Ordinary Differential Equations

Fabian Immler

February 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

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*: $(norm\ (x + y))^2 + (norm\ (x - y))^2 = 2 * (norm\ x)^2 + 2 * (norm\ y)^2$
⟨*proof*⟩

locale *no-real-inner*
begin

lift-definition *fstzero*::(*real*∗*real*) ⇒$_L$ (*real*∗*real*) **is** $\lambda(x, y).\ (x, 0)$
⟨*proof*⟩

lemma [*simp*]: *fstzero* $(a, b) = (a, 0)$
⟨*proof*⟩

lift-definition *zerosnd*::(*real*∗*real*) ⇒$_L$ (*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 $(norm\ (?x + ?y))^2 + (norm\ (?x - ?y))^2 = 2 * (norm\ ?x)^2 + 2 * (norm\ ?y)^2$

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

end

1.2 Topology

1.3 Vector Spaces

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

1.4 Reals

1.5 Balls

sometimes $(?y \in ball\ ?x\ ?e) = (dist\ ?x\ ?y < ?e)$ 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 $\bigwedge i.\ i \in Basis \Longrightarrow bounded\ ((\lambda x.\ x \cdot i)\ `\ X)$
obtains *a b* **where** $X \subseteq cbox\ a\ b$
⟨*proof*⟩

lemma
bounded-euclideanI:
assumes $\bigwedge i.\ i \in Basis \Longrightarrow bounded\ ((\lambda x.\ x \cdot i)\ `\ X)$
shows *bounded X*
⟨*proof*⟩

1.7 Intervals

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

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

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

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

lemma *bdd-above-is-intervalI*: *bdd-above I* **if** *is-interval I a* $\leq$ *b a* $\in$ *I b* $\notin$ *I* **for** *I*::*real set*
⟨*proof*⟩

lemma *bdd-below-is-intervalI*: *bdd-below I* **if** *is-interval I a* $\leq$ *b a* $\notin$ *I b* $\in$ *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* $\in$ *X*
shows *eventually* (λ*e*. *cball x e* $\subseteq$ *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* $\in$ *S*
shows ((λ*x*. *if P x then f x else g x*) *has-derivative F'*) (*at x within S*)
⟨*proof*⟩

lemma *norm-le-in-cubeI*: *norm x* $\leq$ *norm y*
if $\bigwedge$*i*. *i* $\in$ *Basis* $\Longrightarrow$ *abs* (*x* $\cdot$ *i*) $\leq$ *abs* (*y* $\cdot$ *i*) **for** *x y*
⟨*proof*⟩

lemma *has-derivative-partials-euclidean-convexI*:
fixes *f*::*'a*::*euclidean-space* $\Rightarrow$ *'b*::*real-normed-vector*
assumes *f'*: $\bigwedge$*i x xi*. *i* $\in$ *Basis* $\Longrightarrow$ (∀ *j*∈*Basis*. *x* $\cdot$ *j* $\in$ *X j*) $\Longrightarrow$ *xi* = *x* $\cdot$ *i* $\Longrightarrow$
((λ*p*. *f* (*x* + (*p* − *x* $\cdot$ *i*) $*_R$ *i*)) *has-vector-derivative f' i x*) (*at xi within X i*)
assumes *df-cont*: $\bigwedge$*i*. *i* $\in$ *Basis* $\Longrightarrow$ (*f' i* ⟶ (*f' i x*)) (*at x within* {*x*. ∀ *j*∈*Basis*. *x* $\cdot$ *j* $\in$ *X j*})
assumes $\bigwedge$*i*. *i* $\in$ *Basis* $\Longrightarrow$ *x* $\cdot$ *i* $\in$ *X i*
assumes $\bigwedge$*i*. *i* $\in$ *Basis* $\Longrightarrow$ *convex* (*X i*)
shows (*f has-derivative* (λ*h*. $\sum$*j*∈*Basis*. (*h* $\cdot$ *j*) $*_R$ *f' j x*)) (*at x within* {*x*. ∀ *j*∈*Basis*. *x* $\cdot$ *j* $\in$ *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 has-derivative Df x*) (*at x*)
assumes *f'*: ((λ*p*. *f* (*x* + (*p* − *x* · *i*) $*_R$ *i*) · *b*) *has-real-derivative f' x i b*) (*at* (*x* · *i*))
shows *Df x i* · *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 integrable-on s*
and *g integrable-on s*
and $\bigwedge x$. *x* ∈ *s* $\Longrightarrow$ *f x* ≤ *g x*
shows *integral s f* ≤ *integral s g*
⟨*proof*⟩

lemma
integral-ivl-bound:
fixes *l u*::$'a$::*ordered-euclidean-space*
assumes $\bigwedge x\ h'$. *h'* ∈ {*t0* .. *h*} $\Longrightarrow$ *x* ∈ {*t0* .. *h*} $\Longrightarrow$ (*h'* − *t0*) $*_R$ *f x* ∈ {*l* .. *u*}
assumes *t0* ≤ *h*
assumes *f-int*: *f integrable-on* {*t0* .. *h*}
shows *integral* {*t0* .. *h*} *f* ∈ {*l* .. *u*}
⟨*proof*⟩

lemma
add-integral-ivl-bound:
fixes *l u*::$'a$::*ordered-euclidean-space*
assumes $\bigwedge x\ h'$. *h'* ∈ {*t0* .. *h*} $\Longrightarrow$ *x* ∈ {*t0* .. *h*} $\Longrightarrow$ (*h'* − *t0*) $*_R$ *f x* ∈ {*l* − *x0* .. *u* − *x0*}
assumes *t0* ≤ *h*
assumes *f-int*: *f integrable-on* {*t0* .. *h*}
shows *x0* + *integral* {*t0* .. *h*} *f* ∈ {*l* .. *u*}
⟨*proof*⟩

1.15 conditionally complete lattice

1.16 Lists

lemma
Ball-set-Cons[*simp*]: (∀ *a*∈*set-Cons x y*. *P a*) ⟷ (∀ *a*∈*x*. ∀ *b*∈*y*. *P* (*a*#*b*))

⟨*proof*⟩

lemma *set-cons-eq-empty*[*iff*]: *set-Cons a b* = {} ⟷ *a* = {} ∨ *b* = {}
⟨*proof*⟩

lemma *listset-eq-empty-iff*[*iff*]: *listset XS* = {} ⟷ {} ∈ *set XS*
⟨*proof*⟩

lemma *sing-in-sings*[*simp*]: [*x*] ∈ (λ*x*. [*x*]) ‘ *xd* ⟷ *x* ∈ *xd*
⟨*proof*⟩

lemma *those-eq-None-set-iff*: *those xs* = *None* ⟷ *None* ∈ *set xs*
⟨*proof*⟩

lemma *those-eq-Some-lengthD*: *those xs* = *Some ys* ⟹ *length xs* = *length ys*
⟨*proof*⟩

lemma *those-eq-Some-map-Some-iff*: *those xs* = *Some ys* ⟷ (*xs* = *map Some ys*) (**is** *?l* ⟷ *?r*)
⟨*proof*⟩

1.17 Set(sum)

1.18 Max

1.19 Uniform Limit

1.20 Bounded Linear Functions

lift-definition *comp3*::— TODO: name?
(′*c*::*real-normed-vector* ⇒$_L$ ′*d*::*real-normed-vector*) ⇒ (′*b*::*real-normed-vector* ⇒$_L$ ′*c*) ⇒$_L$ ′*b* ⇒$_L$ ′*d* **is**
λ(*cd*::(′*c* ⇒$_L$ ′*d*)) (*bc*::′*b* ⇒$_L$ ′*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*::*real-normed-vector* ⇒$_L$ ′*c*::*real-normed-vector*) ⇒ (′*b*::*real-normed-vector* ⇒$_L$ ′*c*) ⇒ (′*a* × ′*b*) ⇒$_L$ ′*c*
is λ*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

⟨*ML*⟩

1.22 point reflection

definition *preflect*::*'a*::*real-vector* ⇒ *'a* ⇒ *'a* **where** *preflect* ≡ λ*t0 t*. *2* ∗$_R$ *t0* − *t*

lemma *preflect-preflect*[*simp*]: *preflect t0* (*preflect t0 t*) = *t*
⟨*proof*⟩

lemma *preflect-preflect-image*[*simp*]: *preflect t0* ‘ *preflect t0* ‘ *S* = *S*
⟨*proof*⟩

lemma *is-interval-preflect*[*simp*]: *is-interval* (*preflect t0* ‘ *S*) ⟷ *is-interval S*
⟨*proof*⟩

lemma *iv-in-preflect-image*[*intro*, *simp*]: *t0* ∈ *T* ⟹ *t0* ∈ *preflect t0* ‘ *T*
⟨*proof*⟩

lemma *preflect-tendsto*[*tendsto-intros*]:
fixes *l*::*'a*::*real-normed-vector*
shows (*g* ⟶ *l*) *F* ⟹ (*h* ⟶ *m*) *F* ⟹ ((λ*x*. *preflect* (*g x*) (*h x*)) ⟶ *preflect l m*) *F*
⟨*proof*⟩

lemma *continuous-preflect*[*continuous-intros*]:
fixes *a*::*'a*::*real-normed-vector*
shows *continuous* (*at a within A*) (*preflect t0*)
⟨*proof*⟩

lemma
fixes *t0*::*'a*::*ordered-real-vector*
shows *preflect-le*[*simp*]: *t0* ≤ *preflect t0 b* ⟷ *b* ≤ *t0*
and *le-preflect*[*simp*]: *preflect t0 b* ≤ *t0* ⟷ *t0* ≤ *b*
and *antimono-preflect*: *antimono* (*preflect t0*)
and *preflect-le-preflect*[*simp*]: *preflect t0 a* ≤ *preflect t0 b* ⟷ *b* ≤ *a*
and *preflect-eq-cancel*[*simp*]: *preflect t0 a* = *preflect t0 b* ⟷ *a* = *b*
⟨*proof*⟩

lemma *preflect-eq-point-iff*[*simp*]: *t0* = *preflect t0 s* ⟷ *t0* = *s preflect t0 s* = *t0* ⟷ *t0* = *s*
⟨*proof*⟩

lemma *preflect-minus-self*[*simp*]: *preflect t0 s* − *t0* = *t0* − *s*
⟨*proof*⟩

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 $f{::}'a{::}ordered\text{-}euclidean\text{-}space{\Rightarrow}'b{::}euclidean\text{-}space$
assumes *fderiv*: $\bigwedge x.\ x \in D \Longrightarrow (f\ has\text{-}derivative\ J\ x)\ (at\ x\ within\ D)$
assumes *line-in*: $\bigwedge x.\ [\![0 \le x;\ x \le 1]\!] \Longrightarrow a + x *_R u \in D$
shows $(\exists\, t{\in}Basis{\to}\{0{<}..{<}1\}.\ (f\ (a\ +\ u)\ -\ f\ a) = (\sum i{\in}Basis.\ (J\ (a\ +\ t\ i\ *_R\ u)\ u\ \cdot\ i)\ *_R\ i))$
⟨*proof*⟩

lemma *MVT-ivl*:
fixes $f{::}'a{::}ordered\text{-}euclidean\text{-}space{\Rightarrow}'b{::}ordered\text{-}euclidean\text{-}space$
assumes *fderiv*: $\bigwedge x.\ x \in D \Longrightarrow (f\ has\text{-}derivative\ J\ x)\ (at\ x\ within\ D)$
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 $f{::}'a{::}ordered\text{-}euclidean\text{-}space{\Rightarrow}'b{::}ordered\text{-}euclidean\text{-}space$
assumes *fderiv*: $(\bigwedge x.\ x \in D \Longrightarrow (f\ has\text{-}derivative\ J\ x)\ (at\ x\ within\ D))$
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* (λ*x*. *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*: ((λ*x*. *x*) *has-vderiv-on* (λ*x*. *1*)) *T*
and *has-vderiv-on-const*: ((λ*x*. *c*) *has-vderiv-on* (λ*x*. *0*)) *T*
⟨*proof*⟩

lemma [*THEN has-vderiv-on-eq-rhs*, *derivative-intros*]:
fixes *f*::*real* ⇒ *'a*::*real-normed-vector*
assumes (*f has-vderiv-on f'*) *T*
shows *has-vderiv-on-uminus*: ((λ*x*. − *f x*) *has-vderiv-on* (λ*x*. − *f' x*)) *T*
⟨*proof*⟩

lemma [*THEN has-vderiv-on-eq-rhs*, *derivative-intros*]:
fixes *f g*::*real* ⇒ *'a*::*real-normed-vector*
assumes (*f has-vderiv-on f'*) *T*
assumes (*g has-vderiv-on g'*) *T*
shows *has-vderiv-on-add*: ((λ*x*. *f x* + *g x*) *has-vderiv-on* (λ*x*. *f' x* + *g' x*)) *T*
and *has-vderiv-on-diff*: ((λ*x*. *f x* − *g x*) *has-vderiv-on* (λ*x*. *f' x* − *g' x*)) *T*
⟨*proof*⟩

lemma [*THEN has-vderiv-on-eq-rhs*, *derivative-intros*]:
fixes *f*::*real* ⇒ *real* **and** *g*::*real* ⇒ *'a*::*real-normed-vector*
assumes (*f has-vderiv-on f'*) *T*
assumes (*g has-vderiv-on g'*) *T*
shows *has-vderiv-on-scaleR*: ((λ*x*. *f x* ∗*R* *g x*) *has-vderiv-on* (λ*x*. *f x* ∗*R* *g' x* + *f' x* ∗*R* *g x*)) *T*
⟨*proof*⟩

lemma [*THEN has-vderiv-on-eq-rhs*, *derivative-intros*]:
fixes *f g*::*real* ⇒ *'a*::*real-normed-algebra*
assumes (*f has-vderiv-on f'*) *T*
assumes (*g has-vderiv-on g'*) *T*
shows *has-vderiv-on-mult*: ((λ*x*. *f x* ∗ *g x*) *has-vderiv-on* (λ*x*. *f x* ∗ *g' x* + *f' x* ∗ *g x*)) *T*
⟨*proof*⟩

lemma *has-vderiv-on-ln*[*THEN has-vderiv-on-eq-rhs*, *derivative-intros*]:
fixes *g*::*real* ⇒ *real*
assumes ⋀*x*. *x* ∈ *s* ⟹ *0* < *g x*
assumes (*g has-vderiv-on g'*) *s*
shows ((λ*x*. *ln* (*g x*)) *has-vderiv-on* (λ*x*. *g' x* / *g x*)) *s*
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus'*:
fixes *f* :: *real* ⇒ *'a*::*banach*
shows *a* ≤ *b* ⟹ (*f has-vderiv-on f'*) {*a* .. *b*} ⟹ (*f' has-integral* (*f b* − *f a*)) {*a* .. *b*}
⟨*proof*⟩

lemma *has-vderiv-on-If*:
assumes *U* = *S* ∪ *T*
assumes (*f has-vderiv-on f'*) (*S* ∪ (*closure T* ∩ *closure S*))
assumes (*g has-vderiv-on g'*) (*T* ∪ (*closure T* ∩ *closure S*))
assumes ⋀*x*. *x* ∈ *closure T* ⟹ *x* ∈ *closure S* ⟹ *f x* = *g x*

assumes $\bigwedge x.\ x \in \mathit{closure}\ T \Longrightarrow x \in \mathit{closure}\ S \Longrightarrow f'\ x = g'\ x$
shows $((\lambda t.\ \mathit{if}\ t \in S\ \mathit{then}\ f\ t\ \mathit{else}\ g\ t)\ \mathit{has\text{-}vderiv\text{-}on}\ (\lambda t.\ \mathit{if}\ t \in S\ \mathit{then}\ f'\ t\ \mathit{else}\ g'\ t))\ U$
⟨*proof*⟩

lemma *mvt-very-simple-closed-segmentE*:
fixes f::*real*⇒*real*
assumes (f *has-vderiv-on* f') (*closed-segment* a b)
obtains y **where** $y \in$ *closed-segment* a b $f\ b - f\ a = (b - a) * f'\ y$
⟨*proof*⟩

lemma *mvt-simple-closed-segmentE*:
fixes f::*real*⇒*real*
assumes (f *has-vderiv-on* f') (*closed-segment* a b)
assumes $a \neq b$
obtains y **where** $y \in$ *open-segment* a b $f\ b - f\ a = (b - a) * f'\ y$
⟨*proof*⟩

lemma *differentiable-bound-general-open-segment*:
fixes a :: *real*
and b :: *real*
and f :: *real* ⇒ *'a*::*real-normed-vector*
and f' :: *real* ⇒ *'a*
assumes *continuous-on* (*closed-segment* a b) f
assumes *continuous-on* (*closed-segment* a b) g
and (f *has-vderiv-on* f') (*open-segment* a b)
and (g *has-vderiv-on* g') (*open-segment* a b)
and $\bigwedge x.\ x \in$ *open-segment* a $b \Longrightarrow$ *norm* $(f'\ x) \leq g'\ x$
shows *norm* $(f\ b - f\ a) \leq$ *abs* $(g\ b - g\ a)$
⟨*proof*⟩

lemma *has-vderiv-on-union*:
assumes (f *has-vderiv-on* g) ($s \cup$ *closure* $s \cap$ *closure* t)
assumes (f *has-vderiv-on* g) ($t \cup$ *closure* $s \cap$ *closure* t)
shows (f *has-vderiv-on* g) ($s \cup t$)
⟨*proof*⟩

lemma *has-vderiv-on-union-closed*:
assumes (f *has-vderiv-on* g) s
assumes (f *has-vderiv-on* g) t
assumes *closed* s *closed* t
shows (f *has-vderiv-on* g) ($s \cup t$)
⟨*proof*⟩

lemma *vderiv-on-continuous-on*: (f *has-vderiv-on* f') $S \Longrightarrow$ *continuous-on* S f
⟨*proof*⟩

lemma *has-vderiv-on-cong*[*cong*]:
assumes $\bigwedge x.\ x \in S \Longrightarrow f\ x = g\ x$

assumes $\bigwedge x.\ x \in S \Longrightarrow f'\ x = g'\ x$
assumes $S = T$
shows $(f\ \textit{has-vderiv-on}\ f')\ S = (g\ \textit{has-vderiv-on}\ g')\ T$
⟨*proof*⟩

lemma *has-vderiv-eq*:
assumes $(f\ \textit{has-vderiv-on}\ f')\ S$
assumes $\bigwedge x.\ x \in S \Longrightarrow f\ x = g\ x$
assumes $\bigwedge x.\ x \in S \Longrightarrow f'\ x = g'\ x$
assumes $S = T$
shows $(g\ \textit{has-vderiv-on}\ g')\ T$
⟨*proof*⟩

lemma *has-vderiv-on-compose'*:
assumes $(f\ \textit{has-vderiv-on}\ f')\ (g\ `\ T)$
assumes $(g\ \textit{has-vderiv-on}\ g')\ T$
shows $((\lambda x.\ f\ (g\ x))\ \textit{has-vderiv-on}\ (\lambda x.\ g'\ x *_R f'\ (g\ x)))\ T$
⟨*proof*⟩

lemma *has-vderiv-on-compose2*:
assumes $(f\ \textit{has-vderiv-on}\ f')\ S$
assumes $(g\ \textit{has-vderiv-on}\ g')\ T$
assumes $\bigwedge t.\ t \in T \Longrightarrow g\ t \in S$
shows $((\lambda x.\ f\ (g\ x))\ \textit{has-vderiv-on}\ (\lambda x.\ g'\ x *_R f'\ (g\ x)))\ T$
⟨*proof*⟩

lemma *has-vderiv-on-singleton*: $(y\ \textit{has-vderiv-on}\ y')\ \{t0\}$
⟨*proof*⟩

lemma
has-vderiv-on-zero-constant:
assumes *convex s*
assumes $(f\ \textit{has-vderiv-on}\ (\lambda h.\ 0))\ s$
obtains c **where** $\bigwedge x.\ x \in s \Longrightarrow f\ x = c$
⟨*proof*⟩

lemma *bounded-vderiv-on-imp-lipschitz*:
assumes $(f\ \textit{has-vderiv-on}\ f')\ X$
assumes *convex*: *convex X*
assumes $\bigwedge x.\ x \in X \Longrightarrow \textit{norm}\ (f'\ x) \leq C\ 0 \leq C$
shows $C-\textit{lipschitz-on}\ X\ f$
⟨*proof*⟩

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* ⇒ *'b*::*real-normed-vector*) ⇒ *'b* ⇒ *real* ⇒ *real* ⇒ *bool*— TODO: generalize?
(**infixr** *has'-ivl'-integral 46*)
where (*f has-ivl-integral y*) *a b* ⟷ (*if a* ≤ *b then* (*f has-integral y*) {*a* .. *b*} *else* (*f has-integral* − *y*) {*b* .. *a*})

definition *ivl-integral*::*real* ⇒ *real* ⇒ (*real* ⇒ *'a*) ⇒ *'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* ≥ *b* ⟹ *integral* {*a*..*b*} *f* = *0 a* > *b* ⟹ *integral* {*a*..*b*} *f* = *0*
⟨*proof*⟩

lemma *ivl-integral-unique*: (*f has-ivl-integral y*) *a b* ⟹ *ivl-integral a b f* = *y*
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus-ivl-integral*:
fixes *f* :: *real* ⇒ *'a*::*banach*
shows (*f has-vderiv-on f'*) (*closed-segment a b*) ⟹ (*f' has-ivl-integral f b* − *f a*) *a b*
⟨*proof*⟩

lemma
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *f integrable-on* (*closed-segment a b*)
shows *indefinite-ivl-integral-continuous*:
continuous-on (*closed-segment a b*) (λ*x*. *ivl-integral a x f*)
continuous-on (*closed-segment b a*) (λ*x*. *ivl-integral a x f*)
⟨*proof*⟩

lemma
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *f integrable-on* (*closed-segment a b*)
assumes *c* ∈ *closed-segment a b*
shows *indefinite-ivl-integral-continuous-subset*:
continuous-on (*closed-segment a b*) (λ*x*. *ivl-integral c x f*)
⟨*proof*⟩

lemma *real-Icc-closed-segment*: **fixes** *a b*::*real* **shows** *a* ≤ *b* ⟹ {*a* .. *b*} = *closed-segment a b*
⟨*proof*⟩

