.Analysis
HOL–Decision-Procs.Approximation
../ODE-Auxiliarities

begin

1.23 (Counter)Example of Mean Value Theorem in Euclidean Space

There is no exact analogon of the mean value theorem in the multivariate case!

lemma *MVT-wrong: assumes*

$\bigwedge J a u (f::real*real \Rightarrow real*real).$
 $(\bigwedge x. FDERIV f x :> J x) \Longrightarrow$
 $(\exists t \in \{0 <.. < 1\}. f (a + u) - f a = J (a + t *R u) u)$

shows *False*

<proof>

lemma *MVT-corrected:*

fixes $f::'a::ordered-euclidean-space \Rightarrow 'b::euclidean-space$
assumes $fderiv: \bigwedge x. x \in D \Longrightarrow (f \text{ has-derivative } J x) \text{ (at } x \text{ within } D)$
assumes $line-in: \bigwedge x. \llbracket 0 \leq x; x \leq 1 \rrbracket \Longrightarrow a + x *R u \in D$
shows $(\exists t \in Basis \rightarrow \{0 <.. < 1\}. (f (a + u) - f a) = (\sum i \in Basis. (J (a + t i *R u) u \cdot i) *R i))$

<proof>

lemma *MVT-ivl:*

fixes $f::'a::ordered-euclidean-space \Rightarrow 'b::ordered-euclidean-space$
assumes $fderiv: \bigwedge x. x \in D \Longrightarrow (f \text{ has-derivative } J x) \text{ (at } x \text{ within } D)$
assumes $J-ivl: \bigwedge x. x \in D \Longrightarrow J x u \in \{J0 .. J1\}$
assumes $line-in: \bigwedge x. x \in \{0..1\} \Longrightarrow a + x *R u \in D$
shows $f (a + u) - f a \in \{J0..J1\}$

<proof>

lemma *MVT:*

shows
 $\bigwedge J J0 J1 a u (f::real*real \Rightarrow real*real).$
 $(\bigwedge x. FDERIV f x :> J x) \Longrightarrow$
 $(\bigwedge x. J x u \in \{J0 .. J1\}) \Longrightarrow$
 $f (a + u) - f a \in \{J0 .. J1\}$

<proof>

lemma *MVT-ivl':*

fixes $f::'a::ordered-euclidean-space \Rightarrow 'b::ordered-euclidean-space$
assumes $fderiv: (\bigwedge x. x \in D \Longrightarrow (f \text{ has-derivative } J x) \text{ (at } x \text{ within } D))$
assumes $J-ivl: \bigwedge x. x \in D \Longrightarrow J x (a - b) \in \{J0..J1\}$
assumes $line-in: \bigwedge x. x \in \{0..1\} \Longrightarrow b + x *R (a - b) \in D$
shows $f a \in \{f b + J0..f b + J1\}$

<proof>

```

end
theory
  Vector-Derivative-On
imports
  HOL-Analysis.Analysis
begin

```

1.24 Vector derivative on a set

- TODO: also for the other derivatives?!
- TODO: move to repository and rewrite assumptions of common lemmas?

definition

```

has-vderiv-on :: (real ⇒ 'a::real-normed-vector) ⇒ (real ⇒ 'a) ⇒ real set ⇒ bool
(infix (has'-vderiv'-on) 50)

```

where

```

(f has-vderiv-on f') S ⟷ (∀ x ∈ S. (f has-vector-derivative f' x) (at x within S))

```

```

lemma has-vderiv-on-empty[intro, simp]: (f has-vderiv-on f') {}
⟨proof⟩

```

lemma has-vderiv-on-subset:

```

assumes (f has-vderiv-on f') S
assumes T ⊆ S
shows (f has-vderiv-on f') T
⟨proof⟩

```

lemma has-vderiv-on-compose:

```

assumes (f has-vderiv-on f') (g ' T)
assumes (g has-vderiv-on g') T
shows (f o g has-vderiv-on (λx. g' x *R f' (g x))) T
⟨proof⟩

```

lemma has-vderiv-on-open:

```

assumes open T
shows (f has-vderiv-on f') T ⟷ (∀ t ∈ T. (f has-vector-derivative f' t) (at t))
⟨proof⟩

```

lemma has-vderiv-on-eq-rhs:— TODO: integrate intro derivative-eq-intros

```

(f has-vderiv-on g') T ⟹ (∧x. x ∈ T ⟹ g' x = f' x) ⟹ (f has-vderiv-on f')
T
⟨proof⟩

```

lemma [THEN has-vderiv-on-eq-rhs, derivative-intros]:

```

shows has-vderiv-on-id: ((λx. x) has-vderiv-on (λx. 1)) T
and has-vderiv-on-const: ((λx. c) has-vderiv-on (λx. 0)) T
⟨proof⟩

```

lemma [*THEN has-vderiv-on-eq-rhs, derivative-intros*]:
fixes $f::real \Rightarrow 'a::real\text{-normed-vector}$
assumes $(f \text{ has-vderiv-on } f')$ T
shows *has-vderiv-on-uminus*: $((\lambda x. - f x) \text{ has-vderiv-on } (\lambda x. - f' x))$ T
 $\langle \text{proof} \rangle$

lemma [*THEN has-vderiv-on-eq-rhs, derivative-intros*]:
fixes $f g::real \Rightarrow 'a::real\text{-normed-vector}$
assumes $(f \text{ has-vderiv-on } f')$ T
assumes $(g \text{ has-vderiv-on } g')$ T
shows *has-vderiv-on-add*: $((\lambda x. f x + g x) \text{ has-vderiv-on } (\lambda x. f' x + g' x))$ T
and *has-vderiv-on-diff*: $((\lambda x. f x - g x) \text{ has-vderiv-on } (\lambda x. f' x - g' x))$ T
 $\langle \text{proof} \rangle$

lemma [*THEN has-vderiv-on-eq-rhs, derivative-intros*]:
fixes $f::real \Rightarrow real$ **and** $g::real \Rightarrow 'a::real\text{-normed-vector}$
assumes $(f \text{ has-vderiv-on } f')$ T
assumes $(g \text{ has-vderiv-on } g')$ T
shows *has-vderiv-on-scaleR*: $((\lambda x. f x *_{\mathbb{R}} g x) \text{ has-vderiv-on } (\lambda x. f x *_{\mathbb{R}} g' x + f' x *_{\mathbb{R}} g x))$ T
 $\langle \text{proof} \rangle$

lemma [*THEN has-vderiv-on-eq-rhs, derivative-intros*]:
fixes $f g::real \Rightarrow 'a::real\text{-normed-algebra}$
assumes $(f \text{ has-vderiv-on } f')$ T
assumes $(g \text{ has-vderiv-on } g')$ T
shows *has-vderiv-on-mult*: $((\lambda x. f x * g x) \text{ has-vderiv-on } (\lambda x. f x * g' x + f' x * g x))$ T
 $\langle \text{proof} \rangle$

lemma *has-vderiv-on-ln*[*THEN has-vderiv-on-eq-rhs, derivative-intros*]:
fixes $g::real \Rightarrow real$
assumes $\bigwedge x. x \in s \implies 0 < g x$
assumes $(g \text{ has-vderiv-on } g')$ s
shows $((\lambda x. \ln (g x)) \text{ has-vderiv-on } (\lambda x. g' x / g x))$ s
 $\langle \text{proof} \rangle$

lemma *fundamental-theorem-of-calculus'*:
fixes $f :: real \Rightarrow 'a::banach$
shows $a \leq b \implies (f \text{ has-vderiv-on } f') \{a .. b\} \implies (f' \text{ has-integral } (f b - f a)) \{a .. b\}$
 $\langle \text{proof} \rangle$

lemma *has-vderiv-on-If*:
assumes $U = S \cup T$
assumes $(f \text{ has-vderiv-on } f') (S \cup (\text{closure } T \cap \text{closure } S))$
assumes $(g \text{ has-vderiv-on } g') (T \cup (\text{closure } T \cap \text{closure } S))$
assumes $\bigwedge x. x \in \text{closure } T \implies x \in \text{closure } S \implies f x = g x$

assumes $\bigwedge x. x \in \text{closure } T \implies x \in \text{closure } S \implies f' x = g' x$
shows $((\lambda t. \text{if } t \in S \text{ then } f t \text{ else } g t) \text{ has-vderiv-on } (\lambda t. \text{if } t \in S \text{ then } f' t \text{ else } g' t)) U$
 <proof>

lemma *mt-very-simple-closed-segmentE*:

fixes $f :: \text{real} \Rightarrow \text{real}$
assumes $(f \text{ has-vderiv-on } f')$ $(\text{closed-segment } a b)$
obtains y **where** $y \in \text{closed-segment } a b$ $f b - f a = (b - a) * f' y$
 <proof>

lemma *mt-simple-closed-segmentE*:

fixes $f :: \text{real} \Rightarrow \text{real}$
assumes $(f \text{ has-vderiv-on } f')$ $(\text{closed-segment } a b)$
assumes $a \neq b$
obtains y **where** $y \in \text{open-segment } a b$ $f b - f a = (b - a) * f' y$
 <proof>

lemma *differentiable-bound-general-open-segment*:

fixes $a :: \text{real}$
and $b :: \text{real}$
and $f :: \text{real} \Rightarrow 'a :: \text{real-normed-vector}$
and $f' :: \text{real} \Rightarrow 'a$
assumes $\text{continuous-on } (\text{closed-segment } a b) f$
assumes $\text{continuous-on } (\text{closed-segment } a b) g$
and $(f \text{ has-vderiv-on } f')$ $(\text{open-segment } a b)$
and $(g \text{ has-vderiv-on } g')$ $(\text{open-segment } a b)$
and $\bigwedge x. x \in \text{open-segment } a b \implies \text{norm } (f' x) \leq g' x$
shows $\text{norm } (f b - f a) \leq \text{abs } (g b - g a)$
 <proof>

lemma *has-vderiv-on-union*:

assumes $(f \text{ has-vderiv-on } g) (s \cup \text{closure } s \cap \text{closure } t)$
assumes $(f \text{ has-vderiv-on } g) (t \cup \text{closure } s \cap \text{closure } t)$
shows $(f \text{ has-vderiv-on } g) (s \cup t)$
 <proof>

lemma *has-vderiv-on-union-closed*:

assumes $(f \text{ has-vderiv-on } g) s$
assumes $(f \text{ has-vderiv-on } g) t$
assumes $\text{closed } s$ $\text{closed } t$
shows $(f \text{ has-vderiv-on } g) (s \cup t)$
 <proof>

lemma *vderiv-on-continuous-on*: $(f \text{ has-vderiv-on } f') S \implies \text{continuous-on } S f$

<proof>

lemma *has-vderiv-on-cong[cong]*:

assumes $\bigwedge x. x \in S \implies f x = g x$

```

assumes  $\bigwedge x. x \in S \implies f' x = g' x$ 
assumes  $S = T$ 
shows  $(f \text{ has-vderiv-on } f') S = (g \text{ has-vderiv-on } g') T$ 
 $\langle \text{proof} \rangle$ 

lemma has-vderiv-eq:
assumes  $(f \text{ has-vderiv-on } f') S$ 
assumes  $\bigwedge x. x \in S \implies f x = g x$ 
assumes  $\bigwedge x. x \in S \implies f' x = g' x$ 
assumes  $S = T$ 
shows  $(g \text{ has-vderiv-on } g') T$ 
 $\langle \text{proof} \rangle$ 

lemma has-vderiv-on-compose':
assumes  $(f \text{ has-vderiv-on } f') (g \text{ ' } T)$ 
assumes  $(g \text{ has-vderiv-on } g') T$ 
shows  $((\lambda x. f (g x)) \text{ has-vderiv-on } (\lambda x. g' x *_R f' (g x))) T$ 
 $\langle \text{proof} \rangle$ 

lemma has-vderiv-on-compose2:
assumes  $(f \text{ has-vderiv-on } f') S$ 
assumes  $(g \text{ has-vderiv-on } g') T$ 
assumes  $\bigwedge t. t \in T \implies g t \in S$ 
shows  $((\lambda x. f (g x)) \text{ has-vderiv-on } (\lambda x. g' x *_R f' (g x))) T$ 
 $\langle \text{proof} \rangle$ 

lemma has-vderiv-on-singleton:  $(y \text{ has-vderiv-on } y') \{t0\}$ 
 $\langle \text{proof} \rangle$ 

lemma
has-vderiv-on-zero-constant:
assumes convex  $s$ 
assumes  $(f \text{ has-vderiv-on } (\lambda h. 0)) s$ 
obtains  $c$  where  $\bigwedge x. x \in s \implies f x = c$ 
 $\langle \text{proof} \rangle$ 

lemma bounded-vderiv-on-imp-lipschitz:
assumes  $(f \text{ has-vderiv-on } f') X$ 
assumes convex: convex  $X$ 
assumes  $\bigwedge x. x \in X \implies \text{norm } (f' x) \leq C \ 0 \leq C$ 
shows  $C\text{-lipschitz-on } X f$ 
 $\langle \text{proof} \rangle$ 

end
theory Interval-Integral-HK
imports Vector-Derivative-On
begin

```

1.25 interval integral

— TODO: move to repo ?!
 — TODO: replace with Bochner Integral?! But FTC for Bochner requires continuity and euclidean space!

definition *has-ivl-integral* ::

(*real* \Rightarrow '*b*::*real-normed-vector*) \Rightarrow '*b* \Rightarrow *real* \Rightarrow *real* \Rightarrow *bool* — TODO: generalize?
 (**infixr** *has'-ivl'-integral* 46)
where (*f has-ivl-integral y*) *a b* \iff (*if a \leq b then (f has-integral y) {a .. b} else (f has-integral - y) {b .. a}*)

definition *ivl-integral*::*real* \Rightarrow *real* \Rightarrow (*real* \Rightarrow '*a*) \Rightarrow '*a*::*real-normed-vector*

where *ivl-integral a b f* = *integral {a .. b} f* - *integral {b .. a} f*

lemma *integral-emptyI[simp]*:

fixes *a b*::*real*

shows *a \geq b \implies integral {a..b} f = 0* *a > b \implies integral {a..b} f = 0*

<proof>

lemma *ivl-integral-unique*: (*f has-ivl-integral y*) *a b* \implies *ivl-integral a b f = y*

<proof>

lemma *fundamental-theorem-of-calculus-ivl-integral*:

fixes *f* :: *real* \Rightarrow '*a*::*banach*

shows (*f has-vderiv-on f'*) (*closed-segment a b*) \implies (*f' has-ivl-integral f b - f a*)
a b

<proof>

lemma

fixes *f* :: *real* \Rightarrow '*a*::*banach*

assumes *f integrable-on (closed-segment a b)*

shows *indefinite-ivl-integral-continuous*:

continuous-on (closed-segment a b) ($\lambda x. ivl-integral a x f$)

continuous-on (closed-segment b a) ($\lambda x. ivl-integral a x f$)

<proof>

lemma

fixes *f* :: *real* \Rightarrow '*a*::*banach*

assumes *f integrable-on (closed-segment a b)*

assumes *c \in closed-segment a b*

shows *indefinite-ivl-integral-continuous-subset*:

continuous-on (closed-segment a b) ($\lambda x. ivl-integral c x f$)

<proof>

lemma *real-Icc-closed-segment*: **fixes** *a b*::*real* **shows** *a \leq b \implies {a .. b} = closed-segment*

a b

<proof>

lemma *ivl-integral-zero[simp]*: *ivl-integral a a f = 0*

<proof>

lemma *ivl-integral-cong*:

assumes $\bigwedge x. x \in \text{closed-segment } a \ b \implies g \ x = f \ x$

assumes $a = c \ b = d$

shows $\text{ivl-integral } a \ b \ f = \text{ivl-integral } c \ d \ g$

<proof>

lemma *ivl-integral-diff*:

$f \text{ integrable-on } (\text{closed-segment } s \ t) \implies g \text{ integrable-on } (\text{closed-segment } s \ t) \implies$

$\text{ivl-integral } s \ t \ (\lambda x. f \ x - g \ x) = \text{ivl-integral } s \ t \ f - \text{ivl-integral } s \ t \ g$

<proof>

lemma *ivl-integral-norm-bound-ivl-integral*:

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes $f \text{ integrable-on } (\text{closed-segment } a \ b)$

and $g \text{ integrable-on } (\text{closed-segment } a \ b)$

and $\bigwedge x. x \in \text{closed-segment } a \ b \implies \text{norm } (f \ x) \leq g \ x$

shows $\text{norm } (\text{ivl-integral } a \ b \ f) \leq \text{abs } (\text{ivl-integral } a \ b \ g)$

<proof>

lemma *ivl-integral-norm-bound-integral*:

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes $f \text{ integrable-on } (\text{closed-segment } a \ b)$

and $g \text{ integrable-on } (\text{closed-segment } a \ b)$

and $\bigwedge x. x \in \text{closed-segment } a \ b \implies \text{norm } (f \ x) \leq g \ x$

shows $\text{norm } (\text{ivl-integral } a \ b \ f) \leq \text{integral } (\text{closed-segment } a \ b) \ g$

<proof>

lemma *norm-ivl-integral-le*:

fixes $f :: \text{real} \Rightarrow \text{real}$

assumes $f \text{ integrable-on } (\text{closed-segment } a \ b)$

and $g \text{ integrable-on } (\text{closed-segment } a \ b)$

and $\bigwedge x. x \in \text{closed-segment } a \ b \implies f \ x \leq g \ x$

and $\bigwedge x. x \in \text{closed-segment } a \ b \implies 0 \leq f \ x$

shows $\text{abs } (\text{ivl-integral } a \ b \ f) \leq \text{abs } (\text{ivl-integral } a \ b \ g)$

<proof>

lemma *ivl-integral-const* [*simp*]:

shows $\text{ivl-integral } a \ b \ (\lambda x. c) = (b - a) *_{\mathbb{R}} c$

<proof>

lemma *ivl-integral-has-vector-derivative*:

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes $\text{continuous-on } (\text{closed-segment } a \ b) \ f$

and $x \in \text{closed-segment } a \ b$

shows $((\lambda u. \text{ivl-integral } a \ u \ f) \text{ has-vector-derivative } f \ x) \text{ (at } x \text{ within closed-segment } a \ b)$

<proof>

lemma *ivl-integral-has-vderiv-on:*

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes *continuous-on (closed-segment a b) f*

shows $((\lambda u. \text{ivl-integral } a \ u \ f) \text{ has-vderiv-on } f) \text{ (closed-segment } a \ b)$

<proof>

lemma *ivl-integral-has-vderiv-on-subset-segment:*

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes *continuous-on (closed-segment a b) f*

and $c \in \text{closed-segment } a \ b$

shows $((\lambda u. \text{ivl-integral } c \ u \ f) \text{ has-vderiv-on } f) \text{ (closed-segment } a \ b)$

<proof>

lemma *ivl-integral-has-vector-derivative-subset:*

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes *continuous-on (closed-segment a b) f*

and $x \in \text{closed-segment } a \ b$

and $c \in \text{closed-segment } a \ b$

shows $((\lambda u. \text{ivl-integral } c \ u \ f) \text{ has-vector-derivative } f \ x) \text{ (at } x \text{ within closed-segment } a \ b)$

<proof>

lemma

compact-interval-eq-Inf-Sup:

fixes $A::\text{real set}$

assumes *is-interval A compact A A \neq {}*

shows $A = \{\text{Inf } A \ .. \ \text{Sup } A\}$

<proof>

lemma *ivl-integral-has-vderiv-on-compact-interval:*

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes *continuous-on A f*

and $c \in A \text{ is-interval } A \text{ compact } A$

shows $((\lambda u. \text{ivl-integral } c \ u \ f) \text{ has-vderiv-on } f) \ A$

<proof>

lemma *ivl-integral-has-vector-derivative-compact-interval:*

fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$

assumes *continuous-on A f*

and *is-interval A compact A x \in A c \in A*

shows $((\lambda u. \text{ivl-integral } c \ u \ f) \text{ has-vector-derivative } f \ x) \text{ (at } x \text{ within } A)$

<proof>

lemma *ivl-integral-combine:*

fixes $f::\text{real} \Rightarrow 'a::\text{banach}$

assumes *f integrable-on (closed-segment a b)*

assumes *f integrable-on (closed-segment b c)*

assumes *f integrable-on (closed-segment a c)*

shows $ivl\text{-integral } a \ b \ f + ivl\text{-integral } b \ c \ f = ivl\text{-integral } a \ c \ f$
 ⟨proof⟩

lemma *integral-equation-swap-initial-value*:

fixes $x :: real \Rightarrow 'a :: banach$
assumes $\bigwedge t. t \in \text{closed-segment } t0 \ t1 \implies x \ t = x \ t0 + ivl\text{-integral } t0 \ t \ (\lambda t. f \ t$
 $(x \ t))$
assumes $t: t \in \text{closed-segment } t0 \ t1$
assumes $int: (\lambda t. f \ t \ (x \ t)) \text{ integrable-on closed-segment } t0 \ t1$
shows $x \ t = x \ t1 + ivl\text{-integral } t1 \ t \ (\lambda t. f \ t \ (x \ t))$
 ⟨proof⟩

lemma *has-integral-nonpos*:

fixes $f :: 'n :: euclidean-space \Rightarrow real$
assumes $(f \text{ has-integral } i) \ s$
and $\forall x \in s. f \ x \leq 0$
shows $i \leq 0$
 ⟨proof⟩

lemma *has-ivl-integral-nonneg*:

fixes $f :: real \Rightarrow real$
assumes $(f \text{ has-ivl-integral } i) \ a \ b$
and $\bigwedge x. a \leq x \implies x \leq b \implies 0 \leq f \ x$
and $\bigwedge x. b \leq x \implies x \leq a \implies f \ x \leq 0$
shows $0 \leq i$
 ⟨proof⟩

lemma *has-ivl-integral-ivl-integral*:

$f \text{ integrable-on } (\text{closed-segment } a \ b) \longleftrightarrow (f \text{ has-ivl-integral } (ivl\text{-integral } a \ b \ f)) \ a$
 b
 ⟨proof⟩

lemma *ivl-integral-nonneg*:

fixes $f :: real \Rightarrow real$
assumes $f \text{ integrable-on } (\text{closed-segment } a \ b)$
and $\bigwedge x. a \leq x \implies x \leq b \implies 0 \leq f \ x$
and $\bigwedge x. b \leq x \implies x \leq a \implies f \ x \leq 0$
shows $0 \leq ivl\text{-integral } a \ b \ f$
 ⟨proof⟩

lemma *ivl-integral-bound*:

fixes $f :: real \Rightarrow 'a :: banach$
assumes $\text{continuous-on } (\text{closed-segment } a \ b) \ f$
assumes $\bigwedge t. t \in (\text{closed-segment } a \ b) \implies \text{norm } (f \ t) \leq B$
shows $\text{norm } (ivl\text{-integral } a \ b \ f) \leq B * \text{abs } (b - a)$
 ⟨proof⟩

lemma *ivl-integral-minus-sets*:

fixes $f :: real \Rightarrow 'a :: banach$

shows f integrable-on (closed-segment c a) \implies f integrable-on (closed-segment c b) \implies f integrable-on (closed-segment a b) \implies
 ivl -integral c a f $-$ ivl -integral c b f $=$ ivl -integral b a f
 <proof>

lemma *ivl-integral-minus-sets'*:

fixes $f::real \Rightarrow 'a::banach$

shows f integrable-on (closed-segment a c) \implies f integrable-on (closed-segment b c) \implies f integrable-on (closed-segment a b) \implies
 ivl -integral a c f $-$ ivl -integral b c f $=$ ivl -integral a b f
 <proof>

end

theory *Gronwall*

imports *Vector-Derivative-On*

begin

1.26 Gronwall

lemma *derivative-quotient-bound*:

assumes g -deriv-on: (g has-vderiv-on g') $\{a .. b\}$

assumes frac-le: $\bigwedge t. t \in \{a .. b\} \implies g' t / g t \leq K$

assumes g' -cont: continuous-on $\{a .. b\}$ g'

assumes g -pos: $\bigwedge t. t \in \{a .. b\} \implies g t > 0$

assumes t -in: $t \in \{a .. b\}$

shows $g t \leq g a * \exp (K * (t - a))$

<proof>

lemma *derivative-quotient-bound-left*:

assumes g -deriv-on: (g has-vderiv-on g') $\{a .. b\}$

assumes frac-ge: $\bigwedge t. t \in \{a .. b\} \implies K \leq g' t / g t$

assumes g' -cont: continuous-on $\{a .. b\}$ g'

assumes g -pos: $\bigwedge t. t \in \{a .. b\} \implies g t > 0$

assumes t -in: $t \in \{a..b\}$

shows $g t \leq g b * \exp (K * (t - b))$

<proof>

lemma *gronwall-general*:

fixes g K C a b **and** $t::real$

defines $G \equiv \lambda t. C + K * \text{integral } \{a..t\} (\lambda s. g s)$

assumes g -le- G : $\bigwedge t. t \in \{a..b\} \implies g t \leq G t$

assumes g -cont: continuous-on $\{a..b\}$ g

assumes g -nonneg: $\bigwedge t. t \in \{a..b\} \implies 0 \leq g t$

assumes pos: $0 < C$ $K > 0$

assumes $t \in \{a..b\}$

shows $g t \leq C * \exp (K * (t - a))$

<proof>

lemma *gronwall-general-left*:

fixes $g K C a b$ **and** $t::real$
defines $G \equiv \lambda t. C + K * integral \{t..b\} (\lambda s. g s)$
assumes $g-le-G: \bigwedge t. t \in \{a..b\} \implies g t \leq G t$
assumes $g-cont: continuous-on \{a..b\} g$
assumes $g-nonneg: \bigwedge t. t \in \{a..b\} \implies 0 \leq g t$
assumes $pos: 0 < C K > 0$
assumes $t \in \{a..b\}$
shows $g t \leq C * exp (-K * (t - b))$
 $\langle proof \rangle$

lemma *gronwall-general-segment*:

fixes $a b::real$
assumes $\bigwedge t. t \in closed-segment a b \implies g t \leq C + K * integral (closed-segment a t) g$
and $continuous-on (closed-segment a b) g$
and $\bigwedge t. t \in closed-segment a b \implies 0 \leq g t$
and $0 < C$
and $0 < K$
and $t \in closed-segment a b$
shows $g t \leq C * exp (K * abs (t - a))$
 $\langle proof \rangle$

lemma *gronwall-more-general-segment*:

fixes $a b c::real$
assumes $\bigwedge t. t \in closed-segment a b \implies g t \leq C + K * integral (closed-segment c t) g$
and $cont: continuous-on (closed-segment a b) g$
and $\bigwedge t. t \in closed-segment a b \implies 0 \leq g t$
and $0 < C$
and $0 < K$
and $t: t \in closed-segment a b$
and $c: c \in closed-segment a b$
shows $g t \leq C * exp (K * abs (t - c))$
 $\langle proof \rangle$

lemma *gronwall*:

fixes $g K C$ **and** $t::real$
defines $G \equiv \lambda t. C + K * integral \{0..t\} (\lambda s. g s)$
assumes $g-le-G: \bigwedge t. 0 \leq t \implies t \leq a \implies g t \leq G t$
assumes $g-cont: continuous-on \{0..a\} g$
assumes $g-nonneg: \bigwedge t. 0 \leq t \implies t \leq a \implies 0 \leq g t$
assumes $pos: 0 < C 0 < K$
assumes $0 \leq t t \leq a$
shows $g t \leq C * exp (K * t)$
 $\langle proof \rangle$

lemma *gronwall-left*:

fixes $g K C$ **and** $t::real$
defines $G \equiv \lambda t. C + K * integral \{t..0\} (\lambda s. g s)$

assumes $g\text{-le-}G$: $\bigwedge t. a \leq t \implies t \leq 0 \implies g\ t \leq G\ t$
assumes $g\text{-cont}$: *continuous-on* $\{a..0\}$ g
assumes $g\text{-nonneg}$: $\bigwedge t. a \leq t \implies t \leq 0 \implies 0 \leq g\ t$
assumes pos : $0 < C\ 0 < K$
assumes $a \leq t\ t \leq 0$
shows $g\ t \leq C * \exp(-K * t)$
<proof>

end

2 Initial Value Problems

theory *Initial-Value-Problem*

imports

../ODE-Auxiliarities
../Library/Interval-Integral-HK
../Library/Gronwall

begin

lemma *clamp-le[simp]*: $x \leq a \implies \text{clamp } a\ b\ x = a$ **for** $x::'a::\text{ordered-euclidean-space}$
<proof>

lemma *clamp-ge[simp]*: $a \leq b \implies b \leq x \implies \text{clamp } a\ b\ x = b$ **for** $x::'a::\text{ordered-euclidean-space}$
<proof>

abbreviation *cfuncset* :: $'a::\text{topological-space set} \Rightarrow 'b::\text{metric-space set} \Rightarrow ('a \Rightarrow_C 'b)$ *set*

