Formalization of Multivariable Calculus in Mizar

Kazuhisa Nakasho
Yamaguchi University

Logic Seminar @ University of Hawaii at Manoa
13:00 - 14:00, November 16, 2022

Table of contents

Part 1. Quick overview of the Mizar project

- Mizar Hands-on Tutorial (CICM2016) + α

Part 2. Formalization of multivariable calculus

- Focus on implicit function theorem

Quick overview of the Mizar project

- Features
- Library
- Language
- Tools

What is Mizar Project

- Mizar is a proof assistant system.
- The development was started by Andrzej Trybulec (†2013) in 1973.
  - ITP 2023 (July 31 - Aug. 4) celebrates 50th anniversary in Bialystok.
- Research has been conducted primarily at Bialystok University, with collaborations at Shinshu University and the University of Alberta.
- Mizar project manages:
  - Language: tries to mimic standard mathematical practice.
  - Verification engine: designed to preserve human understanding of proof steps.
  - Library: Mizar Mathematical Library (MML)
  - Journal: Journal of Formalized Mathematics
  - Utilities: editor(emacs/vscode), browse/search engine on the Web

QED manifesto

Mizar aims to be a QED manifesto tool

- QED manifesto was proposed by Robert Boyer et al. in 1993.
- The purpose of QED manifesto is to create computer-based database of all mathematical knowledge with strictly formalized statements and machine verified proofs.
- Significance: guaranteeing the rigor of mathematics, formal verification of industrial systems, promotion of mathematics education, preservation of mathematical knowledge, maintenance of mathematical culture, etc...

Features of the Mizar Project

Mizar has long been regarded as one of the four major proof assistants (Mizar, HOL light, Isabelle, Coq):

- High readability: Declarative and structured proof style is adopted so that mathematicians can read the proofs without knowing its syntax. This style was inherited by Isabelle/Isar and Lean.
- Huge legacy: 3.1 million lines of libraries, written in first-order logic, included in the TPTP library and used in CADE ATP system competition.
- Publication: All articles are peer-reviewed and edited by the Mizar committee before being included in the library and published in Journal of Formalized Mathematics.
  - The publication system has motivated researchers to formalize mathematics.

Mizar Mathematical Library - MML

- A systematic collection of articles started around 1989
- Scale of MML
  - includes 1,400 articles written by over 250 authors
  - over 60,000 theorems
  - over 12,000 definitions
  - over 800 schemes
  - over 13,000 registrations
  - over 3.1 millions lines of code
- The library is based on the axioms of Tarski-Grothendieck set theory

Contents of MML (set theory and foundations, data structure)

Articles	Contents
TARSKI,BOOLE,XBOOLE_*,ZFMISC_1	set operation, direct product, power sets, equality of sets
CARD_* ORDINAL*	ordinal numbers, cardinality, direct product of set family
RELAT_* ,FUNCT_*, FUNCOP	relation, function(domain, range, composition, image, inverse)
STRUCT_* ,ALGSTR_*,BINOP_*	basic structure, algebraic structure, binary operation
FINSEQ_*, RFINSEQ*, RVSUM_*	finite sequence and its operations
WAYBEL_*	directed sets, nets, ideals, filters
GOEDELCP	Gödel Completeness Theorem

Contents of MML (arithmetic)

Articles	Contents
NAT_*, INT_*, NTALOG_1, WSIERP_1	natural number, integers, prime factorization, remainders, Euclidean reciprocity, Chinese remainder theorem
NUMBERS, ARYTM_*,REAL XREAL_*,XXREAL_*,COMLEX1,XCMPLX*	Definition and operations on $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{R} \cup \{\pm\infty\}$
ABSVALUE,RINFSUP*,SUPINF_*,	absolute values, upper and lower bounds/limits

Contents of MML (algebra)

Articles	Contents
GROUP_*, GR_CY_*	groups theory
RMOD_*, ZMOD_*	module over rings
ECPF_*	elliptic curve
POLYNOM*	polynomial rings, fundamental theorems of algebra
YELLOW_*	lattice theory

Contents of MML (analysis)

Articles	Contents
RFUNCT_*,FCONT_*, FDIFF_*, FUNCSDOM, RSSAPCE*	sequences and series of $\mathbb{R}$, $\mathbb{C}$, trigonometric functions, De Moivre's formula, Taylor expansion, Bolzano-Weierstrass theorem, Cauchy sequences, Cauchy's theorem, Heine-Borel's theorem
INTEGRAL_* INTEGR_*	fundamental theorem of calculus, Darboux's theorem, Riemannian integrals, differential equations
MEASURE*, MESFUNC*, MESFUN*	measure theory, Lebesgue integrals, Egorov's theorem, Fatou's theorem, bounded convergence theorem, L_p space, Fubini's theorem
NDIFF_*,PDIFF_*	differentiation and partial differentiation on normed spaces
PROB_*,RANDOM_*,DIST_*	Probability and random variables, Stochastics

Contents of MML (linear algebra and functional analysis)