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

⟨*proof*⟩

lemma *ivl-integral-cong*:
assumes $\bigwedge x.\ x \in \textit{closed-segment}\ a\ b \Longrightarrow g\ x = f\ x$
assumes $a = c\ b = d$
shows *ivl-integral a b f* = *ivl-integral c d g*
⟨*proof*⟩

lemma *ivl-integral-diff*:
f integrable-on (*closed-segment s t*) ⟹ *g integrable-on* (*closed-segment s t*) ⟹
ivl-integral s t (λ*x*. *f x* − *g x*) = *ivl-integral s t f* − *ivl-integral s t g*
⟨*proof*⟩

lemma *ivl-integral-norm-bound-ivl-integral*:
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *f integrable-on* (*closed-segment a b*)
and *g integrable-on* (*closed-segment a b*)
and $\bigwedge x.\ x \in \textit{closed-segment}\ a\ b \Longrightarrow \textit{norm}\ (f\ x) \leq g\ x$
shows *norm* (*ivl-integral a b f*) ≤ *abs* (*ivl-integral a b g*)
⟨*proof*⟩

lemma *ivl-integral-norm-bound-integral*:
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *f integrable-on* (*closed-segment a b*)
and *g integrable-on* (*closed-segment a b*)
and $\bigwedge x.\ x \in \textit{closed-segment}\ a\ b \Longrightarrow \textit{norm}\ (f\ x) \leq g\ x$
shows *norm* (*ivl-integral a b f*) ≤ *integral* (*closed-segment a b*) *g*
⟨*proof*⟩

lemma *norm-ivl-integral-le*:
fixes *f* :: *real* ⇒ *real*
assumes *f integrable-on* (*closed-segment a b*)
and *g integrable-on* (*closed-segment a b*)
and $\bigwedge x.\ x \in \textit{closed-segment}\ a\ b \Longrightarrow f\ x \leq g\ x$
and $\bigwedge x.\ x \in \textit{closed-segment}\ a\ b \Longrightarrow 0 \leq f\ x$
shows *abs* (*ivl-integral a b f*) ≤ *abs* (*ivl-integral a b g*)
⟨*proof*⟩

lemma *ivl-integral-const* [*simp*]:
shows *ivl-integral a b* (λ*x*. *c*) = (*b* − *a*) ∗$_R$ *c*
⟨*proof*⟩

lemma *ivl-integral-has-vector-derivative*:
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *continuous-on* (*closed-segment a b*) *f*
and *x* ∈ *closed-segment a b*
shows ((λ*u*. *ivl-integral a u f*) *has-vector-derivative f x*) (*at x within closed-segment a b*)
⟨*proof*⟩

lemma *ivl-integral-has-vderiv-on*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *continuous-on* (*closed-segment a b*) *f*
shows ((λ*u*. *ivl-integral a u f*) *has-vderiv-on f*) (*closed-segment a b*)
⟨*proof*⟩

lemma *ivl-integral-has-vderiv-on-subset-segment*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *continuous-on* (*closed-segment a b*) *f*
and $c \in$ *closed-segment a b*
shows ((λ*u*. *ivl-integral c u f*) *has-vderiv-on f*) (*closed-segment a b*)
⟨*proof*⟩

lemma *ivl-integral-has-vector-derivative-subset*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *continuous-on* (*closed-segment a b*) *f*
and $x \in$ *closed-segment a b*
and $c \in$ *closed-segment a b*
shows ((λ*u*. *ivl-integral c u f*) *has-vector-derivative f x*) (*at x within closed-segment a b*)
⟨*proof*⟩

lemma
compact-interval-eq-Inf-Sup:
fixes *A*::*real set*
assumes *is-interval A compact A* $A \neq \{\}$
shows $A = \{Inf\ A\ ..\ Sup\ A\}$
⟨*proof*⟩

lemma *ivl-integral-has-vderiv-on-compact-interval*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *continuous-on A f*
and $c \in A$ *is-interval A compact A*
shows ((λ*u*. *ivl-integral c u f*) *has-vderiv-on f*) *A*
⟨*proof*⟩

lemma *ivl-integral-has-vector-derivative-compact-interval*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *continuous-on A f*
and *is-interval A compact A* $x \in A$ $c \in A$
shows ((λ*u*. *ivl-integral c u f*) *has-vector-derivative f x*) (*at x within A*)
⟨*proof*⟩

lemma *ivl-integral-combine*:
fixes *f*::*real* ⇒ *'a*::*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-integral a b f + ivl-integral b c f = ivl-integral a c f*
⟨*proof*⟩

lemma *integral-equation-swap-initial-value*:
fixes *x*::*real*⇒'*a*::*banach*
assumes ⋀*t*. *t* ∈ *closed-segment t0 t1* ⟹ *x t* = *x t0* + *ivl-integral t0 t* (λ*t*. *f t* (*x t*))
assumes *t*: *t* ∈ *closed-segment t0 t1*
assumes *int*: (λ*t*. *f t* (*x t*)) *integrable-on closed-segment t0 t1*
shows *x t* = *x t1* + *ivl-integral t1 t* (λ*t*. *f t* (*x t*))
⟨*proof*⟩

lemma *has-integral-nonpos*:
fixes *f* :: '*n*::*euclidean-space* ⇒ *real*
assumes (*f has-integral i*) *s*
and ∀ *x*∈*s*. *f x* ≤ *0*
shows *i* ≤ *0*
⟨*proof*⟩

lemma *has-ivl-integral-nonneg*:
fixes *f* :: *real* ⇒ *real*
assumes (*f has-ivl-integral i*) *a b*
and ⋀*x*. *a* ≤ *x* ⟹ *x* ≤ *b* ⟹ *0* ≤ *f x*
and ⋀*x*. *b* ≤ *x* ⟹ *x* ≤ *a* ⟹ *f x* ≤ *0*
shows *0* ≤ *i*
⟨*proof*⟩

lemma *has-ivl-integral-ivl-integral*:
f integrable-on (*closed-segment a b*) ⟷ (*f has-ivl-integral* (*ivl-integral a b f*)) *a b*
⟨*proof*⟩

lemma *ivl-integral-nonneg*:
fixes *f* :: *real* ⇒ *real*
assumes *f integrable-on* (*closed-segment a b*)
and ⋀*x*. *a* ≤ *x* ⟹ *x* ≤ *b* ⟹ *0* ≤ *f x*
and ⋀*x*. *b* ≤ *x* ⟹ *x* ≤ *a* ⟹ *f x* ≤ *0*
shows *0* ≤ *ivl-integral a b f*
⟨*proof*⟩

lemma *ivl-integral-bound*:
fixes *f*::*real* ⇒ '*a*::*banach*
assumes *continuous-on* (*closed-segment a b*) *f*
assumes ⋀*t*. *t* ∈ (*closed-segment a b*) ⟹ *norm* (*f t*) ≤ *B*
shows *norm* (*ivl-integral a b f*) ≤ *B* ∗ *abs* (*b* − *a*)
⟨*proof*⟩

lemma *ivl-integral-minus-sets*:
fixes *f*::*real* ⇒ '*a*::*banach*
shows *f integrable-on* (*closed-segment c a*) ⟹ *f integrable-on* (*closed-segment c*

$b) \Longrightarrow f$ *integrable-on* (*closed-segment a b*) $\Longrightarrow$
ivl-integral c a f − ivl-integral c b f = ivl-integral b a f
⟨*proof*⟩

lemma *ivl-integral-minus-sets'*:
fixes *f*::*real* ⇒ *'a*::*banach*
shows *f integrable-on* (*closed-segment a c*) $\Longrightarrow$ *f integrable-on* (*closed-segment b c*) $\Longrightarrow$ *f integrable-on* (*closed-segment a b*) $\Longrightarrow$
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*: $\bigwedge t.\ t \in \{a\ ..\ b\} \Longrightarrow g'\ t\ /\ g\ t \leq K$
assumes *g'-cont*: *continuous-on* {*a .. b*} *g'*
assumes *g-pos*: $\bigwedge t.\ t \in \{a\ ..\ b\} \Longrightarrow g\ t > 0$
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*: $\bigwedge t.\ t \in \{a\ ..\ b\} \Longrightarrow K \leq g'\ t\ /\ g\ t$
assumes *g'-cont*: *continuous-on* {*a .. b*} *g'*
assumes *g-pos*: $\bigwedge t.\ t \in \{a\ ..\ b\} \Longrightarrow g\ t > 0$
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 $G \equiv \lambda t.\ C + K * integral\ \{a..t\}\ (\lambda s.\ g\ s)$
assumes *g-le-G*: $\bigwedge t.\ t \in \{a..b\} \Longrightarrow g\ t \leq G\ t$
assumes *g-cont*: *continuous-on* {*a..b*} *g*
assumes *g-nonneg*: $\bigwedge t.\ t \in \{a..b\} \Longrightarrow 0 \leq g\ t$
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 $G \equiv \lambda t.\ C + K * \mathit{integral}\ \{t..b\}\ (\lambda s.\ g\ s)$
assumes *g-le-G*: $\bigwedge t.\ t \in \{a..b\} \Longrightarrow g\ t \leq G\ t$
assumes *g-cont*: *continuous-on* $\{a..b\}$ *g*
assumes *g-nonneg*: $\bigwedge t.\ t \in \{a..b\} \Longrightarrow 0 \leq g\ t$
assumes *pos*: $0 < C\ K > 0$
assumes $t \in \{a..b\}$
shows $g\ t \leq C * \mathit{exp}\ (-K * (t - b))$
⟨*proof*⟩

lemma *gronwall-general-segment*:
fixes *a b*::*real*
assumes $\bigwedge t.\ t \in \mathit{closed\text{-}segment}\ a\ b \Longrightarrow g\ t \leq C + K * \mathit{integral}\ (\mathit{closed\text{-}segment}\ a\ t)\ g$
and *continuous-on* (*closed-segment a b*) *g*
and $\bigwedge t.\ t \in \mathit{closed\text{-}segment}\ a\ b \Longrightarrow 0 \leq g\ t$
and $0 < C$
and $0 < K$
and $t \in \mathit{closed\text{-}segment}\ a\ b$
shows $g\ t \leq C * \mathit{exp}\ (K * \mathit{abs}\ (t - a))$
⟨*proof*⟩

lemma *gronwall-more-general-segment*:
fixes *a b c*::*real*
assumes $\bigwedge t.\ t \in \mathit{closed\text{-}segment}\ a\ b \Longrightarrow g\ t \leq C + K * \mathit{integral}\ (\mathit{closed\text{-}segment}\ c\ t)\ g$
and *cont*: *continuous-on* (*closed-segment a b*) *g*
and $\bigwedge t.\ t \in \mathit{closed\text{-}segment}\ a\ b \Longrightarrow 0 \leq g\ t$
and $0 < C$
and $0 < K$
and *t*: $t \in \mathit{closed\text{-}segment}\ a\ b$
and *c*: $c \in \mathit{closed\text{-}segment}\ a\ b$
shows $g\ t \leq C * \mathit{exp}\ (K * \mathit{abs}\ (t - c))$
⟨*proof*⟩

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

lemma *gronwall-left*:
fixes *g K C* **and** *t*::*real*
defines $G \equiv \lambda t.\ C + K * \mathit{integral}\ \{t..0\}\ (\lambda s.\ g\ s)$
assumes *g-le-G*: $\bigwedge t.\ a \leq t \Longrightarrow t \leq 0 \Longrightarrow g\ t \leq G\ t$

assumes *g-cont*: *continuous-on* $\{a..0\}$ *g*
assumes *g-nonneg*: $\bigwedge t.\ a \leq t \Longrightarrow t \leq 0 \Longrightarrow 0 \leq g\ t$
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*]: $x \leq a \Longrightarrow clamp\ a\ b\ x = a$ **for** *x*::'*a*::*ordered-euclidean-space*
⟨*proof*⟩

lemma *clamp-ge*[*simp*]: $a \leq b \Longrightarrow b \leq x \Longrightarrow clamp\ a\ b\ x = b$ **for** *x*::'*a*::*ordered-euclidean-space*
⟨*proof*⟩

abbreviation *cfuncset* :: '*a*::*topological-space set* ⇒ '*b*::*metric-space set* ⇒ ('*a* ⇒$_C$ '*b*) *set*
(**infixr** →$_C$ *60*)
where $A \to_C B \equiv PiC\ A\ (\lambda\text{-}.\ B)$

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

lemma *closed-segment-subset-interval*: *is-interval* $T \Longrightarrow a \in T \Longrightarrow b \in T \Longrightarrow$ *closed-segment a b* $\subseteq T$
⟨*proof*⟩

definition *half-open-segment*::'*a*::*real-vector* ⇒ '*a* ⇒ '*a set* ((*1*{-−−<-}))
where *half-open-segment a b* $= \{a -- b\} - \{b\}$

lemma *half-open-segment-real*:
fixes *a b*::*real*
shows $\{a --< b\} = (if\ a \leq b\ then\ \{a\ ..< b\}\ else\ \{b <..\ a\})$
⟨*proof*⟩

lemma *closure-half-open-segment*:
fixes *a b*::*real*
shows *closure* $\{a --< b\} = (if\ a = b\ then\ \{\}\ else\ \{a -- b\})$
⟨*proof*⟩

lemma *half-open-segment-subset*[*intro*, *simp*]:
$\{t0--<t1\} \subseteq \{t0 --\ t1\}$
$x \in \{t0--<t1\} \Longrightarrow x \in \{t0 --\ t1\}$
⟨*proof*⟩

lemma *half-open-segment-closed-segmentI*:
$t \in \{t0 --\ t1\} \Longrightarrow t \neq t1 \Longrightarrow t \in \{t0 --<\ t1\}$
⟨*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** $x \in \{t0--<t1\} \Longrightarrow \{t0--x\} \subseteq \{t0--<t1\}$
⟨*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*: $i \in Basis \Longrightarrow (\sum x{\in}Basis.\ x \cdot i) = 1$
⟨*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* $(- r *_R One)$ $(r *_R One{::}'a{::}euclidean\text{-}space) \subseteq$
cball 0 $(sqrt(DIM('a)) * r)$
⟨*proof*⟩

2.1 Solutions of IVPs

definition
solves-ode :: $(real \Rightarrow 'a{::}real\text{-}normed\text{-}vector) \Rightarrow (real \Rightarrow 'a \Rightarrow 'a) \Rightarrow real\ set \Rightarrow$

$'a\ set \Rightarrow bool$
(**infix** (*solves'-ode*) *50*)
where
$(y\ solves\text{-}ode\ f)\ T\ X \longleftrightarrow (y\ has\text{-}vderiv\text{-}on\ (\lambda t.\ f\ t\ (y\ t)))\ T \wedge y \in T \to X$

lemma *solves-odeI*:
assumes *solves-ode-vderivD*: $(y\ has\text{-}vderiv\text{-}on\ (\lambda t.\ f\ t\ (y\ t)))\ T$
and *solves-ode-domainD*: $\bigwedge t.\ t \in T \Longrightarrow y\ t \in X$
shows $(y\ solves\text{-}ode\ f)\ T\ X$
⟨*proof*⟩

lemma *solves-odeD*:
assumes $(y\ solves\text{-}ode\ f)\ T\ X$
shows *solves-ode-vderivD*: $(y\ has\text{-}vderiv\text{-}on\ (\lambda t.\ f\ t\ (y\ t)))\ T$
and *solves-ode-domainD*: $\bigwedge t.\ t \in T \Longrightarrow y\ t \in X$
⟨*proof*⟩

lemma *solves-ode-continuous-on*: $(y\ solves\text{-}ode\ f)\ T\ X \Longrightarrow continuous\text{-}on\ T\ y$
⟨*proof*⟩

lemma *solves-ode-congI*:
assumes $(x\ solves\text{-}ode\ f)\ T\ X$
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 $(y\ solves\text{-}ode\ g)\ S\ Y$
⟨*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 $(x\ solves\text{-}ode\ f)\ T\ X \longleftrightarrow (y\ solves\text{-}ode\ g)\ S\ Y$
⟨*proof*⟩

lemma *solves-ode-on-subset*:
assumes $(x\ solves\text{-}ode\ f)\ S\ Y$
assumes $T \subseteq S\ Y \subseteq X$
shows $(x\ solves\text{-}ode\ f)\ T\ X$
⟨*proof*⟩

lemma *preflect-solution*:
assumes $t0 \in T$
assumes *sol*: $((\lambda t.\ x\ (preflect\ t0\ t))\ solves\text{-}ode\ (\lambda t\ x.\ -\ f\ (preflect\ t0\ t)\ x))$
$(preflect\ t0\ `\ T)\ X$
shows $(x\ solves\text{-}ode\ f)\ T\ X$
⟨*proof*⟩

lemma *solution-preflect*:

assumes *t0* ∈ *T*
assumes *sol*: (*x solves-ode f*) *T X*
shows ((λ*t*. *x* (*preflect t0 t*)) *solves-ode* (λ*t x*. − *f* (*preflect t0 t*) *x*)) (*preflect t0* ‘ *T*) *X*
⟨*proof*⟩

lemma *solution-eq-preflect-solution*:
assumes *t0* ∈ *T*
shows (*x solves-ode f*) *T X* ⟷ ((λ*t*. *x* (*preflect t0 t*)) *solves-ode* (λ*t x*. − *f* (*preflect t0 t*) *x*)) (*preflect t0* ‘ *T*) *X*
⟨*proof*⟩

lemma *shift-autonomous-solution*:
assumes *sol*: (*x solves-ode f*) *T X*
assumes *auto*: ⋀*s t*. *s* ∈ *T* ⟹ *f s* (*x s*) = *f t* (*x s*)
shows ((λ*t*. *x* (*t* + *t0*)) *solves-ode f*) ((λ*t*. *t* − *t0*) ‘ *T*) *X*
⟨*proof*⟩

lemma *solves-ode-singleton*: *y t0* ∈ *X* ⟹ (*y solves-ode f*) {*t0*} *X*
⟨*proof*⟩

2.1.1 Connecting solutions

lemma *connection-solves-ode*:
assumes *x*: (*x solves-ode f*) *T X*
assumes *y*: (*y solves-ode g*) *S Y*
assumes *conn-T*: *closure S* ∩ *closure T* ⊆ *T*
assumes *conn-S*: *closure S* ∩ *closure T* ⊆ *S*
assumes *conn-x*: ⋀*t*. *t* ∈ *closure S* ⟹ *t* ∈ *closure T* ⟹ *x t* = *y t*
assumes *conn-f*: ⋀*t*. *t* ∈ *closure S* ⟹ *t* ∈ *closure T* ⟹ *f t* (*y t*) = *g t* (*y t*)
shows ((λ*t*. *if t* ∈ *T then x t else y t*) *solves-ode* (λ*t*. *if t* ∈ *T then f t else g t*)) (*T* ∪ *S*) (*X* ∪ *Y*)
⟨*proof*⟩

lemma
solves-ode-subset-range:
assumes *x*: (*x solves-ode f*) *T X*
assumes *s*: *x* ‘ *T* ⊆ *Y*
shows (*x solves-ode f*) *T Y*
⟨*proof*⟩

2.2 unique solution with initial value

definition
usolves-ode-from :: (*real* ⇒ *'a*::*real-normed-vector*) ⇒ (*real* ⇒ *'a* ⇒ *'a*) ⇒ *real* ⇒ *real set* ⇒ *'a set* ⇒ *bool*
(((-) *usolves'-ode* (-) *from* (-)) [*10*, *10*, *10*] *10*)
— TODO: no idea about mixfix and precedences, check this!
where

$(y$ *usolves-ode f from t0*$)\ T\ X \longleftrightarrow (y$ *solves-ode f*$)\ T\ X \wedge t0 \in T \wedge$ *is-interval* $T \wedge$
$(\forall z\ T'.\ t0 \in T' \wedge$ *is-interval* $T' \wedge\ T' \subseteq T \wedge (z$ *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
$((\lambda\text{-}.\ 0{::}real)$ *usolves-ode* $(\lambda\text{-}.$ *sqrt*$)$ *from 0*$)\ \{0..\}\ \{0\}$
$((\lambda t.\ t^2\ /\ 4)$ *solves-ode* $(\lambda\text{-}.$ *sqrt*$))\ \{0..\}\ \{0..\}$
$(\lambda t.\ t^2\ /\ 4)\ 0 = (\lambda\text{-}.\ 0{::}real)\ 0$
⟨*proof*⟩

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

lemma *usolves-odeD*:
assumes $(y$ *usolves-ode f from t0*$)\ T\ X$
shows $(y$ *solves-ode f*$)\ T\ X$
and $t0 \in T$
and *is-interval* T
and $\bigwedge z\ T'\ t.\ t0 \in T' \Longrightarrow$ *is-interval* $T' \Longrightarrow T' \subseteq T \Longrightarrow (z$ *solves-ode f*$)\ T'\ X \Longrightarrow z\ t0 = y\ t0 \Longrightarrow t \in T' \Longrightarrow z\ t = y\ t$
⟨*proof*⟩

lemma *usolves-ode-rawI*:
assumes $(y$ *solves-ode f*$)\ T\ X\ t0 \in T$ *is-interval* T
assumes $\bigwedge z\ T'\ t.\ t0 \in T' \Longrightarrow$ *is-interval* $T' \Longrightarrow T' \subseteq T \Longrightarrow (z$ *solves-ode f*$)\ T'\ X \Longrightarrow z\ t0 = y\ t0 \Longrightarrow t \in T' \Longrightarrow z\ t = y\ t$
shows $(y$ *usolves-ode f from t0*$)\ T\ X$
⟨*proof*⟩

lemma *usolves-odeI*:
assumes $(y$ *solves-ode f*$)\ T\ X\ t0 \in T$ *is-interval* T
assumes *usol*: $\bigwedge z\ t.\ \{t0 --\ t\} \subseteq T \Longrightarrow (z$ *solves-ode f*$)\ \{t0 --\ t\}\ X \Longrightarrow z\ t0 = y\ t0 \Longrightarrow z\ t = y\ t$
shows $(y$ *usolves-ode f from t0*$)\ T\ X$
⟨*proof*⟩