(**infixr** \rightarrow_C 60)

where $A \rightarrow_C B \equiv \text{PiC } A\ (\lambda\cdot. B)$

lemma *closed-segment-translation-zero*: $z \in \{z + a \text{ -- } z + b\} \longleftrightarrow 0 \in \{a \text{ -- } b\}$
<proof>

lemma *closed-segment-subset-interval*: $\text{is-interval } T \implies a \in T \implies b \in T \implies \text{closed-segment } a\ b \subseteq T$
<proof>

definition *half-open-segment*:: $'a::\text{real-vector} \Rightarrow 'a \Rightarrow 'a\ \text{set } ((1\ \{\text{---}<\cdot\}))$

where $\text{half-open-segment } a\ b = \{a \text{ -- } b\} - \{b\}$

lemma *half-open-segment-real*:

fixes $a\ b::\text{real}$

shows $\{a \text{ ---} < b\} = (\text{if } a \leq b \text{ then } \{a \text{ ..} < b\} \text{ else } \{b <.. a\})$

<proof>

lemma *closure-half-open-segment*:

fixes $a\ b::\text{real}$

shows $\text{closure } \{a \text{ ---} < b\} = (\text{if } a = b \text{ then } \{\} \text{ else } \{a \text{ -- } b\})$

<proof>

lemma *half-open-segment-subset*[*intro, simp*]:

$\{t0--<t1\} \subseteq \{t0 -- t1\}$
 $x \in \{t0--<t1\} \implies x \in \{t0 -- t1\}$
(*proof*)

lemma *half-open-segment-closed-segmentI*:

$t \in \{t0 -- t1\} \implies t \neq t1 \implies t \in \{t0 --<t1\}$
(*proof*)

lemma *islimpt-half-open-segment*:

fixes $t0\ t1\ s::real$
assumes $t0 \neq t1\ s \in \{t0--t1\}$
shows $s\ islimpt\ \{t0--<t1\}$
(*proof*)

lemma

mem-half-open-segment-eventually-in-closed-segment:
fixes $t::real$
assumes $t \in \{t0--<t1\}$
shows $\forall_F\ t1'\ in\ at\ t1'\ within\ \{t0--<t1\}. t \in \{t0--t1\}$
(*proof*)

lemma *closed-segment-half-open-segment-subsetI*:

fixes $x::real$ **shows** $x \in \{t0--<t1\} \implies \{t0--x\} \subseteq \{t0--<t1\}$
(*proof*)

lemma *dist-component-le*:

fixes $x\ y::'a::euclidean-space$
assumes $i \in Basis$
shows $dist\ (x \cdot i)\ (y \cdot i) \leq dist\ x\ y$
(*proof*)

lemma *sum-inner-Basis-one*: $i \in Basis \implies (\sum_{x \in Basis} x \cdot i) = 1$

(*proof*)

lemma *cball-in-cbox*:

fixes $y::'a::euclidean-space$
shows $cball\ y\ r \subseteq cbox\ (y - r *_{R}\ One)\ (y + r *_{R}\ One)$
(*proof*)

lemma *centered-cbox-in-cball*:

shows $cbox\ (-r *_{R}\ One)\ (r *_{R}\ One::'a::euclidean-space) \subseteq$
 $cball\ 0\ (sqrt(DIM('a)) * r)$
(*proof*)

2.1 Solutions of IVPs

definition

solves-ode :: (real \Rightarrow 'a::real-normed-vector) \Rightarrow (real \Rightarrow 'a \Rightarrow 'a) \Rightarrow real set \Rightarrow 'a set \Rightarrow bool

(**infix** (*solves'-ode*) 50)

where

(*y solves-ode f*) $T X \longleftrightarrow$ (*y has-vderiv-on* ($\lambda t. f t (y t)$)) $T \wedge y \in T \rightarrow X$

lemma *solves-odeI*:

assumes *solves-ode-vderivD*: (*y has-vderiv-on* ($\lambda t. f t (y t)$)) T

and *solves-ode-domainD*: $\bigwedge t. t \in T \Longrightarrow y t \in X$

shows (*y solves-ode f*) $T X$

<proof>

lemma *solves-odeD*:

assumes (*y solves-ode f*) $T X$

shows *solves-ode-vderivD*: (*y has-vderiv-on* ($\lambda t. f t (y t)$)) T

and *solves-ode-domainD*: $\bigwedge t. t \in T \Longrightarrow y t \in X$

<proof>

lemma *solves-ode-continuous-on*: (*y solves-ode f*) $T X \Longrightarrow$ *continuous-on* $T y$

<proof>

lemma *solves-ode-congI*:

assumes (*x solves-ode f*) $T X$

assumes $\bigwedge t. t \in T \Longrightarrow x t = y t$

assumes $\bigwedge t. t \in T \Longrightarrow f t (x t) = g t (x t)$

assumes $T = S \ X = Y$

shows (*y solves-ode g*) $S Y$

<proof>

lemma *solves-ode-cong[cong]*:

assumes $\bigwedge t. t \in T \Longrightarrow x t = y t$

assumes $\bigwedge t. t \in T \Longrightarrow f t (x t) = g t (x t)$

assumes $T = S \ X = Y$

shows (*x solves-ode f*) $T X \longleftrightarrow$ (*y solves-ode g*) $S Y$

<proof>

lemma *solves-ode-on-subset*:

assumes (*x solves-ode f*) $S Y$

assumes $T \subseteq S \ Y \subseteq X$

shows (*x solves-ode f*) $T X$

<proof>

lemma *preflect-solution*:

assumes $t0 \in T$

assumes *sol*: ($\lambda t. x$ (*preflect* $t0 t$)) *solves-ode* ($\lambda t. x. - f$ (*preflect* $t0 t$) x)

(*preflect* $t0$ ' T) X

shows (*x solves-ode f*) $T X$

<proof>

lemma *solution-preflect*:

assumes $t0 \in T$

assumes $sol: (x \text{ solves-ode } f) T X$

shows $((\lambda t. x \text{ (preflect } t0 \ t)}) \text{ solves-ode } (\lambda t x. - f \text{ (preflect } t0 \ t) \ x)) \text{ (preflect } t0 \ ' T) X$
<proof>

lemma *solution-eq-preflect-solution*:

assumes $t0 \in T$

shows $(x \text{ solves-ode } f) T X \longleftrightarrow ((\lambda t. x \text{ (preflect } t0 \ t)}) \text{ solves-ode } (\lambda t x. - f \text{ (preflect } t0 \ t) \ x)) \text{ (preflect } t0 \ ' T) X$
<proof>

lemma *shift-autonomous-solution*:

assumes $sol: (x \text{ solves-ode } f) T X$

assumes $auto: \bigwedge s t. s \in T \implies f s (x s) = f t (x s)$

shows $((\lambda t. x \text{ (} t + t0)) \text{ solves-ode } f) ((\lambda t. t - t0) \ ' T) X$

<proof>

lemma *solves-ode-singleton*: $y \ t0 \in X \implies (y \text{ solves-ode } f) \{t0\} X$

<proof>

2.1.1 Connecting solutions

lemma *connection-solves-ode*:

assumes $x: (x \text{ solves-ode } f) T X$

assumes $y: (y \text{ solves-ode } g) S Y$

assumes $conn-T: \text{closure } S \cap \text{closure } T \subseteq T$

assumes $conn-S: \text{closure } S \cap \text{closure } T \subseteq S$

assumes $conn-x: \bigwedge t. t \in \text{closure } S \implies t \in \text{closure } T \implies x \ t = y \ t$

assumes $conn-f: \bigwedge t. t \in \text{closure } S \implies t \in \text{closure } T \implies f \ t (y \ t) = g \ t (y \ t)$

shows $((\lambda t. \text{if } t \in T \text{ then } x \ t \text{ else } y \ t) \text{ solves-ode } (\lambda t. \text{if } t \in T \text{ then } f \ t \text{ else } g \ t)) (T \cup S) (X \cup Y)$

<proof>

lemma

solves-ode-subset-range:

assumes $x: (x \text{ solves-ode } f) T X$

assumes $s: x \ ' T \subseteq Y$

shows $(x \text{ solves-ode } f) T Y$

<proof>

2.2 unique solution with initial value

definition

$usolves\text{-ode-from} :: (\text{real} \Rightarrow 'a :: \text{real-normed-vector}) \Rightarrow (\text{real} \Rightarrow 'a \Rightarrow 'a) \Rightarrow \text{real} \Rightarrow \text{real set} \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$

$(((-) \text{ usolves}'\text{-ode } (-) \text{ from } (-)) [10, 10, 10] 10)$

— TODO: no idea about mixfix and precedences, check this!

where

$(y \text{ usolves-ode } f \text{ from } t0) \ T \ X \longleftrightarrow (y \text{ solves-ode } f) \ T \ X \wedge t0 \in T \wedge \text{is-interval } T \wedge$
 $(\forall z \ T'. \ t0 \in T' \wedge \text{is-interval } T' \wedge T' \subseteq T \wedge (z \text{ solves-ode } f) \ T' \ X \longrightarrow z \ t0 =$
 $y \ t0 \longrightarrow (\forall t \in T'. \ z \ t = y \ t))$

uniqueness of solution can depend on domain X :

lemma

$((\lambda-. \ 0::\text{real}) \text{ usolves-ode } (\lambda-. \ \text{sqrt}) \text{ from } 0) \ \{0..\} \ \{0\}$
 $((\lambda t. \ t^2 / 4) \text{ solves-ode } (\lambda-. \ \text{sqrt})) \ \{0..\} \ \{0..\}$
 $(\lambda t. \ t^2 / 4) \ 0 = (\lambda-. \ 0::\text{real}) \ 0$
 $\langle \text{proof} \rangle$

TODO: show that if solution stays in interior, then domain can be enlarged!
(?)

lemma *usolves-odeD*:

assumes $(y \text{ usolves-ode } f \text{ from } t0) \ T \ X$
shows $(y \text{ solves-ode } f) \ T \ X$
and $t0 \in T$
and $\text{is-interval } T$
and $\bigwedge z \ T' \ t. \ t0 \in T' \implies \text{is-interval } T' \implies T' \subseteq T \implies (z \text{ solves-ode } f) \ T' \ X$
 $\implies z \ t0 = y \ t0 \implies t \in T' \implies z \ t = y \ t$
 $\langle \text{proof} \rangle$

lemma *usolves-ode-rawI*:

assumes $(y \text{ solves-ode } f) \ T \ X \ t0 \in T \ \text{is-interval } T$
assumes $\bigwedge z \ T' \ t. \ t0 \in T' \implies \text{is-interval } T' \implies T' \subseteq T \implies (z \text{ solves-ode } f)$
 $T' \ X \implies z \ t0 = y \ t0 \implies t \in T' \implies z \ t = y \ t$
shows $(y \text{ usolves-ode } f \text{ from } t0) \ T \ X$
 $\langle \text{proof} \rangle$

lemma *usolves-odeI*:

assumes $(y \text{ solves-ode } f) \ T \ X \ t0 \in T \ \text{is-interval } T$
assumes $\text{usol}: \bigwedge z \ t. \ \{t0 \ _ \ t\} \subseteq T \implies (z \text{ solves-ode } f) \ \{t0 \ _ \ t\} \ X \implies z \ t0$
 $= y \ t0 \implies z \ t = y \ t$
shows $(y \text{ usolves-ode } f \text{ from } t0) \ T \ X$
 $\langle \text{proof} \rangle$

lemma *is-interval-singleton[intro,simp]*: $\text{is-interval } \{t0\}$

$\langle \text{proof} \rangle$

lemma *usolves-ode-singleton*: $x \ t0 \in X \implies (x \text{ usolves-ode } f \text{ from } t0) \ \{t0\} \ X$

$\langle \text{proof} \rangle$

lemma *usolves-ode-congI*:

assumes $x: (x \text{ usolves-ode } f \text{ from } t0) \ T \ X$
assumes $\bigwedge t. \ t \in T \implies x \ t = y \ t$
assumes $\bigwedge t \ y. \ t \in T \implies y \in X \implies f \ t \ y = g \ t \ y$ — TODO: weaken this assumption?!

assumes $t0 = s0$
assumes $T = S$
assumes $X = Y$
shows $(y \text{ solves-ode } g \text{ from } s0) S Y$
 $\langle \text{proof} \rangle$

lemma *usolves-ode-cong*[cong]:

assumes $\bigwedge t. t \in T \implies x t = y t$
assumes $\bigwedge t y. t \in T \implies y \in X \implies f t y = g t y$ — TODO: weaken this assumption?!
assumes $t0 = s0$
assumes $T = S$
assumes $X = Y$
shows $(x \text{ solves-ode } f \text{ from } t0) T X \longleftrightarrow (y \text{ solves-ode } g \text{ from } s0) S Y$
 $\langle \text{proof} \rangle$

lemma *shift-autonomous-unique-solution*:

assumes *usol*: $(x \text{ solves-ode } f \text{ from } t0) T X$
assumes *auto*: $\bigwedge s t x. x \in X \implies f s x = f t x$
shows $((\lambda t. x (t + t0 - t1)) \text{ solves-ode } f \text{ from } t1) ((+) (t1 - t0) ' T) X$
 $\langle \text{proof} \rangle$

lemma *three-intervals-lemma*:

fixes $a b c :: \text{real}$
assumes $a: a \in A - B$
and $b: b \in B - A$
and $c: c \in A \cap B$
and *iA*: *is-interval* A **and** *iB*: *is-interval* B
and *aI*: $a \in I$
and *bI*: $b \in I$
and *iI*: *is-interval* I
shows $c \in I$
 $\langle \text{proof} \rangle$

lemma *connection-usolves-ode*:

assumes $x: (x \text{ solves-ode } f \text{ from } tx) T X$
assumes $y: \bigwedge t. t \in \text{closure } S \cap \text{closure } T \implies (y \text{ solves-ode } g \text{ from } t) S X$
assumes *conn-T*: $\text{closure } S \cap \text{closure } T \subseteq T$
assumes *conn-S*: $\text{closure } S \cap \text{closure } T \subseteq S$
assumes *conn-t*: $t \in \text{closure } S \cap \text{closure } T$
assumes *conn-x*: $\bigwedge t. t \in \text{closure } S \implies t \in \text{closure } T \implies x t = y t$
assumes *conn-f*: $\bigwedge t x. t \in \text{closure } S \implies t \in \text{closure } T \implies x \in X \implies f t x = g t x$
shows $((\lambda t. \text{if } t \in T \text{ then } x t \text{ else } y t) \text{ solves-ode } (\lambda t. \text{if } t \in T \text{ then } f t \text{ else } g t) \text{ from } tx) (T \cup S) X$
 $\langle \text{proof} \rangle$

lemma *usolves-ode-union-closed*:

assumes x : $(x \text{ solves-ode } f \text{ from } tx) T X$
assumes y : $\bigwedge t. t \in \text{closure } S \cap \text{closure } T \implies (x \text{ solves-ode } f \text{ from } t) S X$
assumes $\text{conn-}T$: $\text{closure } S \cap \text{closure } T \subseteq T$
assumes $\text{conn-}S$: $\text{closure } S \cap \text{closure } T \subseteq S$
assumes $\text{conn-}t$: $t \in \text{closure } S \cap \text{closure } T$
shows $(x \text{ solves-ode } f \text{ from } tx) (T \cup S) X$
 $\langle \text{proof} \rangle$

lemma *usolves-ode-solves-odeI*:
assumes $(x \text{ solves-ode } f \text{ from } tx) T X$
assumes $(y \text{ solves-ode } f) T X y tx = x tx$
shows $(y \text{ solves-ode } f \text{ from } tx) T X$
 $\langle \text{proof} \rangle$

lemma *usolves-ode-subset-range*:
assumes x : $(x \text{ solves-ode } f \text{ from } t0) T X$
assumes r : $x \text{ ' } T \subseteq Y \text{ and } Y \subseteq X$
shows $(x \text{ solves-ode } f \text{ from } t0) T Y$
 $\langle \text{proof} \rangle$

2.3 ivp on interval

context
fixes $t0 t1::\text{real}$ **and** T
defines $T \equiv \text{closed-segment } t0 t1$
begin

lemma *is-solution-ext-cont*:
 $\text{continuous-on } T x \implies (\text{ext-cont } x (\text{min } t0 t1) (\text{max } t0 t1) \text{ solves-ode } f) T X =$
 $(x \text{ solves-ode } f) T X$
 $\langle \text{proof} \rangle$

lemma *solution-fixed-point*:
fixes $x::\text{real} \Rightarrow 'a::\text{banach}$
assumes x : $(x \text{ solves-ode } f) T X$ **and** t : $t \in T$
shows $x t0 + \text{ivl-integral } t0 t (\lambda t. f t (x t)) = x t$
 $\langle \text{proof} \rangle$

lemma *solution-fixed-point-left*:
fixes $x::\text{real} \Rightarrow 'a::\text{banach}$
assumes x : $(x \text{ solves-ode } f) T X$ **and** t : $t \in T$
shows $x t1 - \text{ivl-integral } t t1 (\lambda t. f t (x t)) = x t$
 $\langle \text{proof} \rangle$

lemma *solution-fixed-pointI*:
fixes $x::\text{real} \Rightarrow 'a::\text{banach}$
assumes cont-f : $\text{continuous-on } (T \times X) (\lambda(t, x). f t x)$
assumes cont-x : $\text{continuous-on } T x$
assumes defined : $\bigwedge t. t \in T \implies x t \in X$

assumes $fp: \bigwedge t. t \in T \implies x\ t = x\ t0 + ivl\text{-integral}\ t0\ t\ (\lambda t. f\ t\ (x\ t))$
shows $(x\ \text{solves-ode}\ f)\ T\ X$
 $\langle proof \rangle$
end

lemma *solves-ode-half-open-segment-continuation*:
fixes $f::real \Rightarrow 'a \Rightarrow 'a::banach$
assumes $ode: (x\ \text{solves-ode}\ f)\ \{t0\ \text{--<}\ t1\}\ X$
assumes $continuous: continuous\text{-on}\ (\{t0\ \text{--}\ t1\} \times X)\ (\lambda(t, x). f\ t\ x)$
assumes $compact\ X$
assumes $t0 \neq t1$
obtains $l\ \text{where}$
 $(x \longrightarrow l)\ \text{(at}\ t1\ \text{within}\ \{t0\ \text{--<}\ t1\})$
 $((\lambda t. \text{if}\ t = t1\ \text{then}\ l\ \text{else}\ x\ t)\ \text{solves-ode}\ f)\ \{t0\ \text{--}\ t1\}\ X$
 $\langle proof \rangle$

2.4 Picard-Lindelof on set of functions into closed set

locale *continuous-rhs* = **fixes** $T\ X\ f$
assumes $continuous: continuous\text{-on}\ (T \times X)\ (\lambda(t, x). f\ t\ x)$
begin

lemma *continuous-rhs-comp*[*continuous-intros*]:
assumes [*continuous-intros*]: $continuous\text{-on}\ S\ g$
assumes [*continuous-intros*]: $continuous\text{-on}\ S\ h$
assumes $g\ 'S \subseteq T$
assumes $h\ 'S \subseteq X$
shows $continuous\text{-on}\ S\ (\lambda x. f\ (g\ x)\ (h\ x))$
 $\langle proof \rangle$

end

locale *global-lipschitz* =
fixes $T\ X\ f$ **and** $L::real$
assumes $lipschitz: \bigwedge t. t \in T \implies L\text{-lipschitz-on}\ X\ (\lambda x. f\ t\ x)$

locale *closed-domain* =
fixes X **assumes** $closed: closed\ X$

locale *interval* = **fixes** $T::real\ set$
assumes $interval: is\text{-interval}\ T$
begin

lemma *closed-segment-subset-domain*: $t0 \in T \implies t \in T \implies closed\text{-segment}\ t0\ t \subseteq T$
 $\langle proof \rangle$

lemma *closed-segment-subset-domainI*: $t0 \in T \implies t \in T \implies s \in closed\text{-segment}$

$t0\ t \implies s \in T$
 ⟨proof⟩

lemma *convex*[*intro, simp*]: *convex* T
and *connected*[*intro, simp*]: *connected* T
 ⟨proof⟩

end

locale *nonempty-set* = **fixes** T **assumes** *nonempty-set*: $T \neq \{\}$

locale *compact-interval* = *interval* + *nonempty-set* T +
assumes *compact-time*: *compact* T
begin

definition *tmin* = *Inf* T
definition *tmax* = *Sup* T

lemma
shows *tmin*: $t \in T \implies tmin \leq t\ tmin \in T$
and *tmax*: $t \in T \implies t \leq tmax\ tmax \in T$
 ⟨proof⟩

lemma *tmin-le-tmax*[*intro, simp*]: $tmin \leq tmax$
 ⟨proof⟩

lemma *T-def*: $T = \{tmin .. tmax\}$
 ⟨proof⟩

lemma *mem-T-I*[*intro, simp*]: $tmin \leq t \implies t \leq tmax \implies t \in T$
 ⟨proof⟩

end

locale *self-mapping* = *interval* T **for** T +
fixes *t0*::*real* **and** *x0* $f\ X$
assumes *iv-defined*: $t0 \in T\ x0 \in X$
assumes *self-mapping*:
 $\bigwedge x\ t.\ t \in T \implies x\ t0 = x0 \implies x \in \text{closed-segment } t0\ t \rightarrow X \implies$
continuous-on (*closed-segment* $t0\ t$) $x \implies x\ t0 + \text{ivl-integral } t0\ t (\lambda t.\ f\ t\ (x$
 $t)) \in X$
begin

sublocale *nonempty-set* T ⟨proof⟩

lemma *closed-segment-iv-subset-domain*: $t \in T \implies \text{closed-segment } t0\ t \subseteq T$
 ⟨proof⟩

end

locale *unique-on-closed* =
compact-interval T +
self-mapping T $t0$ $x0$ f X +
continuous-rhs T X f +
closed-domain X +
global-lipschitz T X f L **for** $t0::real$ **and** T **and** $x0::'a::banach$ **and** f X L
begin

lemma *T-split*: $T = \{tmin .. t0\} \cup \{t0 .. tmax\}$
<proof>

lemma *L-nonneg*: $0 \leq L$
<proof>

Picard Iteration

definition *P-inner* **where** *P-inner* x $t = x0 + iwl$ -integral $t0$ t $(\lambda t. f$ t $(x$ $t))$

definition *P::(real \Rightarrow_C 'a) \Rightarrow (real \Rightarrow_C 'a)*
where *P* $x = (SOME$ $g::real \Rightarrow_C$ $'a.$
 $(\forall t \in T. g$ $t = P$ -inner x $t) \wedge$
 $(\forall t \leq tmin. g$ $t = P$ -inner x $tmin) \wedge$
 $(\forall t \geq tmax. g$ $t = P$ -inner x $tmax))$

lemma *cont-P-inner-iwl*:
 $x \in T \rightarrow_C X \implies$ *continuous-on* $\{tmin..tmax\}$ $(P$ -inner $(apply$ -bcontfun $x))$
<proof>

lemma *P-inner-t0[simp]*: *P-inner* g $t0 = x0$
<proof>

lemma *t0-cs-tmin-tmax*: $t0 \in \{tmin--tmax\}$ **and** *cs-tmin-tmax-subset*: $\{tmin--tmax\} \subseteq T$
<proof>

lemma
P-eqs:
assumes $x \in T \rightarrow_C X$
shows *P-eq-P-inner*: $t \in T \implies P$ x $t = P$ -inner x t
and *P-le-tmin*: $t \leq tmin \implies P$ x $t = P$ -inner x $tmin$
and *P-ge-tmax*: $t \geq tmax \implies P$ x $t = P$ -inner x $tmax$
<proof>

lemma *P-if-eq*:
 $x \in T \rightarrow_C X \implies$
 P x $t = (if$ $tmin \leq t \wedge t \leq tmax$ **then** *P-inner* x t **else if** $t \geq tmax$ **then** *P-inner* x $tmax$ **else** *P-inner* x $tmin)$
<proof>

lemma *dist-P-le*:

assumes $y: y \in T \rightarrow_C X$ **and** $z: z \in T \rightarrow_C X$

assumes $le: \bigwedge t. tmin \leq t \implies t \leq tmax \implies dist (P\text{-inner } y \ t) (P\text{-inner } z \ t) \leq R$

assumes $0 \leq R$

shows $dist (P \ y \ t) (P \ z \ t) \leq R$

<proof>

lemma *P-def'*:

assumes $t \in T$

assumes $fixed\text{-point} \in T \rightarrow_C X$

shows $(P \ fixed\text{-point}) \ t = x0 + ivl\text{-integral } t0 \ t (\lambda x. f \ x \ (fixed\text{-point} \ x))$

<proof>

definition $iter\text{-space} = PiC \ T \ ((\lambda \cdot. X)(t0 := \{x0\}))$

lemma *iter-spaceI*:

assumes $g \in T \rightarrow_C X$ $g \ t0 = x0$

shows $g \in iter\text{-space}$

<proof>

lemma *iter-spaceD*:

assumes $g \in iter\text{-space}$

shows $g \in T \rightarrow_C X$ $apply\text{-bcontfun } g \ t0 = x0$

<proof>

lemma *const-in-iter-space*: $const\text{-bcontfun } x0 \in iter\text{-space}$

<proof>

lemma *closed-iter-space*: $closed \ iter\text{-space}$

<proof>

lemma *iter-space-notempty*: $iter\text{-space} \neq \{\}$

<proof>

lemma *clamp-in-eq[simp]*: **fixes** $a \ x \ b :: real$ **shows** $a \leq x \implies x \leq b \implies clamp \ a \ b \ x = x$

<proof>

lemma *P-self-mapping*:

assumes $in\text{-space}: g \in iter\text{-space}$

shows $P \ g \in iter\text{-space}$

<proof>

lemma *continuous-on-T*: $continuous\text{-on } \{tmin .. tmax\} \ g \implies continuous\text{-on } T \ g$

<proof>

lemma *T-closed-segment-subsetI[intro, simp]*: $t \in \{tmin .. tmax\} \implies t \in T$

and $T\text{-subsetI}[intro, simp]$: $tmin \leq t \implies t \leq tmax \implies t \in T$

<proof>

lemma *t0-mem-closed-segment*[*intro, simp*]: $t0 \in \{tmin \dots tmax\}$
<proof>

lemma *tmin-le-t0*[*intro, simp*]: $tmin \leq t0$
and *tmax-ge-t0*[*intro, simp*]: $tmax \geq t0$
<proof>

lemma *apply-bcontfun-solution-fixed-point*:
assumes *ode*: (*apply-bcontfun x solves-ode f*) $T X$
assumes *iv*: $x\ t0 = x0$
assumes *t*: $t \in T$
shows $P\ x\ t = x\ t$
<proof>

lemma
solution-in-iter-space:
assumes *ode*: (*apply-bcontfun z solves-ode f*) $T X$
assumes *iv*: $z\ t0 = x0$
shows $z \in \text{iter-space}$ (**is** $?z \in -$)
<proof>

end

locale *unique-on-bounded-closed* = *unique-on-closed* +
assumes *lipschitz-bound*: $\bigwedge s\ t. s \in T \implies t \in T \implies \text{abs } (s - t) * L < 1$
begin

lemma *lipschitz-bound-maxmin*: $(tmax - tmin) * L < 1$
<proof>

lemma *lipschitz-P*:
shows $((tmax - tmin) * L)$ -*lipschitz-on iter-space P*
<proof>

lemma *fixed-point-unique*: $\exists! x \in \text{iter-space}. P\ x = x$
<proof>

definition *fixed-point where*
fixed-point = (*THE* $x. x \in \text{iter-space} \wedge P\ x = x$)

lemma *fixed-point'*:
fixed-point $\in \text{iter-space} \wedge P\ \text{fixed-point} = \text{fixed-point}$
<proof>

lemma *fixed-point*:
fixed-point $\in \text{iter-space}$ $P\ \text{fixed-point} = \text{fixed-point}$

<proof>

lemma *fixed-point-equality'*: $x \in \text{iter-space} \wedge P x = x \implies \text{fixed-point} = x$
<proof>

lemma *fixed-point-equality*: $x \in \text{iter-space} \implies P x = x \implies \text{fixed-point} = x$
<proof>

lemma *fixed-point-iv*: $\text{fixed-point } t0 = x0$
and *fixed-point-domain*: $x \in T \implies \text{fixed-point } x \in X$
<proof>

lemma *fixed-point-has-vderiv-on*: (*fixed-point has-vderiv-on* ($\lambda t. f t (\text{fixed-point } t)$))
 T
<proof>

lemma *fixed-point-solution*:
shows (*fixed-point solves-ode* f) $T X$
<proof>

2.4.1 Unique solution

lemma *solves-ode-equals-fixed-point*:
assumes *ode*: ($x \text{ solves-ode } f$) $T X$
assumes *iv*: $x t0 = x0$
assumes *t*: $t \in T$
shows $x t = \text{fixed-point } t$
<proof>

lemma *solves-ode-on-closed-segment-equals-fixed-point*:
assumes *ode*: ($x \text{ solves-ode } f$) $\{t0 \text{ -- } t1\} X$
assumes *iv*: $x t0 = x0$
assumes *subset*: $\{t0 \text{ -- } t1\} \subseteq T$
assumes *t-mem*: $t \in \{t0 \text{ -- } t1\}$
shows $x t = \text{fixed-point } t$
<proof>

lemma *unique-solution*:
assumes *ivp1*: ($x \text{ solves-ode } f$) $T X x t0 = x0$
assumes *ivp2*: ($y \text{ solves-ode } f$) $T X y t0 = x0$
assumes *t* $\in T$
shows $x t = y t$
<proof>

lemma *fixed-point-usolves-ode*: (*fixed-point usolves-ode* f from $t0$) $T X$
<proof>

end

lemma *closed-segment-Un*:
fixes $a\ b\ c::\text{real}$
assumes $b \in \text{closed-segment } a\ c$
shows $\text{closed-segment } a\ b \cup \text{closed-segment } b\ c = \text{closed-segment } a\ c$
 $\langle \text{proof} \rangle$

lemma *closed-segment-closed-segment-subset*:
fixes $s::\text{real}$ **and** $i::\text{nat}$
assumes $s \in \text{closed-segment } a\ b$
assumes $a \in \text{closed-segment } c\ d$ $b \in \text{closed-segment } c\ d$
shows $s \in \text{closed-segment } c\ d$
 $\langle \text{proof} \rangle$

context *unique-on-closed* **begin**

context— solution until $t1$
fixes $t1::\text{real}$
assumes $\text{mem-}t1: t1 \in T$
begin

lemma *subdivide-count-ex*: $\exists n. L * \text{abs } (t1 - t0) / (\text{Suc } n) < 1$
 $\langle \text{proof} \rangle$

definition *subdivide-count* = $(\text{SOME } n. L * \text{abs } (t1 - t0) / \text{Suc } n < 1)$

lemma *subdivide-count*: $L * \text{abs } (t1 - t0) / \text{Suc } \text{subdivide-count} < 1$
 $\langle \text{proof} \rangle$

lemma *subdivide-lipschitz*:
assumes $|s - t| \leq \text{abs } (t1 - t0) / \text{Suc } \text{subdivide-count}$
shows $|s - t| * L < 1$
 $\langle \text{proof} \rangle$

lemma *subdivide-lipschitz-lemma*:
assumes $st: s \in \{a \text{ --- } b\}$ $t \in \{a \text{ --- } b\}$
assumes $\text{abs } (b - a) \leq \text{abs } (t1 - t0) / \text{Suc } \text{subdivide-count}$
shows $|s - t| * L < 1$
 $\langle \text{proof} \rangle$

definition *step* = $(t1 - t0) / \text{Suc } \text{subdivide-count}$

lemma *last-step*: $t0 + \text{real } (\text{Suc } \text{subdivide-count}) * \text{step} = t1$
 $\langle \text{proof} \rangle$

lemma *step-in-segment*:
assumes $0 \leq i$ $i \leq \text{real } (\text{Suc } \text{subdivide-count})$
shows $t0 + i * \text{step} \in \text{closed-segment } t0\ t1$
 $\langle \text{proof} \rangle$

lemma *subset-T1*:

fixes $s::\text{real}$ **and** $i::\text{nat}$

assumes $s \in \text{closed-segment } t0 \ (t0 + i * \text{step})$

assumes $i \leq \text{Suc } \text{subdivide-count}$

shows $s \in \{t0 \ \text{--} \ t1\}$

<proof>

lemma *subset-T*: $\{t0 \ \text{--} \ t1\} \subseteq T$ **and** *subset-TI*: $s \in \{t0 \ \text{--} \ t1\} \implies s \in T$

<proof>

primrec *psolution*:: $\text{nat} \Rightarrow \text{real} \Rightarrow 'a$ **where**

psolution 0 $t = x0$

| *psolution* (Suc i) $t = \text{unique-on-bounded-closed.fixed-point}$

$(t0 + \text{real } i * \text{step}) \ \{t0 + \text{real } i * \text{step} \ \text{--} \ t0 + \text{real } (\text{Suc } i) * \text{step}\}$

$(\text{psolution } i \ (t0 + \text{real } i * \text{step})) \ f \ X \ t$

definition *psolutions* $t = \text{psolution } (\text{LEAST } i. \ t \in \text{closed-segment } (t0 + \text{real } (i - 1) * \text{step}) \ (t0 + \text{real } i * \text{step})) \ t$

lemma *psolutions-usolves-until-step*:

assumes *i-le*: $i \leq \text{Suc } \text{subdivide-count}$

shows $(\text{psolutions } \text{usolves-ode } f \ \text{from } t0) \ (\text{closed-segment } t0 \ (t0 + \text{real } i * \text{step}))$

X

<proof>

lemma *psolutions-usolves-ode*: $(\text{psolutions } \text{usolves-ode } f \ \text{from } t0) \ \{t0 \ \text{--} \ t1\} \ X$

<proof>

end

definition *solution* $t = (\text{if } t \leq t0 \ \text{then } \text{psolutions } tmin \ t \ \text{else } \text{psolutions } tmax \ t)$

lemma *solution-eq-left*: $tmin \leq t \implies t \leq t0 \implies \text{solution } t = \text{psolutions } tmin \ t$

<proof>

lemma *solution-eq-right*: $t0 \leq t \implies t \leq tmax \implies \text{solution } t = \text{psolutions } tmax \ t$

<proof>

lemma *solution-usolves-ode*: $(\text{solution } \text{usolves-ode } f \ \text{from } t0) \ T \ X$

<proof>

lemma *solution-solves-ode*: $(\text{solution } \text{solves-ode } f) \ T \ X$

<proof>

lemma *solution-iv[simp]*: $\text{solution } t0 = x0$

<proof>

end

2.5 Picard-Lindelof for $X = UNIV$

```

locale unique-on-strip =
  compact-interval  $T$  +
  continuous-rhs  $T$   $UNIV$   $f$  +
  global-lipschitz  $T$   $UNIV$   $f$   $L$ 
  for  $t0$  and  $T$  and  $f::real \Rightarrow 'a \Rightarrow 'a::banach$  and  $L$  +
  assumes iv-time:  $t0 \in T$ 
begin

sublocale unique-on-closed  $t0$   $T$   $x0$   $f$   $UNIV$   $L$  for  $x0$ 
   $\langle proof \rangle$ 

end

```

2.6 Picard-Lindelof on cylindric domain

```

locale solution-in-cylinder =
  continuous-rhs  $T$  cball  $x0$   $b$   $f$  +
  compact-interval  $T$ 
  for  $t0$   $T$   $x0$   $b$  and  $f::real \Rightarrow 'a \Rightarrow 'a::banach$  +
  fixes  $X$   $B$ 
  defines  $X \equiv cball$   $x0$   $b$ 
  assumes initial-time-in:  $t0 \in T$ 
  assumes norm-f:  $\bigwedge x t. t \in T \Longrightarrow x \in X \Longrightarrow norm (f t x) \leq B$ 
  assumes b-pos:  $b \geq 0$ 
  assumes e-bounded:  $\bigwedge t. t \in T \Longrightarrow dist t t0 \leq b / B$ 
begin

lemmas cylinder =  $X$ -def

lemma B-nonneg:  $B \geq 0$ 
   $\langle proof \rangle$ 

lemma in-bounds-derivativeI:
  assumes  $t \in T$ 
  assumes init:  $x t0 = x0$ 
  assumes cont: continuous-on (closed-segment  $t0$   $t$ )  $x$ 
  assumes solves: ( $x$  has-vderiv-on ( $\lambda s. f s (y s)$ )) (open-segment  $t0$   $t$ )
  assumes y-bounded:  $\bigwedge \xi. \xi \in \text{closed-segment } t0 t \Longrightarrow x \xi \in X \Longrightarrow y \xi \in X$ 
  shows  $x t \in cball$   $x0$  ( $B * abs (t - t0)$ )
   $\langle proof \rangle$ 

lemma in-bounds-derivative-globalI:
  assumes  $t \in T$ 
  assumes init:  $x t0 = x0$ 
  assumes cont: continuous-on (closed-segment  $t0$   $t$ )  $x$ 
  assumes solves: ( $x$  has-vderiv-on ( $\lambda s. f s (y s)$ )) (open-segment  $t0$   $t$ )
  assumes y-bounded:  $\bigwedge \xi. \xi \in \text{closed-segment } t0 t \Longrightarrow x \xi \in X \Longrightarrow y \xi \in X$ 

```

shows $x t \in X$
<proof>

lemma *integral-in-bounds*:

assumes $t \in T$ $x t0 = x0$ $x \in \{t0 \text{ -- } t\} \rightarrow X$
assumes *cont*[*continuous-intros*]: *continuous-on* ($\{t0 \text{ -- } t\}$) x
shows $x t0 + ivl\text{-integral } t0 t (\lambda t. f t (x t)) \in X$ (**is** - + ?*ix* $t \in X$)
<proof>

lemma *solves-in-cone*:

assumes $t \in T$
assumes *init*: $x t0 = x0$
assumes *cont*: *continuous-on* (*closed-segment* $t0 t$) x
assumes *solves*: (x *has-vderiv-on* ($\lambda s. f s (x s)$)) (*open-segment* $t0 t$)
shows $x t \in cball\ x0 (B * abs (t - t0))$
<proof>

lemma *is-solution-in-cone*:

assumes $t \in T$
assumes *sol*: (x *solves-ode* f) (*closed-segment* $t0 t$) Y **and** *iv*: $x t0 = x0$
shows $x t \in cball\ x0 (B * abs (t - t0))$
<proof>

lemma *cone-subset-domain*:

assumes $t \in T$
shows $cball\ x0 (B * |t - t0|) \subseteq X$
<proof>

lemma *is-solution-in-domain*:

assumes $t \in T$
assumes *sol*: (x *solves-ode* f) (*closed-segment* $t0 t$) Y **and** *iv*: $x t0 = x0$
shows $x t \in X$
<proof>

lemma *solves-ode-on-subset-domain*:

assumes *sol*: (x *solves-ode* f) $S Y$ **and** *iv*: $x t0 = x0$
and *ivl*: $t0 \in S$ *is-interval* $S S \subseteq T$
shows (x *solves-ode* f) $S X$
<proof>

lemma *usolves-ode-on-subset*:

assumes x : (x *usolves-ode* f *from* $t0$) $T X$ **and** *iv*: $x t0 = x0$
assumes $t0 \in S$ *is-interval* $S S \subseteq T X \subseteq Y$
shows (x *usolves-ode* f *from* $t0$) $S Y$
<proof>

lemma *usolves-ode-on-superset-domain*:

assumes (x *usolves-ode* f *from* $t0$) $T X$ **and** *iv*: $x t0 = x0$
assumes $X \subseteq Y$

```

shows (x solves-ode f from t0) T Y
  ⟨proof⟩

end

locale unique-on-cylinder =
  solution-in-cylinder t0 T x0 b f X B +
  global-lipschitz T X f L
for t0 T x0 b X f B L
begin

sublocale unique-on-closed t0 T x0 f X L
  ⟨proof⟩

end

locale derivative-on-prod =
  fixes T X and f::real ⇒ 'a::banach ⇒ 'a and f': real × 'a ⇒ (real × 'a) ⇒ 'a
  assumes f': ∧tx. tx ∈ T × X ⇒ ((λ(t, x). f t x) has-derivative (f' tx)) (at tx
  within (T × X))
begin

lemma f'-comp[derivative-intros]:
  (g has-derivative g') (at s within S) ⇒ (h has-derivative h') (at s within S) ⇒
  s ∈ S ⇒ (∧x. x ∈ S ⇒ g x ∈ T) ⇒ (∧x. x ∈ S ⇒ h x ∈ X) ⇒
  ((λx. f (g x) (h x)) has-derivative (λy. f' (g s, h s) (g' y, h' y))) (at s within S)
  ⟨proof⟩

lemma derivative-on-prod-subset:
  assumes X' ⊆ X
  shows derivative-on-prod T X' f f'
  ⟨proof⟩

end

end
theory Picard-Lindeloeuf-Qualitative
imports Initial-Value-Problem
begin

```

2.7 Picard-Lindeloeuf On Open Domains

2.7.1 Local Solution with local Lipschitz

```

lemma cball-eq-closed-segment-real:
  fixes x e::real
  shows cball x e = (if e ≥ 0 then {x - e .. x + e} else {})
  ⟨proof⟩

```

lemma *cube-in-cball*:

fixes $x\ y :: 'a::\text{euclidean-space}$

assumes $r > 0$

assumes $\bigwedge i. i \in \text{Basis} \implies \text{dist } (x \cdot i) (y \cdot i) \leq r / \text{sqrt}(\text{DIM}('a))$

shows $y \in \text{cball } x\ r$

$\langle \text{proof} \rangle$

lemma *cbox-in-cball'*:

fixes $x :: 'a::\text{euclidean-space}$

assumes $0 < r$

shows $\exists b > 0. b \leq r \wedge (\exists B. B = (\sum_{i \in \text{Basis}} b *_R i) \wedge (\forall y \in \text{cbox } (x - B) (x + B). y \in \text{cball } x\ r))$

$\langle \text{proof} \rangle$

lemma *Pair1-in-Basis*: $i \in \text{Basis} \implies (i, 0) \in \text{Basis}$

and *Pair2-in-Basis*: $i \in \text{Basis} \implies (0, i) \in \text{Basis}$

$\langle \text{proof} \rangle$

lemma *le-real-sqrt-sumsq' [simp]*: $y \leq \text{sqrt } (x * x + y * y)$

$\langle \text{proof} \rangle$

lemma *cball-Pair-split-subset*: $\text{cball } (a, b)\ c \subseteq \text{cball } a\ c \times \text{cball } b\ c$

$\langle \text{proof} \rangle$

lemma *cball-times-subset*: $\text{cball } a\ (c/2) \times \text{cball } b\ (c/2) \subseteq \text{cball } (a, b)\ c$

$\langle \text{proof} \rangle$

lemma *eventually-bound-pairE*:

assumes $\text{isCont } f\ (t0, x0)$

obtains B **where**

$B \geq 1$

$\text{eventually } (\lambda e. \forall x \in \text{cball } t0\ e \times \text{cball } x0\ e. \text{norm } (f\ x) \leq B) \text{ (at-right } 0)$

$\langle \text{proof} \rangle$

lemma

eventually-in-cballs:

assumes $d > 0\ c > 0$

shows $\text{eventually } (\lambda e. \text{cball } t0\ (c * e) \times (\text{cball } x0\ e) \subseteq \text{cball } (t0, x0)\ d) \text{ (at-right } 0)$

$\langle \text{proof} \rangle$

lemma *cball-eq-sing'*:

fixes $x :: 'a::\{\text{metric-space}, \text{perfect-space}\}$

shows $\text{cball } x\ e = \{y\} \iff e = 0 \wedge x = y$

$\langle \text{proof} \rangle$

locale *ll-on-open = interval T for T +*

fixes $f :: \text{real} \Rightarrow 'a::\{\text{banach}, \text{heine-borel}\} \Rightarrow 'a$ **and** X

assumes *local-lipschitz*: *local-lipschitz* $T\ X\ f$

assumes *cont*: $\bigwedge x. x \in X \implies \text{continuous-on } T (\lambda t. f t x)$
assumes *open-domain*[*intro!*, *simp*]: *open* T *open* X
begin

all flows on closed segments

definition *csols* **where**
csols $t0\ x0 = \{(x, t1). \{t0--t1\} \subseteq T \wedge x\ t0 = x0 \wedge (x \text{ solves-ode } f) \{t0--t1\} X\}$

the maximal existence interval

definition *existence-ivl* $t0\ x0 = (\bigcup (x, t1) \in \text{csols } t0\ x0 . \{t0--t1\})$

witness flow

definition *csol* $t0\ x0 = (\text{SOME } csol. \forall t \in \text{existence-ivl } t0\ x0. (csol\ t, t) \in \text{csols } t0\ x0)$

unique flow

definition *flow* **where** *flow* $t0\ x0 = (\lambda t. \text{if } t \in \text{existence-ivl } t0\ x0 \text{ then } csol\ t0\ x0\ t\ t \text{ else } 0)$

end

locale *ll-on-open-it* =
general?— TODO: why is this qualification necessary? It seems only because of *ll-on-open-it* $T\ f\ X$
ll-on-open + **fixes** $t0::\text{real}$
— if possible, all development should be done with $t0$ as explicit parameter for initial time: then it can be instantiated with 0 for autonomous ODEs

context *ll-on-open* **begin**

sublocale *ll-on-open-it* **where** $t0 = t0$ **for** $t0$ $\langle \text{proof} \rangle$

sublocale *continuous-rhs* $T\ X\ f$
 $\langle \text{proof} \rangle$

end

context *ll-on-open-it* **begin**

lemma *ll-on-open-rev*[*intro*, *simp*]: *ll-on-open* (*preflect* $t0$ ‘ T) ($\lambda t. - f$ (*preflect* $t0\ t$)) X
 $\langle \text{proof} \rangle$

lemma *eventually-lipschitz*:
assumes $t0 \in T\ x0 \in X\ c > 0$
obtains L **where**
eventually ($\lambda u. \forall t' \in \text{cball } t0\ (c * u) \cap T$).

L-lipschitz-on (*cball* $x0$ $u \cap X$) ($\lambda y. f t' y$) (*at-right* 0)
 ⟨*proof*⟩

lemmas *continuous-on-Times-f* = *continuous*

lemmas *continuous-on-f* = *continuous-rhs-comp*

lemma

lipschitz-on-compact:

assumes *compact* K $K \subseteq T$

assumes *compact* Y $Y \subseteq X$

obtains L **where** $\bigwedge t. t \in K \implies L\text{-lipschitz-on } Y (f t)$

⟨*proof*⟩

lemma *csols-empty-iff*: $csols\ t0\ x0 = \{\}$ $\longleftrightarrow t0 \notin T \vee x0 \notin X$

⟨*proof*⟩

lemma *csols-notempty*: $t0 \in T \implies x0 \in X \implies csols\ t0\ x0 \neq \{\}$

⟨*proof*⟩

lemma *existence-ivl-empty-iff[simp]*: $existence\text{-ivl}\ t0\ x0 = \{\}$ $\longleftrightarrow t0 \notin T \vee x0 \notin X$

⟨*proof*⟩

lemma *existence-ivl-empty1[simp]*: $t0 \notin T \implies existence\text{-ivl}\ t0\ x0 = \{\}$

and *existence-ivl-empty2[simp]*: $x0 \notin X \implies existence\text{-ivl}\ t0\ x0 = \{\}$

⟨*proof*⟩

lemma *flow-undefined*:

shows $t0 \notin T \implies flow\ t0\ x0 = (\lambda-. 0)$

$x0 \notin X \implies flow\ t0\ x0 = (\lambda-. 0)$

⟨*proof*⟩

lemma (*in ll-on-open*) *flow-eq-in-existence-ivlI*:

assumes $\bigwedge u. x0 \in X \implies u \in existence\text{-ivl}\ t0\ x0 \longleftrightarrow g\ u \in existence\text{-ivl}\ s0\ x0$

assumes $\bigwedge u. x0 \in X \implies u \in existence\text{-ivl}\ t0\ x0 \implies flow\ t0\ x0\ u = flow\ s0\ x0 (g\ u)$

shows $flow\ t0\ x0 = (\lambda t. flow\ s0\ x0 (g\ t))$

⟨*proof*⟩

2.7.2 Global maximal flow with local Lipschitz

lemma *local-unique-solution*:

assumes *iv-defined*: $t0 \in T$ $x0 \in X$

obtains $et\ ex\ B\ L$

where $et > 0$ $0 < ex$ $cball\ t0\ et \subseteq T$ $cball\ x0\ ex \subseteq X$

unique-on-cylinder $t0$ ($cball\ t0\ et$) $x0\ ex\ f\ B\ L$

⟨*proof*⟩

lemma *mem-existence-ivl-iv-defined*:

assumes $t \in \text{existence-ivl } t0 \ x0$

shows $t0 \in T \ x0 \in X$

<proof>

lemma *csol-mem-csols*:

assumes $t \in \text{existence-ivl } t0 \ x0$

shows $(\text{csol } t0 \ x0 \ t, t) \in \text{csols } t0 \ x0$

<proof>

lemma *csol*:

assumes $t \in \text{existence-ivl } t0 \ x0$

shows $t \in T \ \{t0 \ -- \ t\} \subseteq T \ \text{csol } t0 \ x0 \ t \ t0 = x0 \ (\text{csol } t0 \ x0 \ t \ \text{solves-ode } f)$
 $\{t0 \ -- \ t\} \ X$

<proof>

lemma *existence-ivl-initial-time-iff[simp]*: $t0 \in \text{existence-ivl } t0 \ x0 \longleftrightarrow t0 \in T \wedge x0 \in X$

<proof>

lemma *existence-ivl-initial-time*: $t0 \in T \implies x0 \in X \implies t0 \in \text{existence-ivl } t0 \ x0$

<proof>

lemmas *mem-existence-ivl-subset = csol(1)*

lemma *existence-ivl-subset*:

$\text{existence-ivl } t0 \ x0 \subseteq T$

<proof>

lemma *is-interval-existence-ivl[intro, simp]*: *is-interval* (*existence-ivl* $t0 \ x0$)

<proof>

lemma *connected-existence-ivl[intro, simp]*: *connected* (*existence-ivl* $t0 \ x0$)

<proof>

lemma *in-existence-between-zeroI*:

$t \in \text{existence-ivl } t0 \ x0 \implies s \in \{t0 \ -- \ t\} \implies s \in \text{existence-ivl } t0 \ x0$

<proof>

lemma *segment-subset-existence-ivl*:

assumes $s \in \text{existence-ivl } t0 \ x0 \ t \in \text{existence-ivl } t0 \ x0$

shows $\{s \ -- \ t\} \subseteq \text{existence-ivl } t0 \ x0$

<proof>

lemma *flow-initial-time-if*: $\text{flow } t0 \ x0 \ t0 = (\text{if } t0 \in T \wedge x0 \in X \text{ then } x0 \text{ else } 0)$

<proof>

lemma *flow-initial-time[simp]*: $t0 \in T \implies x0 \in X \implies \text{flow } t0 \ x0 \ t0 = x0$

<proof>

lemma *open-existence-ivl*[*intro, simp*]: *open* (*existence-ivl t0 x0*)
 ⟨*proof*⟩

lemma *csols-unique*:
assumes $(x, t1) \in csols\ t0\ x0$
assumes $(y, t2) \in csols\ t0\ x0$
shows $\forall t \in \{t0 \text{ -- } t1\} \cap \{t0 \text{ -- } t2\}. x\ t = y\ t$
 ⟨*proof*⟩

lemma *csol-unique*:
assumes $t1: t1 \in existence-ivl\ t0\ x0$
assumes $t2: t2 \in existence-ivl\ t0\ x0$
assumes $t: t \in \{t0 \text{ -- } t1\} \cap \{t0 \text{ -- } t2\}$
shows $csol\ t0\ x0\ t1\ t = csol\ t0\ x0\ t2\ t$
 ⟨*proof*⟩

lemma *flow-vderiv-on-left*:
 (*flow t0 x0 has-vderiv-on* ($\lambda x. f\ x\ (flow\ t0\ x0\ x)$)) (*existence-ivl t0 x0* $\cap \{..t0\}$)
 ⟨*proof*⟩

lemma *flow-vderiv-on-right*:
 (*flow t0 x0 has-vderiv-on* ($\lambda x. f\ x\ (flow\ t0\ x0\ x)$)) (*existence-ivl t0 x0* $\cap \{t0..\}$)
 ⟨*proof*⟩

lemma *flow-usolves-ode*:
assumes *iv-defined*: $t0 \in T\ x0 \in X$
shows (*flow t0 x0 usolves-ode f from t0*) (*existence-ivl t0 x0*) X
 ⟨*proof*⟩

lemma *flow-solves-ode*: $t0 \in T \implies x0 \in X \implies (flow\ t0\ x0\ solves-ode\ f)$ (*existence-ivl t0 x0*) X
 ⟨*proof*⟩

lemma *equals-flowI*:
assumes $t0 \in T'$
is-interval T'
 $T' \subseteq existence-ivl\ t0\ x0$
 (*z solves-ode f*) $T'\ X$
 $z\ t0 = flow\ t0\ x0\ t0\ t \in T'$
shows $z\ t = flow\ t0\ x0\ t$
 ⟨*proof*⟩

lemma *existence-ivl-maximal-segment*:
assumes (*x solves-ode f*) $\{t0 \text{ -- } t\}\ X\ x\ t0 = x0$
assumes $\{t0 \text{ -- } t\} \subseteq T$
shows $t \in existence-ivl\ t0\ x0$
 ⟨*proof*⟩

lemma *existence-ivl-maximal-interval*:

assumes $(x \text{ solves-ode } f) S X x t0 = x0$

assumes $t0 \in S \text{ is-interval } S S \subseteq T$

shows $S \subseteq \text{existence-ivl } t0 x0$

<proof>

lemma *maximal-existence-flow*:

assumes $\text{sol}: (x \text{ solves-ode } f) K X$ **and** $\text{iv}: x t0 = x0$

assumes $\text{is-interval } K$

assumes $t0 \in K$

assumes $K \subseteq T$

shows $K \subseteq \text{existence-ivl } t0 x0 \wedge t. t \in K \implies \text{flow } t0 x0 t = x t$

<proof>

lemma *maximal-existence-flowI*:

assumes $(x \text{ has-vderiv-on } (\lambda t. f t (x t))) K$

assumes $\wedge t. t \in K \implies x t \in X$

assumes $x t0 = x0$

assumes $K: \text{is-interval } K t0 \in K K \subseteq T$

shows $K \subseteq \text{existence-ivl } t0 x0 \wedge t. t \in K \implies \text{flow } t0 x0 t = x t$