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

lemma *usolves-ode-singleton*: $x\ t0 \in X \Longrightarrow (x$ *usolves-ode f from t0*$)\ \{t0\}\ X$
⟨*proof*⟩

lemma *usolves-ode-congI*:
assumes x: $(x$ *usolves-ode f from t0*$)\ T\ X$
assumes $\bigwedge t.\ t \in T \Longrightarrow x\ t = y\ t$
assumes $\bigwedge t\ y.\ t \in T \Longrightarrow y \in X \Longrightarrow f\ t\ y = g\ t\ y$— TODO: weaken this assumption?!
assumes $t0 = s0$

assumes $T = S$
assumes $X = Y$
shows $(y$ *usolves-ode* g *from* $s0)$ S Y
⟨*proof*⟩

lemma *usolves-ode-cong*[*cong*]:
assumes $\bigwedge t.\ t \in T \Longrightarrow x\ t = y\ t$
assumes $\bigwedge t\ y.\ t \in T \Longrightarrow y \in X \Longrightarrow f\ t\ y = g\ t\ y$— TODO: weaken this assumption?!
assumes $t0 = s0$
assumes $T = S$
assumes $X = Y$
shows $(x$ *usolves-ode* f *from* $t0)$ T $X \longleftrightarrow (y$ *usolves-ode* g *from* $s0)$ S Y
⟨*proof*⟩

lemma *shift-autonomous-unique-solution*:
assumes *usol*: $(x$ *usolves-ode* f *from* $t0)$ T X
assumes *auto*: $\bigwedge s\ t\ x.\ x \in X \Longrightarrow f\ s\ x = f\ t\ x$
shows $((\lambda t.\ x\ (t + t0 - t1))$ *usolves-ode* f *from* $t1)$ $((+)\ (t1 - t0)\ `\ T)$ X
⟨*proof*⟩

lemma *three-intervals-lemma*:
fixes $a\ b\ c$::*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 $c \in I$
⟨*proof*⟩

lemma *connection-usolves-ode*:
assumes *x*: $(x$ *usolves-ode* f *from* $tx)$ T X
assumes *y*: $\bigwedge t.\ t \in$ *closure* $S \cap$ *closure* $T \Longrightarrow (y$ *usolves-ode* g *from* $t)$ S X
assumes *conn-T*: *closure* $S \cap$ *closure* $T \subseteq T$
assumes *conn-S*: *closure* $S \cap$ *closure* $T \subseteq S$
assumes *conn-t*: $t \in$ *closure* $S \cap$ *closure* T
assumes *conn-x*: $\bigwedge t.\ t \in$ *closure* $S \Longrightarrow t \in$ *closure* $T \Longrightarrow x\ t = y\ t$
assumes *conn-f*: $\bigwedge t\ x.\ t \in$ *closure* $S \Longrightarrow t \in$ *closure* $T \Longrightarrow x \in X \Longrightarrow f\ t\ x = g\ t\ x$
shows $((\lambda t.$ *if* $t \in T$ *then* $x\ t$ *else* $y\ t)$ *usolves-ode* $(\lambda t.$ *if* $t \in T$ *then* $f\ t$ *else* $g\ t)$ *from* $tx)$ $(T \cup S)$ X
⟨*proof*⟩

lemma *usolves-ode-union-closed*:
assumes *x*: $(x$ *usolves-ode* f *from* $tx)$ T X

assumes *y*: $\bigwedge t.\ t \in \mathit{closure}\ S \cap \mathit{closure}\ T \Longrightarrow (x\ \mathit{usolves\text{-}ode}\ f\ \mathit{from}\ t)\ S\ X$
assumes *conn-T*: $\mathit{closure}\ S \cap \mathit{closure}\ T \subseteq T$
assumes *conn-S*: $\mathit{closure}\ S \cap \mathit{closure}\ T \subseteq S$
assumes *conn-t*: $t \in \mathit{closure}\ S \cap \mathit{closure}\ T$
shows $(x\ \mathit{usolves\text{-}ode}\ f\ \mathit{from}\ tx)\ (T \cup S)\ X$
⟨*proof*⟩

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

lemma *usolves-ode-subset-range*:
assumes *x*: $(x\ \mathit{usolves\text{-}ode}\ f\ \mathit{from}\ t0)\ T\ X$
assumes *r*: $x\ `\ T \subseteq Y$ **and** $Y \subseteq X$
shows $(x\ \mathit{usolves\text{-}ode}\ f\ \mathit{from}\ t0)\ T\ Y$
⟨*proof*⟩

2.3 ivp on interval

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

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

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

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

lemma *solution-fixed-pointI*:
fixes *x*:: *real* ⇒ *'a*::*banach*
assumes *cont-f*: $\mathit{continuous\text{-}on}\ (T \times X)\ (\lambda(t, x).\ f\ t\ x)$
assumes *cont-x*: $\mathit{continuous\text{-}on}\ T\ x$
assumes *defined*: $\bigwedge t.\ t \in T \Longrightarrow x\ t \in X$
assumes *fp*: $\bigwedge t.\ t \in T \Longrightarrow x\ t = x\ t0 + \mathit{ivl\text{-}integral}\ t0\ t\ (\lambda t.\ f\ t\ (x\ t))$

shows (*x solves-ode f*) *T X*
⟨*proof*⟩

end

lemma *solves-ode-half-open-segment-continuation*:
fixes *f*::*real* ⇒ *'a* ⇒ *'a*::*banach*
assumes *ode*: (*x solves-ode f*) {*t0* −−< *t1*} *X*
assumes *continuous*: *continuous-on* ({*t0* −− *t1*} × *X*) (λ(*t*, *x*). *f t x*)
assumes *compact X*
assumes *t0* ≠ *t1*
obtains *l* **where**
(*x* ⟶ *l*) (*at t1 within* {*t0* −−< *t1*})
((λ*t*. *if t* = *t1 then l else x t*) *solves-ode f*) {*t0* −− *t1*} *X*
⟨*proof*⟩

2.4 Picard-Lindeloef on set of functions into closed set

locale *continuous-rhs* = **fixes** *T X f*
assumes *continuous*: *continuous-on* (*T* × *X*) (λ(*t*, *x*). *f t x*)
begin

lemma *continuous-rhs-comp*[*continuous-intros*]:
assumes [*continuous-intros*]: *continuous-on S g*
assumes [*continuous-intros*]: *continuous-on S h*
assumes *g* ‘ *S* ⊆ *T*
assumes *h* ‘ *S* ⊆ *X*
shows *continuous-on S* (λ*x*. *f* (*g x*) (*h x*))
⟨*proof*⟩

end

locale *global-lipschitz* =
fixes *T X f* **and** *L*::*real*
assumes *lipschitz*: ⋀*t*. *t* ∈ *T* ⟹ *L*−*lipschitz-on X* (λ*x*. *f t x*)

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

locale *interval* = **fixes** *T*::*real set*
assumes *interval*: *is-interval T*
begin

lemma *closed-segment-subset-domain*: *t0* ∈ *T* ⟹ *t* ∈ *T* ⟹ *closed-segment t0 t* ⊆ *T*
⟨*proof*⟩

lemma *closed-segment-subset-domainI*: *t0* ∈ *T* ⟹ *t* ∈ *T* ⟹ *s* ∈ *closed-segment t0 t* ⟹ *s* ∈ *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*: $t \in T \Longrightarrow tmin \leq t\ tmin \in T$
and *tmax*: $t \in T \Longrightarrow t \leq tmax\ tmax \in T$
⟨*proof*⟩

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

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

lemma *mem-T-I*[*intro*, *simp*]: $tmin \leq t \Longrightarrow t \leq tmax \Longrightarrow t \in T$
⟨*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*:
$\bigwedge x\ t.\ t \in T \Longrightarrow x\ t0 = x0 \Longrightarrow x \in$ *closed-segment t0 t* $\rightarrow X \Longrightarrow$
continuous-on (*closed-segment t0 t*) $x \Longrightarrow x\ t0 +$ *ivl-integral t0 t* ($\lambda t.\ f\ t\ (x\ t)) \in X$
begin

sublocale *nonempty-set T* ⟨*proof*⟩

lemma *closed-segment-iv-subset-domain*: $t \in T \Longrightarrow$ *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* + *ivl-integral t0 t* (λ*t*. *f t* (*x t*))

definition *P*::(*real* ⇒$_C$ *'a*) ⇒ (*real* ⇒$_C$ *'a*)
  where *P x* = (*SOME g*::*real*⇒$_C$ *'a*.
    (∀ *t* ∈ *T*. *g t* = *P-inner x t*) ∧
    (∀ *t*≤*tmin*. *g t* = *P-inner x tmin*) ∧
    (∀ *t*≥*tmax*. *g t* = *P-inner x tmax*))

lemma *cont-P-inner-ivl*:
  x ∈ *T* →$_C$ *X* ⟹ *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* ∈ {*tmin*−−*tmax*} **and** *cs-tmin-tmax-subset*: {*tmin*−−*tmax*} ⊆ *T*
  ⟨*proof*⟩

lemma
  P-eqs:
  assumes *x* ∈ *T* →$_C$ *X*
  shows *P-eq-P-inner*: *t* ∈ *T* ⟹ *P x t* = *P-inner x t*
    and *P-le-tmin*: *t* ≤ *tmin* ⟹ *P x t* = *P-inner x tmin*
    and *P-ge-tmax*: *t* ≥ *tmax* ⟹ *P x t* = *P-inner x tmax*
  ⟨*proof*⟩

lemma *P-if-eq*:
  x ∈ *T* →$_C$ *X* ⟹
    P x t = (*if tmin* ≤ *t* ∧ *t* ≤ *tmax then P-inner x t else if t* ≥ *tmax then P-inner x tmax else P-inner x tmin*)
  ⟨*proof*⟩

lemma *dist-P-le*:

assumes *y*: $y \in T \to_C X$ **and** *z*: $z \in T \to_C X$
assumes *le*: $\bigwedge t.\ tmin \le t \Longrightarrow t \le tmax \Longrightarrow dist\ (P\text{-}inner\ y\ t)\ (P\text{-}inner\ z\ t) \le R$
assumes $0 \le R$
shows $dist\ (P\ y\ t)\ (P\ z\ t) \le R$
⟨*proof*⟩

lemma *P-def'*:
assumes $t \in T$
assumes $fixed\text{-}point \in T \to_C X$
shows $(P\ fixed\text{-}point)\ t = x0 + ivl\text{-}integral\ t0\ t\ (\lambda x.\ f\ x\ (fixed\text{-}point\ x))$
⟨*proof*⟩

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

lemma *iter-spaceI*:
assumes $g \in T \to_C X\ g\ t0 = x0$
shows $g \in iter\text{-}space$
⟨*proof*⟩

lemma *iter-spaceD*:
assumes $g \in iter\text{-}space$
shows $g \in T \to_C X\ apply\text{-}bcontfun\ g\ t0 = x0$
⟨*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** $a \le x \Longrightarrow x \le b \Longrightarrow clamp\ a\ b\ x = x$
⟨*proof*⟩

lemma *P-self-mapping*:
assumes *in-space*: $g \in iter\text{-}space$
shows $P\ g \in iter\text{-}space$
⟨*proof*⟩

lemma *continuous-on-T*: $continuous\text{-}on\ \{tmin\ ..\ tmax\}\ g \Longrightarrow continuous\text{-}on\ T\ g$
⟨*proof*⟩

lemma *T-closed-segment-subsetI*[*intro*, *simp*]: $t \in \{tmin--tmax\} \Longrightarrow t \in T$
and *T-subsetI*[*intro*, *simp*]: $tmin \le t \Longrightarrow t \le tmax \Longrightarrow t \in T$
⟨*proof*⟩

lemma *t0-mem-closed-segment*[*intro*, *simp*]: $t0 \in \{tmin -- tmax\}$
⟨*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*: $(apply\text{-}bcontfun\ x\ solves\text{-}ode\ f)\ T\ X$
assumes *iv*: $x\ t0 = x0$
assumes *t*: $t \in T$
shows $P\ x\ t = x\ t$
⟨*proof*⟩

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

end

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

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

lemma *lipschitz-P*:
shows $((tmax - tmin) * L)\text{-}lipschitz\text{-}on\ iter\text{-}space\ P$
⟨*proof*⟩

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

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

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

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

lemma *fixed-point-equality'*: $x \in \textit{iter-space} \wedge P\ x = x \Longrightarrow \textit{fixed-point} = x$
⟨*proof*⟩

lemma *fixed-point-equality*: $x \in \textit{iter-space} \Longrightarrow P\ x = x \Longrightarrow \textit{fixed-point} = x$
⟨*proof*⟩

lemma *fixed-point-iv*: *fixed-point t0* = *x0*
and *fixed-point-domain*: $x \in T \Longrightarrow \textit{fixed-point}\ x \in X$
⟨*proof*⟩

lemma *fixed-point-has-vderiv-on*: (*fixed-point has-vderiv-on* (λ*t*. *f t* (*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 solves-ode f*) *T X*
assumes *iv*: *x t0* = *x0*
assumes *t*: $t \in T$
shows *x t* = *fixed-point t*
⟨*proof*⟩

lemma *solves-ode-on-closed-segment-equals-fixed-point*:
assumes *ode*: (*x solves-ode f*) {*t0* −− *t1'*} *X*
assumes *iv*: *x t0* = *x0*
assumes *subset*: {*t0*−−*t1'*} ⊆ *T*
assumes *t-mem*: $t \in \{t0--t1'\}$
shows *x t* = *fixed-point t*
⟨*proof*⟩

lemma *unique-solution*:
assumes *ivp1*: (*x solves-ode f*) *T X x t0* = *x0*
assumes *ivp2*: (*y 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*::*real*
assumes *b* ∈ *closed-segment a c*
shows *closed-segment a b* ∪ *closed-segment b c* = *closed-segment a c*
⟨*proof*⟩

lemma *closed-segment-closed-segment-subset*:
fixes *s*::*real* **and** *i*::*nat*
assumes *s* ∈ *closed-segment a b*
assumes *a* ∈ *closed-segment c d b* ∈ *closed-segment c d*
shows *s* ∈ *closed-segment c d*
⟨*proof*⟩

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

context— solution until t1
fixes *t1*::*real*
assumes *mem-t1*: *t1* ∈ *T*
begin

lemma *subdivide-count-ex*: ∃ *n*. *L* ∗ *abs* (*t1* − *t0*) / (*Suc n*) < *1*
⟨*proof*⟩

definition *subdivide-count* = (*SOME n*. *L* ∗ *abs* (*t1* − *t0*) / *Suc n* < *1*)

lemma *subdivide-count*: *L* ∗ *abs* (*t1* − *t0*) / *Suc subdivide-count* < *1*
⟨*proof*⟩

lemma *subdivide-lipschitz*:
assumes |*s* − *t*| ≤ *abs* (*t1* − *t0*) / *Suc subdivide-count*
shows |*s* − *t*| ∗ *L* < *1*
⟨*proof*⟩

lemma *subdivide-lipschitz-lemma*:
assumes *st*: *s* ∈ {*a* −− *b*} *t* ∈ {*a* −− *b*}
assumes *abs* (*b* − *a*) ≤ *abs* (*t1* − *t0*) / *Suc subdivide-count*
shows |*s* − *t*| ∗ *L* < *1*
⟨*proof*⟩

definition *step* = (*t1* − *t0*) / *Suc subdivide-count*

lemma *last-step*: *t0* + *real* (*Suc subdivide-count*) ∗ *step* = *t1*
⟨*proof*⟩

lemma *step-in-segment*:
assumes *0* ≤ *i i* ≤ *real* (*Suc subdivide-count*)
shows *t0* + *i* ∗ *step* ∈ *closed-segment t0 t1*
⟨*proof*⟩

lemma *subset-T1*:
fixes *s*::*real* **and** *i*::*nat*
assumes $s \in$ *closed-segment t0* $(t0 + i * step)$
assumes $i \leq$ *Suc subdivide-count*
shows $s \in \{t0 -- t1\}$
⟨*proof*⟩

lemma *subset-T*: $\{t0 -- t1\} \subseteq T$ **and** *subset-TI*: $s \in \{t0 -- t1\} \Longrightarrow s \in T$
⟨*proof*⟩

primrec *psolution*::*nat* ⇒ *real* ⇒ *'a* **where**
psolution 0 t = *x0*
| *psolution* (*Suc i*) *t* = *unique-on-bounded-closed.fixed-point*
(*t0* + *real i* ∗ *step*) {*t0* + *real i* ∗ *step* −− *t0* + *real* (*Suc i*) ∗ *step*}
(*psolution i* (*t0* + *real i* ∗ *step*)) *f X t*

definition *psolutions t* = *psolution* (*LEAST i*. *t* ∈ *closed-segment* (*t0* + *real* (*i* − *1*) ∗ *step*) (*t0* + *real i* ∗ *step*)) *t*

lemma *psolutions-usolves-until-step*:
assumes *i-le*: $i \leq$ *Suc subdivide-count*
shows (*psolutions usolves-ode f from t0*) (*closed-segment t0* (*t0* + *real i* ∗ *step*)) *X*
⟨*proof*⟩

lemma *psolutions-usolves-ode*: (*psolutions usolves-ode f from t0*) {*t0* −− *t1*} *X*
⟨*proof*⟩

end

definition *solution t* = (*if t* ≤ *t0 then psolutions tmin t else psolutions tmax t*)

lemma *solution-eq-left*: *tmin* ≤ *t* ⟹ *t* ≤ *t0* ⟹ *solution t* = *psolutions tmin t*
⟨*proof*⟩

lemma *solution-eq-right*: *t0* ≤ *t* ⟹ *t* ≤ *tmax* ⟹ *solution t* = *psolutions tmax t*
⟨*proof*⟩

lemma *solution-usolves-ode*: (*solution usolves-ode f from t0*) *T X*
⟨*proof*⟩

lemma *solution-solves-ode*: (*solution solves-ode f*) *T X*
⟨*proof*⟩

lemma *solution-iv*[*simp*]: *solution t0* = *x0*
⟨*proof*⟩

end

2.5 Picard-Lindeloef 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*
  ⟨*proof*⟩

end

2.6 Picard-Lindeloef 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*: $B \geq 0$
⟨*proof*⟩

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*: $\bigwedge \xi.\ \xi \in$ *closed-segment t0 t* $\Longrightarrow x\ \xi \in X \Longrightarrow y\ \xi \in X$
  shows $x\ t \in$ *cball x0* ($B * abs\ (t - t0)$)
⟨*proof*⟩

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*: $\bigwedge \xi.\ \xi \in$ *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 $t \in T\ x\ t0 = x0\ x \in \{t0 --\ t\} \to X$
assumes *cont*[*continuous-intros*]: *continuous-on* $(\{t0 --\ t\})\ x$
shows $x\ t0 +$ *ivl-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 usolves-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*) ⟹
((λ*x*. *f* (*g x*) (*h x*)) *has-derivative* (λ*y*. *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-Lindeloef-Qualitative*
imports *Initial-Value-Problem*
begin

2.7 Picard-Lindeloef 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 $x\ y :: 'a::euclidean\text{-}space$
assumes $r > 0$
assumes $\bigwedge i.\ i \in Basis \Longrightarrow dist\ (x \cdot i)\ (y \cdot i) \leq r\ /\ sqrt(DIM('a))$
shows $y \in cball\ x\ r$
⟨*proof*⟩

lemma *cbox-in-cball'*:
fixes $x::'a::euclidean\text{-}space$
assumes $0 < r$
shows $\exists b > 0.\ b \leq r \wedge (\exists B.\ B = (\sum i \in Basis.\ b *_R i) \wedge (\forall y \in cbox\ (x - B)\ (x + B).\ y \in cball\ x\ r))$
⟨*proof*⟩

lemma *Pair1-in-Basis*: $i \in Basis \Longrightarrow (i, 0) \in Basis$
and *Pair2-in-Basis*: $i \in Basis \Longrightarrow (0, i) \in Basis$
⟨*proof*⟩

lemma *le-real-sqrt-sumsq'* [*simp*]: $y \leq sqrt\ (x * x + y * y)$
⟨*proof*⟩

lemma *cball-Pair-split-subset*: $cball\ (a, b)\ c \subseteq cball\ a\ c \times cball\ b\ c$
⟨*proof*⟩

lemma *cball-times-subset*: $cball\ a\ (c/2) \times cball\ b\ (c/2) \subseteq cball\ (a, b)\ c$
⟨*proof*⟩

lemma *eventually-bound-pairE*:
assumes $isCont\ f\ (t0, x0)$
obtains B **where**
$B \geq 1$
$eventually\ (\lambda e.\ \forall x \in cball\ t0\ e \times cball\ x0\ e.\ norm\ (f\ x) \leq B)\ (at\text{-}right\ 0)$
⟨*proof*⟩

lemma
eventually-in-cballs:
assumes $d > 0\ c > 0$
shows $eventually\ (\lambda e.\ cball\ t0\ (c * e) \times (cball\ x0\ e) \subseteq cball\ (t0, x0)\ d)\ (at\text{-}right\ 0)$
⟨*proof*⟩

lemma *cball-eq-sing'*:
fixes $x :: 'a::\{metric\text{-}space, perfect\text{-}space\}$
shows $cball\ x\ e = \{y\} \longleftrightarrow e = 0 \wedge x = y$
⟨*proof*⟩

locale *ll-on-open* = *interval* T **for** T +
fixes $f::real \Rightarrow 'a::\{banach, heine\text{-}borel\} \Rightarrow 'a$ **and** X
assumes *local-lipschitz*: $local\text{-}lipschitz\ T\ X\ f$

assumes *cont*: $\bigwedge x.\ x \in X \Longrightarrow$ *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 = $\{(x,\ t1).\ \{t0--t1\} \subseteq T \wedge x\ t0 = x0 \wedge (x$ *solves-ode* $f)\ \{t0--t1\}\ X\}$

the maximal existence interval

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

witness flow

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

unique flow

definition *flow* **where** *flow t0 x0* = $(\lambda t.$ *if* $t \in$ *existence-ivl t0 x0 then csol t0 x0 t t 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*::*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* ⟨*proof*⟩

sublocale *continuous-rhs T X f*
⟨*proof*⟩

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
⟨*proof*⟩

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

$L-$*lipschitz-on* (*cball x0 u* $\cap$ *X*) (λ*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* $\Longrightarrow$ *L*$-$*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* $\Longrightarrow$ *x0* $\in$ *X* $\Longrightarrow$ *csols t0 x0* $\neq$ {}
⟨*proof*⟩

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

lemma *existence-ivl-empty1*[*simp*]: *t0* $\notin$ *T* $\Longrightarrow$ *existence-ivl t0 x0* = {}
and *existence-ivl-empty2*[*simp*]: *x0* $\notin$ *X* $\Longrightarrow$ *existence-ivl t0 x0* = {}
⟨*proof*⟩

lemma *flow-undefined*:
shows *t0* $\notin$ *T* $\Longrightarrow$ *flow t0 x0* = (λ-. *0*)
x0 $\notin$ *X* $\Longrightarrow$ *flow t0 x0* = (λ-. *0*)
⟨*proof*⟩

lemma (**in** *ll-on-open*) *flow-eq-in-existence-ivlI*:
assumes $\bigwedge$*u. x0* $\in$ *X* $\Longrightarrow$ *u* $\in$ *existence-ivl t0 x0* $\longleftrightarrow$ *g u* $\in$ *existence-ivl s0 x0*
assumes $\bigwedge$*u. x0* $\in$ *X* $\Longrightarrow$ *u* $\in$ *existence-ivl t0 x0* $\Longrightarrow$ *flow t0 x0 u* = *flow s0 x0* (*g u*)
shows *flow t0 x0* = (λ*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 $t \in$ *existence-ivl t0 x0*
shows *t0* $\in$ *T x0* $\in$ *X*
⟨*proof*⟩

lemma *csol-mem-csols*:
assumes $t \in$ *existence-ivl t0 x0*
shows (*csol t0 x0 t*, *t*) $\in$ *csols t0 x0*
⟨*proof*⟩

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

lemma *existence-ivl-initial-time-iff*[*simp*]: *t0* $\in$ *existence-ivl t0 x0* $\longleftrightarrow$ *t0* $\in$ *T* $\wedge$ *x0* $\in$ *X*
⟨*proof*⟩

lemma *existence-ivl-initial-time*: *t0* $\in$ *T* $\Longrightarrow$ *x0* $\in$ *X* $\Longrightarrow$ *t0* $\in$ *existence-ivl t0 x0*
⟨*proof*⟩

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

lemma *existence-ivl-subset*:
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 $\in$ *existence-ivl t0 x0* $\Longrightarrow$ *s* $\in$ $\{t0 --$ *t*$\}$ $\Longrightarrow$ *s* $\in$ *existence-ivl t0 x0*
⟨*proof*⟩

lemma *segment-subset-existence-ivl*:
assumes *s* $\in$ *existence-ivl t0 x0 t* $\in$ *existence-ivl t0 x0*
shows $\{s -- t\} \subseteq$ *existence-ivl t0 x0*
⟨*proof*⟩

lemma *flow-initial-time-if*: *flow t0 x0 t0* = (*if t0* $\in$ *T* $\wedge$ *x0* $\in$ *X then x0 else 0*)
⟨*proof*⟩

lemma *flow-initial-time*[*simp*]: *t0* $\in$ *T* $\Longrightarrow$ *x0* $\in$ *X* $\Longrightarrow$ *flow t0 x0 t0* = *x0*
⟨*proof*⟩

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

lemma *csols-unique*:
  assumes (*x*, *t1*) ∈ *csols t0 x0*
  assumes (*y*, *t2*) ∈ *csols t0 x0*
  shows ∀ *t* ∈ {*t0* −− *t1*} ∩ {*t0* −− *t2*}. *x t* = *y t*
⟨*proof*⟩

lemma *csol-unique*:
  assumes *t1*: *t1* ∈ *existence-ivl t0 x0*
  assumes *t2*: *t2* ∈ *existence-ivl t0 x0*
  assumes *t*: *t* ∈ {*t0* −− *t1*} *t* ∈ {*t0* −− *t2*}
  shows *csol t0 x0 t1 t* = *csol t0 x0 t2 t*
  ⟨*proof*⟩

lemma *flow-vderiv-on-left*:
  (*flow t0 x0 has-vderiv-on* (λ*x*. *f x* (*flow t0 x0 x*))) (*existence-ivl t0 x0* ∩ {..*t0*})
  ⟨*proof*⟩

lemma *flow-vderiv-on-right*:
  (*flow t0 x0 has-vderiv-on* (λ*x*. *f x* (*flow t0 x0 x*))) (*existence-ivl t0 x0* ∩ {*t0*..})
  ⟨*proof*⟩

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

lemma *flow-solves-ode*: *t0* ∈ *T* ⟹ *x0* ∈ *X* ⟹ (*flow t0 x0 solves-ode f*) (*existence-ivl t0 x0*) *X*
  ⟨*proof*⟩

lemma *equals-flowI*:
  assumes *t0* ∈ *T′*
    is-interval T′
    T′ ⊆ *existence-ivl t0 x0*
    (*z solves-ode f*) *T′ X*
    z t0 = *flow t0 x0 t0 t* ∈ *T′*
  shows *z t* = *flow t0 x0 t*
⟨*proof*⟩

lemma *existence-ivl-maximal-segment*:
  assumes (*x solves-ode f*) {*t0* −− *t*} *X x t0* = *x0*
  assumes {*t0* −− *t*} ⊆ *T*
  shows *t* ∈ *existence-ivl t0 x0*
  ⟨*proof*⟩

lemma *existence-ivl-maximal-interval*:
assumes (*x solves-ode f*) *S X x t0* = *x0*
assumes *t0* ∈ *S is-interval S S* ⊆ *T*
shows *S* ⊆ *existence-ivl t0 x0*
⟨*proof*⟩

lemma *maximal-existence-flow*:
assumes *sol*: (*x solves-ode f*) *K X* **and** *iv*: *x t0* = *x0*
assumes *is-interval K*
assumes *t0* ∈ *K*
assumes *K* ⊆ *T*
shows *K* ⊆ *existence-ivl t0 x0* ⋀*t*. *t* ∈ *K* ⟹ *flow t0 x0 t* = *x t*
⟨*proof*⟩

lemma *maximal-existence-flowI*:
assumes (*x has-vderiv-on* (λ*t*. *f t* (*x t*))) *K*
assumes ⋀*t*. *t* ∈ *K* ⟹ *x t* ∈ *X*
assumes *x t0* = *x0*
assumes *K*: *is-interval K t0* ∈ *K K* ⊆ *T*
shows *K* ⊆ *existence-ivl t0 x0* ⋀*t*. *t* ∈ *K* ⟹ *flow t0 x0 t* = *x t*
⟨*proof*⟩

lemma *flow-in-domain*: *t* ∈ *existence-ivl t0 x0* ⟹ *flow t0 x0 t* ∈ *X*
⟨*proof*⟩

lemma (**in** *ll-on-open*)
assumes *t* ∈ *existence-ivl s x*
assumes *x* ∈ *X*
assumes *auto*: ⋀*s t x*. *x* ∈ *X* ⟹ *f s x* = *f t x*
assumes *T* = *UNIV*
shows *mem-existence-ivl-shift-autonomous1*: *t* − *s* ∈ *existence-ivl 0 x*
and *flow-shift-autonomous1*: *flow s x t* = *flow 0 x* (*t* − *s*)
⟨*proof*⟩

lemma (**in** *ll-on-open*)
assumes *t* − *s* ∈ *existence-ivl 0 x*
assumes *x* ∈ *X*
assumes *auto*: ⋀*s t x*. *x* ∈ *X* ⟹ *f s x* = *f t x*
assumes *T* = *UNIV*
shows *mem-existence-ivl-shift-autonomous2*: *t* ∈ *existence-ivl s x*
and *flow-shift-autonomous2*: *flow s x t* = *flow 0 x* (*t* − *s*)
⟨*proof*⟩

lemma
flow-eq-rev:
assumes *t* ∈ *existence-ivl t0 x0*
shows *preflect t0 t* ∈ *ll-on-open.existence-ivl* (*preflect t0* ‘ *T*) (λ*t*. − *f* (*preflect t0 t*)) *X t0 x0*
flow t0 x0 t = *ll-on-open.flow* (*preflect t0* ‘ *T*) (λ*t*. − *f* (*preflect t0 t*)) *X t0 x0*

(*preflect t0 t*)
⟨*proof*⟩

lemma (**in** *ll-on-open*)
shows *rev-flow-eq*: $t \in$ *ll-on-open.existence-ivl* (*preflect t0* ‘ *T*) ($\lambda t.\ -f$ (*preflect t0 t*)) *X t0 x0* $\Longrightarrow$
ll-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-on-open.existence-ivl* (*preflect t0* ‘ *T*) ($\lambda t.\ -f$ (*preflect t0 t*)) *X t0 x0* $\longleftrightarrow$ *preflect t0 t* ∈ *existence-ivl t0 x0*
⟨*proof*⟩

lemma
shows *rev-existence-ivl-eq*: *ll-on-open.existence-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-ivl t0 x0* = *preflect t0* ‘ *ll-on-open.existence-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* ⇒ *'a* ⇒ *'a* **is** *blinfun-apply* ⟨*proof*⟩
lift-definition *Blinop*::(*'a*::*real-normed-vector* ⇒ *'a*) ⇒ *'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* ⇒ *'a* ⇒ *'a*]]

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

lift-definition *norm-blinop* :: *'a blinop* ⇒ *real* **is** *norm* ⟨*proof*⟩

lift-definition *minus-blinop* :: *'a blinop* ⇒ *'a blinop* ⇒ *'a blinop* **is** *minus* ⟨*proof*⟩

lift-definition *dist-blinop* :: *'a blinop* ⇒ *'a blinop* ⇒ *real* **is** *dist* ⟨*proof*⟩

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

definition *open-blinop* :: *'a blinop set* ⇒ *bool* **where**
open-blinop U = (∀ *x*∈*U*. ∀ *F* (*x'*, *y*) *in uniformity*. *x'* = *x* ⟶ *y* ∈ *U*)

lift-definition *uminus-blinop* :: *'a blinop* ⇒ *'a blinop* **is** *uminus* ⟨*proof*⟩

lift-definition *zero-blinop* :: *'a blinop* **is** *0* ⟨*proof*⟩

lift-definition *plus-blinop* :: *'a blinop* ⇒ *'a blinop* ⇒ *'a blinop* **is** *plus* ⟨*proof*⟩

lift-definition *scaleR-blinop*::*real* ⇒ *'a blinop* ⇒ *'a blinop* **is** *scaleR* ⟨*proof*⟩

lift-definition *sgn-blinop* :: *'a blinop* ⇒ *'a blinop* **is** *sgn* ⟨*proof*⟩

instance
⟨*proof*⟩
end

lemma *bounded-bilinear-blinop-apply*: *bounded-bilinear* (*$*)
⟨*proof*⟩

interpretation *blinop*: *bounded-bilinear* (*$*)
⟨*proof*⟩

lemma *blinop-eqI*: (⋀*i*. *x $ i* = *y $ i*) ⟹ *x* = *y*
⟨*proof*⟩

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*
⟨*proof*⟩

instance *blinop* :: (*euclidean-space*) *heine-borel*
⟨*proof*⟩

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* ⇒ *'a blinop* ⇒ *'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* ⇒ *'b*::*banach*
and *Df*::*'a* ⇒ *nat* ⇒ *'a* ⇒ *'a* ⇒ *'b*
assumes *n* > *0*
assumes *Df-Nil*: ⋀*a x*. *Df a 0 H H* = *f a*
assumes *Df-Cons*: ⋀*a i d*. *a* ∈ *closed-segment X* (*X* + *H*) ⟹ *i* < *n* ⟹
((λ*a*. *Df a i H H*) *has-derivative* (*Df a* (*Suc i*) *H*)) (*at a within G*)
assumes *cs*: *closed-segment X* (*X* + *H*) ⊆ *G*
defines *i* ≡ λ*x*.

$((1 - x) \hat{} (n - 1) / fact\ (n - 1)) *_R Df\ (X + x *_R H)\ n\ H\ H$
shows *multivariate-Taylor-has-integral*:
$(i\ has\text{-}integral\ f\ (X + H) - (\sum i{<}n.\ (1 / fact\ i) *_R Df\ X\ i\ H\ H))\ \{0..1\}$
and *multivariate-Taylor*:
$f\ (X + H) = (\sum i{<}n.\ (1 / fact\ i) *_R Df\ X\ i\ H\ H) + integral\ \{0..1\}\ i$
and *multivariate-Taylor-integrable*:
$i\ integrable\text{-}on\ \{0..1\}$
⟨*proof*⟩

4.1 Symmetric second derivative

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

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

lemma *symmetric-second-derivative-within*:
assumes $a \in G$
assumes $\bigwedge s\ t.\ s \in \{0..1\} \Longrightarrow t \in \{0..1\} \Longrightarrow a + s *_R i + t *_R j \in G$
shows $f''\ i\ j = f''\ j\ i$
⟨*proof*⟩

end

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

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

lemma *symmetric-second-derivative*:
assumes $a \in interior\ G$
shows $f''\ i\ j = f''\ j\ i$
⟨*proof*⟩

end

lemma
uniform-explicit-remainder-Taylor-1:
fixes $f::'a::\{banach,heine\text{-}borel,perfect\text{-}space\} \Rightarrow 'b::banach$
assumes $f'[derivative\text{-}intros]$: $\bigwedge x.\ x \in G \Longrightarrow (f\ has\text{-}derivative\ blinfun\text{-}apply\ (f'\ x))\ (at\ x)$
assumes $f'\text{-}cont$: $\bigwedge x.\ x \in G \Longrightarrow isCont\ f'\ x$
assumes *open* G
assumes $J \neq \{\}$ *compact* J $J \subseteq G$
assumes $e > 0$
obtains $d\ R$
where $d > 0$
$\bigwedge x\ z.\ f\ z = f\ x + f'\ x\ (z - x) + R\ x\ z$
$\bigwedge x\ y.\ x \in J \Longrightarrow y \in J \Longrightarrow dist\ x\ y < d \Longrightarrow norm\ (R\ x\ y) \leq e * dist\ x\ y$
continuous-on $(G \times G)$ $(\lambda(a,\ b).\ R\ a\ b)$
⟨*proof*⟩

TODO: rename, duplication?

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

lemma *symmetric-second-derivative-within*:
assumes $a \in G$ *open* G
assumes $\bigwedge s\ t.\ s \in \{0..1\} \Longrightarrow t \in \{0..1\} \Longrightarrow a + s *_R i + t *_R j \in G$
shows $f''\ i\ j = f''\ j\ i$
⟨*proof*⟩

end

locale *second-derivative-on-open* =
fixes $f::'a::real\text{-}normed\text{-}vector \Rightarrow 'b::banach$
and $f' :: 'a \Rightarrow 'a \Rightarrow 'b$
and $f'' :: 'a \Rightarrow 'a \Rightarrow 'b$
and $a :: 'a$

and $G :: 'a\ set$
assumes *first-fderiv*[*derivative-intros*]:
$\bigwedge a.\ a \in G \Longrightarrow (f\ has\text{-}derivative\ f'\ a)\ (at\ a)$
assumes *in-G*: $a \in G$ **and** *open-G*: *open G*
assumes *second-fderiv*[*derivative-intros*]:
$((\lambda x.\ f'\ x\ i)\ has\text{-}derivative\ f''\ i)\ (at\ a)$
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-Lindeloef-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*: $x = y \longleftrightarrow (\forall i.\ blinfun\text{-}apply\ x\ i = blinfun\text{-}apply\ y\ i)$
⟨*proof*⟩

notation *id-blinfun* (1_L)

lemma *blinfun-inverse-left*:
fixes f::$'a$::*euclidean-space* $\Rightarrow_L$ $'a$ **and** f'
shows $f\ o_L\ f' = 1_L \longleftrightarrow f'\ o_L\ f = 1_L$
⟨*proof*⟩

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

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

named-theorems *integrable-on-simps*

lemma *integrable-on-refl-ivl*[*intro*, *simp*]: *g integrable-on* {*b* .. (*b*::'*b*::*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*]: (λ*x*. *c*) *integrable-on closed-segment a* (*b*::*real*)
⟨*proof*⟩

lemma *integrable-ident-ivl*[*intro*, *simp*]: (λ*x*. *x*) *integrable-on closed-segment a* (*b*::*real*)
and *integrable-ident-cbox*[*intro*, *simp*]: (λ*x*. *x*) *integrable-on cbox a* (*b*::*real*)
⟨*proof*⟩

lemma *content-closed-segment-real*:
fixes *a b*::*real*
shows *content* (*closed-segment a b*) = *abs* (*b* − *a*)
⟨*proof*⟩

lemma *integral-const-closed-segment*:
fixes *a b*::*real*
shows *integral* (*closed-segment a b*) (λ*x*. *c*) = *abs* (*b* − *a*) $*_R$ *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*::*real*
shows *f integrable-on X* ⟹ (λ*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*: $dist\ (t0 + et)\ t0 = norm\ et$
⟨*proof*⟩

lemma *double-nonneg-le*:
fixes *a*::*real*
shows $a * 2 \leq b \Longrightarrow a \geq 0 \Longrightarrow a \leq b$
⟨*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
$0 < t\ 0 < u$
$cball\ t0\ t \subseteq existence\text{-}ivl\ t0\ x0$
$cball\ x0\ (2 * u) \subseteq X$
$\bigwedge t'.\ t' \in cball\ t0\ t \Longrightarrow L\text{-}lipschitz\text{-}on\ (cball\ x0\ (2 * u))\ (f\ t')$
$\bigwedge x.\ x \in cball\ x0\ u \Longrightarrow (flow\ t0\ x\ usolves\text{-}ode\ f\ from\ t0)\ (cball\ t0\ t)\ (cball\ x\ u)$
$\bigwedge x.\ x \in cball\ x0\ u \Longrightarrow cball\ x\ u \subseteq X$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *flow-has-vderiv-on*: (*flow t0 x0 has-vderiv-on* (λ*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* ∈ *U t0* ∈ *V* **and** *subs*: *U* ⊆ *T V* ⊆ *T*
assumes *ivls*: *is-interval U is-interval V*
assumes *iv*: *x t0* = *y t0*
assumes *mem*: *t* ∈ *U t* ∈ *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* ∈ *U* **and** *subs*: *U* ⊆ *T*
assumes *ivls*: *is-interval U*
assumes *iv*: *x t0* = *y t0*
assumes *mem*: *t* ∈ *U*
shows *x t* = *y t*
⟨*proof*⟩

lemma
assumes *s*: *s* ∈ *existence-ivl t0 x0*
assumes *t*: *t* + *s* ∈ *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* ∈ *existence-ivl t0 x0*
⟨*proof*⟩

lemma
assumes *t*: *t* ∈ *existence-ivl t0 x0*
shows *flows-reverse*: *flow t* (*flow t0 x0 t*) *t0* = *x0*
and *existence-ivl-reverse*: *t0* ∈ *existence-ivl t* (*flow t0 x0 t*)
⟨*proof*⟩

lemma *flow-has-derivative*:
assumes *t* ∈ *existence-ivl t0 x0*
shows (*flow t0 x0 has-derivative* (λ*i*. *i* ∗$_R$ *f t* (*flow t0 x0 t*))) (*at t*)
⟨*proof*⟩

lemma *flow-has-vector-derivative*:

assumes $t \in$ *existence-ivl t0 x0*
shows (*flow t0 x0 has-vector-derivative f t* (*flow t0 x0 t*)) (*at t*)
⟨*proof*⟩

lemma *flow-has-vector-derivative-at-0*:
assumes*t* ∈ *existence-ivl t0 x0*
shows ((λ*h*. *flow t0 x0* (*t* + *h*)) *has-vector-derivative f t* (*flow t0 x0 t*)) (*at 0*)
⟨*proof*⟩

lemma
assumes $t \in$ *existence-ivl t0 x0*
shows *closed-segment-subset-existence-ivl*: *closed-segment t0 t* ⊆ *existence-ivl t0 x0*
and *ivl-subset-existence-ivl*: {*t0* .. *t*} ⊆ *existence-ivl t0 x0*
and *ivl-subset-existence-ivl′*: {*t* .. *t0*} ⊆ *existence-ivl t0 x0*
⟨*proof*⟩

lemma *flow-fixed-point*:
assumes *t*: *t* ∈ *existence-ivl t0 x0*
shows *flow t0 x0 t* = *x0* + *ivl-integral t0 t* (λ*t*. *f t* (*flow t0 x0 t*))
⟨*proof*⟩

lemma *flow-continuous*: *t* ∈ *existence-ivl t0 x0* ⟹ *continuous* (*at t*) (*flow t0 x0*)
⟨*proof*⟩

lemma *flow-tendsto*: *t* ∈ *existence-ivl t0 x0* ⟹ (*ts* ⟶ *t*) *F* ⟹
((λ*s*. *flow t0 x0* (*ts s*)) ⟶ *flow t0 x0 t*) *F*
⟨*proof*⟩

lemma *flow-continuous-on*: *continuous-on* (*existence-ivl t0 x0*) (*flow t0 x0*)
⟨*proof*⟩

lemma *flow-continuous-on-intro*:
continuous-on s g ⟹
(⋀*xa*. *xa* ∈ *s* ⟹ *g xa* ∈ *existence-ivl t0 x0*) ⟹
continuous-on s (λ*xa*. *flow t0 x0* (*g xa*))
⟨*proof*⟩

lemma *f-flow-continuous*:
assumes *t* ∈ *existence-ivl t0 x0*
shows *isCont* (λ*t*. *f t* (*flow t0 x0 t*)) *t*
⟨*proof*⟩