<proof>

lemma *flow-in-domain*: $t \in \text{existence-ivl } t0 x0 \implies \text{flow } t0 x0 t \in X$

<proof>

lemma (*in ll-on-open*)

assumes $t \in \text{existence-ivl } s x$

assumes $x \in X$

assumes $\text{auto}: \wedge s t x. x \in X \implies f s x = f t x$

assumes $T = \text{UNIV}$

shows $\text{mem-existence-ivl-shift-autonomous1}: t - s \in \text{existence-ivl } 0 x$

and $\text{flow-shift-autonomous1}: \text{flow } s x t = \text{flow } 0 x (t - s)$

<proof>

lemma (*in ll-on-open*)

assumes $t - s \in \text{existence-ivl } 0 x$

assumes $x \in X$

assumes $\text{auto}: \wedge s t x. x \in X \implies f s x = f t x$

assumes $T = \text{UNIV}$

shows $\text{mem-existence-ivl-shift-autonomous2}: t \in \text{existence-ivl } s x$

and $\text{flow-shift-autonomous2}: \text{flow } s x t = \text{flow } 0 x (t - s)$

<proof>

lemma

flow-eq-rev:

assumes $t \in \text{existence-ivl } t0 x0$

shows $\text{preflect } t0 t \in \text{ll-on-open.existence-ivl } (\text{preflect } t0 ' T) (\lambda t. - f (\text{preflect } t0 t)) X t0 x0$

$\text{flow } t0 x0 t = \text{ll-on-open.flow } (\text{preflect } t0 ' T) (\lambda t. - f (\text{preflect } t0 t)) X t0 x0$

(*preflect t0 t*)
<*proof*>

lemma (in *ll-on-open*)

shows *rev-flow-eq*: $t \in ll\text{-on-open.exists-ivl } (preflect\ t0\ 'T) (\lambda t. - f (preflect\ t0\ t)) X\ t0\ x0 \implies$

$ll\text{-on-open.flow } (preflect\ t0\ 'T) (\lambda t. - f (preflect\ t0\ t)) X\ t0\ x0\ t = flow\ t0\ x0$
(*preflect t0 t*)

and *mem-rev-existence-ivl-eq*:

$t \in ll\text{-on-open.exists-ivl } (preflect\ t0\ 'T) (\lambda t. - f (preflect\ t0\ t)) X\ t0\ x0 \longleftrightarrow$
preflect t0 t \in *existence-ivl t0 x0*

<*proof*>

lemma

shows *rev-existence-ivl-eq*: $ll\text{-on-open.exists-ivl } (preflect\ t0\ 'T) (\lambda t. - f (preflect\ t0\ t)) X\ t0\ x0 =$
preflect t0 'existence-ivl t0 x0

and *existence-ivl-eq-rev*: $existence\text{-ivl } t0\ x0 = preflect\ t0\ 'll\text{-on-open.exists-ivl}$
(preflect t0 'T) (\lambda t. - f (preflect t0 t)) X t0 x0

<*proof*>

end

end

3 Bounded Linear Operator

theory *Bounded-Linear-Operator*

imports

HOL-Analysis.Analysis

begin

typedef (overloaded) *'a blinop* = *UNIV::('a, 'a) blinfun set*

<*proof*>

setup-lifting *type-definition-blinop*

lift-definition *blinop-apply::('a::real-normed-vector) blinop \Rightarrow 'a \Rightarrow 'a* **is** *blinfun-apply* <*proof*>

lift-definition *Blinop::('a::real-normed-vector \Rightarrow 'a) \Rightarrow 'a blinop* **is** *Blinfun* <*proof*>

no-notation *vec-nth* (**infixl** \$ 90)

notation *blinop-apply* (**infixl** \$ 999)

declare [[*coercion blinop-apply :: ('a::real-normed-vector) blinop \Rightarrow 'a \Rightarrow 'a*]]

instantiation *blinop :: (real-normed-vector) real-normed-vector*

begin

lift-definition *norm-blinop :: 'a blinop \Rightarrow real* **is** *norm* <*proof*>

lift-definition *minus-blinop* :: 'a blinop \Rightarrow 'a blinop \Rightarrow 'a blinop **is** *minus* \langle proof \rangle

lift-definition *dist-blinop* :: 'a blinop \Rightarrow 'a blinop \Rightarrow real **is** *dist* \langle proof \rangle

definition *uniformity-blinop* :: ('a blinop \times 'a blinop) filter **where**
uniformity-blinop = (INF e \in {0<..}. principal {(x, y). dist x y < e})

definition *open-blinop* :: 'a blinop set \Rightarrow bool **where**
open-blinop U = ($\forall x \in U. \forall_F (x', y)$ in *uniformity*. $x' = x \longrightarrow y \in U$)

lift-definition *uminus-blinop* :: 'a blinop \Rightarrow 'a blinop **is** *uminus* \langle proof \rangle

lift-definition *zero-blinop* :: 'a blinop **is** 0 \langle proof \rangle

lift-definition *plus-blinop* :: 'a blinop \Rightarrow 'a blinop \Rightarrow 'a blinop **is** *plus* \langle proof \rangle

lift-definition *scaleR-blinop*::real \Rightarrow 'a blinop \Rightarrow 'a blinop **is** *scaleR* \langle proof \rangle

lift-definition *sgn-blinop* :: 'a blinop \Rightarrow 'a blinop **is** *sgn* \langle proof \rangle

instance

\langle proof \rangle

end

lemma *bounded-bilinear-blinop-apply*: bounded-bilinear (\$) \langle proof \rangle

interpretation *blinop*: bounded-bilinear (\$) \langle proof \rangle

lemma *blinop-eqI*: ($\bigwedge i. x \$ i = y \$ i$) \Longrightarrow $x = y$ \langle proof \rangle

lemmas *bounded-linear-apply-blinop*[intro, simp] = *blinop.bounded-linear-left*

declare *blinop.tendsto*[*tendsto-intros*]

declare *blinop.FDERIV*[*derivative-intros*]

declare *blinop.continuous*[*continuous-intros*]

declare *blinop.continuous-on*[*continuous-intros*]

instance *blinop* :: (banach) banach \langle proof \rangle

instance *blinop* :: (euclidean-space) heine-borel \langle proof \rangle

instantiation *blinop*::({real-normed-vector, perfect-space}) real-normed-algebra-1
begin

lift-definition *one-blinop*::'a blinop is id-blinfun <proof>

lemma *blinop-apply-one-blinop[simp]*: 1 \$ x = x

<proof>

lift-definition *times-blinop* :: 'a blinop \Rightarrow 'a blinop \Rightarrow 'a blinop is blinfun-compose

<proof>

lemma *blinop-apply-times-blinop[simp]*: (f * g) \$ x = f \$ (g \$ x)

<proof>

instance

<proof>

end

lemmas *bounded-bilinear-bounded-uniform-limit-intros*[uniform-limit-intros] =

bounded-bilinear.bounded-uniform-limit[OF Bounded-Linear-Operator.bounded-bilinear-blinop-apply]

bounded-bilinear.bounded-uniform-limit[OF Bounded-Linear-Function.bounded-bilinear-blinfun-apply]

bounded-bilinear.bounded-uniform-limit[OF Bounded-Linear-Operator.blinop.flip]

bounded-bilinear.bounded-uniform-limit[OF Bounded-Linear-Function.blinfun.flip]

bounded-linear.uniform-limit[OF blinop.bounded-linear-right]

bounded-linear.uniform-limit[OF blinop.bounded-linear-left]

bounded-linear.uniform-limit[OF bounded-linear-apply-blinop]

no-notation

blinop-apply (infixl \$ 999)

notation *vec-nth* (infixl \$ 90)

end

4 Multivariate Taylor

theory *Multivariate-Taylor*

imports

HOL-Analysis.Analysis

../ODE-Auxiliarities

begin

no-notation *vec-nth* (infixl \$ 90)

notation *blinfun-apply* (infixl \$ 999)

lemma

fixes *f*::'a::real-normed-vector \Rightarrow 'b::banach

and *Df*::'a \Rightarrow nat \Rightarrow 'a \Rightarrow 'a \Rightarrow 'b

assumes *n* > 0

assumes *Df-Nil*: $\bigwedge a x. Df a 0 H H = f a$

assumes *Df-Cons*: $\bigwedge a i d. a \in \text{closed-segment } X (X + H) \Longrightarrow i < n \Longrightarrow$

$((\lambda a. Df a i H H) \text{ has-derivative } (Df a (Suc i) H)) \text{ (at } a \text{ within } G)$

assumes *cs*: *closed-segment* *X* (*X* + *H*) \subseteq *G*

defines *i* \equiv $\lambda x.$

$((1 - x) \wedge (n - 1) / \text{fact } (n - 1)) *_{\mathbb{R}} Df (X + x *_{\mathbb{R}} H) n H H$
shows *multivariate-Taylor-has-integral*:
 $(i \text{ has-integral } f (X + H) - (\sum_{i < n}. (1 / \text{fact } i) *_{\mathbb{R}} Df X i H H)) \{0..1\}$
and *multivariate-Taylor*:
 $f (X + H) = (\sum_{i < n}. (1 / \text{fact } i) *_{\mathbb{R}} Df X i H H) + \text{integral } \{0..1\} i$
and *multivariate-Taylor-integrable*:
 $i \text{ integrable-on } \{0..1\}$
 <proof>

4.1 Symmetric second derivative

lemma *symmetric-second-derivative-aux*:
assumes *first-fderiv[derivative-intros]*:
 $\bigwedge a. a \in G \implies (f \text{ has-derivative } (f' a)) \text{ (at } a \text{ within } G)$
assumes *second-fderiv[derivative-intros]*:
 $\bigwedge i. ((\lambda x. f' x i) \text{ has-derivative } (\lambda j. f'' j i)) \text{ (at } a \text{ within } G)$
assumes $i \neq j \ i \neq 0 \ j \neq 0$
assumes $a \in G$
assumes $\bigwedge s \ t. s \in \{0..1\} \implies t \in \{0..1\} \implies a + s *_{\mathbb{R}} i + t *_{\mathbb{R}} j \in G$
shows $f'' j i = f'' i j$
 <proof>

locale *second-derivative-within* =
fixes $f \ f' \ f'' \ a \ G$
assumes *first-fderiv[derivative-intros]*:
 $\bigwedge a. a \in G \implies (f \text{ has-derivative } \text{blinfun-apply } (f' a)) \text{ (at } a \text{ within } G)$
assumes *in-G*: $a \in G$
assumes *second-fderiv[derivative-intros]*:
 $(f' \text{ has-derivative } \text{blinfun-apply } f'') \text{ (at } a \text{ within } G)$
begin

lemma *symmetric-second-derivative-within*:
assumes $a \in G$
assumes $\bigwedge s \ t. s \in \{0..1\} \implies t \in \{0..1\} \implies a + s *_{\mathbb{R}} i + t *_{\mathbb{R}} j \in G$
shows $f'' i j = f'' j i$
 <proof>

end

locale *second-derivative* =
fixes $f :: 'a :: \text{real-normed-vector} \Rightarrow 'b :: \text{banach}$
and $f' :: 'a \Rightarrow 'a \Rightarrow_L 'b$
and $f'' :: 'a \Rightarrow_L 'a \Rightarrow_L 'b$
and $a :: 'a$
and $G :: 'a \text{ set}$
assumes *first-fderiv[derivative-intros]*:
 $\bigwedge a. a \in G \implies (f \text{ has-derivative } f' a) \text{ (at } a)$
assumes *in-G*: $a \in \text{interior } G$
assumes *second-fderiv[derivative-intros]*:

(*f*' has-derivative *f*'') (at *a*)
begin

lemma *symmetric-second-derivative*:

assumes *a* ∈ interior *G*

shows *f*'' *i j* = *f*'' *j i*

⟨*proof*⟩

end

lemma

uniform-explicit-remainder-Taylor-1:

fixes *f*::'*a*::{*banach,heine-borel,perfect-space*} ⇒ '*b*::*banach*

assumes *f*'[*derivative-intros*]: $\bigwedge x. x \in G \implies (f \text{ has-derivative } \text{blinfun-apply } (f' x)) \text{ (at } x)$

assumes *f*'-cont: $\bigwedge x. x \in G \implies \text{isCont } f' x$

assumes open *G*

assumes *J* ≠ {} compact *J* *J* ⊆ *G*

assumes *e* > 0

obtains *d* *R*

where *d* > 0

$\bigwedge x z. f z = f x + f' x (z - x) + R x z$

$\bigwedge x y. x \in J \implies y \in J \implies \text{dist } x y < d \implies \text{norm } (R x y) \leq e * \text{dist } x y$

continuous-on (*G* × *G*) ($\lambda(a, b). R a b$)

⟨*proof*⟩

TODO: rename, duplication?

locale *second-derivative-within'* =

fixes *f* *f*' *f*'' *a* *G*

assumes *first-fderiv*[*derivative-intros*]:

$\bigwedge a. a \in G \implies (f \text{ has-derivative } f' a) \text{ (at } a \text{ within } G)$

assumes *in-G*: *a* ∈ *G*

assumes *second-fderiv*[*derivative-intros*]:

$\bigwedge i. ((\lambda x. f' x i) \text{ has-derivative } f'' i) \text{ (at } a \text{ within } G)$

begin

lemma *symmetric-second-derivative-within*:

assumes *a* ∈ *G* open *G*

assumes $\bigwedge s t. s \in \{0..1\} \implies t \in \{0..1\} \implies a + s *_R i + t *_R j \in G$

shows *f*'' *i j* = *f*'' *j i*

⟨*proof*⟩

end

locale *second-derivative-on-open* =

fixes *f*::'*a*::*real-normed-vector* ⇒ '*b*::*banach*

and *f*' :: '*a* ⇒ '*a* ⇒ '*b*

and *f*'' :: '*a* ⇒ '*a* ⇒ '*b*

and *a* :: '*a*


```

and  $G :: 'a$  set
assumes first-fderiv[derivative-intros]:
   $\bigwedge a. a \in G \implies (f \text{ has-derivative } f' a) \text{ (at } a)$ 
assumes in-G:  $a \in G$  and open-G: open  $G$ 
assumes second-fderiv[derivative-intros]:
   $((\lambda x. f' x i) \text{ has-derivative } f'' i) \text{ (at } a)$ 
begin

```

```

lemma symmetric-second-derivative:

```

```

  assumes  $a \in G$ 
  shows  $f'' i j = f'' j i$ 
  <proof>

```

```

end

```

```

no-notation

```

```

  blinfun-apply (infixl $ 999)
notation vec-nth (infixl $ 90)

```

```

end

```

5 Flow

```

theory Flow

```

```

imports

```

```

  Picard-Lindelof-Qualitative
  HOL-Library.Diagonal-Subsequence
  ../Library/Bounded-Linear-Operator
  ../Library/Multivariate-Taylor
  ../Library/Interval-Integral-HK

```

```

begin

```

TODO: extend theorems for dependence on initial time

5.1 simp rules for integrability (TODO: move)

```

lemma blinfun-ext:  $x = y \iff (\forall i. \text{blinfun-apply } x \ i = \text{blinfun-apply } y \ i)$ 
  <proof>

```

```

notation id-blinfun ( $1_L$ )

```

```

lemma blinfun-inverse-left:

```

```

  fixes  $f :: 'a :: \text{euclidean-space} \Rightarrow_L 'a$  and  $f'$ 
  shows  $f \circ_L f' = 1_L \iff f' \circ_L f = 1_L$ 
  <proof>

```

```

lemma onorm-zero-blinfun[simp]:  $\text{onorm } (\text{blinfun-apply } 0) = 0$ 
  <proof>

```

lemma *blinfun-compose-1-left*[simp]: $x \circ_L 1_L = x$
and *blinfun-compose-1-right*[simp]: $1_L \circ_L y = y$
 ⟨proof⟩

named-theorems *integrable-on-simps*

lemma *integrable-on-refl-ivl*[intro, simp]: g *integrable-on* $\{b \dots (b::'b::\text{ordered-euclidean-space})\}$
and *integrable-on-refl-closed-segment*[intro, simp]: h *integrable-on closed-segment*
 a a
 ⟨proof⟩

lemma *integrable-const-ivl-closed-segment*[intro, simp]: $(\lambda x. c)$ *integrable-on closed-segment*
 a $(b::\text{real})$
 ⟨proof⟩

lemma *integrable-ident-ivl*[intro, simp]: $(\lambda x. x)$ *integrable-on closed-segment* a $(b::\text{real})$
and *integrable-ident-cbox*[intro, simp]: $(\lambda x. x)$ *integrable-on cbox* a $(b::\text{real})$
 ⟨proof⟩

lemma *content-closed-segment-real*:
fixes a $b::\text{real}$
shows *content* (*closed-segment* a b) = $\text{abs } (b - a)$
 ⟨proof⟩

lemma *integral-const-closed-segment*:
fixes a $b::\text{real}$
shows *integral* (*closed-segment* a b) $(\lambda x. c)$ = $\text{abs } (b - a) *_{\mathbb{R}} c$
 ⟨proof⟩

lemmas [*integrable-on-simps*] =
integrable-on-empty — *empty*
integrable-on-refl *integrable-on-refl-ivl* *integrable-on-refl-closed-segment* — *singleton*
integrable-const *integrable-const-ivl* *integrable-const-ivl-closed-segment* — *constant*
ident-integrable-on *integrable-ident-ivl* *integrable-ident-cbox* — *identity*

lemma *integrable-cmul-real*:
fixes $K::\text{real}$
shows f *integrable-on* $X \implies (\lambda x. K * f x)$ *integrable-on* X
 ⟨proof⟩

lemmas [*integrable-on-simps*] =
integrable-0
integrable-neg
integrable-cmul
integrable-cmul-real
integrable-on-cmult-iff
integrable-on-cmult-left

integrable-on-cmult-right
integrable-on-cdivide
integrable-on-cmult-iff
integrable-on-cmult-left-iff
integrable-on-cmult-right-iff
integrable-on-cdivide-iff
integrable-diff
integrable-add
integrable-sum

lemma *dist-cancel-add1*: $\text{dist } (t0 + et) t0 = \text{norm } et$
 ⟨proof⟩

lemma *double-nonneg-le*:
fixes $a::\text{real}$
shows $a * 2 \leq b \implies a \geq 0 \implies a \leq b$
 ⟨proof⟩

5.2 Nonautonomous IVP on maximal existence interval

context *ll-on-open-it*
begin

context
fixes $x0$
assumes *iv-defined*: $t0 \in T \ x0 \in X$
begin

lemmas *closed-segment-iv-subset-domain* = *closed-segment-subset-domainI*[*OF iv-defined*(1)]

lemma
local-unique-solutions:
obtains $t \ u \ L$
where
 $0 < t \ 0 < u$
 $\text{cball } t0 \ t \subseteq \text{existence-ivl } t0 \ x0$
 $\text{cball } x0 \ (2 * u) \subseteq X$
 $\bigwedge t'. t' \in \text{cball } t0 \ t \implies L\text{-lipschitz-on } (\text{cball } x0 \ (2 * u)) \ (f \ t')$
 $\bigwedge x. x \in \text{cball } x0 \ u \implies (\text{flow } t0 \ x \ \text{usolves-ode } f \ \text{from } t0) \ (\text{cball } t0 \ t) \ (\text{cball } x \ u)$
 $\bigwedge x. x \in \text{cball } x0 \ u \implies \text{cball } x \ u \subseteq X$
 ⟨proof⟩

lemma *Picard-iterate-mem-existence-ivlI*:
assumes $t \in T$
assumes *compact* $C \ x0 \in C \ C \subseteq X$
assumes $\bigwedge y \ s. s \in \{t0 \ \text{--} \ t\} \implies y \ t0 = x0 \implies y \in \{t0 \ \text{--} \ s\} \rightarrow C \implies$
continuous-on $\{t0 \ \text{--} \ s\} \ y \implies$
 $x0 + \text{ivl-integral } t0 \ s \ (\lambda t. f \ t \ (y \ t)) \in C$
shows $t \in \text{existence-ivl } t0 \ x0 \ \bigwedge s. s \in \{t0 \ \text{--} \ t\} \implies \text{flow } t0 \ x0 \ s \in C$

<proof>

lemma *flow-has-vderiv-on*: (*flow* *t0* *x0* *has-vderiv-on* ($\lambda t. f t$ (*flow* *t0* *x0* *t*))) (*existence-ivl* *t0* *x0*)
<proof>

lemmas *flow-has-vderiv-on-compose*[*derivative-intros*] =
has-vderiv-on-compose2[*OF* *flow-has-vderiv-on*, *THEN* *has-vderiv-on-eq-rhs*]

end

lemma *unique-on-intersection*:
assumes *sols*: (*x solves-ode* *f*) *U* *X* (*y solves-ode* *f*) *V* *X*
assumes *iv-mem*: *t0* \in *U* *t0* \in *V* **and** *subs*: *U* \subseteq *T* *V* \subseteq *T*
assumes *ivls*: *is-interval* *U* *is-interval* *V*
assumes *iv*: *x* *t0* = *y* *t0*
assumes *mem*: *t* \in *U* *t* \in *V*
shows *x* *t* = *y* *t*
<proof>

lemma *unique-solution*:
assumes *sols*: (*x solves-ode* *f*) *U* *X* (*y solves-ode* *f*) *U* *X*
assumes *iv-mem*: *t0* \in *U* **and** *subs*: *U* \subseteq *T*
assumes *ivls*: *is-interval* *U*
assumes *iv*: *x* *t0* = *y* *t0*
assumes *mem*: *t* \in *U*
shows *x* *t* = *y* *t*
<proof>

lemma
assumes *s*: *s* \in *existence-ivl* *t0* *x0*
assumes *t*: *t* + *s* \in *existence-ivl* *s* (*flow* *t0* *x0* *s*)
shows *flow-trans*: *flow* *t0* *x0* (*s* + *t*) = *flow* *s* (*flow* *t0* *x0* *s*) (*s* + *t*)
and *existence-ivl-trans*: *s* + *t* \in *existence-ivl* *t0* *x0*
<proof>

lemma
assumes *t*: *t* \in *existence-ivl* *t0* *x0*
shows *flows-reverse*: *flow* *t* (*flow* *t0* *x0* *t*) *t0* = *x0*
and *existence-ivl-reverse*: *t0* \in *existence-ivl* *t* (*flow* *t0* *x0* *t*)
<proof>

lemma *flow-has-derivative*:
assumes *t* \in *existence-ivl* *t0* *x0*
shows (*flow* *t0* *x0* *has-derivative* ($\lambda i. i *_R f t$ (*flow* *t0* *x0* *t*))) (*at* *t*)
<proof>

lemma *flow-has-vector-derivative*:

assumes $t \in \text{existence-ivl } t0 \ x0$
shows $(\text{flow } t0 \ x0 \ \text{has-vector-derivative } f \ t \ (\text{flow } t0 \ x0 \ t)) \ (\text{at } t)$
 $\langle \text{proof} \rangle$

lemma *flow-has-vector-derivative-at-0:*

assumes $t \in \text{existence-ivl } t0 \ x0$
shows $((\lambda h. \text{flow } t0 \ x0 \ (t + h)) \ \text{has-vector-derivative } f \ t \ (\text{flow } t0 \ x0 \ t)) \ (\text{at } 0)$
 $\langle \text{proof} \rangle$

lemma

assumes $t \in \text{existence-ivl } t0 \ x0$
shows *closed-segment-subset-existence-ivl:* $\text{closed-segment } t0 \ t \subseteq \text{existence-ivl } t0 \ x0$
and *ivl-subset-existence-ivl:* $\{t0 \ .. \ t\} \subseteq \text{existence-ivl } t0 \ x0$
and *ivl-subset-existence-ivl':* $\{t \ .. \ t0\} \subseteq \text{existence-ivl } t0 \ x0$
 $\langle \text{proof} \rangle$

lemma *flow-fixed-point:*

assumes $t: t \in \text{existence-ivl } t0 \ x0$
shows $\text{flow } t0 \ x0 \ t = x0 + \text{ivl-integral } t0 \ t \ (\lambda t. f \ t \ (\text{flow } t0 \ x0 \ t))$
 $\langle \text{proof} \rangle$

lemma *flow-continuous:* $t \in \text{existence-ivl } t0 \ x0 \implies \text{continuous } (\text{at } t) \ (\text{flow } t0 \ x0)$
 $\langle \text{proof} \rangle$

lemma *flow-tendsto:* $t \in \text{existence-ivl } t0 \ x0 \implies (ts \longrightarrow t) \ F \implies$
 $((\lambda s. \text{flow } t0 \ x0 \ (ts \ s)) \longrightarrow \text{flow } t0 \ x0 \ t) \ F$
 $\langle \text{proof} \rangle$

lemma *flow-continuous-on:* $\text{continuous-on } (\text{existence-ivl } t0 \ x0) \ (\text{flow } t0 \ x0)$
 $\langle \text{proof} \rangle$

lemma *flow-continuous-on-intro:*

continuous-on $s \ g \implies$
 $(\bigwedge xa. xa \in s \implies g \ xa \in \text{existence-ivl } t0 \ x0) \implies$
continuous-on $s \ (\lambda xa. \text{flow } t0 \ x0 \ (g \ xa))$
 $\langle \text{proof} \rangle$

lemma *f-flow-continuous:*

assumes $t \in \text{existence-ivl } t0 \ x0$
shows $\text{isCont } (\lambda t. f \ t \ (\text{flow } t0 \ x0 \ t)) \ t$
 $\langle \text{proof} \rangle$

lemma *exponential-initial-condition:*

assumes $y0: t \in \text{existence-ivl } t0 \ y0$
assumes $z0: t \in \text{existence-ivl } t0 \ z0$
assumes $Y \subseteq X$
assumes *remain:* $\bigwedge s. s \in \text{closed-segment } t0 \ t \implies \text{flow } t0 \ y0 \ s \in Y$
 $\bigwedge s. s \in \text{closed-segment } t0 \ t \implies \text{flow } t0 \ z0 \ s \in Y$

assumes *lipschitz*: $\bigwedge s. s \in \text{closed-segment } t0 \ t \implies K\text{-lipschitz-on } Y \ (f \ s)$
shows $\text{norm } (\text{flow } t0 \ y0 \ t - \text{flow } t0 \ z0 \ t) \leq \text{norm } (y0 - z0) * \exp ((K + 1) * \text{abs } (t - t0))$
 $\langle \text{proof} \rangle$

lemma

existence-ivl-cballs:

assumes *iv-defined*: $t0 \in T \ x0 \in X$

obtains $t \ u \ L$

where

$\bigwedge y. y \in \text{cball } x0 \ u \implies \text{cball } t0 \ t \subseteq \text{existence-ivl } t0 \ y$

$\bigwedge s \ y. y \in \text{cball } x0 \ u \implies s \in \text{cball } t0 \ t \implies \text{flow } t0 \ y \ s \in \text{cball } y \ u$

$L\text{-lipschitz-on } (\text{cball } t0 \ t \times \text{cball } x0 \ u) \ (\lambda(t, x). \text{flow } t0 \ x \ t)$

$\bigwedge y. y \in \text{cball } x0 \ u \implies \text{cball } y \ u \subseteq X$

$0 < t \ t < u$

$\langle \text{proof} \rangle$

context

fixes $x0$

assumes *iv-defined*: $t0 \in T \ x0 \in X$

begin

lemma *existence-ivl-notempty*: $\text{existence-ivl } t0 \ x0 \neq \{\}$

$\langle \text{proof} \rangle$

lemma *initial-time-bounds*:

shows $\text{bdd-above } (\text{existence-ivl } t0 \ x0) \implies t0 < \text{Sup } (\text{existence-ivl } t0 \ x0)$ (**is** ?a $\implies -$)

and $\text{bdd-below } (\text{existence-ivl } t0 \ x0) \implies \text{Inf } (\text{existence-ivl } t0 \ x0) < t0$ (**is** ?b $\implies -$)