lemma *exponential-initial-condition*:
assumes *y0*: *t* ∈ *existence-ivl t0 y0*
assumes *z0*: *t* ∈ *existence-ivl t0 z0*
assumes *Y* ⊆ *X*
assumes *remain*: ⋀*s*. *s* ∈ *closed-segment t0 t* ⟹ *flow t0 y0 s* ∈ *Y*
⋀*s*. *s* ∈ *closed-segment t0 t* ⟹ *flow t0 z0 s* ∈ *Y*

assumes *lipschitz*: $\bigwedge s.\ s \in$ *closed-segment t0 t* $\Longrightarrow$ *K−lipschitz-on Y* (*f s*)
shows *norm* (*flow t0 y0 t* − *flow t0 z0 t*) ≤ *norm* (*y0* − *z0*) ∗ *exp* ((*K* + *1*) ∗ *abs* (*t* − *t0*))
⟨*proof*⟩

lemma
existence-ivl-cballs:
assumes *iv-defined*: *t0* ∈ *T x0* ∈ *X*
obtains *t u L*
where
$\bigwedge y.\ y \in$ *cball x0 u* $\Longrightarrow$ *cball t0 t* ⊆ *existence-ivl t0 y*
$\bigwedge s\ y.\ y \in$ *cball x0 u* $\Longrightarrow$ *s* ∈ *cball t0 t* $\Longrightarrow$ *flow t0 y s* ∈ *cball y u*
L−lipschitz-on (*cball t0 t*×*cball x0 u*) (λ(*t*, *x*). *flow t0 x t*)
$\bigwedge y.\ y \in$ *cball x0 u* $\Longrightarrow$ *cball y u* ⊆ *X*
0 < *t 0* < *u*
⟨*proof*⟩

context
fixes *x0*
assumes *iv-defined*: *t0* ∈ *T x0* ∈ *X*
begin

lemma *existence-ivl-notempty*: *existence-ivl t0 x0* ≠ {}
⟨*proof*⟩

lemma *initial-time-bounds*:
shows *bdd-above* (*existence-ivl t0 x0*) $\Longrightarrow$ *t0* < *Sup* (*existence-ivl t0 x0*) (**is** *?a* $\Longrightarrow$ -)
and *bdd-below* (*existence-ivl t0 x0*) $\Longrightarrow$ *Inf* (*existence-ivl t0 x0*) < *t0* (**is** *?b* $\Longrightarrow$ -)
⟨*proof*⟩

lemma
flow-leaves-compact-ivl-right:
assumes *bdd*: *bdd-above* (*existence-ivl t0 x0*)
defines *b* ≡ *Sup* (*existence-ivl t0 x0*)
assumes *b* ∈ *T*
assumes *compact K*
assumes *K* ⊆ *X*
obtains *t* **where** *t* ≥ *t0 t* ∈ *existence-ivl t0 x0 flow t0 x0 t* ∉ *K*
⟨*proof*⟩

lemma
flow-leaves-compact-ivl-left:
assumes *bdd*: *bdd-below* (*existence-ivl t0 x0*)
defines *b* ≡ *Inf* (*existence-ivl t0 x0*)
assumes *b* ∈ *T*
assumes *compact K*
assumes *K* ⊆ *X*

obtains t **where** $t \leq t0\ t \in$ *existence-ivl t0 x0 flow t0 x0 t* $\notin K$
⟨*proof*⟩

lemma
sup-existence-maximal:
assumes $\bigwedge t.\ t0 \leq t \Longrightarrow t \in$ *existence-ivl t0 x0* $\Longrightarrow$ *flow t0 x0 t* $\in K$
assumes *compact K K* $\subseteq X$
assumes *bdd-above* (*existence-ivl t0 x0*)
shows *Sup* (*existence-ivl t0 x0*) $\notin T$
⟨*proof*⟩

lemma
inf-existence-minimal:
assumes $\bigwedge t.\ t \leq t0 \Longrightarrow t \in$ *existence-ivl t0 x0* $\Longrightarrow$ *flow t0 x0 t* $\in K$
assumes *compact K K* $\subseteq X$
assumes *bdd-below* (*existence-ivl t0 x0*)
shows *Inf* (*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*: $\bigwedge t.\ t \in T' \Longrightarrow t \in$ *existence-ivl t0 x0* $\Longrightarrow$ *flow t0 x0 t* $\in K$
assumes *K*: *compact K K* $\subseteq X$
shows $T' \subseteq$ *existence-ivl t0 x0*
⟨*proof*⟩

lemma
mem-compact-implies-subset-existence-interval:
assumes *iv-defined*: $t0 \in T\ x0 \in X$
assumes *mem-compact*: $\bigwedge t.\ t \in T \Longrightarrow t \in$ *existence-ivl t0 x0* $\Longrightarrow$ *flow t0 x0 t* $\in K$
assumes *K*: *compact K K* $\subseteq X$
shows $T \subseteq$ *existence-ivl t0 x0*
⟨*proof*⟩

lemma
global-right-existence-ivl-explicit:
assumes $b \geq t0$
assumes *b*: $b \in$ *existence-ivl t0 x0*
obtains $d\ K$ **where** $d > 0\ K > 0$
ball x0 d $\subseteq X$
$\bigwedge y.\ y \in$ *ball x0 d* $\Longrightarrow b \in$ *existence-ivl t0 y*
$\bigwedge t\ y.\ y \in$ *ball x0 d* $\Longrightarrow t \in \{t0\ ..\ b\} \Longrightarrow$
dist (*flow t0 x0 t*) (*flow t0 y t*) $\leq$ *dist x0 y* $*$ *exp* ($K *$ *abs* ($t - t0$))

⟨*proof*⟩

lemma
global-left-existence-ivl-explicit:
assumes $b \leq t0$
assumes *b*: $b \in$ *existence-ivl t0 x0*
assumes *iv-defined*: $t0 \in T\ x0 \in X$
obtains $d\ K$ **where** $d > 0\ K > 0$
ball x0 d $\subseteq X$
$\bigwedge y.\ y \in$ *ball x0 d* $\Longrightarrow b \in$ *existence-ivl t0 y*
$\bigwedge t\ y.\ y \in$ *ball x0 d* $\Longrightarrow t \in \{b\ ..\ t0\} \Longrightarrow$ *dist* (*flow t0 x0 t*) (*flow t0 y t*) $\leq$ *dist x0 y* $*$ *exp* ($K *$ *abs* ($t - t0$))
⟨*proof*⟩

lemma
global-existence-ivl-explicit:
assumes *a*: $a \in$ *existence-ivl t0 x0*
assumes *b*: $b \in$ *existence-ivl t0 x0*
assumes *le*: $a \leq b$
obtains $d\ K$ **where** $d > 0\ K > 0$
ball x0 d $\subseteq X$
$\bigwedge y.\ y \in$ *ball x0 d* $\Longrightarrow a \in$ *existence-ivl t0 y*
$\bigwedge y.\ y \in$ *ball x0 d* $\Longrightarrow b \in$ *existence-ivl t0 y*
$\bigwedge t\ y.\ y \in$ *ball x0 d* $\Longrightarrow t \in \{a\ ..\ b\} \Longrightarrow$
dist (*flow t0 x0 t*) (*flow t0 y t*) $\leq$ *dist x0 y* $*$ *exp* ($K *$ *abs* ($t - t0$))
⟨*proof*⟩

lemma *eventually-exponential-separation*:
assumes *a*: $a \in$ *existence-ivl t0 x0*
assumes *b*: $b \in$ *existence-ivl t0 x0*
assumes *le*: $a \leq b$
obtains K **where** $K > 0\ \forall_F\ y$ *in at x0*. $\forall\, t \in \{a..b\}$. *dist* (*flow t0 x0 t*) (*flow t0 y t*) $\leq$ *dist x0 y* $*$ *exp* ($K * |t - t0|$)
⟨*proof*⟩

lemma *eventually-mem-existence-ivl*:
assumes *b*: $b \in$ *existence-ivl t0 x0*
shows $\forall_F\ x$ *in at x0*. $b \in$ *existence-ivl t0 x*
⟨*proof*⟩

lemma *uniform-limit-flow*:
assumes *a*: $a \in$ *existence-ivl t0 x0*
assumes *b*: $b \in$ *existence-ivl t0 x0*
assumes *le*: $a \leq b$
shows *uniform-limit* $\{a\ ..\ b\}$ (*flow t0*) (*flow t0 x0*) (*at x0*)
⟨*proof*⟩

lemma *eventually-at-fst*:
assumes *eventually* P (*at* (*fst x*))

assumes P (fst x)
shows *eventually* ($\lambda h.$ P (fst h)) (at x)
⟨*proof*⟩

lemma *eventually-at-snd*:
assumes *eventually* P (at (snd x))
assumes P (snd x)
shows *eventually* ($\lambda h.$ P (snd h)) (at x)
⟨*proof*⟩

lemma
shows *open-state-space*: *open* (*Sigma* X (*existence-ivl t0*))
and *flow-continuous-on-state-space*:
continuous-on (*Sigma* X (*existence-ivl t0*)) ($\lambda(x,\ t).$ *flow t0* x t)
⟨*proof*⟩

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

lemma *flow-isCont-state-space*: $t \in$ *existence-ivl t0 x0* $\Longrightarrow$ *isCont* ($\lambda(x,\ t).$ *flow t0* x t) (*x0*, t)
⟨*proof*⟩

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

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

lemma *flow-unique*:
assumes $t \in$ *existence-ivl t0 x0*
assumes *phi t0* $=$ *x0*
assumes $\bigwedge t.\ t \in$ *existence-ivl t0 x0* $\Longrightarrow$ (*phi has-vector-derivative f t* (*phi t*)) (*at* t)
assumes $\bigwedge t.\ t \in$ *existence-ivl t0 x0* $\Longrightarrow$ *phi* $t \in X$
shows *flow t0 x0* t $=$ *phi* t
⟨*proof*⟩

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

assumes (*phi has-vderiv-on* (λ*t*. *f t* (*phi t*))) (*existence-ivl t0 x0*)
assumes ⋀*t*. *t* ∈ *existence-ivl t0 x0* ⟹ *phi t* ∈ *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* ∈ *J*
assumes *J-subset*: *J* ⊆ *F.existence-ivl 0 x0*
assumes *J-ivl*: *is-interval J*
assumes *F-lipschitz*: ⋀*t*. *t* ∈ *J* ⟹ *K*−*lipschitz-on X* (*F t*)
assumes *K-pos*: *0* < *K*
assumes *F-G-norm-ineq*: ⋀*t x*. *t* ∈ *J* ⟹ *x* ∈ *X* ⟹ *norm* (*F t x* − *G t x*) < *e*
begin

context begin

lemma *F-iv-defined*: *0* ∈ *T1 x0* ∈ *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 ∀ *t* ∈ *J* ∩ *G.existence-ivl 0 x0*. *norm* (*flow0 t* − *Y t*) ≤ *e* / *K* ∗ (*exp*(*K* ∗ |*t*|) − *1*)
⟨*proof*⟩

end

end

locale *auto-ll-on-open* =
fixes *f*::*'a*::{*banach*, *heine-borel*} ⇒ *'a* **and** *X*
assumes *auto-local-lipschitz*: *local-lipschitz UNIV X* (λ-::*real*. *f*)
assumes *auto-open-domain*[*intro*!, *simp*]: *open X*
begin

autonomous flow and existence interval

definition *flow0 x0 t* = *ll-on-open.flow UNIV* (λ-. *f*) *X 0 x0 t*

definition *existence-ivl0 x0* = *ll-on-open.existence-ivl UNIV* (λ-. *f*) *X 0 x0*

sublocale *ll-on-open-it UNIV* λ-. *f X 0*

rewrites *flow* = (λ*t0 x0 t*. *flow0 x0* (*t* − *t0*))
and *existence-ivl* = (λ*t0 x0*. (+) *t0* ‘ *existence-ivl0 x0*)
and (+) *0* = (λ*x*::*real*. *x*)
and *s* − *0* = *s*
and (λ*x*. *x*) ‘ *S* = *S*
and *s* ∈ (+) *t* ‘ *S* ⟷ *s* − *t* ∈ (*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?
⟨*proof*⟩

lemma *existence-ivl-zero*: *x0* ∈ *X* ⟹ *0* ∈ *existence-ivl0 x0* ⟨*proof*⟩

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 ⋀*t*. *0* ≤ *t* ⟹ *t* ∈ *existence-ivl0 x* ⟹ *flow0 x t* ∈ *K*
assumes *compact K K* ⊆ *X*
assumes *x* ∈ *X t* ≥ *0*
shows *t* ∈ *existence-ivl0 x*
⟨*proof*⟩

lemma
flow-in-compact-left-existence:
assumes ⋀*t*. *t* ≤ *0* ⟹ *t* ∈ *existence-ivl0 x* ⟹ *flow0 x t* ∈ *K*
assumes *compact K K* ⊆ *X*
assumes *x* ∈ *X t* ≤ *0*
shows *t* ∈ *existence-ivl0 x*
⟨*proof*⟩

end

locale *compact-continuously-diff* =
derivative-on-prod T X f λ(*t*, *x*). *f*′ *x* *o*$_L$ *snd-blinfun*
for *T X* **and** *f*::*real* ⇒ *'a*::{*banach*,*perfect-space*,*heine-borel*} ⇒ *'a*
and *f'*::*'a* ⇒ (*'a*, *'a*) *blinfun* +
assumes *compact-domain*: *compact X*
assumes *convex*: *convex X*
assumes *nonempty-domains*: *T* ≠ {} *X* ≠ {}
assumes *continuous-derivative*: *continuous-on X f*′
begin

lemma *ex-onorm-bound*:
∃ *B*. ∀ *x* ∈ *X*. *norm* (*f*′ *x*) ≤ *B*
⟨*proof*⟩

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

lemma *onorm-bound*: **assumes** $x \in X$ **shows** *norm* $(f'\ x) \leq$ *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*::{*banach*, *perfect-space*, *heine-borel*} ⇒ '*a* **and** *f' X*
assumes *open-dom*[*simp*]: *open X*
assumes *derivative-rhs*:
$\bigwedge x.\ x \in X \Longrightarrow$ (*f has-derivative blinfun-apply* (*f' x*)) (*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 has-derivative g'*) (*at x within s*)) $\Longrightarrow$ *g x* $\in X \Longrightarrow$
$((\lambda x.\ f\ (g\ x))$ *has-derivative*
$(\lambda xa.$ *blinfun-apply* (*f'* (*g x*)) (*g'* *xa*)))
(*at x within s*)
⟨*proof*⟩

sublocale *auto-ll-on-open*
⟨*proof*⟩

end — *?x* ∈ *X* ⟹ (*f has-derivative blinfun-apply* (*f′ ?x*)) (*at ?x*)

locale *c1-on-open-euclidean* = *c1-on-open f f′ X*
  for *f*::*'a*::*euclidean-space* ⇒ - **and** *f′ X*
begin
lemma *c1-on-open-euclidean-anchor*: *True* ⟨*proof*⟩

definition *vareq x0 t* = *f′* (*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 ⋀*s*. *s* ∈ *S* ⟹ *a s* ∈ *X*
  assumes ⋀*s*. *s* ∈ *S* ⟹ *b s* ∈ *existence-ivl0* (*a s*)
  shows *continuous-on S* (λ*s*. *vareq* (*a s*) (*b s*))
⟨*proof*⟩

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

context
  fixes *x0*::*'a*
begin

lemma *flow0-defined*: *xa* ∈ *existence-ivl0 x0* ⟹ *flow0 x0 xa* ∈ *X*
  ⟨*proof*⟩

lemma *continuous-on-flow0*: *continuous-on* (*existence-ivl0 x0*) (*flow0 x0*)
  ⟨*proof*⟩

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

lemma *varexivl-eq-exivl*:
  assumes *t* ∈ *existence-ivl0 x0*
  shows *var.existence-ivl x0 t a* = *existence-ivl0 x0*
⟨*proof*⟩

definition *vector-Dflow u0 t* ≡ *var.flow x0 0 u0 t*

qualified abbreviation *Y z t* ≡ *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 $t \in$ *existence-ivl0 x0*
shows *vector-Dflow* $(\alpha *_R a + \beta *_R b)\ t = \alpha *_R$ *vector-Dflow a t* $+ \beta *_R$ *vector-Dflow b t*
⟨*proof*⟩

lemma *linear-vector-Dflow*:
assumes $t \in$ *existence-ivl0 x0*
shows *linear* $(\lambda z.$ *vector-Dflow z t*$)$
⟨*proof*⟩

lemma *bounded-linear-vector-Dflow*:
assumes $t \in$ *existence-ivl0 x0*
shows *bounded-linear* $(\lambda z.$ *vector-Dflow z t*$)$
⟨*proof*⟩

lemma *vector-Dflow-continuous-on-time*: $x0 \in X \Longrightarrow$ *continuous-on* (*existence-ivl0 x0*) $(\lambda t.$ *vector-Dflow z t*$)$
⟨*proof*⟩

proposition *proposition-17-6-weak*:
— from "Differential Equations, Dynamical Systems, and an Introduction to Chaos", Hirsch/Smale/Devaney
assumes $t \in$ *existence-ivl0 x0*
shows $(\lambda y.$ $(Y\ (y - x0)\ t -$ *flow0 x0 t* $-$ *vector-Dflow* $(y - x0)\ t) /_R$ *norm* $(y - x0)) - x0 \to 0$
⟨*proof*⟩

lemma *local-lipschitz-A*:
$OT \subseteq$ *existence-ivl0 x0* $\Longrightarrow$ *local-lipschitz OT* $(OS{::}('a \Rightarrow_L 'a)\ set)$ $(\lambda t.$ (o_L) $(vareq\ x0\ t))$
⟨*proof*⟩

lemma *total-derivative-ll-on-open*:
ll-on-open (*existence-ivl0 x0*) $(\lambda t.$ *blinfun-compose* (*vareq x0 t*)) $(UNIV{::}('a \Rightarrow_L 'a)\ set)$
⟨*proof*⟩

end

end

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

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

tence-ivl0 ?x0.0 ⟹ var.existence-ivl ?x0.0 ?t ?a = existence-ivl0 ?x0.0
assumes *t ∈ existence-ivl0 x0*
shows *mvar.existence-ivl x0 t = (λ-. existence-ivl0 x0)*
⟨*proof*⟩

lemma
assumes *t ∈ existence-ivl0 x0*
shows *continuous-on (UNIV × existence-ivl0 x0) (λ(x, ta). mvar.flow x0 t x ta)*
⟨*proof*⟩

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

lemma *var-eq-mvar*:
assumes *t0 ∈ existence-ivl0 x0*
assumes *t ∈ existence-ivl0 x0*
shows *var.flow x0 t0 i t = mvar.flow x0 t0 id-blinfun t i*
⟨*proof*⟩

lemma *Dflow-zero*[*simp*]: *x ∈ X ⟹ Dflow x 0 = 1$_L$*
⟨*proof*⟩

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 ∈ existence-ivl0 x0*
shows *((λx0. flow0 x0 t) has-derivative (λz. vector-Dflow x0 z t)) (at x0)*
⟨*proof*⟩

lemma *flow-dx-derivative-blinfun*:
assumes *t ∈ existence-ivl0 x0*
shows *((λx. flow0 x t) has-derivative Blinfun (λz. vector-Dflow x0 z t)) (at x0)*
⟨*proof*⟩

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) ξ$_1$ + ξ$_2$ *$_R$ f (flow0 x0 t)*
⟨*proof*⟩