$\langle \text{proof} \rangle$

lemma

flow-leaves-compact-ivl-right:

assumes *bdd*: $\text{bdd-above } (\text{existence-ivl } t0 \ x0)$

defines $b \equiv \text{Sup } (\text{existence-ivl } t0 \ x0)$

assumes $b \in T$

assumes *compact* K

assumes $K \subseteq X$

obtains t **where** $t \geq t0 \ t \in \text{existence-ivl } t0 \ x0 \ \text{flow } t0 \ x0 \ t \notin K$

$\langle \text{proof} \rangle$

lemma

flow-leaves-compact-ivl-left:

assumes *bdd*: $\text{bdd-below } (\text{existence-ivl } t0 \ x0)$

defines $b \equiv \text{Inf } (\text{existence-ivl } t0 \ x0)$

assumes $b \in T$

assumes *compact* K

assumes $K \subseteq X$

obtains t **where** $t \leq t0$ $t \in \text{existence-ivl } t0 \ x0$ $\text{flow } t0 \ x0 \ t \notin K$
(proof)

lemma

sup-existence-maximal:

assumes $\bigwedge t. t0 \leq t \implies t \in \text{existence-ivl } t0 \ x0 \implies \text{flow } t0 \ x0 \ t \in K$

assumes *compact* $K \ K \subseteq X$

assumes *bdd-above* ($\text{existence-ivl } t0 \ x0$)

shows $\text{Sup } (\text{existence-ivl } t0 \ x0) \notin T$

(proof)

lemma

inf-existence-minimal:

assumes $\bigwedge t. t \leq t0 \implies t \in \text{existence-ivl } t0 \ x0 \implies \text{flow } t0 \ x0 \ t \in K$

assumes *compact* $K \ K \subseteq X$

assumes *bdd-below* ($\text{existence-ivl } t0 \ x0$)

shows $\text{Inf } (\text{existence-ivl } t0 \ x0) \notin T$

(proof)

end

lemma

subset-mem-compact-implies-subset-existence-interval:

assumes *ivl*: $t0 \in T'$ *is-interval* $T' \ T' \subseteq T$

assumes *iv-defined*: $x0 \in X$

assumes *mem-compact*: $\bigwedge t. t \in T' \implies t \in \text{existence-ivl } t0 \ x0 \implies \text{flow } t0 \ x0 \ t \in K$

assumes *K*: *compact* $K \ K \subseteq X$

shows $T' \subseteq \text{existence-ivl } t0 \ x0$

(proof)

lemma

mem-compact-implies-subset-existence-interval:

assumes *iv-defined*: $t0 \in T \ x0 \in X$

assumes *mem-compact*: $\bigwedge t. t \in T \implies t \in \text{existence-ivl } t0 \ x0 \implies \text{flow } t0 \ x0 \ t \in K$

assumes *K*: *compact* $K \ K \subseteq X$

shows $T \subseteq \text{existence-ivl } t0 \ x0$

(proof)

lemma

global-right-existence-ivl-explicit:

assumes $b \geq t0$

assumes $b: b \in \text{existence-ivl } t0 \ x0$

obtains $d \ K$ **where** $d > 0 \ K > 0$

$\text{ball } x0 \ d \subseteq X$

$\bigwedge y. y \in \text{ball } x0 \ d \implies b \in \text{existence-ivl } t0 \ y$

$\bigwedge t y. y \in \text{ball } x0 \ d \implies t \in \{t0 \ .. \ b\} \implies$

$\text{dist } (\text{flow } t0 \ x0 \ t) (\text{flow } t0 \ y \ t) \leq \text{dist } x0 \ y * \exp (K * \text{abs } (t - t0))$

<proof>

lemma

global-left-existence-ivl-explicit:

assumes $b \leq t0$

assumes $b: b \in \text{existence-ivl } t0 \ x0$

assumes *iv-defined*: $t0 \in T \ x0 \in X$

obtains $d \ K$ **where** $d > 0 \ K > 0$

$\text{ball } x0 \ d \subseteq X$

$\bigwedge y. y \in \text{ball } x0 \ d \implies b \in \text{existence-ivl } t0 \ y$

$\bigwedge t y. y \in \text{ball } x0 \ d \implies t \in \{b .. t0\} \implies \text{dist } (\text{flow } t0 \ x0 \ t) (\text{flow } t0 \ y \ t) \leq \text{dist } x0 \ y * \text{exp } (K * \text{abs } (t - t0))$

<proof>

lemma

global-existence-ivl-explicit:

assumes $a: a \in \text{existence-ivl } t0 \ x0$

assumes $b: b \in \text{existence-ivl } t0 \ x0$

assumes $le: a \leq b$

obtains $d \ K$ **where** $d > 0 \ K > 0$

$\text{ball } x0 \ d \subseteq X$

$\bigwedge y. y \in \text{ball } x0 \ d \implies a \in \text{existence-ivl } t0 \ y$

$\bigwedge y. y \in \text{ball } x0 \ d \implies b \in \text{existence-ivl } t0 \ y$

$\bigwedge t y. y \in \text{ball } x0 \ d \implies t \in \{a .. b\} \implies$

$\text{dist } (\text{flow } t0 \ x0 \ t) (\text{flow } t0 \ y \ t) \leq \text{dist } x0 \ y * \text{exp } (K * \text{abs } (t - t0))$

<proof>

lemma *eventually-exponential-separation:*

assumes $a: a \in \text{existence-ivl } t0 \ x0$

assumes $b: b \in \text{existence-ivl } t0 \ x0$

assumes $le: a \leq b$

obtains K **where** $K > 0 \ \forall_F y \text{ in at } x0. \forall t \in \{a..b\}. \text{dist } (\text{flow } t0 \ x0 \ t) (\text{flow } t0 \ y \ t) \leq \text{dist } x0 \ y * \text{exp } (K * |t - t0|)$

<proof>

lemma *eventually-mem-existence-ivl:*

assumes $b: b \in \text{existence-ivl } t0 \ x0$

shows $\forall_F x \text{ in at } x0. b \in \text{existence-ivl } t0 \ x$

<proof>

lemma *uniform-limit-flow:*

assumes $a: a \in \text{existence-ivl } t0 \ x0$

assumes $b: b \in \text{existence-ivl } t0 \ x0$

assumes $le: a \leq b$

shows *uniform-limit* $\{a .. b\} (\text{flow } t0) (\text{flow } t0 \ x0) (\text{at } x0)$

<proof>

lemma *eventually-at-fst:*

assumes *eventually* $P (\text{at } (\text{fst } x))$

assumes P ($\text{fst } x$)
shows *eventually* $(\lambda h. P (\text{fst } h))$ ($\text{at } x$)
 $\langle \text{proof} \rangle$

lemma *eventually-at-snd*:
assumes *eventually* P ($\text{at } (\text{snd } x)$)
assumes P ($\text{snd } x$)
shows *eventually* $(\lambda h. P (\text{snd } h))$ ($\text{at } x$)
 $\langle \text{proof} \rangle$

lemma
shows *open-state-space*: $\text{open } (\text{Sigma } X (\text{existence-ivl } t0))$
and *flow-continuous-on-state-space*:
 $\text{continuous-on } (\text{Sigma } X (\text{existence-ivl } t0)) (\lambda(x, t). \text{flow } t0 \ x \ t)$
 $\langle \text{proof} \rangle$

lemmas *flow-continuous-on-compose*[*continuous-intros*] =
continuous-on-compose-Pair[*OF flow-continuous-on-state-space*]

lemma *flow-isCont-state-space*: $t \in \text{existence-ivl } t0 \ x0 \implies \text{isCont } (\lambda(x, t). \text{flow } t0 \ x \ t)$ ($x0, t$)
 $\langle \text{proof} \rangle$

lemma
flow-absolutely-integrable-on[*integrable-on-simps*]:
assumes $s \in \text{existence-ivl } t0 \ x0$
shows $(\lambda x. \text{norm } (\text{flow } t0 \ x0 \ x))$ *integrable-on closed-segment* $t0 \ s$
 $\langle \text{proof} \rangle$

lemma *existence-ivl-eq-domain*:
assumes *iv-defined*: $t0 \in T \ x0 \in X$
assumes *bnd*: $\bigwedge tm \ tM \ t \ x. tm \in T \implies tM \in T \implies \exists M. \exists L. \forall t \in \{tm .. tM\}. \forall x \in X. \text{norm } (f \ t \ x) \leq M + L * \text{norm } x$
assumes *is-interval* $T \ X = \text{UNIV}$
shows $\text{existence-ivl } t0 \ x0 = T$
 $\langle \text{proof} \rangle$

lemma *flow-unique*:
assumes $t \in \text{existence-ivl } t0 \ x0$
assumes $\text{phi } t0 = x0$
assumes $\bigwedge t. t \in \text{existence-ivl } t0 \ x0 \implies (\text{phi has-vector-derivative } f \ t \ (\text{phi } t))$
 $(\text{at } t)$
assumes $\bigwedge t. t \in \text{existence-ivl } t0 \ x0 \implies \text{phi } t \in X$
shows $\text{flow } t0 \ x0 \ t = \text{phi } t$
 $\langle \text{proof} \rangle$

lemma *flow-unique-on*:
assumes $t \in \text{existence-ivl } t0 \ x0$
assumes $\text{phi } t0 = x0$

```

assumes (phi has-vderiv-on ( $\lambda t. f t (phi t)$ )) (existence-ivl t0 x0)
assumes  $\bigwedge t. t \in \text{existence-ivl } t0 \ x0 \implies phi \ t \in X$ 
shows flow t0 x0 t = phi t
<proof>

end — local-lipschitz T X f

locale two-ll-on-open =
  F: ll-on-open T1 F X + G: ll-on-open T2 G X
  for F T1 G T2 X J x0 +
  fixes e::real and K
  assumes t0-in-J: 0  $\in$  J
  assumes J-subset:  $J \subseteq F.\text{existence-ivl } 0 \ x0$ 
  assumes J-ivl: is-interval J
  assumes F-lipschitz:  $\bigwedge t. t \in J \implies K\text{-lipschitz-on } X \ (F \ t)$ 
  assumes K-pos: 0 < K
  assumes F-G-norm-ineq:  $\bigwedge t \ x. t \in J \implies x \in X \implies \text{norm } (F \ t \ x - G \ t \ x) < e$ 
begin

context begin

lemma F-iv-defined: 0  $\in$  T1 x0  $\in$  X
  <proof>

lemma e-pos: 0 < e
  <proof> definition flow0 t = F.flow 0 x0 t
qualified definition Y t = G.flow 0 x0 t

lemma norm-X-Y-bound:
shows  $\forall t \in J \cap G.\text{existence-ivl } 0 \ x0. \text{norm } (\text{flow0 } t - Y \ t) \leq e / K * (\exp(K * |t|) - 1)$ 
  <proof>

end

end

locale auto-ll-on-open =
  fixes f::'a::{banach, heine-borel}  $\implies$  'a and X
  assumes auto-local-lipschitz: local-lipschitz UNIV X ( $\lambda\cdot.f$ )
  assumes auto-open-domain[intro!, simp]: open X
begin

autonomous flow and existence interval

definition flow0 x0 t = ll-on-open.flow UNIV ( $\lambda\cdot.f$ ) X 0 x0 t

definition existence-ivl0 x0 = ll-on-open.existence-ivl UNIV ( $\lambda\cdot.f$ ) X 0 x0

sublocale ll-on-open-it UNIV  $\lambda\cdot.f$  X 0

```

rewrites $flow = (\lambda t0\ x0\ t.\ flow0\ x0\ (t - t0))$
and $existence-ivl = (\lambda t0\ x0.\ (+)\ t0\ 'existence-ivl0\ x0)$
and $(+)\ 0 = (\lambda x::real.\ x)$
and $s - 0 = s$
and $(\lambda x.\ x)\ 'S = S$
and $s \in (+)\ t\ 'S \iff s - t \in (S::real\ set)$
and $P\ (s + t - s) = P\ (t::real)$ — TODO: why does just the equation not work?
and $P\ (t + s - s) = P\ t$ — TODO: why does just the equation not work?
 $\langle proof \rangle$

lemma $existence-ivl-zero: x0 \in X \implies 0 \in existence-ivl0\ x0$ $\langle proof \rangle$

lemmas $[continuous-intros\ del] = continuous-on-f$

lemmas $continuous-on-f-comp[continuous-intros] = continuous-on-f[OF\ continuous-on-const - subset-UNIV]$

lemma

flow-in-compact-right-existence:

assumes $\bigwedge t.\ 0 \leq t \implies t \in existence-ivl0\ x \implies flow0\ x\ t \in K$

assumes $compact\ K\ K \subseteq X$

assumes $x \in X\ t \geq 0$

shows $t \in existence-ivl0\ x$

$\langle proof \rangle$

lemma

flow-in-compact-left-existence:

assumes $\bigwedge t.\ t \leq 0 \implies t \in existence-ivl0\ x \implies flow0\ x\ t \in K$

assumes $compact\ K\ K \subseteq X$

assumes $x \in X\ t \leq 0$

shows $t \in existence-ivl0\ x$

$\langle proof \rangle$

end

locale $compact-continuously-diff =$

derivative-on-prod $T\ X\ f\ \lambda(t, x).\ f'\ x\ o_L\ snd-blifun$

for $T\ X$ **and** $f::real \Rightarrow 'a::\{banach, perfect-space, heine-borel\} \Rightarrow 'a$

and $f'::'a \Rightarrow ('a, 'a)\ blifun +$

assumes $compact-domain: compact\ X$

assumes $convex: convex\ X$

assumes $nonempty-domains: T \neq \{\}\ X \neq \{\}$

assumes $continuous-derivative: continuous-on\ X\ f'$

begin

lemma $ex-onorm-bound:$

$\exists B.\ \forall x \in X.\ norm\ (f'\ x) \leq B$

$\langle proof \rangle$

definition *onorm-bound* = (SOME B. $\forall x \in X. \text{norm } (f' x) \leq B$)

lemma *onorm-bound*: **assumes** $x \in X$ **shows** $\text{norm } (f' x) \leq \text{onorm-bound}$
<proof>

sublocale *closed-domain* X
<proof>

sublocale *global-lipschitz* T X f *onorm-bound*
<proof>

end — *compact* X

locale *unique-on-compact-continuously-diff* = *self-mapping* +
compact-interval T +
compact-continuously-diff T X f

begin

sublocale *unique-on-closed* t0 T x0 f X *onorm-bound*
<proof>

end

locale *c1-on-open* =

fixes $f::'a::\{\text{banach, perfect-space, heine-borel}\} \Rightarrow 'a$ **and** $f' X$
assumes *open-dom*[*simp*]: *open* X
assumes *derivative-rhs*:
 $\bigwedge x. x \in X \implies (f \text{ has-derivative } \text{blinfun-apply } (f' x)) \text{ (at } x)$
assumes *continuous-derivative*: *continuous-on* X f'

begin

lemmas *continuous-derivative-comp*[*continuous-intros*] =
continuous-on-compose2[OF *continuous-derivative*]

lemma *derivative-tendsto*[*tendsto-intros*]:

assumes [*tendsto-intros*]: $(g \longrightarrow l) F$

and $l \in X$

shows $((\lambda x. f' (g x)) \longrightarrow f' l) F$

<proof>

lemma *c1-on-open-rev*[*intro, simp*]: *c1-on-open* $(-f) (-f')$ X
<proof>

lemma *derivative-rhs-compose*[*derivative-intros*]:

$((g \text{ has-derivative } g') \text{ (at } x \text{ within } s)) \implies g x \in X \implies$

$((\lambda x. f (g x)) \text{ has-derivative}$

$(\lambda xa. \text{blinfun-apply } (f' (g x)) (g' xa)))$

$\text{(at } x \text{ within } s)$

<proof>

sublocale *auto-ll-on-open*
 ⟨*proof*⟩

end — $?x \in X \implies (f \text{ has-derivative } \text{blinfun-apply } (f' ?x)) \text{ (at } ?x)$

locale *c1-on-open-euclidean* = *c1-on-open* $f f' X$
 for $f::'a::\text{euclidean-space} \Rightarrow -$ and $f' X$
begin
lemma *c1-on-open-euclidean-anchor*: *True* ⟨*proof*⟩

definition $\text{vareq } x0 \ t = f' (\text{flow0 } x0 \ t)$

interpretation *var*: *ll-on-open* *existence-ivl0* $x0$ *vareq* $x0$ *UNIV*
 ⟨*proof*⟩

context begin

lemma *continuous-on-A*[*continuous-intros*]:
 assumes *continuous-on* S a
 assumes *continuous-on* S b
 assumes $\bigwedge s. s \in S \implies a \ s \in X$
 assumes $\bigwedge s. s \in S \implies b \ s \in \text{existence-ivl0 } (a \ s)$
 shows *continuous-on* S $(\lambda s. \text{vareq } (a \ s) \ (b \ s))$
 ⟨*proof*⟩

lemmas [*intro*] = *mem-existence-ivl-iv-defined*

context
 fixes $x0::'a$
begin

lemma *flow0-defined*: $xa \in \text{existence-ivl0 } x0 \implies \text{flow0 } x0 \ xa \in X$
 ⟨*proof*⟩

lemma *continuous-on-flow0*: *continuous-on* $(\text{existence-ivl0 } x0)$ $(\text{flow0 } x0)$
 ⟨*proof*⟩

lemmas *continuous-on-flow0-comp*[*continuous-intros*] = *continuous-on-compose2*[*OF*
continuous-on-flow0]

lemma *varexivl-eq-exivl*:
 assumes $t \in \text{existence-ivl0 } x0$
 shows $\text{var.}\text{existence-ivl } x0 \ t \ a = \text{existence-ivl0 } x0$
 ⟨*proof*⟩

definition *vector-Dflow* $u0 \ t \equiv \text{var.}\text{flow } x0 \ 0 \ u0 \ t$

qualified abbreviation $Y \ z \ t \equiv \text{flow0 } (x0 + z) \ t$

Linearity of the solution to the variational equation. TODO: generalize this and some other things for arbitrary linear ODEs

lemma *vector-Dflow-linear*:

assumes $t \in \text{existence-ivl0 } x0$

shows $\text{vector-Dflow } (\alpha *_R a + \beta *_R b) t = \alpha *_R \text{vector-Dflow } a t + \beta *_R \text{vector-Dflow } b t$

<proof>

lemma *linear-vector-Dflow*:

assumes $t \in \text{existence-ivl0 } x0$

shows $\text{linear } (\lambda z. \text{vector-Dflow } z t)$

<proof>

lemma *bounded-linear-vector-Dflow*:

assumes $t \in \text{existence-ivl0 } x0$

shows $\text{bounded-linear } (\lambda z. \text{vector-Dflow } z t)$

<proof>

lemma *vector-Dflow-continuous-on-time*: $x0 \in X \implies \text{continuous-on } (\text{existence-ivl0 } x0) (\lambda t. \text{vector-Dflow } z t)$

<proof>

proposition *proposition-17-6-weak*:

— from "Differential Equations, Dynamical Systems, and an Introduction to Chaos", Hirsch/Smale/Devaney

assumes $t \in \text{existence-ivl0 } x0$

shows $(\lambda y. (Y (y - x0) t - \text{flow0 } x0 t - \text{vector-Dflow } (y - x0) t) /_R \text{norm } (y - x0)) - x0 \rightarrow 0$

<proof>

lemma *local-lipschitz-A*:

$OT \subseteq \text{existence-ivl0 } x0 \implies \text{local-lipschitz } OT (OS::('a \Rightarrow_L 'a) \text{ set}) (\lambda t. (o_L) (\text{vareq } x0 t))$

<proof>

lemma *total-derivative-ll-on-open*:

$\text{ll-on-open } (\text{existence-ivl0 } x0) (\lambda t. \text{blinfun-compose } (\text{vareq } x0 t)) (UNIV::('a \Rightarrow_L 'a) \text{ set})$

<proof>

end

end

sublocale *mvar*: $\text{ll-on-open } \text{existence-ivl0 } x0 \lambda t. \text{blinfun-compose } (\text{vareq } x0 t) UNIV::('a \Rightarrow_L 'a) \text{ set}$ **for** $x0$

<proof>

lemma *mvar-existence-ivl-eq-existence-ivl[simp]*:— TODO: unify with $?t \in \text{exis-$

tence-ivl0 ?x0.0 \implies *var.existence-ivl ?x0.0 ?t ?a = existence-ivl0 ?x0.0*
assumes *t* \in *existence-ivl0 x0*
shows *mvar.existence-ivl x0 t* = (λ -. *existence-ivl0 x0*)
 \langle *proof* \rangle

lemma

assumes *t* \in *existence-ivl0 x0*
shows *continuous-on* (*UNIV* \times *existence-ivl0 x0*) (λ (*x, ta*). *mvar.flow x0 t x ta*)
 \langle *proof* \rangle

definition *Dflow x0* = *mvar.flow x0 0 id-blinfun*

lemma *var-eq-mvar*:

assumes *t0* \in *existence-ivl0 x0*
assumes *t* \in *existence-ivl0 x0*
shows *var.flow x0 t0 i t* = *mvar.flow x0 t0 id-blinfun t i*
 \langle *proof* \rangle

lemma *Dflow-zero[simp]*: $x \in X \implies Dflow x 0 = 1_L$
 \langle *proof* \rangle

5.3 Differentiability of the flow0

U t, i.e. the solution of the variational equation, is the space derivative at the initial value *x0*.

lemma *flow-dx-derivative*:

assumes *t* \in *existence-ivl0 x0*
shows (λ *x0. flow0 x0 t*) *has-derivative* (λ *z. vector-Dflow x0 z t*) (*at x0*)
 \langle *proof* \rangle

lemma *flow-dx-derivative-blinfun*:

assumes *t* \in *existence-ivl0 x0*
shows (λ *x. flow0 x t*) *has-derivative* *Blinfun* (λ *z. vector-Dflow x0 z t*) (*at x0*)
 \langle *proof* \rangle

definition *flowderiv x0 t* = *comp12* (*Dflow x0 t*) (*blinfun-scaleR-left* (*f* (*flow0 x0 t*)))

lemma *flowderiv-eq*: *flowderiv x0 t* (ξ_1, ξ_2) = (*Dflow x0 t*) $\xi_1 + \xi_2 *_R f$ (*flow0 x0 t*)
 \langle *proof* \rangle

lemma *W-continuous-on*: *continuous-on* (*Sigma X existence-ivl0*) (λ (*x0, t*). *Dflow x0 t*)

— TODO: somewhere here is hidden continuity wrt rhs of ODE, extract it!
 \langle *proof* \rangle

lemma *W-continuous-on-comp[continuous-intros]*:

assumes *h*: *continuous-on S h* **and** *g*: *continuous-on S g*

shows $(\bigwedge s. s \in S \implies h s \in X) \implies (\bigwedge s. s \in S \implies g s \in \text{existence-ivl0 } (h s))$
 \implies
continuous-on S $(\lambda s. D\text{flow } (h s) (g s))$
 $\langle \text{proof} \rangle$

lemma *f-flow-continuous-on: continuous-on* $(\text{Sigma } X \text{ existence-ivl0}) (\lambda(x0, t). f$
 $(\text{flow0 } x0 t))$
 $\langle \text{proof} \rangle$

lemma
flow-has-space-derivative:
assumes $t \in \text{existence-ivl0 } x0$
shows $((\lambda x0. \text{flow0 } x0 t) \text{ has-derivative } D\text{flow } x0 t)$ $(\text{at } x0)$
 $\langle \text{proof} \rangle$

lemma
flow-has-flowderiv:
assumes $t \in \text{existence-ivl0 } x0$
shows $((\lambda(x0, t). \text{flow0 } x0 t) \text{ has-derivative } \text{flowderiv } x0 t)$ $(\text{at } (x0, t) \text{ within } S)$
 $\langle \text{proof} \rangle$

lemma *flow0-comp-has-derivative:*
assumes $h: h s \in \text{existence-ivl0 } (g s)$
assumes $[\text{derivative-intros}]: (g \text{ has-derivative } g')$ $(\text{at } s \text{ within } S)$
assumes $[\text{derivative-intros}]: (h \text{ has-derivative } h')$ $(\text{at } s \text{ within } S)$
shows $((\lambda x. \text{flow0 } (g x) (h x)) \text{ has-derivative } (\lambda x. \text{blinfun-apply } (\text{flowderiv } (g s)$
 $(h s)) (g' x, h' x)))$
 $(\text{at } s \text{ within } S)$
 $\langle \text{proof} \rangle$

lemma *flowderiv-continuous-on: continuous-on* $(\text{Sigma } X \text{ existence-ivl0}) (\lambda(x0, t).$
 $\text{flowderiv } x0 t)$
 $\langle \text{proof} \rangle$

lemma *flowderiv-continuous-on-comp[continuous-intros]:*
assumes *continuous-on* S x
assumes *continuous-on* S t
assumes $\bigwedge s. s \in S \implies x s \in X \bigwedge s. s \in S \implies t s \in \text{existence-ivl0 } (x s)$
shows *continuous-on* S $(\lambda xa. \text{flowderiv } (x xa) (t xa))$
 $\langle \text{proof} \rangle$

lemmas $[\text{intro}] = \text{flow-in-domain}$

lemma *vareq-trans: t0* $\in \text{existence-ivl0 } x0 \implies t \in \text{existence-ivl0 } (\text{flow0 } x0 t0) \implies$
 $\text{vareq } (\text{flow0 } x0 t0) t = \text{vareq } x0 (t0 + t)$
 $\langle \text{proof} \rangle$

lemma *diff-existence-ivl-trans:*
 $t0 \in \text{existence-ivl0 } x0 \implies t \in \text{existence-ivl0 } x0 \implies t - t0 \in \text{existence-ivl0 } (\text{flow0}$

$x0\ t0)$ for t
 ⟨proof⟩

lemma *has-vderiv-on-blinfun-compose-right*[*derivative-intros*]:

assumes $(g\ \text{has-vderiv-on}\ g')\ T$
assumes $\bigwedge x. x \in T \implies gd' x = g' x\ o_L\ d$
shows $((\lambda x. g\ x\ o_L\ d)\ \text{has-vderiv-on}\ gd')\ T$
 ⟨proof⟩

lemma *has-vderiv-on-blinfun-compose-left*[*derivative-intros*]:

assumes $(g\ \text{has-vderiv-on}\ g')\ T$
assumes $\bigwedge x. x \in T \implies gd' x = d\ o_L\ g' x$
shows $((\lambda x. d\ o_L\ g\ x)\ \text{has-vderiv-on}\ gd')\ T$
 ⟨proof⟩

lemma *mvar-flow-shift*:

assumes $t0 \in \text{existence-ivl0}\ x0\ t1 \in \text{existence-ivl0}\ x0$
shows $mvar.\text{flow}\ x0\ t0\ d\ t1 = D\text{flow}\ (\text{flow0}\ x0\ t0)\ (t1 - t0)\ o_L\ d$
 ⟨proof⟩

lemma *Dflow-trans*:

assumes $h \in \text{existence-ivl0}\ x0$
assumes $i \in \text{existence-ivl0}\ (\text{flow0}\ x0\ h)$
shows $D\text{flow}\ x0\ (h + i) = D\text{flow}\ (\text{flow0}\ x0\ h)\ i\ o_L\ (D\text{flow}\ x0\ h)$
 ⟨proof⟩

lemma *Dflow-trans-apply*:

assumes $h \in \text{existence-ivl0}\ x0$
assumes $i \in \text{existence-ivl0}\ (\text{flow0}\ x0\ h)$
shows $D\text{flow}\ x0\ (h + i)\ d0 = D\text{flow}\ (\text{flow0}\ x0\ h)\ i\ (D\text{flow}\ x0\ h\ d0)$
 ⟨proof⟩

end — *True*

end

6 Upper and Lower Solutions

theory *Upper-Lower-Solution*

imports *Flow*

begin

Following Walter [1] in section 9

lemma *IVT-min*:

fixes $f :: \text{real} \Rightarrow 'b :: \{\text{linorder-topology}, \text{real-normed-vector}, \text{ordered-real-vector}\}$
 — generalize?
assumes $y: f\ a \leq y\ y \leq f\ b\ a \leq b$
assumes $*: \text{continuous-on}\ \{a .. b\}\ f$

notes $[continuous-intros] = *[THEN\ continuous-on-subset]$
obtains x **where** $a \leq x \leq b \wedge f\ x = y \wedge x'. a \leq x' \implies x' < x \implies f\ x' < y$
 $\langle proof \rangle$

lemma *filtermap-at-left-shift*: $filtermap\ (\lambda x. x - d)\ (at-left\ a) = at-left\ (a - d)::real$
 $\langle proof \rangle$

context

fixes $v\ v'\ w\ w'::real \Rightarrow real$ **and** $t0\ t1\ e::real$
assumes v' : $(v\ has-vderiv-on\ v')\ \{t0 <.. t1\}$
and w' : $(w\ has-vderiv-on\ w')\ \{t0 <.. t1\}$
assumes *pos-ivl*: $t0 < t1$
assumes *e-pos*: $e > 0$ **and** *e-in*: $t0 + e \leq t1$
assumes *less*: $\bigwedge t. t0 < t \implies t < t0 + e \implies v\ t < w\ t$
begin

lemma *first-intersection-crossing-derivatives*:

assumes *na*: $t0 < tg \leq t1 \wedge v\ tg \geq w\ tg$
notes $[continuous-intros] =$
 $vderiv-on-continuous-on[OF\ v',\ THEN\ continuous-on-subset]$
 $vderiv-on-continuous-on[OF\ w',\ THEN\ continuous-on-subset]$
obtains $x0$ **where**
 $t0 < x0 \leq tg$
 $v'\ x0 \geq w'\ x0$
 $v\ x0 = w\ x0$
 $\bigwedge t. t0 < t \implies t < x0 \implies v\ t < w\ t$
 $\langle proof \rangle$

lemma *defect-less*:

assumes *b*: $\bigwedge t. t0 < t \implies t \leq t1 \implies v'\ t - f\ t\ (v\ t) < w'\ t - f\ t\ (w\ t)$
notes $[continuous-intros] =$
 $vderiv-on-continuous-on[OF\ v',\ THEN\ continuous-on-subset]$
 $vderiv-on-continuous-on[OF\ w',\ THEN\ continuous-on-subset]$
shows $\forall t \in \{t0 <.. t1\}. v\ t < w\ t$
 $\langle proof \rangle$

end

lemma *has-derivatives-less-lemma*:

fixes $v\ v'::real \Rightarrow real$
assumes v' : $(v\ has-vderiv-on\ v')\ T$
assumes y' : $(y\ has-vderiv-on\ y')\ T$
assumes *lu*: $\bigwedge t. t \in T \implies t > t0 \implies v'\ t - f\ t\ (v\ t) < y'\ t - f\ t\ (y\ t)$
assumes *lower*: $v\ t0 \leq y\ t0$
assumes *eq-imp*: $v\ t0 = y\ t0 \implies v'\ t0 < y'\ t0$
assumes *t*: $t0 < t \wedge t0 \in T \wedge t \in T$ *is-interval* T
shows $v\ t < y\ t$
 $\langle proof \rangle$

lemma *strict-lower-solution*:
fixes $v\ v' :: \text{real} \Rightarrow \text{real}$
assumes $\text{sol}: (y \text{ solves-ode } f) \ T \ X$
assumes $v': (v \text{ has-vderiv-on } v') \ T$
assumes $\text{lower}: \bigwedge t. t \in T \Longrightarrow t > t0 \Longrightarrow v' \ t < f \ t \ (v \ t)$
assumes $\text{iv}: v \ t0 \leq y \ t0 \ v \ t0 = y \ t0 \Longrightarrow v' \ t0 < f \ t0 \ (y \ t0)$
assumes $t: t0 < t \ t0 \in T \ t \in T \text{ is-interval } T$
shows $v \ t < y \ t$
 $\langle \text{proof} \rangle$

lemma *strict-upper-solution*:
fixes $w \ w' :: \text{real} \Rightarrow \text{real}$
assumes $\text{sol}: (y \text{ solves-ode } f) \ T \ X$
assumes $w': (w \text{ has-vderiv-on } w') \ T$
and $\text{upper}: \bigwedge t. t \in T \Longrightarrow t > t0 \Longrightarrow f \ t \ (w \ t) < w' \ t$
and $\text{iv}: y \ t0 \leq w \ t0 \ y \ t0 = w \ t0 \Longrightarrow f \ t0 \ (y \ t0) < w' \ t0$
assumes $t: t0 < t \ t0 \in T \ t \in T \text{ is-interval } T$
shows $y \ t < w \ t$
 $\langle \text{proof} \rangle$

lemma *uniform-limit-at-within-subset*:
assumes $\text{uniform-limit } S \ x \ l \ (\text{at } t \ \text{within } T)$
assumes $U \subseteq T$
shows $\text{uniform-limit } S \ x \ l \ (\text{at } t \ \text{within } U)$
 $\langle \text{proof} \rangle$

lemma *uniform-limit-le*:
fixes $f :: 'c \Rightarrow 'a \Rightarrow 'b :: \{\text{metric-space, linorder-topology}\}$
assumes $I: I \neq \text{bot}$
assumes $u: \text{uniform-limit } X \ f \ g \ I$
assumes $u': \text{uniform-limit } X \ f' \ g' \ I$
assumes $\forall_F \ i \ \text{in } I. \forall x \in X. f \ i \ x \leq f' \ i \ x$
assumes $x \in X$
shows $g \ x \leq g' \ x$
 $\langle \text{proof} \rangle$

lemma *uniform-limit-le-const*:
fixes $f :: 'c \Rightarrow 'a \Rightarrow 'b :: \{\text{metric-space, linorder-topology}\}$
assumes $I: I \neq \text{bot}$
assumes $u: \text{uniform-limit } X \ f \ g \ I$
assumes $\forall_F \ i \ \text{in } I. \forall x \in X. f \ i \ x \leq h \ x$
assumes $x \in X$
shows $g \ x \leq h \ x$
 $\langle \text{proof} \rangle$

lemma *uniform-limit-ge-const*:
fixes $f :: 'c \Rightarrow 'a \Rightarrow 'b :: \{\text{metric-space, linorder-topology}\}$
assumes $I: I \neq \text{bot}$

```

assumes u: uniform-limit  $X f g I$ 
assumes  $\forall_F i \text{ in } I. \forall x \in X. h x \leq f i x$ 
assumes  $x \in X$ 
shows  $h x \leq g x$ 
<proof>

locale ll-on-open-real = ll-on-open  $T f X$  for  $T f$  and  $X::\text{real set}$ 
begin

lemma lower-solution:
  fixes  $v v' :: \text{real} \Rightarrow \text{real}$ 
  assumes sol: (y solves-ode  $f$ )  $S X$ 
  assumes v': (v has-vderiv-on  $v'$ )  $S$ 
  assumes lower:  $\bigwedge t. t \in S \implies t > t0 \implies v' t < f t (v t)$ 
  assumes iv:  $v t0 \leq y t0$ 
  assumes t:  $t0 \leq t t0 \in S t \in S \text{ is-interval } S S \subseteq T$ 
  shows  $v t \leq y t$ 
<proof>

lemma upper-solution:
  fixes  $v v' :: \text{real} \Rightarrow \text{real}$ 
  assumes sol: (y solves-ode  $f$ )  $S X$ 
  assumes v': (v has-vderiv-on  $v'$ )  $S$ 
  assumes upper:  $\bigwedge t. t \in S \implies t > t0 \implies f t (v t) < v' t$ 
  assumes iv:  $y t0 \leq v t0$ 
  assumes t:  $t0 \leq t t0 \in S t \in S \text{ is-interval } S S \subseteq T$ 
  shows  $y t \leq v t$ 
<proof>

end

end
theory Poincare-Map
imports
  Flow
begin

abbreviation plane  $n c \equiv \{x. x \cdot n = c\}$ 

lemma
  eventually-tendsto-compose-within:
  assumes eventually  $P$  (at  $l$  within  $S$ )
  assumes  $P l$ 
  assumes ( $f \longrightarrow l$ ) (at  $x$  within  $T$ )
  assumes eventually  $(\lambda x. f x \in S)$  (at  $x$  within  $T$ )
  shows eventually  $(\lambda x. P (f x))$  (at  $x$  within  $T$ )
<proof>

lemma

```

eventually-eventually-withinI:— aha...
assumes $\forall_F x$ in at x within A . $P x P x$
shows $\forall_F a$ in at x within S . $\forall_F x$ in at a within A . $P x$
 \langle proof \rangle

lemma *eventually-not-in-closed*:
assumes *closed* P
assumes $f t \notin P t \in T$
assumes *continuous-on* $T f$
shows $\forall_F t$ in at t within T . $f t \notin P$
 \langle proof \rangle

context *ll-on-open-it* **begin**

lemma
existence-ivl-trans':
assumes $t + s \in$ *existence-ivl* $t0 x0$
 $t \in$ *existence-ivl* $t0 x0$
shows $t + s \in$ *existence-ivl* t (*flow* $t0 x0 t$)
 \langle proof \rangle

end

context *auto-ll-on-open*— TODO: generalize to continuous systems
begin

definition *returns-to* :: 'a set \Rightarrow 'a \Rightarrow bool
where *returns-to* $P x \longleftrightarrow (\forall_F t$ in at-right 0 . *flow0* $x t \notin P) \wedge (\exists t > 0$. $t \in$
existence-ivl0 $x \wedge$ *flow0* $x t \in P)$

definition *return-time* :: 'a set \Rightarrow 'a \Rightarrow real
where *return-time* $P x =$
(if *returns-to* $P x$ *then* (*SOME* t .
 $t > 0 \wedge$
 $t \in$ *existence-ivl0* $x \wedge$
flow0 $x t \in P \wedge$
 $(\forall s \in \{0 <..<t\}$. *flow0* $x s \notin P))$ *else* $0)$

lemma *returns-toI*:
assumes $t > 0 t \in$ *existence-ivl0* x *flow0* $x t \in P$
assumes *ev*: $\forall_F t$ in at-right 0 . *flow0* $x t \notin P$
assumes *closed* P
shows *returns-to* $P x$
 \langle proof \rangle

lemma *returns-to-outsideI*:
assumes $t \geq 0 t \in$ *existence-ivl0* x *flow0* $x t \in P$
assumes *ev*: $x \notin P$
assumes *closed* P

shows *returns-to* P x
<proof>

lemma *returns-toE*:

assumes *returns-to* P x

obtains $t0$ $t1$ **where**

$0 < t0$

$t0 \leq t1$

$t1 \in \text{existence-ivl0 } x$

$\text{flow0 } x \ t1 \in P$

$\bigwedge t. 0 < t \implies t < t0 \implies \text{flow0 } x \ t \notin P$

<proof>

lemma *return-time-some*:

assumes *returns-to* P x

shows *return-time* P $x =$

$(\text{SOME } t. t > 0 \wedge t \in \text{existence-ivl0 } x \wedge \text{flow0 } x \ t \in P \wedge (\forall s \in \{0 <..<t\}. \text{flow0 } x \ s \notin P))$

<proof>

lemma *return-time-ex1*:

assumes *returns-to* P x

assumes *closed* P

shows $\exists! t. t > 0 \wedge t \in \text{existence-ivl0 } x \wedge \text{flow0 } x \ t \in P \wedge (\forall s \in \{0 <..<t\}. \text{flow0 } x \ s \notin P)$

<proof>

lemma

return-time-pos-returns-to:

return-time P $x > 0 \implies \text{returns-to } P$ x

<proof>

lemma

assumes *ret*: *returns-to* P x

assumes *closed* P

shows *return-time-pos*: *return-time* P $x > 0$

<proof>

lemma *returns-to-return-time-pos*:

assumes *closed* P

shows *returns-to* P $x \longleftrightarrow \text{return-time } P$ $x > 0$

<proof>

lemma *return-time*:

assumes *ret*: *returns-to* P x

assumes *closed* P

shows *return-time* P $x > 0$

and *return-time-exivl*: *return-time* P $x \in \text{existence-ivl0 } x$

and *return-time-returns*: $\text{flow0 } x$ (*return-time* P x) $\in P$

and *return-time-least*: $\bigwedge s. 0 < s \implies s < \text{return-time } P \ x \implies \text{flow0 } x \ s \notin P$
(proof)

lemma *returns-to-earlierI*:
assumes *ret*: *returns-to* P (*flow0* x t) *closed* P
assumes $t \geq 0$ $t \in \text{existence-ivl0 } x$
assumes *ev*: $\forall_F t$ *in at-right* 0 . *flow0* x $t \notin P$
shows *returns-to* P x
(proof)

lemma *return-time-gt*:
assumes *ret*: *returns-to* P x *closed* P
assumes *flow-not*: $\bigwedge s. 0 < s \implies s \leq t \implies \text{flow0 } x \ s \notin P$
shows $t < \text{return-time } P \ x$
(proof)

lemma *return-time-le*:
assumes *ret*: *returns-to* P x *closed* P
assumes *flow-not*: *flow0* x $t \in P$ $t > 0$
shows *return-time* P $x \leq t$
(proof)

lemma *returns-to-laterI*:
assumes *ret*: *returns-to* P x *closed* P
assumes $t > 0$ $t \in \text{existence-ivl0 } x$
assumes *flow-not*: $\bigwedge s. 0 < s \implies s \leq t \implies \text{flow0 } x \ s \notin P$
shows *returns-to* P (*flow0* x t)
(proof)

lemma *never-returns*:
assumes $\neg \text{returns-to } P \ x$
assumes *closed* P $t \geq 0$ $t \in \text{existence-ivl0 } x$
assumes *ev*: $\forall_F t$ *in at-right* 0 . *flow0* x $t \notin P$
shows $\neg \text{returns-to } P$ (*flow0* x t)
(proof)

lemma *return-time-eqI*:
assumes *closed* P
and *t-pos*: $t > 0$
and *ex*: $t \in \text{existence-ivl0 } x$
and *ret*: *flow0* x $t \in P$
and *least*: $\bigwedge s. 0 < s \implies s < t \implies \text{flow0 } x \ s \notin P$
shows *return-time* P $x = t$
(proof)

lemma *return-time-step*:
assumes *returns-to* P (*flow0* x t)
assumes *closed* P
assumes *flow-not*: $\bigwedge s. 0 < s \implies s \leq t \implies \text{flow0 } x \ s \notin P$

assumes $t: t > 0 \ t \in \text{existence-ivl0 } x$
shows $\text{return-time } P (\text{flow0 } x \ t) = \text{return-time } P \ x - t$
 $\langle \text{proof} \rangle$

definition $\text{poincare-map } P \ x = \text{flow0 } x (\text{return-time } P \ x)$

lemma $\text{poincare-map-step-flow}$:
assumes $\text{ret}: \text{returns-to } P \ x \ \text{closed } P$
assumes $\text{flow-not}: \bigwedge s. \ 0 < s \implies s \leq t \implies \text{flow0 } x \ s \notin P$
assumes $t: t > 0 \ t \in \text{existence-ivl0 } x$
shows $\text{poincare-map } P (\text{flow0 } x \ t) = \text{poincare-map } P \ x$
 $\langle \text{proof} \rangle$

lemma $\text{poincare-map-returns}$:
assumes $\text{returns-to } P \ x \ \text{closed } P$
shows $\text{poincare-map } P \ x \in P$
 $\langle \text{proof} \rangle$

lemma poincare-map-onto :
assumes $\text{closed } P$
assumes $0 < t \ t \in \text{existence-ivl0 } x \ \forall_F \ t \ \text{in at-right } 0. \ \text{flow0 } x \ t \notin P$
assumes $\text{flow0 } x \ t \in P$
shows $\text{poincare-map } P \ x \in \text{flow0 } x \ ' \{0 <.. t\} \cap P$
 $\langle \text{proof} \rangle$

end

lemma isCont-blinfunD :
fixes $f': 'a::\text{metric-space} \Rightarrow 'b::\text{real-normed-vector} \Rightarrow_L 'c::\text{real-normed-vector}$
assumes $\text{isCont } f' \ a \ 0 < e$
shows $\exists d > 0. \ \forall x. \ \text{dist } a \ x < d \longrightarrow \text{onorm } (\lambda v. \ \text{blinfun-apply } (f' \ x) \ v) - \text{blinfun-apply } (f' \ a) \ v < e$
 $\langle \text{proof} \rangle$

proposition $\text{has-derivative-locally-injective-blinfun}$:
fixes $f :: 'n::\text{euclidean-space} \Rightarrow 'm::\text{euclidean-space}$
and $f': 'n \Rightarrow 'n \Rightarrow_L 'm$
and $g': 'm \Rightarrow_L 'n$
assumes $a \in s$
and $\text{open } s$
and $g': g' \ o_L (f' \ a) = 1_L$
and $f': \bigwedge x. \ x \in s \implies (f \ \text{has-derivative } f' \ x) \ (\text{at } x)$
and $c: \text{isCont } f' \ a$
obtains $r \ \text{where } r > 0 \ \text{ball } a \ r \subseteq s \ \text{inj-on } f \ (\text{ball } a \ r)$
 $\langle \text{proof} \rangle$

lift-definition $\text{embed1-blinfun}:: 'a::\text{real-normed-vector} \Rightarrow_L ('a * 'b::\text{real-normed-vector})$
is $\lambda x. (x, 0)$

<proof>

lemma *blinfun-apply-embed1-blinfun[simp]*: *blinfun-apply embed1-blinfun* $x = (x, 0)$

<proof>

lift-definition *embed2-blinfun*::*'a::real-normed-vector* \Rightarrow_L (*'b::real-normed-vector**'*a*)
is $\lambda x. (0, x)$

<proof>

lemma *blinfun-apply-embed2-blinfun[simp]*: *blinfun-apply embed2-blinfun* $x = (0, x)$

<proof>

lemma *blinfun-inverseD*: $f \circ_L f' = 1_L \implies f (f' x) = x$

<proof>

lemmas *continuous-on-open-vimageI* = *continuous-on-open-vimage[THEN iffD1, rule-format]*

lemmas *continuous-on-closed-vimageI* = *continuous-on-closed-vimage[THEN iffD1, rule-format]*

lemma *ball-times-subset*: $\text{ball } a \ (c/2) \times \text{ball } b \ (c/2) \subseteq \text{ball } (a, b) \ c$

<proof>

lemma *linear-inverse-blinop-lemma*:

fixes $w::'a::\{\text{banach, perfect-space}\}$ *blinop*

assumes $\text{norm } w < 1$

shows

summable $(\lambda n. (-1)^{\widehat{n}} *_R w^{\widehat{n}})$ (**is** ?*C*)

$(\sum n. (-1)^{\widehat{n}} *_R w^{\widehat{n}}) * (1 + w) = 1$ (**is** ?*I1*)

$(1 + w) * (\sum n. (-1)^{\widehat{n}} *_R w^{\widehat{n}}) = 1$ (**is** ?*I2*)

$\text{norm } ((\sum n. (-1)^{\widehat{n}} *_R w^{\widehat{n}}) - 1 + w) \leq (\text{norm } w)^2 / (1 - \text{norm } (w))$ (**is** ?*L*)

<proof>

lemma *linear-inverse-blinfun-lemma*:

fixes $w::'a \Rightarrow_L 'a::\{\text{banach, perfect-space}\}$

assumes $\text{norm } w < 1$

obtains *I* **where**

$I \circ_L (1_L + w) = 1_L (1_L + w) \circ_L I = 1_L$

$\text{norm } (I - 1_L + w) \leq (\text{norm } w)^2 / (1 - \text{norm } (w))$

<proof>

definition *invertibles-blinfun* = $\{w. \exists wi. w \circ_L wi = 1_L \wedge wi \circ_L w = 1_L\}$

lemma *blinfun-inverse-open*:— 8.3.2 in Dieudonne, TODO: add continuity and derivative

shows *open* (*invertibles-blinfun*::

$'a::\{\text{banach, perfect-space}\} \Rightarrow_L 'b::\text{banach}$) *set*)

<proof>

lemma *blinfun-compose-assoc*[*ac-simps*]: $a \circ_L b \circ_L c = a \circ_L (b \circ_L c)$
 ⟨*proof*⟩

TODO: move $norm (- ?x) = norm ?x$ to class!

lemma (in *real-normed-vector*) *norm-minus-cancel* [*simp*]: $norm (- x) = norm x$
 ⟨*proof*⟩

TODO: move $norm (?a - ?b) = norm (?b - ?a)$ to class!

lemma (in *real-normed-vector*) *norm-minus-commute*: $norm (a - b) = norm (b - a)$
 ⟨*proof*⟩

instance *euclidean-space* \subseteq *banach*
 ⟨*proof*⟩

lemma *blinfun-apply-Pair-split*:
 $blinfun-apply g (a, b) = blinfun-apply g (a, 0) + blinfun-apply g (0, b)$
 ⟨*proof*⟩

lemma *blinfun-apply-Pair-add2*: $blinfun-apply f (0, a + b) = blinfun-apply f (0, a) + blinfun-apply f (0, b)$
 ⟨*proof*⟩

lemma *blinfun-apply-Pair-add1*: $blinfun-apply f (a + b, 0) = blinfun-apply f (a, 0) + blinfun-apply f (b, 0)$
 ⟨*proof*⟩

lemma *blinfun-apply-Pair-minus2*: $blinfun-apply f (0, a - b) = blinfun-apply f (0, a) - blinfun-apply f (0, b)$
 ⟨*proof*⟩

lemma *blinfun-apply-Pair-minus1*: $blinfun-apply f (a - b, 0) = blinfun-apply f (a, 0) - blinfun-apply f (b, 0)$
 ⟨*proof*⟩

lemma *implicit-function-theorem*:

fixes $f::'a::euclidean-space * 'b::euclidean-space \Rightarrow 'c::euclidean-space$ — TODO: generalize?!

assumes [*derivative-intros*]: $\bigwedge x. x \in S \implies (f \text{ has-derivative } blinfun-apply (f' x))$
 (at x)

assumes $S: (x, y) \in S$ open S

assumes $DIM('c) \leq DIM('b)$

assumes $f'C: isCont f' (x, y)$

assumes $f (x, y) = 0$

assumes $T2: T \circ_L (f' (x, y) \circ_L embed2-blinfun) = 1_L$

assumes $T1: (f' (x, y) \circ_L embed2-blinfun) \circ_L T = 1_L$ — TODO: reduce?!

obtains $u \ e \ r$

where $f (x, u \ x) = 0 \ u \ x = y$

$\bigwedge s. s \in cball \ x \ e \implies f (s, u \ s) = 0$

continuous-on (*cball* x e) u
 $(\lambda t. (t, u t)) \text{ ' cball } x e \subseteq S$
 $e > 0$
 $(u \text{ has-derivative } - T \text{ o}_L f' (x, y) \text{ o}_L \text{ embed1-blinfun}) (at x)$

$r > 0$
 $\bigwedge U v s. v x = y \implies (\bigwedge s. s \in U \implies f (s, v s) = 0) \implies U \subseteq \text{cball } x e \implies$
 $\text{continuous-on } U v \implies s \in U \implies (s, v s) \in \text{ball } (x, y) r \implies u s = v s$
 $\langle \text{proof} \rangle$

lemma *implicit-function-theorem-unique*:

fixes $f::'a::\text{euclidean-space} * 'b::\text{euclidean-space} \Rightarrow 'c::\text{euclidean-space}$ — **TODO**:
 generalize?!

assumes f' [*derivative-intros*]: $\bigwedge x. x \in S \implies (f \text{ has-derivative blinfun-apply } (f' x)) (at x)$

assumes S : $(x, y) \in S$ open S

assumes D : $\text{DIM}('c) \leq \text{DIM}('b)$

assumes $f'C$: *continuous-on* $S f'$

assumes z : $f (x, y) = 0$

assumes $T2$: $T \text{ o}_L (f' (x, y) \text{ o}_L \text{ embed2-blinfun}) = 1_L$

assumes $T1$: $(f' (x, y) \text{ o}_L \text{ embed2-blinfun}) \text{ o}_L T = 1_L$ — **TODO**: reduce?!

obtains $u e$

where $f (x, u x) = 0$ $u x = y$

$\bigwedge s. s \in \text{cball } x e \implies f (s, u s) = 0$

continuous-on (*cball* x e) u

$(\lambda t. (t, u t)) \text{ ' cball } x e \subseteq S$

$e > 0$

$(u \text{ has-derivative } (- T \text{ o}_L f' (x, y) \text{ o}_L \text{ embed1-blinfun})) (at x)$

$\bigwedge s. s \in \text{cball } x e \implies f' (s, u s) \text{ o}_L \text{ embed2-blinfun} \in \text{invertibles-blinfun}$

$\bigwedge U v s. (\bigwedge s. s \in U \implies f (s, v s) = 0) \implies$

$u x = v x \implies$

continuous-on $U v \implies s \in U \implies x \in U \implies U \subseteq \text{cball } x e \implies \text{connected } U$