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!
⟨*proof*⟩

lemma *W-continuous-on-comp*[*continuous-intros*]:
assumes *h*: *continuous-on S h* **and** *g*: *continuous-on S g*

shows $(\bigwedge s.\ s \in S \Longrightarrow h\ s \in X) \Longrightarrow (\bigwedge s.\ s \in S \Longrightarrow g\ s \in \mathit{existence\text{-}ivl0}\ (h\ s)) \Longrightarrow$
continuous-on S $(\lambda s.\ \mathit{Dflow}\ (h\ s)\ (g\ s))$
⟨*proof*⟩

lemma *f-flow-continuous-on*: *continuous-on* $(\mathit{Sigma}\ X\ \mathit{existence\text{-}ivl0})\ (\lambda(x0,\ t).\ f\ (\mathit{flow0}\ x0\ t))$
⟨*proof*⟩

lemma
flow-has-space-derivative:
assumes $t \in \mathit{existence\text{-}ivl0}\ x0$
shows $((\lambda x0.\ \mathit{flow0}\ x0\ t)\ \mathit{has\text{-}derivative}\ \mathit{Dflow}\ x0\ t)\ (\mathit{at}\ x0)$
⟨*proof*⟩

lemma
flow-has-flowderiv:
assumes $t \in \mathit{existence\text{-}ivl0}\ x0$
shows $((\lambda(x0,\ t).\ \mathit{flow0}\ x0\ t)\ \mathit{has\text{-}derivative}\ \mathit{flowderiv}\ x0\ t)\ (\mathit{at}\ (x0,\ t)\ \mathit{within}\ S)$
⟨*proof*⟩

lemma *flow0-comp-has-derivative*:
assumes *h*: $h\ s \in \mathit{existence\text{-}ivl0}\ (g\ s)$
assumes [*derivative-intros*]: $(g\ \mathit{has\text{-}derivative}\ g')\ (\mathit{at}\ s\ \mathit{within}\ S)$
assumes [*derivative-intros*]: $(h\ \mathit{has\text{-}derivative}\ h')\ (\mathit{at}\ s\ \mathit{within}\ S)$
shows $((\lambda x.\ \mathit{flow0}\ (g\ x)\ (h\ x))\ \mathit{has\text{-}derivative}\ (\lambda x.\ \mathit{blinfun\text{-}apply}\ (\mathit{flowderiv}\ (g\ s)\ (h\ s))\ (g'\ x,\ h'\ x)))$
$(\mathit{at}\ s\ \mathit{within}\ S)$
⟨*proof*⟩

lemma *flowderiv-continuous-on*: *continuous-on* $(\mathit{Sigma}\ X\ \mathit{existence\text{-}ivl0})\ (\lambda(x0,\ t).\ \mathit{flowderiv}\ x0\ t)$
⟨*proof*⟩

lemma *flowderiv-continuous-on-comp*[*continuous-intros*]:
assumes *continuous-on S x*
assumes *continuous-on S t*
assumes $\bigwedge s.\ s \in S \Longrightarrow x\ s \in X\ \bigwedge s.\ s \in S \Longrightarrow t\ s \in \mathit{existence\text{-}ivl0}\ (x\ s)$
shows *continuous-on S* $(\lambda xa.\ \mathit{flowderiv}\ (x\ xa)\ (t\ xa))$
⟨*proof*⟩

lemmas [*intro*] = *flow-in-domain*

lemma *vareq-trans*: $t0 \in \mathit{existence\text{-}ivl0}\ x0 \Longrightarrow t \in \mathit{existence\text{-}ivl0}\ (\mathit{flow0}\ x0\ t0) \Longrightarrow \mathit{vareq}\ (\mathit{flow0}\ x0\ t0)\ t = \mathit{vareq}\ x0\ (t0 + t)$
⟨*proof*⟩

lemma *diff-existence-ivl-trans*:
$t0 \in \mathit{existence\text{-}ivl0}\ x0 \Longrightarrow t \in \mathit{existence\text{-}ivl0}\ x0 \Longrightarrow t - t0 \in \mathit{existence\text{-}ivl0}\ (\mathit{flow0}$

x0 t0) **for** *t*
⟨*proof*⟩

lemma *has-vderiv-on-blinfun-compose-right*[*derivative-intros*]:
assumes (*g has-vderiv-on g'*) *T*
assumes ⋀*x*. *x* ∈ *T* ⟹ *gd' x* = *g' x* o_L *d*
shows ((λ*x*. *g x* o_L *d*) *has-vderiv-on gd'*) *T*
⟨*proof*⟩

lemma *has-vderiv-on-blinfun-compose-left*[*derivative-intros*]:
assumes (*g has-vderiv-on g'*) *T*
assumes ⋀*x*. *x* ∈ *T* ⟹ *gd' x* = *d* o_L *g' x*
shows ((λ*x*. *d* o_L *g x*) *has-vderiv-on gd'*) *T*
⟨*proof*⟩

lemma *mvar-flow-shift*:
assumes *t0* ∈ *existence-ivl0 x0 t1* ∈ *existence-ivl0 x0*
shows *mvar.flow x0 t0 d t1* = *Dflow* (*flow0 x0 t0*) (*t1* − *t0*) o_L *d*
⟨*proof*⟩

lemma *Dflow-trans*:
assumes *h* ∈ *existence-ivl0 x0*
assumes *i* ∈ *existence-ivl0* (*flow0 x0 h*)
shows *Dflow x0* (*h* + *i*) = *Dflow* (*flow0 x0 h*) *i* o_L (*Dflow x0 h*)
⟨*proof*⟩

lemma *Dflow-trans-apply*:
assumes *h* ∈ *existence-ivl0 x0*
assumes *i* ∈ *existence-ivl0* (*flow0 x0 h*)
shows *Dflow x0* (*h* + *i*) *d0* = *Dflow* (*flow0 x0 h*) *i* (*Dflow 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* :: *real* ⇒ *'b* :: {*linorder-topology*,*real-normed-vector*,*ordered-real-vector*}
— generalize?
assumes *y*: *f a* ≤ *y y* ≤ *f b a* ≤ *b*
assumes ∗: *continuous-on* {*a* .. *b*} *f*

notes [*continuous-intros*] = ∗[*THEN continuous-on-subset*]
obtains x **where** $a \leq x$ $x \leq b$ $f\ x = y$ $\bigwedge x'.\ a \leq x' \Longrightarrow x' < x \Longrightarrow f\ x' < y$
⟨*proof*⟩

lemma *filtermap-at-left-shift*: *filtermap* $(\lambda x.\ x - d)$ (*at-left* a) = *at-left* ($a - d$::*real*)
⟨*proof*⟩

context
fixes $v\ v'\ w\ w'$::*real* ⇒ *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* ≤ *t1*
assumes *less*: $\bigwedge t.\ t0 < t \Longrightarrow t < t0 + e \Longrightarrow v\ t < w\ t$
begin

lemma *first-intersection-crossing-derivatives*:
assumes *na*: *t0* < *tg* *tg* ≤ *t1* *v tg* ≥ *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* *x0* ≤ *tg*
v' *x0* ≥ w' *x0*
v x0 = *w x0*
$\bigwedge t.\ t0 < t \Longrightarrow t < x0 \Longrightarrow v\ t < w\ t$
⟨*proof*⟩

lemma *defect-less*:
assumes *b*: $\bigwedge t.\ t0 < t \Longrightarrow t \leq t1 \Longrightarrow v'\ t - f\ t\ (v\ t) < w'\ t - f\ t\ (w\ t)$
notes [*continuous-intros*] =
vderiv-on-continuous-on[*OF* v', *THEN continuous-on-subset*]
vderiv-on-continuous-on[*OF* w', *THEN continuous-on-subset*]
shows $\forall\, t \in \{t0 <..\ t1\}.\ v\ t < w\ t$
⟨*proof*⟩

end

lemma *has-derivatives-less-lemma*:
fixes $v\ v'$ ::*real* ⇒ *real*
assumes v': (v *has-vderiv-on* v') T
assumes y': (y *has-vderiv-on* y') T
assumes *lu*: $\bigwedge t.\ t \in T \Longrightarrow t > t0 \Longrightarrow v'\ t - f\ t\ (v\ t) < y'\ t - f\ t\ (y\ t)$
assumes *lower*: *v t0* ≤ *y t0*
assumes *eq-imp*: *v t0* = *y t0* ⟹ v' *t0* < y' *t0*
assumes *t*: *t0* < *t* *t0* ∈ T *t* ∈ T *is-interval* T
shows *v t* < *y t*
⟨*proof*⟩

lemma *strict-lower-solution*:
fixes $v\ v'$ ::*real* ⇒ *real*
assumes *sol*: $(y$ *solves-ode* $f)\ T\ X$
assumes v': $(v$ *has-vderiv-on* $v')\ T$
assumes *lower*: $\bigwedge t.\ t \in T \Longrightarrow t > t0 \Longrightarrow v'\ t < f\ t\ (v\ t)$
assumes *iv*: $v\ t0 \leq y\ t0\ v\ t0 = y\ t0 \Longrightarrow v'\ t0 < f\ t0\ (y\ t0)$
assumes *t*: $t0 < t\ t0 \in T\ t \in T$ *is-interval* T
shows $v\ t < y\ t$
⟨*proof*⟩

lemma *strict-upper-solution*:
fixes $w\ w'$::*real* ⇒ *real*
assumes *sol*: $(y$ *solves-ode* $f)\ T\ X$
assumes w': $(w$ *has-vderiv-on* $w')\ T$
and *upper*: $\bigwedge t.\ t \in T \Longrightarrow t > t0 \Longrightarrow f\ t\ (w\ t) < w'\ t$
and *iv*: $y\ t0 \leq w\ t0\ y\ t0 = w\ t0 \Longrightarrow f\ t0\ (y\ t0) < w'\ t0$
assumes *t*: $t0 < t\ t0 \in T\ t \in T$ *is-interval* T
shows $y\ t < w\ t$
⟨*proof*⟩

lemma *uniform-limit-at-within-subset*:
assumes *uniform-limit* $S\ x\ l$ $(at\ t$ *within* $T)$
assumes $U \subseteq T$
shows *uniform-limit* $S\ x\ l$ $(at\ t$ *within* $U)$
⟨*proof*⟩

lemma *uniform-limit-le*:
fixes f::$'c \Rightarrow\ 'a \Rightarrow\ 'b$::{*metric-space*, *linorder-topology*}
assumes I: $I \neq bot$
assumes u: *uniform-limit* $X\ f\ g\ I$
assumes u': *uniform-limit* $X\ f'\ g'\ I$
assumes $\forall_F\ i$ *in* $I.\ \forall x \in X.\ f\ i\ x \leq f'\ i\ x$
assumes $x \in X$
shows $g\ x \leq g'\ x$
⟨*proof*⟩

lemma *uniform-limit-le-const*:
fixes f::$'c \Rightarrow\ 'a \Rightarrow\ 'b$::{*metric-space*, *linorder-topology*}
assumes I: $I \neq bot$
assumes u: *uniform-limit* $X\ f\ g\ I$
assumes $\forall_F\ i$ *in* $I.\ \forall x \in X.\ f\ i\ x \leq h\ x$
assumes $x \in X$
shows $g\ x \leq h\ x$
⟨*proof*⟩

lemma *uniform-limit-ge-const*:
fixes f::$'c \Rightarrow\ 'a \Rightarrow\ 'b$::{*metric-space*, *linorder-topology*}
assumes I: $I \neq bot$
assumes u: *uniform-limit* $X\ f\ g\ I$

assumes $\forall_F\ i\ in\ I.\ \forall x \in X.\ h\ x \leq f\ i\ x$
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::*real set*
begin

lemma *lower-solution*:
fixes $v\ v'$ ::*real* ⇒ *real*
assumes *sol*: $(y\ solves\text{-}ode\ f)\ S\ X$
assumes v': $(v\ has\text{-}vderiv\text{-}on\ v')\ S$
assumes *lower*: $\bigwedge t.\ t \in S \Longrightarrow t > t0 \Longrightarrow v'\ t < f\ t\ (v\ t)$
assumes *iv*: $v\ t0 \leq y\ t0$
assumes *t*: $t0 \leq t\ t0 \in S\ t \in S\ is\text{-}interval\ S\ S \subseteq T$
shows $v\ t \leq y\ t$
⟨*proof*⟩

lemma *upper-solution*:
fixes $v\ v'$ ::*real* ⇒ *real*
assumes *sol*: $(y\ solves\text{-}ode\ f)\ S\ X$
assumes v': $(v\ has\text{-}vderiv\text{-}on\ v')\ S$
assumes *upper*: $\bigwedge t.\ t \in S \Longrightarrow t > t0 \Longrightarrow f\ t\ (v\ t) < v'\ t$
assumes *iv*: $y\ t0 \leq v\ t0$
assumes *t*: $t0 \leq t\ t0 \in S\ t \in S\ is\text{-}interval\ S\ S \subseteq T$
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* $A.\ P\ x$
⟨*proof*⟩

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* $T.\ f\ t \notin P$
⟨*proof*⟩

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)
⟨*proof*⟩

end

context *auto-ll-on-open*— TODO: generalize to continuous systems
begin

definition *returns-to* ::′*a set* ⇒ ′*a* ⇒ *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* ⇒ ′*a* ⇒ *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*: $t > 0\ t \in$ *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* $P\ x$
⟨*proof*⟩

lemma *returns-to-outsideI*:
assumes *t*: $t \geq 0\ t \in$ *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 ≤ t1
t1 ∈ existence-ivl0 x
flow0 x t1 ∈ P
⋀*t*. *0 < t ⟹ t < t0 ⟹ flow0 x t ∉ P*
⟨*proof*⟩

lemma *return-time-some*:
assumes *returns-to P x*
shows *return-time P x =*
(SOME t. t > 0 ∧ t ∈ existence-ivl0 x ∧ flow0 x t ∈ P ∧ (∀ s ∈ {0<..<t}. flow0 x s ∉ P))
⟨*proof*⟩

lemma *return-time-ex1*:
assumes *returns-to P x*
assumes *closed P*
shows ∃!*t*. *t > 0 ∧ t ∈ existence-ivl0 x ∧ flow0 x t ∈ P ∧ (∀ s ∈ {0<..<t}. flow0 x s ∉ P)*
⟨*proof*⟩

lemma
return-time-pos-returns-to:
return-time P x > 0 ⟹ 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 ⟷ 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 ∈ existence-ivl0 x*
and *return-time-returns*: *flow0 x (return-time P x) ∈ P*
and *return-time-least*: ⋀*s*. *0 < s ⟹ s < return-time P x ⟹ flow0 x s ∉ P*

⟨*proof*⟩

lemma *returns-to-earlierI*:
assumes *ret*: *returns-to P* (*flow0 x t*) *closed P*
assumes $t \geq 0$ $t \in$ *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*: $\bigwedge s.\ 0 < s \Longrightarrow s \leq t \Longrightarrow$ *flow0 x s* $\notin$ *P*
shows $t <$ *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*: $t > 0$ $t \in$ *existence-ivl0 x*
assumes *flow-not*: $\bigwedge s.\ 0 < s \Longrightarrow s \leq t \Longrightarrow$ *flow0 x s* $\notin$ *P*
shows *returns-to P* (*flow0 x t*)
⟨*proof*⟩

lemma *never-returns*:
assumes ¬*returns-to P x*
assumes *closed P* $t \geq 0$ $t \in$ *existence-ivl0 x*
assumes *ev*: $\forall_F$ *t in at-right 0. flow0 x t* $\notin$ *P*
shows ¬*returns-to P* (*flow0 x t*)
⟨*proof*⟩

lemma *return-time-eqI*:
assumes *closed P*
and *t-pos*: $t > 0$
and *ex*: $t \in$ *existence-ivl0 x*
and *ret*: *flow0 x t* $\in$ *P*
and *least*: $\bigwedge s.\ 0 < s \Longrightarrow s < t \Longrightarrow$ *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*: $\bigwedge s.\ 0 < s \Longrightarrow s \leq t \Longrightarrow$ *flow0 x s* $\notin$ *P*
assumes *t*: $t > 0$ $t \in$ *existence-ivl0 x*

shows *return-time P (flow0 x t) = return-time P x − t*
⟨*proof*⟩

definition *poincare-map P x = flow0 x (return-time P x)*

lemma *poincare-map-step-flow*:
assumes *ret*: *returns-to P x closed P*
assumes *flow-not*: ⋀*s*. *0 < s ⟹ s ≤ t ⟹ flow0 x s ∉ P*
assumes *t*: *t > 0 t ∈ existence-ivl0 x*
shows *poincare-map P (flow0 x t) = poincare-map P x*
⟨*proof*⟩

lemma *poincare-map-returns*:
assumes *returns-to P x closed P*
shows *poincare-map P x ∈ P*
⟨*proof*⟩

lemma *poincare-map-onto*:
assumes *closed P*
assumes *0 < t t ∈ existence-ivl0 x* ∀ *F t in at-right 0. flow0 x t ∉ P*
assumes *flow0 x t ∈ P*
shows *poincare-map P x ∈ flow0 x ‘ {0 <.. t} ∩ P*
⟨*proof*⟩

end

lemma *isCont-blinfunD*:
fixes *f′*::*′a*::*metric-space ⇒ ′b*::*real-normed-vector ⇒*$_L$ *′c*::*real-normed-vector*
assumes *isCont f′ a 0 < e*
shows ∃ *d*>*0*. ∀ *x*. *dist a x < d ⟶ onorm* (*λv*. *blinfun-apply* (*f′ x*) *v − blinfun-apply* (*f′ a*) *v*) *< e*
⟨*proof*⟩

proposition *has-derivative-locally-injective-blinfun*:
fixes *f* :: *′n*::*euclidean-space ⇒ ′m*::*euclidean-space*
and *f′*::*′n ⇒ ′n ⇒*$_L$ *′m*
and *g′*::*′m ⇒*$_L$ *′n*
assumes *a ∈ s*
and *open s*
and *g′*: *g′ o*$_L$ (*f′ a*) = *1*$_L$
and *f′*: ⋀*x*. *x ∈ s ⟹* (*f has-derivative f′ x*) (*at x*)
and *c*: *isCont f′ a*
obtains *r* **where** *r > 0 ball a r ⊆ s inj-on f* (*ball a r*)
⟨*proof*⟩

lift-definition *embed1-blinfun*::*′a*::*real-normed-vector ⇒*$_L$ (*′a∗′b*::*real-normed-vector*)
is *λx*. (*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* ⇒$_L$ ('*b*::*real-normed-vector*∗'*a*) **is** λ*x*. (*0*, *x*)
⟨*proof*⟩
lemma *blinfun-apply-embed2-blinfun*[*simp*]: *blinfun-apply embed2-blinfun x* = (*0*, *x*)
⟨*proof*⟩

lemma *blinfun-inverseD*: *f o*$_L$ *f'* = *1*$_L$ ⟹ *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*: *ball a* (*c*/*2*) × *ball b* (*c*/*2*) ⊆ *ball* (*a*, *b*) *c*
⟨*proof*⟩

lemma *linear-inverse-blinop-lemma*:
fixes *w*::'*a*::{*banach*, *perfect-space*} *blinop*
assumes *norm w* < *1*
shows
summable (λ*n*. (−*1*)^*n* ∗$_R$ *w*^*n*) (**is** *?C*)
(∑ *n*. (−*1*)^*n* ∗$_R$ *w*^*n*) ∗ (*1* + *w*) = *1* (**is** *?I1*)
(*1* + *w*) ∗ (∑ *n*. (−*1*)^*n* ∗$_R$ *w*^*n*) = *1* (**is** *?I2*)
norm ((∑ *n*. (−*1*)^*n* ∗$_R$ *w*^*n*) − *1* + *w*) ≤ (*norm w*)^2/(*1* − *norm* (*w*)) (**is** *?L*)
⟨*proof*⟩

lemma *linear-inverse-blinfun-lemma*:
fixes *w*::'*a* ⇒$_L$ '*a*::{*banach*, *perfect-space*}
assumes *norm w* < *1*
obtains *I* **where**
I o$_L$ (*1*$_L$ + *w*) = *1*$_L$ (*1*$_L$ + *w*) *o*$_L$ *I* = *1*$_L$
norm (*I* − *1*$_L$ + *w*) ≤ (*norm w*)^2/(*1* − *norm* (*w*))
⟨*proof*⟩

definition *invertibles-blinfun* = {*w*. ∃ *wi*. *w o*$_L$ *wi* = *1*$_L$ ∧ *wi o*$_L$ *w* = *1*$_L$}

lemma *blinfun-inverse-open*:— 8.3.2 in Dieudonne, TODO: add continuity and derivative
shows *open* (*invertibles-blinfun*::
('*a*::{*banach*, *perfect-space*} ⇒$_L$ '*b*::*banach*) *set*)
⟨*proof*⟩

lemma *blinfun-compose-assoc*[*ac-simps*]: *a o*$_L$ *b o*$_L$ *c* = *a o*$_L$ (*b o*$_L$ *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* ⊆ *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* ⇒ *'c*::*euclidean-space*— TODO: generalize?!
assumes [*derivative-intros*]: ⋀*x*. *x* ∈ *S* ⟹ (*f has-derivative blinfun-apply* (*f' x*)) (*at x*)
assumes *S*: (*x*, *y*) ∈ *S open S*
assumes *DIM*(*'c*) ≤ *DIM*(*'b*)
assumes *f'C*: *isCont f'* (*x*, *y*)
assumes *f* (*x*, *y*) = *0*
assumes *T2*: *T* o_L (*f'* (*x*, *y*) o_L *embed2-blinfun*) = 1_L
assumes *T1*: (*f'* (*x*, *y*) o_L *embed2-blinfun*) o_L *T* = 1_L— TODO: reduce?!
obtains *u e r*
where *f* (*x*, *u x*) = *0 u x* = *y*
⋀*s*. *s* ∈ *cball x e* ⟹ *f* (*s*, *u s*) = *0*
continuous-on (*cball x e*) *u*

$(\lambda t.\ (t,\ u\ t))\ `\ cball\ x\ e \subseteq S$
$e > 0$
$(u\ has\text{-}derivative\ -\ T\ o_L\ f'\ (x,\ y)\ o_L\ embed1\text{-}blinfun)\ (at\ x)$