$\implies \text{open } U \implies u s = v s$

$\langle \text{proof} \rangle$

lemma *uniform-limit-compose*:

assumes ul : *uniform-limit* $T f l F$

assumes uc : *uniformly-continuous-on* $S s$

assumes ev : $\forall_F x$ in $F. f x \text{ ' } T \subseteq S$

assumes $subs$: $l \text{ ' } T \subseteq S$

shows *uniform-limit* $T (\lambda i x. s (f i x)) (\lambda x. s (l x)) F$

$\langle \text{proof} \rangle$

lemma

uniform-limit-in-open:

fixes $l::'a::\text{topological-space} \Rightarrow 'b::\text{heine-borel}$

assumes ul : *uniform-limit* $T f l$ (*at* x)

assumes $cont$: *continuous-on* $T l$

assumes $compact$: *compact* T **and** $T\text{-ne}$: $T \neq \{\}$

assumes B : open B
assumes mem : $l \in T \subseteq B$
shows $\forall_F y$ in at x . $\forall t \in T$. $f y t \in B$
 <proof>

lemma

order-uniform-limitD1:
fixes $l::'a::topological-space \Rightarrow real$ — TODO: generalize?!
assumes ul : uniform-limit $T f l$ (at x)
assumes $cont$: continuous-on $T l$
assumes $compact$: compact T
assumes $less$: $\bigwedge t. t \in T \Longrightarrow l t < b$
shows $\forall_F y$ in at x . $\forall t \in T$. $f y t < b$
 <proof>

lemma

order-uniform-limitD2:
fixes $l::'a::topological-space \Rightarrow real$ — TODO: generalize?!
assumes ul : uniform-limit $T f l$ (at x)
assumes $cont$: continuous-on $T l$
assumes $compact$: compact T
assumes $less$: $\bigwedge t. t \in T \Longrightarrow l t > b$
shows $\forall_F y$ in at x . $\forall t \in T$. $f y t > b$
 <proof>

lemma *continuous-on-avoid-cases*:

fixes $l::'b::topological-space \Rightarrow 'a::linear-continuum-topology$ — TODO: generalize!
assumes $cont$: continuous-on $T l$ **and** $conn$: connected T
assumes $avoid$: $\bigwedge t. t \in T \Longrightarrow l t \neq b$
obtains $\bigwedge t. t \in T \Longrightarrow l t < b \mid \bigwedge t. t \in T \Longrightarrow l t > b$
 <proof>

lemma

order-uniform-limit-ne:
fixes $l::'a::topological-space \Rightarrow real$ — TODO: generalize?!
assumes ul : uniform-limit $T f l$ (at x)
assumes $cont$: continuous-on $T l$
assumes $compact$: compact T **and** $conn$: connected T
assumes ne : $\bigwedge t. t \in T \Longrightarrow l t \neq b$
shows $\forall_F y$ in at x . $\forall t \in T$. $f y t \neq b$
 <proof>

lemma *open-cballE*:

assumes open S $x \in S$
obtains e where $e > 0$ $cball x e \subseteq S$
 <proof>

lemma *pos-half-less*: **fixes** $x::real$ **shows** $x > 0 \Longrightarrow x / 2 < x$
 <proof>

lemma *closed-levelset*: $\text{closed } \{x. s \ x = (c::'a::t1\text{-space})\}$ **if** *continuous-on UNIV s*
 ⟨*proof*⟩

lemma *closed-levelset-within*: $\text{closed } \{x \in S. s \ x = (c::'a::t1\text{-space})\}$ **if** *continuous-on S s closed S*
 ⟨*proof*⟩

context *c1-on-open-euclidean*
begin

lemma *open-existence-ivlE*:
assumes $t \in \text{existence-ivl0 } x \ t \geq 0$
obtains e **where** $e > 0 \ \text{cball } x \ e \times \{0 \ .. \ t + e\} \subseteq \text{Sigma } X \ \text{existence-ivl0}$
 ⟨*proof*⟩

lemmas [*derivative-intros*] = *flow0-comp-has-derivative*

lemma *flow-isCont-state-space-comp*[*continuous-intros*]:
 $t \ x \in \text{existence-ivl0 } (s \ x) \implies \text{isCont } s \ x \implies \text{isCont } t \ x \implies \text{isCont } (\lambda x. \text{flow0 } (s \ x) \ (t \ x)) \ x$
 ⟨*proof*⟩

lemma *closed-plane[simp]*: $\text{closed } \{x. x \cdot i = c\}$
 ⟨*proof*⟩

lemma *flow-tendsto-compose*[*tendsto-intros*]:
assumes $(x \longrightarrow xs) \ F \ (t \longrightarrow ts) \ F$
assumes $ts \in \text{existence-ivl0 } xs$
shows $((\lambda s. \text{flow0 } (x \ s) \ (t \ s)) \longrightarrow \text{flow0 } xs \ ts) \ F$
 ⟨*proof*⟩

lemma *returns-to-implicit-function*:
fixes $s::'a::\text{euclidean-space} \Rightarrow \text{real}$
assumes $rt: \text{returns-to } \{x \in S. s \ x = 0\} \ x \ (\text{is returns-to } ?P \ x)$
assumes $cS: \text{closed } S$
assumes $Ds: \bigwedge x. (s \ \text{has-derivative } \text{blinfun-apply } (Ds \ x)) \ (at \ x)$
assumes $DsC: \text{isCont } Ds \ (\text{poincare-map } ?P \ x)$
assumes $nz: Ds \ (\text{poincare-map } ?P \ x) \ (f \ (\text{poincare-map } ?P \ x)) \neq 0$
obtains $u \ e$
where $s \ (\text{flow0 } x \ (u \ x)) = 0$
 $u \ x = \text{return-time } ?P \ x$
 $(\bigwedge y. y \in \text{cball } x \ e \implies s \ (\text{flow0 } y \ (u \ y)) = 0)$
 $\text{continuous-on } (\text{cball } x \ e) \ u$
 $(\lambda t. (t, u \ t)) \ ' \ \text{cball } x \ e \subseteq \text{Sigma } X \ \text{existence-ivl0}$
 $0 < e \ (u \ \text{has-derivative } (- \ \text{blinfun-scaleR-left } (\text{inverse } (\text{blinfun-apply } (Ds \ (\text{poincare-map } ?P \ x)) \ (f \ (\text{poincare-map } ?P \ x)))) \ o_L$
 $(Ds \ (\text{poincare-map } ?P \ x) \ o_L \ \text{flowderiv } x \ (\text{return-time } ?P \ x)) \ o_L$

embed1-blinfun) (at x)
 ⟨proof⟩

lemma (in *auto-ll-on-open*) *f-tendsto*[*tendsto-intros*]:
 assumes $g1: (g1 \longrightarrow b1)$ (at s within S) and $b1 \in X$
 shows $((\lambda x. f (g1 x)) \longrightarrow f b1)$ (at s within S)
 ⟨proof⟩

lemma *flow-avoids-surface-eventually-at-right-pos*:
 assumes $s x > 0 \vee s x = 0 \wedge \text{blinfun-apply } (Ds x) (f x) > 0$
 assumes $x: x \in X$
 assumes $Ds: \bigwedge x. (s \text{ has-derivative } Ds x)$ (at x)
 assumes $DsC: \bigwedge x. \text{isCont } Ds x$
 shows $\forall_F t \text{ in at-right } 0. s (\text{flow0 } x t) > (0::\text{real})$
 ⟨proof⟩

lemma *flow-avoids-surface-eventually-at-right-neg*:
 assumes $s x < 0 \vee s x = 0 \wedge \text{blinfun-apply } (Ds x) (f x) < 0$
 assumes $x: x \in X$
 assumes $Ds: \bigwedge x. (s \text{ has-derivative } Ds x)$ (at x)
 assumes $DsC: \bigwedge x. \text{isCont } Ds x$
 shows $\forall_F t \text{ in at-right } 0. s (\text{flow0 } x t) < (0::\text{real})$
 ⟨proof⟩

lemma *flow-avoids-surface-eventually-at-right*:
 assumes $x \notin S \vee s x \neq 0 \vee \text{blinfun-apply } (Ds x) (f x) \neq 0$
 assumes $x: x \in X$ and $cS: \text{closed } S$
 assumes $Ds: \bigwedge x. (s \text{ has-derivative } Ds x)$ (at x)
 assumes $DsC: \bigwedge x. \text{isCont } Ds x$
 shows $\forall_F t \text{ in at-right } 0. (\text{flow0 } x t) \notin \{x \in S. s x = (0::\text{real})\}$
 ⟨proof⟩

lemma *eventually-returns-to*:
 fixes $s::'a::\text{euclidean-space} \Rightarrow \text{real}$
 assumes $rt: \text{returns-to } \{x \in S. s x = 0\} x$ (is *returns-to* ? P x)
 assumes $cS: \text{closed } S$
 assumes $Ds: \bigwedge x. (s \text{ has-derivative } \text{blinfun-apply } (Ds x))$ (at x)
 assumes $DsC: \bigwedge x. \text{isCont } Ds x$
 assumes *eventually-inside*: $\forall_F x \text{ in at } (\text{poincare-map } ?P x). s x = 0 \longrightarrow x \in S$
 assumes $nz: Ds (\text{poincare-map } ?P x) (f (\text{poincare-map } ?P x)) \neq 0$
 assumes $nz0: x \notin S \vee s x \neq 0 \vee Ds x (f x) \neq 0$
 shows $\forall_F x \text{ in at } x. \text{returns-to } ?P x$
 ⟨proof⟩

lemma
return-time-isCont-outside:
 fixes $s::'a::\text{euclidean-space} \Rightarrow \text{real}$
 assumes $rt: \text{returns-to } \{x \in S. s x = 0\} x$ (is *returns-to* ? P x)
 assumes $cS: \text{closed } S$

assumes Ds : $\bigwedge x. (s \text{ has-derivative } \text{blinfun-apply } (Ds \ x)) \ (at \ x)$
assumes DsC : $\bigwedge x. \text{isCont } Ds \ x$
assumes *through*: $(Ds \ (\text{poincare-map } ?P \ x)) \ (f \ (\text{poincare-map } ?P \ x)) \neq 0$
assumes *eventually-inside*: $\forall_F \ x \ \text{in} \ \text{at} \ (\text{poincare-map } ?P \ x). \ s \ x = 0 \longrightarrow x \in S$
assumes *outside*: $x \notin S \vee s \ x \neq 0$
shows $\text{isCont} \ (\text{return-time } ?P) \ x$
<proof>

lemma *isCont-poincare-map*:
assumes $\text{isCont} \ (\text{return-time } P) \ x$
returns-to $P \ x \ \text{closed } P$
shows $\text{isCont} \ (\text{poincare-map } P) \ x$
<proof>

lemma *poincare-map-tendsto*:
assumes $(\text{return-time } P \longrightarrow \text{return-time } P \ x) \ (\text{at } x \ \text{within } S)$
returns-to $P \ x \ \text{closed } P$
shows $(\text{poincare-map } P \longrightarrow \text{poincare-map } P \ x) \ (\text{at } x \ \text{within } S)$
<proof>

lemma
return-time-continuous-below:
fixes $s::'a::\text{euclidean-space} \Rightarrow \text{real}$
assumes rt : *returns-to* $\{x \in S. \ s \ x = 0\} \ x \ (\text{is } \text{returns-to } ?P \ x)$
assumes Ds : $\bigwedge x. (s \ \text{has-derivative } \text{blinfun-apply } (Ds \ x)) \ (\text{at } x)$
assumes cS : *closed* S
assumes *eventually-inside*: $\forall_F \ x \ \text{in} \ \text{at} \ (\text{poincare-map } ?P \ x). \ s \ x = 0 \longrightarrow x \in S$
assumes DsC : $\bigwedge x. \text{isCont } Ds \ x$
assumes *through*: $(Ds \ (\text{poincare-map } ?P \ x)) \ (f \ (\text{poincare-map } ?P \ x)) \neq 0$
assumes *inside*: $x \in S \ s \ x = 0 \ Ds \ x \ (f \ x) < 0$
shows *continuous* $(\text{at } x \ \text{within } \{x. \ s \ x \leq 0\}) \ (\text{return-time } ?P)$
<proof>

lemma
return-time-continuous-below-plane:
fixes $s::'a::\text{euclidean-space} \Rightarrow \text{real}$
assumes rt : *returns-to* $\{x \in R. \ x \cdot n = c\} \ x \ (\text{is } \text{returns-to } ?P \ x)$
assumes cR : *closed* R
assumes *through*: $f \ (\text{poincare-map } ?P \ x) \cdot n \neq 0$
assumes R : $x \in R$
assumes *inside*: $x \cdot n = c \ f \ x \cdot n < 0$
assumes *eventually-inside*: $\forall_F \ x \ \text{in} \ \text{at} \ (\text{poincare-map } ?P \ x). \ x \cdot n = c \longrightarrow x \in R$
shows *continuous* $(\text{at } x \ \text{within } \{x. \ x \cdot n \leq c\}) \ (\text{return-time } ?P)$
<proof>

lemma
poincare-map-in-interior-eventually-return-time-equal:
assumes RP : $R \subseteq P$

assumes cP : closed P
assumes cR : closed R
assumes ret : returns-to P x
assumes $evret$: $\forall_F x$ in at x within S . returns-to P x
assumes evR : $\forall_F x$ in at x within S . poincare-map P $x \in R$
shows $\forall_F x$ in at x within S . returns-to R $x \wedge$ return-time P $x =$ return-time R
 x
 <proof>

lemma *poincare-map-in-planeI*:
assumes returns-to (plane n c) $x0$
shows poincare-map (plane n c) $x0 \cdot n = c$
 <proof>

lemma *less-return-time-imp-exivl*:
 $h \in$ existence-ivl0 x' **if** $h \leq$ return-time P x' returns-to P x' closed P $0 \leq h$
 <proof>

lemma *eventually-returns-to-continuousI*:
assumes returns-to P x
assumes closed P
assumes continuous (at x within S) (return-time P)
shows $\forall_F x$ in at x within S . returns-to P x
 <proof>

lemma *return-time-implicit-functionE*:
fixes $s::'a::euclidean-space \Rightarrow real$
assumes rt : returns-to $\{x \in S. s\ x = 0\}$ x (**is** returns-to $?P$ -)
assumes cS : closed S
assumes Ds : $\bigwedge x$. (s has-derivative blinfun-apply (Ds x)) (at x)
assumes DsC : $\bigwedge x$. isCont Ds x
assumes Ds -through: (Ds (poincare-map $?P$ x)) (f (poincare-map $?P$ x)) $\neq 0$
assumes eventually-inside: $\forall_F x$ in at (poincare-map $?P$ x). $s\ x = 0 \longrightarrow x \in S$
assumes outside: $x \notin S \vee s\ x \neq 0$
obtains e' **where**
 $0 < e'$
 $\bigwedge y$. $y \in$ ball x $e' \Longrightarrow$ returns-to $?P$ y
 $\bigwedge y$. $y \in$ ball x $e' \Longrightarrow s$ (flow0 y (return-time $?P$ y)) = 0
 continuous-on (ball x e') (return-time $?P$)
 $(\bigwedge y$. $y \in$ ball x $e' \Longrightarrow Ds$ (poincare-map $?P$ y) o_L flowderiv y (return-time $?P$ y)) o_L embed2-blinfun \in invertibles-blinfun)
 $(\bigwedge U$ v sa .
 $(\bigwedge sa$. $sa \in U \Longrightarrow s$ (flow0 sa (v sa)) = 0) \Longrightarrow
 return-time $?P$ $x = v$ $x \Longrightarrow$
 continuous-on U $v \Longrightarrow sa \in U \Longrightarrow x \in U \Longrightarrow U \subseteq$ ball x $e' \Longrightarrow$ connected
 $U \Longrightarrow$ open $U \Longrightarrow$ return-time $?P$ $sa = v$ sa)
 (return-time $?P$ has-derivative
 - blinfun-scaleR-left (inverse ((Ds (poincare-map $?P$ x)) (f (poincare-map
 $?P$ x)))) o_L

$(Ds \text{ (poincare-map } ?P \ x) \ o_L \ Dflow \ x \ \text{(return-time } ?P \ x)))$
 $(\text{at } x)$

$\langle \text{proof} \rangle$

lemma *return-time-has-derivative:*
fixes $s::'a::\text{euclidean-space} \Rightarrow \text{real}$
assumes rt : *returns-to* $\{x \in S. s \ x = 0\} \ x$ (**is** *returns-to* $?P \ -$)
assumes cS : *closed* S
assumes Ds : $\bigwedge x. (s \ \text{has-derivative} \ \text{blinfun-apply} \ (Ds \ x)) \ (\text{at } x)$
assumes DsC : $\bigwedge x. \ \text{isCont} \ Ds \ x$
assumes $Ds\text{-through}$: $(Ds \ \text{(poincare-map } ?P \ x)) \ (f \ \text{(poincare-map } ?P \ x)) \neq 0$
assumes eventually-inside : $\forall_F \ x \ \text{in} \ \text{at} \ (\text{poincare-map} \ \{x \in S. \ s \ x = 0\} \ x). \ s \ x = 0 \longrightarrow x \in S$
assumes outside : $x \notin S \vee s \ x \neq 0$
shows *return-time* $?P$ *has-derivative*
 $- \ \text{blinfun-scaleR-left} \ (\text{inverse} \ ((Ds \ \text{(poincare-map } ?P \ x)) \ (f \ \text{(poincare-map } ?P \ x)))) \ o_L$
 $(Ds \ \text{(poincare-map } ?P \ x) \ o_L \ Dflow \ x \ \text{(return-time } ?P \ x)))$
 $(\text{at } x)$
 $\langle \text{proof} \rangle$

lemma *return-time-plane-has-derivative-blinfun:*
assumes rt : *returns-to* $\{x \in S. x \cdot i = c\} \ x$ (**is** *returns-to* $?P \ -$)
assumes cS : *closed* S
assumes fnz : $f \ \text{(poincare-map } ?P \ x) \cdot i \neq 0$
assumes eventually-inside : $\forall_F \ x \ \text{in} \ \text{at} \ (\text{poincare-map } ?P \ x). \ x \cdot i = c \longrightarrow x \in S$
assumes outside : $x \notin S \vee x \cdot i \neq c$
shows *return-time* $?P$ *has-derivative*
 $(- \ \text{blinfun-scaleR-left} \ (\text{inverse} \ ((\text{blinfun-inner-left } i) \ (f \ \text{(poincare-map } ?P \ x)))))) \ o_L$
 $(\text{blinfun-inner-left } i \ o_L \ Dflow \ x \ \text{(return-time } ?P \ x))) \ (\text{at } x)$
 $\langle \text{proof} \rangle$

lemma *return-time-plane-has-derivative:*
assumes rt : *returns-to* $\{x \in S. x \cdot i = c\} \ x$ (**is** *returns-to* $?P \ -$)
assumes cS : *closed* S
assumes fnz : $f \ \text{(poincare-map } ?P \ x) \cdot i \neq 0$
assumes eventually-inside : $\forall_F \ x \ \text{in} \ \text{at} \ (\text{poincare-map } ?P \ x). \ x \cdot i = c \longrightarrow x \in S$
assumes outside : $x \notin S \vee x \cdot i \neq c$
shows *return-time* $?P$ *has-derivative*
 $(\lambda h. \ - \ (Dflow \ x \ \text{(return-time } ?P \ x)) \ h \cdot i / (f \ \text{(poincare-map } ?P \ x) \cdot i)) \ (\text{at } x)$
 $\langle \text{proof} \rangle$

definition $D\text{poincare-map } i \ c \ S \ x =$
 $(\lambda h. \ (Dflow \ x \ \text{(return-time } \{x \in S. \ x \cdot i = c\} \ x)) \ h \ -$
 $((Dflow \ x \ \text{(return-time } \{x \in S. \ x \cdot i = c\} \ x)) \ h \cdot i /$
 $(f \ \text{(poincare-map } \{x \in S. \ x \cdot i = c\} \ x) \cdot i)) \ *_R \ f \ \text{(poincare-map } \{x \in S. \ x$
 $\cdot i = c\} \ x))$

definition $D\text{poincare-map}' i c S x =$
 $D\text{flow } x \text{ (return-time } \{x \in S. x \cdot i - c = 0\} x) -$
 $(\text{blinfun-scaleR-left } (f \text{ (poincare-map } \{x \in S. x \cdot i = c\} x)) \text{ } o_L$
 $(\text{blinfun-scaleR-left } (\text{inverse } ((f \text{ (poincare-map } \{x \in S. x \cdot i = c\} x) \cdot i)))) \text{ } o_L$
 $(\text{blinfun-inner-left } i \text{ } o_L \text{ } D\text{flow } x \text{ (return-time } \{x \in S. x \cdot i - c = 0\} x))))$

theorem *poincare-map-plane-has-derivative:*

assumes *rt*: *returns-to* $\{x \in S. x \cdot i = c\} x$ (**is** *returns-to* $?P$ -)

assumes *cS*: *closed* S

assumes *fnz*: $f \text{ (poincare-map } ?P x) \cdot i \neq 0$

assumes *eventually-inside*: $\forall_F x \text{ in at } (\text{poincare-map } ?P x). x \cdot i = c \longrightarrow x \in S$

assumes *outside*: $x \notin S \vee x \cdot i \neq c$

notes [*derivative-intros*] = *return-time-plane-has-derivative*[*OF* *rt* *cS* *fnz* *eventually-inside* *outside*]

shows (*poincare-map* $?P$ *has-derivative* $D\text{poincare-map}' i c S x$) (*at* x)

<proof>

end

end

theory *Reachability-Analysis*

imports

Flow

Poincare-Map

begin

lemma *not-mem-eq-mem-not*: $a \notin A \longleftrightarrow a \in - A$

<proof>

lemma *continuous-orderD*:

fixes $g :: 'b :: t2\text{-space} \Rightarrow 'c :: \text{order-topology}$

assumes *continuous* (*at* x *within* S) g

shows $g x > c \Longrightarrow \forall_F y \text{ in at } x \text{ within } S. g y > c$

$g x < c \Longrightarrow \forall_F y \text{ in at } x \text{ within } S. g y < c$

<proof>

lemma *frontier-halfspace-component-ge*: $n \neq 0 \Longrightarrow \text{frontier } \{x. c \leq x \cdot n\} = \text{plane}$

$n \ c$

<proof>

lemma *closed-Collect-le-within*:

fixes $f g :: 'a :: \text{topological-space} \Rightarrow 'b :: \text{linorder-topology}$

assumes *f*: *continuous-on* $UNIV$ f

and *g*: *continuous-on* $UNIV$ g

and *closed* R

shows *closed* $\{x \in R. f x \leq g x\}$

<proof>

6.1 explicit representation of hyperplanes / halfspaces

datatype 'a sctn = Sctn (normal: 'a) (pstn: real)

definition le-halfspace sctn $x \longleftrightarrow x \cdot \text{normal sctn} \leq \text{pstn sctn}$

definition lt-halfspace sctn $x \longleftrightarrow x \cdot \text{normal sctn} < \text{pstn sctn}$

definition ge-halfspace sctn $x \longleftrightarrow x \cdot \text{normal sctn} \geq \text{pstn sctn}$

definition gt-halfspace sctn $x \longleftrightarrow x \cdot \text{normal sctn} > \text{pstn sctn}$

definition plane-of sctn = $\{x. x \cdot \text{normal sctn} = \text{pstn sctn}\}$

definition above-halfspace sctn = Collect (ge-halfspace sctn)

definition below-halfspace sctn = Collect (le-halfspace sctn)

definition sbelow-halfspace sctn = Collect (lt-halfspace sctn)

definition sabove-halfspace sctn = Collect (gt-halfspace sctn)

6.2 explicit H representation of polytopes (mind Polytopes.thy)

definition below-halfspaces

where below-halfspaces sctns = $\bigcap (\text{below-halfspace } ' \text{ sctns})$

definition sbelow-halfspaces

where sbelow-halfspaces sctns = $\bigcap (\text{sbelow-halfspace } ' \text{ sctns})$

definition above-halfspaces

where above-halfspaces sctns = $\bigcap (\text{above-halfspace } ' \text{ sctns})$

definition sabove-halfspaces

where sabove-halfspaces sctns = $\bigcap (\text{sabove-halfspace } ' \text{ sctns})$

lemmas halfspace-simps =

above-halfspace-def

sabove-halfspace-def

below-halfspace-def

sbelow-halfspace-def

below-halfspaces-def

sbelow-halfspaces-def

above-halfspaces-def

sabove-halfspaces-def

ge-halfspace-def[abs-def]

gt-halfspace-def[abs-def]

le-halfspace-def[abs-def]

lt-halfspace-def[abs-def]

6.3 predicates for reachability analysis

context *c1-on-open-euclidean*

begin

definition *flowpipe* ::

$(('a::\text{euclidean-space}) \times ('a \Rightarrow_L 'a)) \text{ set} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow$
 $('a \times ('a \Rightarrow_L 'a)) \text{ set} \Rightarrow ('a \times ('a \Rightarrow_L 'a)) \text{ set} \Rightarrow \text{bool}$

where *flowpipe* $X0\ hl\ hu\ CX\ X1 \longleftrightarrow 0 \leq hl \wedge hl \leq hu \wedge \text{fst } 'X0 \subseteq X \wedge \text{fst } 'CX \subseteq X \wedge \text{fst } 'X1 \subseteq X \wedge$

$(\forall (x0, d0) \in X0. \forall h \in \{hl .. hu\}.$

$h \in \text{existence-ivl0 } x0 \wedge (\text{flow0 } x0\ h, \text{Dflow } x0\ h\ o_L\ d0) \in X1 \wedge (\forall h' \in \{0 .. h\}. (\text{flow0 } x0\ h', \text{Dflow } x0\ h'\ o_L\ d0) \in CX))$

lemma *flowpipeD*:

assumes *flowpipe* $X0\ hl\ hu\ CX\ X1$

shows *flowpipe-safeD*: $\text{fst } 'X0 \cup \text{fst } 'CX \cup \text{fst } 'X1 \subseteq X$

and *flowpipe-nonneg*: $0 \leq hl \wedge hl \leq hu$

and *flowpipe-exivl*: $hl \leq h \Longrightarrow h \leq hu \Longrightarrow (x0, d0) \in X0 \Longrightarrow h \in \text{existence-ivl0 } x0$

and *flowpipe-discrete*: $hl \leq h \Longrightarrow h \leq hu \Longrightarrow (x0, d0) \in X0 \Longrightarrow (\text{flow0 } x0\ h, \text{Dflow } x0\ h\ o_L\ d0) \in X1$

and *flowpipe-cont*: $hl \leq h \Longrightarrow h \leq hu \Longrightarrow (x0, d0) \in X0 \Longrightarrow 0 \leq h' \Longrightarrow h' \leq h \Longrightarrow (\text{flow0 } x0\ h', \text{Dflow } x0\ h'\ o_L\ d0) \in CX$

<proof>

lemma *flowpipe-source-subset*: *flowpipe* $X0\ hl\ hu\ CX\ X1 \Longrightarrow X0 \subseteq CX$

<proof>

definition *flowsto* $X0\ T\ CX\ X1 \longleftrightarrow$

$(\forall (x0, d0) \in X0. \exists h \in T. h \in \text{existence-ivl0 } x0 \wedge (\text{flow0 } x0\ h, \text{Dflow } x0\ h\ o_L\ d0) \in X1 \wedge (\forall h' \in \text{open-segment } 0\ h. (\text{flow0 } x0\ h', \text{Dflow } x0\ h'\ o_L\ d0) \in CX))$

lemma *flowsto-to-empty-iff[simp]*: *flowsto* $a\ t\ b\ \{\}$ $\longleftrightarrow a = \{\}$

<proof>

lemma *flowsto-from-empty-iff[simp]*: *flowsto* $\{\}\ t\ b\ c$

<proof>

lemma *flowsto-empty-time-iff[simp]*: *flowsto* $a\ \{\}\ b\ c \longleftrightarrow a = \{\}$

<proof>

lemma *flowstoE*:

assumes *flowsto* $X0\ T\ CX\ X1$ $(x0, d0) \in X0$

obtains h **where** $h \in T$ $h \in \text{existence-ivl0 } x0$ $(\text{flow0 } x0\ h, \text{Dflow } x0\ h\ o_L\ d0) \in X1$

$\wedge h'. h' \in \text{open-segment } 0\ h \Longrightarrow (\text{flow0 } x0\ h', \text{Dflow } x0\ h'\ o_L\ d0) \in CX$

<proof>

lemma *flowsto-safeD*: *flowsto* $X0\ T\ CX\ X1 \Longrightarrow \text{fst } 'X0 \subseteq X$

<proof>

lemma *flowsto-union*:

assumes 1: *flowsto* $X0\ T\ CX\ Y$ **and** 2: *flowsto* $Z\ S\ CZ\ W$

shows *flowsto* $(X0 \cup Z)\ (T \cup S)\ (CX \cup CZ)\ (Y \cup W)$

<proof>

lemma *flowsto-subset*:

assumes *flowsto* $X0\ T\ CX\ Y$

assumes $Z \subseteq X0\ T \subseteq S\ CX \subseteq CZ\ Y \subseteq W$

shows *flowsto* $Z\ S\ CZ\ W$

<proof>

lemmas *flowsto-unionI* = *flowsto-subset*[*OF flowsto-union*]

lemma *flowsto-unionE*:

assumes *flowsto* $X0\ T\ CX\ (Y \cup Z)$

obtains $X1\ X2$ **where** $X0 = X1 \cup X2$ *flowsto* $X1\ T\ CX\ Y$ *flowsto* $X2\ T\ CX\ Z$

<proof>

lemma *flowsto-trans*:

assumes A : *flowsto* $A\ S\ B\ C$ **and** C : *flowsto* $C\ T\ D\ E$

shows *flowsto* $A\ \{s + t \mid s \in S \wedge t \in T\}\ (B \cup D \cup C)\ E$

<proof>

lemma *flowsto-step*:

assumes A : *flowsto* $A\ S\ B\ C$

assumes D : *flowsto* $D\ T\ E\ F$

shows *flowsto* $A\ (S \cup \{s + t \mid s \in S \wedge t \in T\})\ (B \cup E \cup C \cap D)\ (C - D \cup F)$

<proof>

lemma

flowsto-stepI:

flowsto $X0\ U\ B\ C \implies$

flowsto $D\ T\ E\ F \implies$

$Z \subseteq X0 \implies$

$(\bigwedge s. s \in U \implies s \in S) \implies$

$(\bigwedge s\ t. s \in U \implies t \in T \implies s + t \in S) \implies$

$B \cup E \cup D \cap C \subseteq CZ \implies C - D \cup F \subseteq W \implies$ *flowsto* $Z\ S\ CZ\ W$

<proof>

lemma *flowsto-imp-flowsto*:

flowpipe $Y\ h\ h\ CY\ Z \implies$ *flowsto* $Y\ \{h\}\ (CY)\ Z$

<proof>

lemma *connected-below-halfspace*:

assumes $x \in$ *below-halfspace* *sctn*

assumes $x \in S$ *connected* S

assumes $S \cap \text{plane-of sctn} = \{\}$
shows $S \subseteq \text{below-halfspace sctn}$
 ⟨proof⟩

lemma

inter-Collect-eq-empty:

assumes $\bigwedge x. x \in X0 \implies \neg g x$ **shows** $X0 \cap \text{Collect } g = \{\}$
 ⟨proof⟩

6.4 Poincare Map

lemma *closed-plane-of[simp]: closed (plane-of sctn)*
 ⟨proof⟩

definition *poincare-mapsto* $P X0 S CX Y \longleftrightarrow (\forall (x, d) \in X0.$
returns-to $P x \wedge \text{fst } ' X0 \subseteq S \wedge$
return-time P *differentiable at x within S*) \wedge
 $(\exists D. (\text{poincare-map } P \text{ has-derivative } \text{blinfun-apply } D) (\text{at } x \text{ within } S) \wedge$
 $(\text{poincare-map } P x, D \text{ o}_L d) \in Y) \wedge$
 $(\forall t \in \{0 <.. < \text{return-time } P x\}. \text{flow0 } x t \in CX))$

lemma *poincare-mapsto-empty[simp]:*
poincare-mapsto $P \{\} S CX Y$
 ⟨proof⟩

lemma *flowsto-eventually-mem-cont:*
assumes *flowsto* $X0 T CX Y (x, d) \in X0 T \subseteq \{0 <..\}$
shows $\forall_F t$ *in at-right 0.* $(\text{flow0 } x t, D\text{flow } x t \text{ o}_L d) \in CX$
 ⟨proof⟩

lemma *frontier-aux-lemma:*

fixes $R :: 'n::\text{euclidean-space set}$
assumes *closed* $R R \subseteq \{x. x \cdot n = c\}$ **and** *[simp]:* $n \neq 0$
shows *frontier* $\{x \in R. c \leq x \cdot n\} = \{x \in R. c = x \cdot n\}$
 ⟨proof⟩

lemma *blinfun-minus-comp-distrib:* $(a - b) \text{ o}_L c = (a \text{ o}_L c) - (b \text{ o}_L c)$
 ⟨proof⟩

lemma *flowpipe-split-at-above-halfspace:*

assumes *flowpipe* $X0 \text{ hl } t CX Y \text{ fst } ' X0 \cap \{x. x \cdot n \geq c\} = \{\}$ **and** *[simp]:* $n \neq 0$
assumes *cR:* *closed* R **and** *Rs:* $R \subseteq \text{plane } n c$
assumes *PDP:* $\bigwedge x d. (x, d) \in CX \implies x \cdot n = c \implies (x,$
 $d - (\text{blinfun-scaleR-left } (f (x)) \text{ o}_L (\text{blinfun-scaleR-left } (\text{inverse } (f x \cdot n)) \text{ o}_L$
 $(\text{blinfun-inner-left } n \text{ o}_L d)))) \in \text{PDP}$
assumes *PDP-nz:* $\bigwedge x d. (x, d) \in \text{PDP} \implies f x \cdot n \neq 0$
assumes *PDP-inR:* $\bigwedge x d. (x, d) \in \text{PDP} \implies x \in R$

assumes *PDP-in*: $\bigwedge x d. (x, d) \in PDP \implies \forall_F x \text{ in at } x \text{ within plane } n \ c. x \in R$
obtains $X1 \ X2$ **where** $X0 = X1 \cup X2$
flowsto $X1 \ \{0 <..t\} \ (CX \cap \{x. x \cdot n < c\} \times UNIV) \ (CX \cap \{x \in R. x \cdot n = c\} \times UNIV)$
flowsto $X2 \ \{hl .. t\} \ (CX \cap \{x. x \cdot n < c\} \times UNIV) \ (Y \cap (\{x. x \cdot n < c\} \times UNIV))$
poincare-mapsto $\{x \in R. x \cdot n = c\} \ X1 \ UNIV \ (fst \ ' \ CX \cap \{x. x \cdot n < c\}) \ PDP$
<proof>

lemma *poincare-map-has-derivative-step*:

assumes *Deriv*: (*poincare-map* P *has-derivative* *blinfun-apply* D) (at (*flow0* $x0$ h))
assumes *ret*: *returns-to* $P \ x0$
assumes *cont*: *continuous* (at $x0$ *within* S) (*return-time* P)
assumes *less*: $0 \leq h \ h < \text{return-time } P \ x0$
assumes *cP*: *closed* P **and** $x0: x0 \in S$
shows ($\lambda x. \text{poincare-map } P \ x$) *has-derivative* ($D \ o_L \ D\text{flow } x0 \ h$) (at $x0$ *within* S)
<proof>

lemma *poincare-mapsto-trans*:

assumes *poincare-mapsto* $p1 \ X0 \ S \ \ \ CX \ P1$
assumes *poincare-mapsto* $p2 \ P1 \ UNIV \ CY \ P2$
assumes $CX \cup CY \cup \text{fst } ' \ P1 \subseteq CZ$
assumes $p2 \cap (CX \cup \text{fst } ' \ P1) = \{\}$
assumes [*intro*, *simp*]: *closed* $p1$
assumes [*intro*, *simp*]: *closed* $p2$
assumes *cont*: $\bigwedge x d. (x, d) \in X0 \implies \text{continuous} \ (at \ x \ \text{within } S) \ (\text{return-time } p2)$
shows *poincare-mapsto* $p2 \ X0 \ S \ CZ \ P2$
<proof>

lemma *flowsto-poincare-trans*:— TODO: the proof is close to $\llbracket \text{poincare-mapsto } ?p1.0 \ ?X0.0 \ ?S \ ?CX \ ?P1.0; \text{poincare-mapsto } ?p2.0 \ ?P1.0 \ UNIV \ ?CY \ ?P2.0; \ ?CX \cup \ ?CY \cup \text{fst } ' \ ?P1.0 \subseteq \ ?CZ; \ ?p2.0 \cap (\ ?CX \cup \text{fst } ' \ ?P1.0) = \{\}; \ \text{closed } ?p1.0; \ \text{closed } ?p2.0; \ \bigwedge x d. (x, d) \in \ ?X0.0 \implies \text{continuous} \ (at \ x \ \text{within } \ ?S) \ (\text{return-time } ?p2.0) \rrbracket \implies \text{poincare-mapsto } ?p2.0 \ ?X0.0 \ ?S \ ?CZ \ ?P2.0$

assumes *f*: *flowsto* $X0 \ T \ \ \ CX \ P1$
assumes *poincare-mapsto* $p2 \ P1 \ UNIV \ CY \ P2$
assumes *nn*: $\bigwedge t. t \in T \implies t \geq 0$
assumes $\text{fst } ' \ CX \cup CY \cup \text{fst } ' \ P1 \subseteq CZ$
assumes $p2 \cap (\text{fst } ' \ CX \cup \text{fst } ' \ P1) = \{\}$
assumes [*intro*, *simp*]: *closed* $p2$
assumes *cont*: $\bigwedge x d. (x, d) \in X0 \implies \text{continuous} \ (at \ x \ \text{within } S) \ (\text{return-time } p2)$
assumes *subset*: $\text{fst } ' \ X0 \subseteq S$
shows *poincare-mapsto* $p2 \ X0 \ S \ CZ \ P2$
<proof>

6.5 conditions for continuous return time

definition *section s Ds S* \longleftrightarrow
 $(\forall x. (s \text{ has-derivative blinfun-apply } (Ds x)) (at x)) \wedge$
 $(\forall x. isCont Ds x) \wedge$
 $(\forall x \in S. s x = (0::real) \longrightarrow Ds x (f x) \neq 0) \wedge$
 $closed S \wedge S \subseteq X$

lemma *sectionD*:

assumes *section s Ds S*
shows $(s \text{ has-derivative blinfun-apply } (Ds x)) (at x)$
 $isCont Ds x$
 $x \in S \implies s x = 0 \implies Ds x (f x) \neq 0$
 $closed S S \subseteq X$
 $\langle proof \rangle$

definition *transversal p* $\longleftrightarrow (\forall x \in p. \forall_F t \text{ in at-right } 0. flow0 x t \notin p)$

lemma *transversalD*: $transversal p \implies x \in p \implies \forall_F t \text{ in at-right } 0. flow0 x t \notin p$
 $\langle proof \rangle$

lemma *transversal-section*:

fixes $c::real$
assumes *section s Ds S*
shows $transversal \{x \in S. s x = 0\}$
 $\langle proof \rangle$

lemma *section-closed[intro, simp]*: $section s Ds S \implies closed \{x \in S. s x = 0\}$
 $\langle proof \rangle$

lemma *return-time-continuous-belowI*:

assumes *ft: flowsto X0 T CX X1*
assumes *pos*: $\bigwedge t. t \in T \implies t > 0$
assumes *X0*: $fst \text{ ' } X0 \subseteq \{x \in S. s x = 0\}$
assumes *CX*: $fst \text{ ' } CX \cap \{x \in S. s x = 0\} = \{\}$
assumes *X1*: $fst \text{ ' } X1 \subseteq \{x \in S. s x = 0\}$
assumes *sec*: *section s Ds S*
assumes *nz*: $\bigwedge x. x \in S \implies s x = 0 \implies Ds x (f x) \neq 0$
assumes *Dneg*: $(\lambda x. (Ds x) (f x)) \text{ ' } fst \text{ ' } X0 \subseteq \{..<0\}$
assumes *rel-int*: $\bigwedge x. x \in fst \text{ ' } X1 \implies \forall_F x \text{ in at } x. s x = 0 \longrightarrow x \in S$
assumes $(x, d) \in X0$
shows *continuous (at x within {x. s x ≤ 0}) (return-time {x ∈ S. s x = 0})*
 $\langle proof \rangle$

end

end

theory *Flow-Congs*

imports *Reachability-Analysis*
begin

lemma *lipschitz-on-congI*:
 assumes L' -lipschitz-on s' g'
 assumes $s' = s$
 assumes $L' \leq L$
 assumes $\bigwedge x y. x \in s \implies g' x = g x$
 shows L -lipschitz-on s g
 \langle *proof* \rangle

lemma *local-lipschitz-congI*:
 assumes *local-lipschitz* s' t' g'
 assumes $s' = s$
 assumes $t' = t$
 assumes $\bigwedge x y. x \in s \implies y \in t \implies g' x y = g x y$
 shows *local-lipschitz* s t g
 \langle *proof* \rangle

context *ll-on-open-it*— TODO: do this more generically for *ll-on-open-it*
begin

context **fixes** S Y g **assumes** *cong*: $X = Y$ $T = S$ $\bigwedge x t. x \in Y \implies t \in S \implies f$
 $t x = g t x$
begin

lemma *ll-on-open-congI*: *ll-on-open* S g Y
 \langle *proof* \rangle

lemma *existence-ivl-subsetI*:
 assumes $t: t \in$ *existence-ivl* $t0$ $x0$
 shows $t \in$ *ll-on-open.existence-ivl* S g Y $t0$ $x0$
 \langle *proof* \rangle

lemma *existence-ivl-cong*:
 shows *existence-ivl* $t0$ $x0 =$ *ll-on-open.existence-ivl* S g Y $t0$ $x0$
 \langle *proof* \rangle

lemma *flow-cong*:
 assumes $t \in$ *existence-ivl* $t0$ $x0$
 shows *flow* $t0$ $x0$ $t =$ *ll-on-open.flow* S g Y $t0$ $x0$ t
 \langle *proof* \rangle

end

end

context *auto-ll-on-open* **begin**

context fixes $Y g$ **assumes** $\text{cong}: X = Y \wedge x t. x \in Y \implies f x = g x$
begin

lemma $\text{auto-ll-on-open-congI}$: $\text{auto-ll-on-open } g Y$
 $\langle \text{proof} \rangle$

lemma $\text{existence-ivl0-cong}$:
shows $\text{existence-ivl0 } x0 = \text{auto-ll-on-open.existence-ivl0 } g Y x0$
 $\langle \text{proof} \rangle$

lemma flow0-cong :
assumes $t \in \text{existence-ivl0 } x0$
shows $\text{flow0 } x0 t = \text{auto-ll-on-open.flow0 } g Y x0 t$
 $\langle \text{proof} \rangle$

end

end

context $\text{c1-on-open-euclidean}$ **begin**

context fixes $Y g$ **assumes** $\text{cong}: X = Y \wedge x t. x \in Y \implies f x = g x$
begin

lemma f' - cong : $(g \text{ has-derivative } \text{blinfun-apply } (f' x)) \text{ (at } x) \text{ if } x \in Y$
 $\langle \text{proof} \rangle$

lemma $\text{c1-on-open-euclidean-congI}$: $\text{c1-on-open-euclidean } g f' Y$
 $\langle \text{proof} \rangle$

lemma vareq-cong : $\text{vareq } x0 t = \text{c1-on-open-euclidean.vareq } g f' Y x0 t$
if $t \in \text{existence-ivl0 } x0$
 $\langle \text{proof} \rangle$

lemma $D\text{flow-cong}$:
assumes $t \in \text{existence-ivl0 } x0$
shows $D\text{flow } x0 t = \text{c1-on-open-euclidean.Dflow } g f' Y x0 t$
 $\langle \text{proof} \rangle$

lemma flowsto-congI1 :
assumes $\text{flowsto } A B C D$
shows $\text{c1-on-open-euclidean.flowsto } g f' Y A B C D$
 $\langle \text{proof} \rangle$

lemma flowsto-congI2 :
assumes $\text{c1-on-open-euclidean.flowsto } g f' Y A B C D$
shows $\text{flowsto } A B C D$
 $\langle \text{proof} \rangle$

lemma *flowsto-congI*: $\text{flowsto } A B C D = \text{c1-on-open-euclidean.flowsto } g f' Y A B C D$
<proof>

lemma
returns-to-congI1:
assumes *returns-to* $A x$
shows *auto-ll-on-open.returns-to* $g Y A x$
<proof>

lemma
returns-to-congI2:
assumes *auto-ll-on-open.returns-to* $g Y x A$
shows *returns-to* $x A$
<proof>

lemma *returns-to-cong*: *auto-ll-on-open.returns-to* $g Y A x = \text{returns-to } A x$
<proof>

lemma
return-time-cong:
shows *return-time* $A x = \text{auto-ll-on-open.return-time } g Y A x$
<proof>

lemma *poincare-mapsto-congI1*:
assumes *poincare-mapsto* $A B C D E$ *closed* A
shows *c1-on-open-euclidean.poincare-mapsto* $g Y A B C D E$
<proof>

lemma *poincare-mapsto-congI2*:
assumes *c1-on-open-euclidean.poincare-mapsto* $g Y A B C D E$ *closed* A
shows *poincare-mapsto* $A B C D E$
<proof>

lemma *poincare-mapsto-cong*: *closed* $A \implies$
poincare-mapsto $A B C D E = \text{c1-on-open-euclidean.poincare-mapsto } g Y A B C D E$
<proof>

end

end

end

theory *Cones*

imports

HOL-Analysis.Analysis

Triangle.Triangle

../ODE-Auxiliarities
begin

lemma *arcsin-eq-zero-iff[simp]*: $-1 \leq x \implies x \leq 1 \implies \arcsin x = 0 \iff x = 0$
 ⟨proof⟩

definition *conemem* :: 'a::real-vector \Rightarrow 'a \Rightarrow real \Rightarrow 'a **where** *conemem* u v t =
 $\cos t *_{\mathbb{R}} u + \sin t *_{\mathbb{R}} v$

definition *conesegment* u v = *conemem* u v ' {0.. pi / 2}

lemma

bounded-linear-image-conemem:

assumes *bounded-linear* F

shows F (conemem u v t) = conemem (F u) (F v) t

⟨proof⟩

lemma

bounded-linear-image-conesegment:

assumes *bounded-linear* F

shows F ' conesegment u v = conesegment (F u) (F v)

⟨proof⟩

lemma *discriminant*: $a * x^2 + b * x + c = (0::real) \implies 0 \leq b^2 - 4 * a * c$
 ⟨proof⟩

lemma *quadratic-eq-factoring*:

assumes D: $D = b^2 - 4 * a * c$

assumes nn: $0 \leq D$

assumes x1: $x_1 = (-b + \text{sqrt } D) / (2 * a)$

assumes x2: $x_2 = (-b - \text{sqrt } D) / (2 * a)$

assumes a: $a \neq 0$

shows $a * x^2 + b * x + c = a * (x - x_1) * (x - x_2)$

⟨proof⟩

lemma *quadratic-eq-zeroes-iff*:

assumes D: $D = b^2 - 4 * a * c$

assumes x1: $x_1 = (-b + \text{sqrt } D) / (2 * a)$

assumes x2: $x_2 = (-b - \text{sqrt } D) / (2 * a)$

assumes a: $a \neq 0$

shows $a * x^2 + b * x + c = 0 \iff (D \geq 0 \wedge (x = x_1 \vee x = x_2))$ (is ?z \iff -)

⟨proof⟩

lemma *quadratic-ex-zero-iff*:

$(\exists x. a * x^2 + b * x + c = 0) \iff (a \neq 0 \wedge b^2 - 4 * a * c \geq 0 \vee a = 0 \wedge (b = 0 \implies c = 0))$

for a b c::real

⟨proof⟩

lemma *Cauchy-Schwarz-eq-iff*:

shows $(\text{inner } x \ y)^2 = \text{inner } x \ x * \text{inner } y \ y \longleftrightarrow ((\exists k. x = k *_R y) \vee y = 0)$
<proof>

lemma *Cauchy-Schwarz-strict-ineq*:

$(\text{inner } x \ y)^2 < \text{inner } x \ x * \text{inner } y \ y$ **if** $y \neq 0 \wedge k. x \neq k *_R y$
<proof>

lemma *Cauchy-Schwarz-eq2-iff*:

$|\text{inner } x \ y| = \text{norm } x * \text{norm } y \longleftrightarrow ((\exists k. x = k *_R y) \vee y = 0)$
<proof>

lemma *Cauchy-Schwarz-strict-ineq2*:

$|\text{inner } x \ y| < \text{norm } x * \text{norm } y$ **if** $y \neq 0 \wedge k. x \neq k *_R y$
<proof>

lemma *gt-minus-one-absI*: $\text{abs } k < 1 \implies -1 < k$ **for** $k::\text{real}$

<proof>

lemma *gt-one-absI*: $\text{abs } k < 1 \implies k < 1$ **for** $k::\text{real}$

<proof>

lemma *abs-impossible*:

$|y1| < x1 \implies |y2| < x2 \implies x1 * x2 + y1 * y2 \neq 0$ **for** $x1 \ x2::\text{real}$
<proof>

lemma *vangle-eq-arctan-minus*: — TODO: generalize?!

assumes $ij: i \in \text{Basis } j \in \text{Basis}$ **and** $ij\text{-neg}: i \neq j$

assumes $xy1: |y1| < x1$

assumes $xy2: |y2| < x2$

assumes $\text{less}: y2 / x2 > y1 / x1$

shows $\text{vangle } (x1 *_R i + y1 *_R j) (x2 *_R i + y2 *_R j) = \text{arctan } (y2 / x2) - \text{arctan } (y1 / x1)$

(**is** $\text{vangle } ?u \ ?v = -$)

<proof>

lemma *vangle-le-pi2*: $0 \leq u \cdot v \implies \text{vangle } u \ v \leq \text{pi}/2$

<proof>

lemma *inner-eq-vangle*: $u \cdot v = \cos (\text{vangle } u \ v) * (\text{norm } u * \text{norm } v)$

<proof>

lemma *vangle-scaleR-self*:

$\text{vangle } (k *_R v) \ v = (\text{if } k = 0 \vee v = 0 \text{ then } \text{pi} / 2 \text{ else if } k > 0 \text{ then } 0 \text{ else } \text{pi})$

$\text{vangle } v \ (k *_R v) = (\text{if } k = 0 \vee v = 0 \text{ then } \text{pi} / 2 \text{ else if } k > 0 \text{ then } 0 \text{ else } \text{pi})$

<proof>

lemma *vangle-scaleR*:

$\text{vangle } (k *_R v) \ w = \text{vangle } v \ w \ \text{vangle } w \ (k *_R v) = \text{vangle } w \ v$ **if** $k > 0$

<proof>

lemma *cos-vangle-eq-zero-iff-vangle:*

$\cos (\text{vangle } u \ v) = 0 \iff (u = 0 \vee v = 0 \vee u \cdot v = 0)$

<proof>

lemma *ortho-imp-angle-pi-half:* $u \cdot v = 0 \implies \text{vangle } u \ v = \text{pi} / 2$

<proof>

lemma *arccos-eq-zero-iff:* $\arccos x = 0 \iff x = 1$ **if** $-1 \leq x \leq 1$

<proof>

lemma *vangle-eq-zeroD:* $\text{vangle } u \ v = 0 \implies (\exists k. v = k *_{\mathbb{R}} u)$

<proof>

lemma *less-one-multI:*— TODO: also in AA!

fixes $e \ x :: \text{real}$

shows $e \leq 1 \implies 0 < x \implies x < 1 \implies e * x < 1$

<proof>

lemma *conemem-expansion-estimate:*

fixes $u \ v \ u' \ v' :: 'a :: \text{euclidean-space}$

assumes $t \in \{0 .. \text{pi} / 2\}$

assumes *angle-pos:* $0 < \text{vangle } u \ v \ \text{vangle } u \ v < \text{pi} / 2$

assumes *angle-le:* $(\text{vangle } u' \ v') \leq (\text{vangle } u \ v)$

assumes $\text{norm } u = 1 \ \text{norm } v = 1$

shows $\text{norm } (\text{conemem } u' \ v' \ t) \geq \min (\text{norm } u') (\text{norm } v') * \text{norm } (\text{conemem } u \ v \ t)$

<proof>

lemma *conemem-commute:* $\text{conemem } a \ b \ t = \text{conemem } b \ a \ (\text{pi} / 2 - t)$ **if** $0 \leq t \leq \text{pi} / 2$

<proof>

lemma *conesegment-commute:* $\text{conesegment } a \ b = \text{conesegment } b \ a$

<proof>

definition *conefield* $u \ v = \text{cone hull } (\text{conesegment } u \ v)$

lemma *conefield-alt-def:* $\text{conefield } u \ v = \text{cone hull } \{u - v\}$

<proof>

lemma

bounded-linear-image-cone-hull:

assumes *bounded-linear* F

shows $F \text{ ` } (\text{cone hull } T) = \text{cone hull } (F \text{ ` } T)$

<proof>

lemma

bounded-linear-image-conefield:

assumes *bounded-linear* F

shows $F \text{ ` } \text{conefield } u \ v = \text{conefield } (F \ u) \ (F \ v)$

<proof>

lemma *conefield-commute:* $\text{conefield } x \ y = \text{conefield } y \ x$

<proof>

lemma *convex-conefield:* $\text{convex } (\text{conefield } x \ y)$

<proof>

lemma *conefield-scaleRI:* $v \in \text{conefield } (r \ *_R \ x) \ y$ **if** $v \in \text{conefield } x \ y \ r > 0$

<proof>

lemma *conefield-scaleRD:* $v \in \text{conefield } x \ y$ **if** $v \in \text{conefield } (r \ *_R \ x) \ y \ r > 0$

<proof>

lemma *conefield-scaleR:* $\text{conefield } (r \ *_R \ x) \ y = \text{conefield } x \ y$ **if** $r > 0$

<proof>

lemma *conefield-expansion-estimate:*

fixes $u \ v::'a::\text{euclidean-space}$ **and** $F::'a \Rightarrow 'a$

assumes $t \in \{0 \ .. \ \text{pi} / 2\}$

assumes *angle-pos:* $0 < \text{vangle } u \ v \ \text{vangle } u \ v < \text{pi} / 2$

assumes *angle-le:* $\text{vangle } (F \ u) \ (F \ v) \leq \text{vangle } u \ v$

assumes *bounded-linear* F

assumes $x \in \text{conefield } u \ v$

shows $\text{norm } (F \ x) \geq \min (\text{norm } (F \ u) / \text{norm } u) (\text{norm } (F \ v) / \text{norm } v) * \text{norm } x$

<proof>

lemma *conefield-rightI:*

assumes *ij:* $i \in \text{Basis } j \in \text{Basis}$ **and** *ij-neq:* $i \neq j$

assumes $y \in \{y1 \ .. \ y2\}$

shows $(i + y \ *_R \ j) \in \text{conefield } (i + y1 \ *_R \ j) \ (i + y2 \ *_R \ j)$

<proof>

lemma *conefield-right-vangleI:*

assumes *ij:* $i \in \text{Basis } j \in \text{Basis}$ **and** *ij-neq:* $i \neq j$

assumes $y \in \{y1 \ .. \ y2\} \ y1 < y2$

shows $(i + y \ *_R \ j) \in \text{conefield } (i + y1 \ *_R \ j) \ (i + y2 \ *_R \ j)$

<proof>

lemma *cone-conefield[intro, simp]:* $\text{cone } (\text{conefield } x \ y)$

<proof>

lemma *conefield-mk-rightI:*

assumes *ij:* $i \in \text{Basis } j \in \text{Basis}$ **and** *ij-neq:* $i \neq j$

assumes $(i + (y / x) *_{\mathbb{R}} j) \in \text{conefield } (i + (y1 / x1) *_{\mathbb{R}} j) (i + (y2 / x2) *_{\mathbb{R}} j)$
assumes $x > 0 \ x1 > 0 \ x2 > 0$
shows $(x *_{\mathbb{R}} i + y *_{\mathbb{R}} j) \in \text{conefield } (x1 *_{\mathbb{R}} i + y1 *_{\mathbb{R}} j) (x2 *_{\mathbb{R}} i + y2 *_{\mathbb{R}} j)$
 <proof>

lemma *conefield-prod3I*:
assumes $x > 0 \ x1 > 0 \ x2 > 0$
assumes $y1 / x1 \leq y / x \ y / x \leq y2 / x2$
shows $(x, y, 0) \in (\text{conefield } (x1, y1, 0) (x2, y2, 0))::(\text{real}*\text{real}*\text{real}) \text{ set}$
 <proof>

end

7 Linear ODE

theory *Linear-ODE*

imports

../IVP/Flow

Bounded-Linear-Operator

Multivariate-Taylor

begin

lemma

exp-scaleR-has-derivative-right[derivative-intros]:

fixes $f::\text{real} \Rightarrow \text{real}$

assumes $(f \text{ has-derivative } f') \text{ (at } x \text{ within } s)$

shows $((\lambda x. \text{exp } (f x *_{\mathbb{R}} A)) \text{ has-derivative } (\lambda h. f' h *_{\mathbb{R}} (\text{exp } (f x *_{\mathbb{R}} A) * A)))$
(at } x \text{ within } s)

<proof>

context

fixes $A::'a::\{\text{banach},\text{perfect-space}\} \text{ blinop}$

begin

definition *linode-solution* $t0 \ x0 = (\lambda t. \text{exp } ((t - t0) *_{\mathbb{R}} A) \ x0)$

lemma *linode-solution-solves-ode*:

(linode-solution $t0 \ x0$ *solves-ode* $(\lambda-. A)$ *UNIV UNIV* *linode-solution* $t0 \ x0 \ t0 =$
 $x0$

<proof>

lemma *(linode-solution* $t0 \ x0$ *usolves-ode* $(\lambda-. A)$ *from* $t0)$ *UNIV UNIV*

<proof>

end

end

theory *ODE-Analysis*

imports

Library/MVT-Ex

IVP/Flow

IVP/Upper-Lower-Solution

IVP/Reachability-Analysis

IVP/Flow-Congs

IVP/Cones

Library/Linear-ODE

begin

end

References

- [1] W. Walter. *Ordinary Differential Equations*. Springer, 1 edition, 1998.