$r > 0$
$\bigwedge U\ v\ s.\ v\ x = y \Longrightarrow (\bigwedge s.\ s \in U \Longrightarrow f\ (s,\ v\ s) = 0) \Longrightarrow U \subseteq cball\ x\ e \Longrightarrow$
$continuous\text{-}on\ U\ v \Longrightarrow s \in U \Longrightarrow (s,\ v\ s) \in ball\ (x,\ y)\ r \Longrightarrow u\ s = v\ s$
⟨*proof*⟩

lemma *implicit-function-theorem-unique*:
fixes f::$'a$::*euclidean-space* $*$ $'b$::*euclidean-space* $\Rightarrow$ $'c$::*euclidean-space*— TODO: generalize?!
assumes f'[*derivative-intros*]: $\bigwedge x.\ x \in S \Longrightarrow (f\ has\text{-}derivative\ blinfun\text{-}apply\ (f'\ x))\ (at\ x)$
assumes *S*: $(x,\ y) \in S\ open\ S$
assumes *D*: $DIM('c) \leq DIM('b)$
assumes $f'C$: $continuous\text{-}on\ S\ f'$
assumes *z*: $f\ (x,\ y) = 0$
assumes *T2*: $T\ o_L\ (f'\ (x,\ y)\ o_L\ embed2\text{-}blinfun) = 1_L$
assumes *T1*: $(f'\ (x,\ y)\ o_L\ embed2\text{-}blinfun)\ o_L\ T = 1_L$— TODO: reduce?!
obtains *u e*
where $f\ (x,\ u\ x) = 0\ u\ x = y$
$\bigwedge s.\ s \in cball\ x\ e \Longrightarrow f\ (s,\ u\ s) = 0$
$continuous\text{-}on\ (cball\ x\ e)\ u$
$(\lambda t.\ (t,\ u\ t))\ `\ cball\ x\ e \subseteq S$
$e > 0$
$(u\ has\text{-}derivative\ (-\ T\ o_L\ f'\ (x,\ y)\ o_L\ embed1\text{-}blinfun))\ (at\ x)$
$\bigwedge s.\ s \in cball\ x\ e \Longrightarrow f'\ (s,\ u\ s)\ o_L\ embed2\text{-}blinfun \in invertibles\text{-}blinfun$
$\bigwedge U\ v\ s.\ (\bigwedge s.\ s \in U \Longrightarrow f\ (s,\ v\ s) = 0) \Longrightarrow$
$u\ x = v\ x \Longrightarrow$
$continuous\text{-}on\ U\ v \Longrightarrow s \in U \Longrightarrow x \in U \Longrightarrow U \subseteq cball\ x\ e \Longrightarrow connected\ U \Longrightarrow open\ U \Longrightarrow u\ s = v\ s$
⟨*proof*⟩

lemma *uniform-limit-compose*:
assumes *ul*: *uniform-limit T f l F*
assumes *uc*: *uniformly-continuous-on S s*
assumes *ev*: $\forall_F\ x\ in\ F.\ f\ x\ `\ T \subseteq S$
assumes *subs*: $l\ `\ T \subseteq S$
shows *uniform-limit T* $(\lambda i\ x.\ s\ (f\ i\ x))\ (\lambda x.\ s\ (l\ x))\ F$
⟨*proof*⟩

lemma
uniform-limit-in-open:
fixes l::$'a$::*topological-space*$\Rightarrow$$'b$::*heine-borel*
assumes *ul*: *uniform-limit T f l* $(at\ x)$
assumes *cont*: *continuous-on T l*
assumes *compact*: *compact T* **and** *T-ne*: $T \neq \{\}$
assumes *B*: *open B*

assumes *mem*: l ‘ $T \subseteq B$
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*⇒*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*⇒*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* ⇒ *'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*⇒*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*∈*S*
obtains *e* **where** *e*>*0 cball x e* ⊆ *S*
⟨*proof*⟩

lemma *pos-half-less*: **fixes** *x*::*real* **shows** $x > 0 \Longrightarrow x\ /\ 2 < x$
⟨*proof*⟩

lemma *closed-levelset*: *closed* $\{x.\ s\ x = (c::'a::t1\text{-}space)\}$ **if** *continuous-on UNIV s*
⟨*proof*⟩

lemma *closed-levelset-within*: *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 $t \in$ *existence-ivl0 x* $t \geq 0$
obtains *e* **where** $e > 0$ *cball x e* $\times \{0\ ..\ t + e\} \subseteq$ *Sigma X existence-ivl0*
⟨*proof*⟩

lemmas [*derivative-intros*] = *flow0-comp-has-derivative*

lemma *flow-isCont-state-space-comp*[*continuous-intros*]:
$t\ x \in$ *existence-ivl0* $(s\ x) \Longrightarrow$ *isCont s x* $\Longrightarrow$ *isCont t x* $\Longrightarrow$ *isCont* $(\lambda x.$ *flow0* $(s\ x)\ (t\ x))\ x$
⟨*proof*⟩

lemma *closed-plane*[*simp*]: *closed* $\{x.\ x \cdot i = c\}$
⟨*proof*⟩

lemma *flow-tendsto-compose*[*tendsto-intros*]:
assumes $(x \longrightarrow xs)\ F$ $(t \longrightarrow ts)\ F$
assumes $ts \in$ *existence-ivl0 xs*
shows $((\lambda s.$ *flow0* $(x\ s)\ (t\ s)) \longrightarrow$ *flow0 xs ts*$)\ F$
⟨*proof*⟩

lemma *returns-to-implicit-function*:
fixes $s::'a::euclidean\text{-}space \Rightarrow real$
assumes *rt*: *returns-to* $\{x \in S.\ s\ x = 0\}\ x$ (**is** *returns-to ?P x*)
assumes *cS*: *closed S*
assumes *Ds*: $\bigwedge x.$ $(s$ *has-derivative blinfun-apply* $(Ds\ x))$ $(at\ x)$
assumes *DsC*: *isCont Ds* $(poincare\text{-}map\ ?P\ x)$
assumes *nz*: *Ds* $(poincare\text{-}map\ ?P\ x)$ $(f\ (poincare\text{-}map\ ?P\ x)) \neq 0$
obtains *u e*
where $s\ (flow0\ x\ (u\ x)) = 0$
$u\ x =$ *return-time ?P x*
$(\bigwedge y.\ y \in cball\ x\ e \Longrightarrow s\ (flow0\ y\ (u\ y)) = 0)$
continuous-on $(cball\ x\ e)\ u$
$(\lambda t.\ (t,\ u\ t))$ ‘ $cball\ x\ e \subseteq$ *Sigma X existence-ivl0*
$0 < e$ $(u$ *has-derivative* $(-$ *blinfun-scaleR-left*
$(inverse\ (blinfun\text{-}apply\ (Ds\ (poincare\text{-}map\ ?P\ x))\ (f\ (poincare\text{-}map\ ?P\ x))))\ o_L$
$(Ds\ (poincare\text{-}map\ ?P\ x)\ o_L\ flowderiv\ x\ (return\text{-}time\ ?P\ x))\ o_L$ *embed1-blinfun*$))$ $(at\ x)$

⟨*proof*⟩

lemma (**in** *auto-ll-on-open*) *f-tendsto*[*tendsto-intros*]:
assumes *g1*: (*g1* ⟶ *b1*) (*at s within S*) **and** *b1* ∈ *X*
shows ((λ*x*. *f* (*g1 x*)) ⟶ *f b1*) (*at s within S*)
⟨*proof*⟩

lemma *flow-avoids-surface-eventually-at-right-pos*:
assumes *s x* > *0* ∨ *s x* = *0* ∧ *blinfun-apply* (*Ds x*) (*f x*) > *0*
assumes *x*: *x* ∈ *X*
assumes *Ds*: ⋀*x*. (*s has-derivative Ds x*) (*at x*)
assumes *DsC*: ⋀*x*. *isCont Ds x*
shows ∀ *F t in at-right 0*. *s* (*flow0 x t*) > (*0*::*real*)
⟨*proof*⟩

lemma *flow-avoids-surface-eventually-at-right-neg*:
assumes *s x* < *0* ∨ *s x* = *0* ∧ *blinfun-apply* (*Ds x*) (*f x*) < *0*
assumes *x*: *x* ∈ *X*
assumes *Ds*: ⋀*x*. (*s has-derivative Ds x*) (*at x*)
assumes *DsC*: ⋀*x*. *isCont Ds x*
shows ∀ *F t in at-right 0*. *s* (*flow0 x t*) < (*0*::*real*)
⟨*proof*⟩

lemma *flow-avoids-surface-eventually-at-right*:
assumes *x* ∉ *S* ∨ *s x* ≠ *0* ∨ *blinfun-apply* (*Ds x*) (*f x*) ≠ *0*
assumes *x*: *x* ∈ *X* **and** *cS*: *closed S*
assumes *Ds*: ⋀*x*. (*s has-derivative Ds x*) (*at x*)
assumes *DsC*: ⋀*x*. *isCont Ds x*
shows ∀ *F t in at-right 0*. (*flow0 x t*) ∉ {*x* ∈ *S*. *s x* = (*0*::*real*)}
⟨*proof*⟩

lemma *eventually-returns-to*:
fixes *s*::′*a*::*euclidean-space* ⇒ *real*
assumes *rt*: *returns-to* {*x* ∈ *S*. *s x* = *0*} *x* (**is** *returns-to ?P x*)
assumes *cS*: *closed S*
assumes *Ds*: ⋀*x*. (*s has-derivative blinfun-apply* (*Ds x*)) (*at x*)
assumes *DsC*: ⋀*x*. *isCont Ds x*
assumes *eventually-inside*: ∀ *F x in at* (*poincare-map ?P x*). *s x* = *0* ⟶ *x* ∈ *S*
assumes *nz*: *Ds* (*poincare-map ?P x*) (*f* (*poincare-map ?P x*)) ≠ *0*
assumes *nz0*: *x* ∉ *S* ∨ *s x* ≠ *0* ∨ *Ds x* (*f x*) ≠ *0*
shows ∀ *F x in at x*. *returns-to ?P x*
⟨*proof*⟩

lemma
return-time-isCont-outside:
fixes *s*::′*a*::*euclidean-space* ⇒ *real*
assumes *rt*: *returns-to* {*x* ∈ *S*. *s x* = *0*} *x* (**is** *returns-to ?P x*)
assumes *cS*: *closed S*
assumes *Ds*: ⋀*x*. (*s has-derivative blinfun-apply* (*Ds x*)) (*at x*)

assumes *DsC*: $\bigwedge x.$ *isCont Ds x*
assumes *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*: $x \notin S \vee s\ x \neq 0$
shows *isCont* (*return-time ?P*) *x*
⟨*proof*⟩

lemma *isCont-poincare-map*:
assumes *isCont* (*return-time P*) *x*
returns-to P x closed P
shows *isCont* (*poincare-map P*) *x*
⟨*proof*⟩

lemma *poincare-map-tendsto*:
assumes (*return-time P* $\longrightarrow$ *return-time P x*) (*at x within S*)
returns-to P x closed P
shows (*poincare-map P* $\longrightarrow$ *poincare-map P x*) (*at x within S*)
⟨*proof*⟩

lemma
return-time-continuous-below:
fixes *s*::*'a*::*euclidean-space* $\Rightarrow$ *real*
assumes *rt*: *returns-to* $\{x \in S.\ s\ x = 0\}$ *x* (**is** *returns-to ?P x*)
assumes *Ds*: $\bigwedge x.$ (*s has-derivative blinfun-apply* (*Ds x*)) (*at x*)
assumes *cS*: *closed S*
assumes *eventually-inside*: $\forall_F$ *x in at* (*poincare-map ?P x*). *s x* $= 0 \longrightarrow x \in S$
assumes *DsC*: $\bigwedge x.$ *isCont Ds x*
assumes *through*: (*Ds* (*poincare-map ?P x*)) (*f* (*poincare-map ?P x*)) $\neq 0$
assumes *inside*: $x \in S$ *s x* $= 0$ *Ds x* (*f x*) < 0
shows *continuous* (*at x within* $\{x.\ s\ x \leq 0\}$) (*return-time ?P*)
⟨*proof*⟩

lemma
return-time-continuous-below-plane:
fixes *s*::*'a*::*euclidean-space* $\Rightarrow$ *real*
assumes *rt*: *returns-to* $\{x \in R.\ x \cdot n = c\}$ *x* (**is** *returns-to ?P x*)
assumes *cR*: *closed R*
assumes *through*: *f* (*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 in at* (*poincare-map ?P x*). $x \cdot n = c \longrightarrow x \in R$
shows *continuous* (*at x within* $\{x.\ x \cdot n \leq c\}$) (*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* (*poincare-map ?P x*) o_L *Dflow x* (*return-time ?P x*)))

(*at x*)
⟨*proof*⟩

lemma *return-time-has-derivative*:
fixes *s*::′*a*::*euclidean-space* ⇒ *real*
assumes *rt*: *returns-to* {*x* ∈ *S*. *s x* = *0*} *x* (**is** *returns-to ?P* -)
assumes *cS*: *closed S*
assumes *Ds*: ⋀*x*. (*s has-derivative blinfun-apply* (*Ds x*)) (*at x*)
assumes *DsC*: ⋀*x*. *isCont Ds x*
assumes *Ds-through*: (*Ds* (*poincare-map ?P x*)) (*f* (*poincare-map ?P x*)) ≠ *0*
assumes *eventually-inside*: ∀ *F x in at* (*poincare-map* {*x* ∈ *S*. *s x* = *0*} *x*). *s x* = *0* ⟶ *x* ∈ *S*
assumes *outside*: *x* ∉ *S* ∨ *s x* ≠ *0*
shows (*return-time ?P has-derivative*
− *blinfun-scaleR-left* (*inverse* ((*Ds* (*poincare-map ?P x*)) (*f* (*poincare-map ?P x*)))) o_L
(*Ds* (*poincare-map ?P x*) o_L *Dflow x* (*return-time ?P x*)))
(*at x*)
⟨*proof*⟩

lemma *return-time-plane-has-derivative-blinfun*:
assumes *rt*: *returns-to* {*x* ∈ *S*. *x* ∙ *i* = *c*} *x* (**is** *returns-to ?P* -)
assumes *cS*: *closed S*
assumes *fnz*: *f* (*poincare-map ?P x*) ∙ *i* ≠ *0*
assumes *eventually-inside*: ∀ *F x in at* (*poincare-map ?P x*). *x* ∙ *i* = *c* ⟶ *x* ∈ *S*
assumes *outside*: *x* ∉ *S* ∨ *x* ∙ *i* ≠ *c*
shows (*return-time ?P has-derivative*
(− *blinfun-scaleR-left* (*inverse* ((*blinfun-inner-left i*) (*f* (*poincare-map ?P x*)))) o_L
(*blinfun-inner-left i* o_L *Dflow x* (*return-time ?P x*)))) (*at x*)
⟨*proof*⟩

lemma *return-time-plane-has-derivative*:
assumes *rt*: *returns-to* {*x* ∈ *S*. *x* ∙ *i* = *c*} *x* (**is** *returns-to ?P* -)
assumes *cS*: *closed S*
assumes *fnz*: *f* (*poincare-map ?P x*) ∙ *i* ≠ *0*
assumes *eventually-inside*: ∀ *F x in at* (*poincare-map ?P x*). *x* ∙ *i* = *c* ⟶ *x* ∈ *S*
assumes *outside*: *x* ∉ *S* ∨ *x* ∙ *i* ≠ *c*
shows (*return-time ?P has-derivative*
(λ*h*. − (*Dflow x* (*return-time ?P x*)) *h* ∙ *i* / (*f* (*poincare-map ?P x*) ∙ *i*))) (*at x*)
⟨*proof*⟩

definition *Dpoincare-map i c S x* =
(λ*h*. (*Dflow x* (*return-time* {*x* ∈ *S*. *x* ∙ *i* = *c*} *x*)) *h* −
((*Dflow x* (*return-time* {*x* ∈ *S*. *x* ∙ *i* = *c*} *x*)) *h* ∙ *i* /
(*f* (*poincare-map* {*x* ∈ *S*. *x* ∙ *i* = *c*} *x*) ∙ *i*)) ∗$_R$ *f* (*poincare-map* {*x* ∈ *S*. *x* ∙ *i* = *c*} *x*))

definition *Dpoincare-map*′ *i c S x* =

Dflow x (*return-time* {$x \in S.\ x \cdot i - c = 0$} *x*) −
(*blinfun-scaleR-left* (*f* (*poincare-map* {$x \in S.\ x \cdot i = c$} *x*)) o_L
(*blinfun-scaleR-left* (*inverse* ((*f* (*poincare-map* {$x \in S.\ x \cdot i = c$} *x*) · *i*))) o_L
(*blinfun-inner-left i* o_L *Dflow x* (*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* (*poincare-map ?P x*) · *i* ≠ *0*
assumes *eventually-inside*: $\forall_F$ *x in at* (*poincare-map ?P x*). *x* · *i* = *c* ⟶ *x* ∈ *S*
assumes *outside*: *x* ∉ *S* ∨ *x* · *i* ≠ *c*
notes [*derivative-intros*] = *return-time-plane-has-derivative*[*OF rt cS fnz eventually-inside outside*]
shows (*poincare-map ?P has-derivative Dpoincare-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* ∉ *A* ⟷ *a* ∈ − *A*
⟨*proof*⟩

lemma *continuous-orderD*:
fixes *g*::′*b*::*t2-space* ⇒ ′*c*::*order-topology*
assumes *continuous* (*at x within S*) *g*
shows *g x* > *c* ⟹ $\forall_F$ *y in at x within S*. *g y* > *c*
g x < *c* ⟹ $\forall_F$ *y in at x within S*. *g y* < *c*
⟨*proof*⟩

lemma *frontier-halfspace-component-ge*: *n* ≠ *0* ⟹ *frontier* {*x*. *c* ≤ *x* · *n*} = *plane n c*
⟨*proof*⟩

lemma *closed-Collect-le-within*:
fixes *f g* :: ′*a* :: *topological-space* ⇒ ′*b*::*linorder-topology*
assumes *f*: *continuous-on UNIV f*
and *g*: *continuous-on UNIV g*
and *closed R*
shows *closed* {*x* ∈ *R*. *f x* ≤ *g x*}
⟨*proof*⟩

6.1 explicit representation of hyperplanes / halfspaces

datatype *'a sctn = Sctn (normal: 'a) (pstn: real)*

definition *le-halfspace sctn x ⟷ x · normal sctn ≤ pstn sctn*

definition *lt-halfspace sctn x ⟷ x · normal sctn < pstn sctn*

definition *ge-halfspace sctn x ⟷ x · normal sctn ≥ pstn sctn*

definition *gt-halfspace sctn x ⟷ x · normal sctn > pstn sctn*

definition *plane-of sctn = {x. x · normal sctn = 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 = ⋂(below-halfspace ‘ sctns)*

definition *sbelow-halfspaces*
where *sbelow-halfspaces sctns = ⋂(sbelow-halfspace ‘ sctns)*

definition *above-halfspaces*
where *above-halfspaces sctns = ⋂(above-halfspace ‘ sctns)*

definition *sabove-halfspaces*
where *sabove-halfspaces sctns = ⋂(sabove-halfspace ‘ 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::*euclidean-space*) × ('a ⇒$_L$ 'a)) *set* ⇒ *real* ⇒ *real* ⇒
('a × ('a ⇒$_L$ 'a)) *set* ⇒ ('a × ('a ⇒$_L$ 'a)) *set* ⇒ *bool*
where *flowpipe X0 hl hu CX X1* ⟷ $0 \le hl \wedge hl \le hu \wedge$ *fst* ‘ *X0* ⊆ *X* ∧ *fst* ‘ *CX* ⊆ *X* ∧ *fst* ‘ *X1* ⊆ *X* ∧
(∀ (*x0*, *d0*) ∈ *X0*. ∀ *h* ∈ {*hl* .. *hu*}.
h ∈ *existence-ivl0 x0* ∧ (*flow0 x0 h*, *Dflow x0 h* o_L *d0*) ∈ *X1* ∧ (∀ *h'* ∈ {0 .. *h*}. (*flow0 x0 h'*, *Dflow x0 h'* o_L *d0*) ∈ *CX*))

lemma *flowpipeD*:
assumes *flowpipe X0 hl hu CX X1*
shows *flowpipe-safeD*: *fst* ‘ *X0* ∪ *fst* ‘ *CX* ∪ *fst* ‘ *X1* ⊆ *X*
and *flowpipe-nonneg*: $0 \le hl\ hl \le hu$
and *flowpipe-exivl*: $hl \le h \Longrightarrow h \le hu \Longrightarrow$ (*x0*, *d0*) ∈ *X0* ⟹ *h* ∈ *existence-ivl0 x0*
and *flowpipe-discrete*: $hl \le h \Longrightarrow h \le hu \Longrightarrow$ (*x0*, *d0*) ∈ *X0* ⟹ (*flow0 x0 h*, *Dflow x0 h* o_L *d0*) ∈ *X1*
and *flowpipe-cont*: $hl \le h \Longrightarrow h \le hu \Longrightarrow$ (*x0*, *d0*) ∈ *X0* ⟹ $0 \le h' \Longrightarrow h' \le h \Longrightarrow$ (*flow0 x0 h'*, *Dflow x0 h'* o_L *d0*) ∈ *CX*
⟨*proof*⟩

lemma *flowpipe-source-subset*: *flowpipe X0 hl hu CX X1* ⟹ *X0* ⊆ *CX*
⟨*proof*⟩

definition *flowsto X0 T CX X1* ⟷
(∀ (*x0*, *d0*) ∈ *X0*. ∃ *h* ∈ *T*. *h* ∈ *existence-ivl0 x0* ∧ (*flow0 x0 h*, *Dflow x0 h* o_L *d0*) ∈ *X1* ∧ (∀ *h'* ∈ *open-segment 0 h*. (*flow0 x0 h'*, *Dflow x0 h'* o_L *d0*) ∈ *CX*))

lemma *flowsto-to-empty-iff*[*simp*]: *flowsto a t b* {} ⟷ *a* = {}
⟨*proof*⟩

lemma *flowsto-from-empty-iff*[*simp*]: *flowsto* {} *t b c*
⟨*proof*⟩

lemma *flowsto-empty-time-iff*[*simp*]: *flowsto a* {} *b c* ⟷ *a* = {}
⟨*proof*⟩

lemma *flowstoE*:
assumes *flowsto X0 T CX X1* (*x0*, *d0*) ∈ *X0*
obtains *h* **where** *h* ∈ *T h* ∈ *existence-ivl0 x0* (*flow0 x0 h*, *Dflow x0 h* o_L *d0*) ∈ *X1*
⋀*h'*. *h'* ∈ *open-segment 0 h* ⟹ (*flow0 x0 h'*, *Dflow x0 h'* o_L *d0*) ∈ *CX*
⟨*proof*⟩

lemma *flowsto-safeD*: *flowsto X0 T CX X1* ⟹ *fst* ‘ *X0* ⊆ *X*

⟨*proof*⟩

lemma *flowsto-union*:
assumes *1*: *flowsto X0 T CX Y* **and** *2*: *flowsto Z S CZ W*
shows *flowsto* (*X0* ∪ *Z*) (*T* ∪ *S*) (*CX* ∪ *CZ*) (*Y* ∪ *W*)
⟨*proof*⟩

lemma *flowsto-subset*:
assumes *flowsto X0 T CX Y*
assumes *Z* ⊆ *X0 T* ⊆ *S CX* ⊆ *CZ Y* ⊆ *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* ∪ *Z*)
obtains *X1 X2* **where** *X0* = *X1* ∪ *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* |*s t*. *s* ∈ *S* ∧ *t* ∈ *T*} (*B* ∪ *D* ∪ *C*) *E*
⟨*proof*⟩

lemma *flowsto-step*:
assumes *A*: *flowsto A S B C*
assumes *D*: *flowsto D T E F*
shows *flowsto A* (*S* ∪ {*s* + *t* |*s t*. *s* ∈ *S* ∧ *t* ∈ *T*}) (*B* ∪ *E* ∪ *C* ∩ *D*) (*C* − *D* ∪ *F*)
⟨*proof*⟩

lemma
flowsto-stepI:
flowsto X0 U B C ⟹
flowsto D T E F ⟹
Z ⊆ *X0* ⟹
(⋀*s*. *s* ∈ *U* ⟹ *s* ∈ *S*) ⟹
(⋀*s t*. *s* ∈ *U* ⟹ *t* ∈ *T* ⟹ *s* + *t* ∈ *S*) ⟹
B ∪ *E* ∪ *D* ∩ *C* ⊆ *CZ* ⟹ *C* − *D* ∪ *F* ⊆ *W* ⟹ *flowsto Z S CZ W*
⟨*proof*⟩

lemma *flowsto-imp-flowsto*:
flowpipe Y h h CY Z ⟹ *flowsto Y* {*h*} (*CY*) *Z*
⟨*proof*⟩

lemma *connected-below-halfspace*:
assumes *x* ∈ *below-halfspace sctn*
assumes *x* ∈ *S connected S*

assumes $S \cap$ *plane-of sctn* $= \{\}$
shows $S \subseteq$ *below-halfspace sctn*
⟨*proof*⟩

lemma
inter-Collect-eq-empty:
assumes $\bigwedge x.\ x \in X0 \Longrightarrow \neg\ g\ x$ **shows** $X0 \cap$ *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 $\wedge$ *fst* ‘ *X0* $\subseteq S \wedge$
(*return-time P differentiable at x within S*) $\wedge$
$(\exists D.$ (*poincare-map P has-derivative blinfun-apply D*) (*at x within S*) $\wedge$
(*poincare-map P x*, *D* o_L *d*) $\in Y) \wedge$
$(\forall\, t \in \{0<..<$*return-time P x*$\}.$ *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.* (*flow0 x t*, *Dflow x t* o_L *d*) $\in CX$
⟨*proof*⟩

lemma *frontier-aux-lemma*:
fixes *R* :: *'n*::*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*: $(a - b)\ o_L\ c = (a\ o_L\ c) - (b\ o_L\ c)$
⟨*proof*⟩

lemma *flowpipe-split-at-above-halfspace*:
assumes *flowpipe X0 hl t CX Y fst* ‘ *X0* $\cap \{x.\ x \cdot n \geq c\} = \{\}$ **and** [*simp*]: $n \neq 0$
assumes *cR*: *closed R* **and** *Rs*: $R \subseteq$ *plane n c*
assumes *PDP*: $\bigwedge x\ d.\ (x,\ d) \in CX \Longrightarrow x \cdot n = c \Longrightarrow (x,$
$d -$ (*blinfun-scaleR-left* (*f* (*x*)) o_L (*blinfun-scaleR-left* (*inverse* (*f x* $\cdot$ *n*)) o_L (*blinfun-inner-left n* o_L *d*)))) $\in$ *PDP*
assumes *PDP-nz*: $\bigwedge x\ d.\ (x,\ d) \in PDP \Longrightarrow f\ x \cdot n \neq 0$
assumes *PDP-inR*: $\bigwedge x\ d.\ (x,\ d) \in PDP \Longrightarrow x \in R$

assumes *PDP-in*: $\bigwedge x\ d.\ (x,\ d) \in PDP \Longrightarrow \forall_F\ x\ in\ at\ x\ within\ plane\ n\ c.\ x \in R$
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*: $0 \le h\ h <$ *return-time P x0*
assumes *cP*: *closed P* **and** *x0*: $x0 \in S$
shows $((\lambda x.$ *poincare-map P x*$)$ *has-derivative* $(D\ o_L\ Dflow\ x0\ h))$ (*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 $CX \cup CY \cup fst\ `\ P1 \subseteq CZ$
assumes $p2 \cap (CX \cup fst\ `\ P1) = \{\}$
assumes [*intro*, *simp*]: *closed p1*
assumes [*intro*, *simp*]: *closed p2*
assumes *cont*: $\bigwedge x\ d.\ (x,\ d) \in X0 \Longrightarrow$ *continuous* (*at x within S*) (*return-time p2*)
shows *poincare-mapsto p2 X0 S CZ P2*
⟨*proof*⟩

lemma *flowsto-poincare-trans*:— TODO: the proof is close to ⟦*poincare-mapsto ?p1.0 ?X0.0 ?S ?CX ?P1.0*; *poincare-mapsto ?p2.0 ?P1.0 UNIV ?CY ?P2.0*; $?CX \cup ?CY \cup fst\ `\ ?P1.0 \subseteq ?CZ$; $?p2.0 \cap (?CX \cup fst\ `\ ?P1.0) = \{\}$; *closed ?p1.0*; *closed ?p2.0*; $\bigwedge x\ d.\ (x,\ d) \in ?X0.0 \Longrightarrow$ *continuous* (*at x within ?S*) (*return-time ?p2.0*)⟧ ⟹ *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*: $\bigwedge t.\ t \in T \Longrightarrow t \ge 0$
assumes $fst\ `\ CX \cup CY \cup fst\ `\ P1 \subseteq CZ$
assumes $p2 \cap (fst\ `\ CX \cup fst\ `\ P1) = \{\}$
assumes [*intro*, *simp*]: *closed p2*
assumes *cont*: $\bigwedge x\ d.\ (x,\ d) \in X0 \Longrightarrow$ *continuous* (*at x within S*) (*return-time p2*)
assumes *subset*: $fst\ `\ X0 \subseteq S$
shows *poincare-mapsto p2 X0 S CZ P2*
⟨*proof*⟩

6.5 conditions for continuous return time

definition *section s Ds S* $\longleftrightarrow$
$(\forall x.\ (s\ \textit{has-derivative blinfun-apply}\ (Ds\ x))\ (at\ x)) \wedge$
$(\forall x.\ isCont\ Ds\ x) \wedge$
$(\forall x \in S.\ s\ x = (0::real) \longrightarrow Ds\ x\ (f\ x) \neq 0) \wedge$
$closed\ S \wedge S \subseteq X$

lemma *sectionD*:
assumes *section s Ds S*
shows $(s\ \textit{has-derivative blinfun-apply}\ (Ds\ x))\ (at\ x)$
$isCont\ Ds\ x$
$x \in S \Longrightarrow s\ x = 0 \Longrightarrow Ds\ x\ (f\ x) \neq 0$
$closed\ S\ S \subseteq X$
⟨*proof*⟩

definition *transversal p* $\longleftrightarrow (\forall x \in p.\ \forall_F\ t\ in\ at\text{-}right\ 0.\ flow0\ x\ t \notin p)$

lemma *transversalD*: *transversal p* $\Longrightarrow x \in p \Longrightarrow \forall_F\ t\ in\ at\text{-}right\ 0.\ flow0\ x\ t \notin p$
⟨*proof*⟩

lemma *transversal-section*:
fixes *c*::*real*
assumes *section s Ds S*
shows *transversal* $\{x \in S.\ s\ x = 0\}$
⟨*proof*⟩

lemma *section-closed*[*intro*, *simp*]: *section s Ds S* $\Longrightarrow$ *closed* $\{x \in S.\ s\ x = 0\}$
⟨*proof*⟩

lemma *return-time-continuous-belowI*:
assumes *ft*: *flowsto X0 T CX X1*
assumes *pos*: $\bigwedge t.\ t \in T \Longrightarrow t > 0$
assumes *X0*: *fst ‘ X0* $\subseteq \{x \in S.\ s\ x = 0\}$
assumes *CX*: *fst ‘ CX* $\cap \{x \in S.\ s\ x = 0\} = \{\}$
assumes *X1*: *fst ‘ X1* $\subseteq \{x \in S.\ s\ x = 0\}$
assumes *sec*: *section s Ds S*
assumes *nz*: $\bigwedge x.\ x \in S \Longrightarrow s\ x = 0 \Longrightarrow Ds\ x\ (f\ x) \neq 0$
assumes *Dneg*: $(\lambda x.\ (Ds\ x)\ (f\ x))$ *‘ fst ‘ X0* $\subseteq \{..<0\}$
assumes *rel-int*: $\bigwedge x.\ x \in$ *fst ‘ X1* $\Longrightarrow \forall_F\ x\ in\ at\ x.\ s\ x = 0 \longrightarrow x \in S$
assumes $(x, d) \in X0$
shows *continuous* $(at\ x\ within\ \{x.\ s\ x \leq 0\})$ *(return-time* $\{x \in S.\ s\ x = 0\}$)
⟨*proof*⟩

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 $\bigwedge x\ y.\ x \in s \Longrightarrow g'\ x = g\ x$
shows L−*lipschitz-on* s g
⟨*proof*⟩

lemma *local-lipschitz-congI*:
assumes *local-lipschitz* s' t' g'
assumes $s' = s$
assumes $t' = t$
assumes $\bigwedge x\ y.\ x \in s \Longrightarrow y \in t \Longrightarrow g'\ x\ y = g\ x\ y$
shows *local-lipschitz* s t g
⟨*proof*⟩

context *ll-on-open-it*— TODO: do this more generically for *ll-on-open-it*
begin

context fixes S Y g **assumes** *cong*: $X = Y$ $T = S$ $\bigwedge x\ t.\ x \in Y \Longrightarrow t \in S \Longrightarrow f\ t\ x = g\ t\ x$
begin

lemma *ll-on-open-congI*: *ll-on-open* S g Y
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *existence-ivl-cong*:
shows *existence-ivl t0 x0* = *ll-on-open.existence-ivl* S g Y *t0 x0*
⟨*proof*⟩

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*
⟨*proof*⟩

end

end

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

context fixes Y g **assumes** *cong*: $X = Y$ $\bigwedge x\ t.\ x \in Y \Longrightarrow f\ x = g\ x$

begin

lemma *auto-ll-on-open-congI*: *auto-ll-on-open g Y*
⟨*proof*⟩

lemma *existence-ivl0-cong*:
shows *existence-ivl0 x0 = auto-ll-on-open.existence-ivl0 g Y x0*
⟨*proof*⟩

lemma *flow0-cong*:
assumes *t ∈ existence-ivl0 x0*
shows *flow0 x0 t = auto-ll-on-open.flow0 g Y x0 t*
⟨*proof*⟩

end

end

context *c1-on-open-euclidean* **begin**

context fixes *Y g* **assumes** *cong*: $X = Y \bigwedge x\ t.\ x \in Y \Longrightarrow f\ x = g\ x$
begin

lemma *f'-cong*: (*g has-derivative blinfun-apply* (*f' x*)) (*at x*) **if** *x ∈ Y*
⟨*proof*⟩

lemma *c1-on-open-euclidean-congI*: *c1-on-open-euclidean g f' Y*
⟨*proof*⟩

lemma *vareq-cong*: *vareq x0 t = c1-on-open-euclidean.vareq g f' Y x0 t*
if *t ∈ existence-ivl0 x0*
⟨*proof*⟩

lemma *Dflow-cong*:
assumes *t ∈ existence-ivl0 x0*
shows *Dflow x0 t = c1-on-open-euclidean.Dflow g f' Y x0 t*
⟨*proof*⟩

lemma *flowsto-congI1*:
assumes *flowsto A B C D*
shows *c1-on-open-euclidean.flowsto g f' Y A B C D*
⟨*proof*⟩

lemma *flowsto-congI2*:
assumes *c1-on-open-euclidean.flowsto g f' Y A B C D*
shows *flowsto A B C D*
⟨*proof*⟩

lemma *flowsto-congI*: *flowsto A B C D = 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 = returns-to A x*
⟨*proof*⟩

lemma
return-time-cong:
shows *return-time A x = 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 ⟹*
poincare-mapsto A B C D E = 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*]: $-1 \leq x \Longrightarrow x \leq 1 \Longrightarrow \arcsin x = 0 \longleftrightarrow x = 0$
⟨*proof*⟩

definition *conemem* :: *'a*::*real-vector* ⇒ *'a* ⇒ *real* ⇒ *'a* **where** *conemem u v t* = *cos t* $*_R$ *u* + *sin t* $*_R$ *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*: $a * x^2 + b * x + c = (0::real) \Longrightarrow 0 \leq b^2 - 4 * a * c$
⟨*proof*⟩

lemma *quadratic-eq-factoring*:
assumes *D*: $D = b^2 - 4 * a * c$
assumes *nn*: $0 \leq D$
assumes *x1*: $x_1 = (-b + sqrt\ D) / (2 * a)$
assumes *x2*: $x_2 = (-b - sqrt\ D) / (2 * a)$
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*: $x_1 = (-b + sqrt\ D) / (2 * a)$
assumes *x2*: $x_2 = (-b - sqrt\ D) / (2 * a)$
assumes *a*: $a \neq 0$
shows $a * x^2 + b * x + c = 0 \longleftrightarrow (D \geq 0 \wedge (x = x_1 \vee x = x_2))$ (**is** *?z* ⟷ -)
⟨*proof*⟩

lemma *quadratic-ex-zero-iff*:
$(\exists x.\ a * x^2 + b * x + c = 0) \longleftrightarrow (a \neq 0 \wedge b^2 - 4 * a * c \geq 0 \vee a = 0 \wedge (b = 0 \longrightarrow c = 0))$
for *a b c*::*real*
⟨*proof*⟩

lemma *Cauchy-Schwarz-eq-iff*:
shows $(inner\ x\ y)^2 = inner\ x\ x * inner\ y\ y \longleftrightarrow ((\exists k.\ x = k *_R y) \vee y = 0)$
⟨*proof*⟩

lemma *Cauchy-Schwarz-strict-ineq*:
$(inner\ x\ y)^2 < inner\ x\ x * inner\ y\ y$ **if** $y \neq 0$ $\bigwedge k.\ x \neq k *_R y$
⟨*proof*⟩

lemma *Cauchy-Schwarz-eq2-iff*:
$|inner\ x\ y| = norm\ x * norm\ y \longleftrightarrow ((\exists k.\ x = k *_R y) \vee y = 0)$
⟨*proof*⟩

lemma *Cauchy-Schwarz-strict-ineq2*:
$|inner\ x\ y| < norm\ x * norm\ y$ **if** $y \neq 0$ $\bigwedge k.\ x \neq k *_R y$
⟨*proof*⟩

lemma *gt-minus-one-absI*: $abs\ k < 1 \Longrightarrow -1 < k$ **for** *k*::*real*
⟨*proof*⟩
lemma *gt-one-absI*: $abs\ k < 1 \Longrightarrow k < 1$ **for** *k*::*real*
⟨*proof*⟩

lemma *abs-impossible*:
$|y1| < x1 \Longrightarrow |y2| < x2 \Longrightarrow x1 * x2 + y1 * y2 \neq 0$ **for** *x1 x2*::*real*
⟨*proof*⟩

lemma *vangle-eq-arctan-minus*:— TODO: generalize?!
assumes *ij*: $i \in Basis\ j \in Basis$ **and** *ij-neq*: $i \neq j$
assumes *xy1*: $|y1| < x1$
assumes *xy2*: $|y2| < x2$
assumes *less*: $y2 / x2 > y1 / x1$
shows $vangle\ (x1 *_R i + y1 *_R j)\ (x2 *_R i + y2 *_R j) = arctan\ (y2 / x2) - arctan\ (y1 / x1)$
(**is** *vangle ?u ?v* = -)
⟨*proof*⟩

lemma *vangle-le-pi2*: $0 \leq u \cdot v \Longrightarrow vangle\ u\ v \leq pi/2$
⟨*proof*⟩

lemma *inner-eq-vangle*: $u \cdot v = cos\ (vangle\ u\ v) * (norm\ u * norm\ v)$
⟨*proof*⟩

lemma *vangle-scaleR-self*:
$vangle\ (k *_R v)\ v = (if\ k = 0 \vee v = 0\ then\ pi / 2\ else\ if\ k > 0\ then\ 0\ else\ pi)$
$vangle\ v\ (k *_R v) = (if\ k = 0 \vee v = 0\ then\ pi / 2\ else\ if\ k > 0\ then\ 0\ else\ pi)$
⟨*proof*⟩

lemma *vangle-scaleR*:
$vangle\ (k *_R v)\ w = vangle\ v\ w\ vangle\ w\ (k *_R v) = vangle\ w\ v$ **if** $k > 0$
⟨*proof*⟩

lemma *cos-vangle-eq-zero-iff-vangle*:
cos (*vangle u v*) = *0* ⟷ (*u* = *0* ∨ *v* = *0* ∨ *u* ⋅ *v* = *0*)
⟨*proof*⟩

lemma *ortho-imp-angle-pi-half*: *u* ⋅ *v* = *0* ⟹ *vangle u v* = *pi* / *2*
⟨*proof*⟩

lemma *arccos-eq-zero-iff*: *arccos x* = *0* ⟷ *x* = *1* **if** −*1* ≤ *x x* ≤ *1*
⟨*proof*⟩

lemma *vangle-eq-zeroD*: *vangle u v* = *0* ⟹ (∃ *k*. *v* = *k* ∗$_R$ *u*)
⟨*proof*⟩

lemma *less-one-multI*:— TODO: also in AA!
fixes *e x*::*real*
shows *e* ≤ *1* ⟹ *0* < *x* ⟹ *x* < *1* ⟹ *e* ∗ *x* < *1*
⟨*proof*⟩

lemma *conemem-expansion-estimate*:
fixes *u v u′ v′*::*'a*::*euclidean-space*
assumes *t* ∈ {*0* .. *pi* / *2*}
assumes *angle-pos*: *0* < *vangle u v vangle u v* < *pi* / *2*
assumes *angle-le*: (*vangle u′ v′*) ≤ (*vangle u v*)
assumes *norm u* = *1 norm v* = *1*
shows *norm* (*conemem u′ v′ t*) ≥ *min* (*norm u′*) (*norm v′*) ∗ *norm* (*conemem u v t*)
⟨*proof*⟩

lemma *conemem-commute*: *conemem a b t* = *conemem b a* (*pi* / *2* − *t*) **if** *0* ≤ *t t* ≤ *pi* / *2*
⟨*proof*⟩

lemma *conesegment-commute*: *conesegment a b* = *conesegment b a*
⟨*proof*⟩

definition *conefield u v* = *cone hull* (*conesegment u v*)

lemma *conefield-alt-def*: *conefield u v* = *cone hull* {*u*−−*v*}
⟨*proof*⟩

lemma
bounded-linear-image-cone-hull:
assumes *bounded-linear F*
shows *F* ‘ (*cone hull T*) = *cone hull* (*F* ‘ *T*)
⟨*proof*⟩

lemma
bounded-linear-image-conefield:
assumes *bounded-linear F*
shows *F ‘ conefield u v = conefield (F u) (F v)*
⟨*proof*⟩

lemma *conefield-commute*: *conefield x y = conefield y x*
⟨*proof*⟩

lemma *convex-conefield*: *convex (conefield x y)*
⟨*proof*⟩

lemma *conefield-scaleRI*: *v ∈ conefield (r *$_R$ x) y* **if** *v ∈ conefield x y r > 0*
⟨*proof*⟩

lemma *conefield-scaleRD*: *v ∈ conefield x y* **if** *v ∈ conefield (r *$_R$ x) y r > 0*
⟨*proof*⟩

lemma *conefield-scaleR*: *conefield (r *$_R$ x) y = conefield x y* **if** *r > 0*
⟨*proof*⟩

lemma *conefield-expansion-estimate*:
fixes *u v*::*'a*::*euclidean-space* **and** *F*::*'a ⇒ 'a*
assumes *t ∈ {0 .. pi / 2}*
assumes *angle-pos*: *0 < vangle u v vangle u v < pi / 2*
assumes *angle-le*: *vangle (F u) (F v) ≤ vangle u v*
assumes *bounded-linear F*
assumes *x ∈ conefield u v*
shows *norm (F x) ≥ min (norm (F u)/norm u) (norm (F v)/norm v) * norm x*
⟨*proof*⟩

lemma *conefield-rightI*:
assumes *ij*: *i ∈ Basis j ∈ Basis* **and** *ij-neq*: *i ≠ j*
assumes *y ∈ {y1 .. y2}*
shows *(i + y *$_R$ j) ∈ conefield (i + y1 *$_R$ j) (i + y2 *$_R$ j)*
⟨*proof*⟩

lemma *conefield-right-vangleI*:
assumes *ij*: *i ∈ Basis j ∈ Basis* **and** *ij-neq*: *i ≠ j*
assumes *y ∈ {y1 .. y2} y1 < y2*
shows *(i + y *$_R$ j) ∈ conefield (i + y1 *$_R$ j) (i + y2 *$_R$ j)*
⟨*proof*⟩

lemma *cone-conefield*[*intro*, *simp*]: *cone (conefield x y)*
⟨*proof*⟩

lemma *conefield-mk-rightI*:
assumes *ij*: *i ∈ Basis j ∈ Basis* **and** *ij-neq*: *i ≠ j*
assumes *(i + (y / x) *$_R$ j) ∈ conefield (i + (y1 / x1) *$_R$ j) (i + (y2 / x2) *$_R$ j)*

assumes $x > 0$ $x1 > 0$ $x2 > 0$
shows $(x *_R i + y *_R j) \in$ *conefield* $(x1 *_R i + y1 *_R j)$ $(x2 *_R i + y2 *_R j)$
⟨*proof*⟩

lemma *conefield-prod3I*:
assumes $x > 0$ $x1 > 0$ $x2 > 0$
assumes $y1 / x1 \leq y / x$ $y / x \leq y2 / x2$
shows $(x, y, 0) \in$ (*conefield* $(x1, y1, 0)$ $(x2, y2, 0)$::(*real*∗*real*∗*real*) *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::*real* ⇒ *real*
assumes (f *has-derivative* f') (*at x within s*)
shows ((λ*x*. *exp* ($f\ x *_R A$)) *has-derivative* (λ*h*. $f'\ h *_R$ (*exp* ($f\ x *_R A$) ∗ A)))
(*at x within s*)
⟨*proof*⟩

context
fixes A::'*a*::{*banach*,*perfect-space*} *blinop*
begin

definition *linode-solution t0 x0* = (λ*t*. *exp* ((*t* − *t0*) $*_R$ A) *x0*)

lemma *linode-solution-solves-ode*:
(*linode-solution t0 x0 solves-ode* (λ-. A)) *UNIV UNIV linode-solution t0 x0 t0 = x0*
⟨*proof*⟩

lemma (*linode-solution t0 x0 usolves-ode* (λ-. 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.