Articles	Contents
VFUNCT*,LOPBAN_*	function spaces, bounded linear operators, Uniform boundedness theorem, open mapping theorem, closed graph theorem
RLVECT_*,NORMSP_*,VECTSP_*, PRVECT_*, BHSP_*, HAHNBAN	vector space, normed space, Banach space, Hilbert space, Hahn-Banach theorem

Contents of MML (geometry)

Articles	Contents
JORDAN*	Jordan closed curve theorem
METRIC_*	metric space
EUCLID*	Euclidean space
TOPS_*	topological space
TOPMETR	compact space
TOPREAL*	real number line
BROUWER*	Brouwer's fixed point theorem

Contents of MML (computer science)

Articles	Contents
DESCIP_1	DES encryption
CIRCUIT*,FTACELL1, GFACIRC*	circuit, parallel circuit
PERTRI*	Petri nets
MORPH_01	image processing and morphology
AMI_*,SCM*	abstract computing machine

Representative formalizations in MML

- Jordan closed curve theorem (2005)
  - Every simple closed curve on a plane divides the plane into two region.
  - Artur Korniłowicz: A Proof of the Jordan Curve Theorem via the Brouwer Fixed Point Theorem
- Lattice theory
  - Formalized almost all of "Gierz et al.: A Compendium of Continuous Lattices"
  - Robbins’ problem, etc...
  - Adam Grabowski: Mechanizing Complemented Lattices Within Mizar Type System
- Formalizing 100 theorems (Freek Wiedijk)
  - Mizar formalized 69/100 theorems.
  - Fermat's Last Theorem and The Isoperimetric Theorems are left now.

Key features of the Mizar language

- Classical first-order logic
  - Free second-order variables (e.g. the induction scheme) are supported
- A declarative style of writing proofs (="have" statement of Lean/Isabelle).
  - Jaśkowski-style natural deduction
- Highly structured
  - Types of inheritance
  - Explicit / implicit definition, re-definition
- Designed to be mathematician-friendly and verifiable.
  - "A subset" of standard English used in mathematical texts
  - Prefix, postfix, infix notations for predicates as well as parenthetical notations for functors

Logical connectives and quantifiers

Logical formula	Mizar expresion
$\neg\alpha$	not α
$\alpha \wedge \beta$	α & β
$\alpha \vee \beta$	α or β
$\alpha \rightarrow \beta$	α implies β
$\alpha \leftrightarrow \beta$	α iff β
$\exists_x \alpha$	ex x st α
$\forall_x \alpha$	for x holds α
$\forall_{x:\alpha} \beta$	for x st α holds β

Quantified variables

- Usage of quantified variables:

```
for x being set holds ...
ex y being real number st ...
```

- Global type assignment with ***reserve*** statement:

```
reserve x,y for real number;
```

  - One does not have to mention the type of x or y in quantified formulas.
  - Mizar implicitly applies universal quantifiers to formulas if needed.

Constructors in Mizar

- **pred** (predicate) constructs formulae.
- **func** (functor) constructors terms.
- **mode** constructs types. Every variable or term has a unique type.
- **attr** (adjectives) gives restrictions to types.
- **struct** constructs data structures.

Example of predicate definitions

- Definition of "devides" predicate

```
definition
  let i1,i2 be Integer;
  pred i1 divides i2 means
  ex i3 be Integer st i2 = i1 * i3;
end;
```

Example of functor definitions

1. Explicit version

```
definition
  let z be Complex;
  func |.z.| -> Real equals
  sqrt ((Re z)^2 + (Im z)^2);
end;
```

2. Implicit version

```
definition
  let f,g be Function;
  func <:f,g:> -> Function means
  dom it = dom f /\ dom g &
  for x being object st x in dom it holds it.x = [f.x,g.x];
end;
```

Examples of mode definitions

1. Modes that define a type with an explicit definiens:

```
definition
  let K,L be Ring;
  let J be Function of K,L;
  let V be for LeftMod of K, W be LeftMod of L;
  mode Homomorphism of J,V,W -> Function of V,W means
  (for x,y being Vector of V holds it.(x+y) = it.x+it.y) &
  for a being Scalar of K, x being Vector of V holds it.(a*x) = J.a*it.x;
end;
```

2. Modes defined as a collection of adjectives associated with an already defined radix type:

```
definition
  let G,H be AddGroup;
  mode Homomorphism of G,H is additive Function of G,H;
end;
```

Examples of attribute definitions

1. Without implicit parameters:

```
definition
  let R be Relation;
  attr R is well_founded means
  for Y being set st Y c= field R & Y <> {}
  ex a being set st a in Y & R-Seg a misses Y;
end;
```

2. With an implicit parameter:

```
definition
  let n be Nat, X be set;
  attr X is n-at_most_dimensional means
  for x being set st x in X holds card x c= n+1;
end;
```

Example of structure definitions

- Structures model mathematical notions like groups, topological spaces, categories, etc. which are usually represented as tuples.
- Mizar supports multiple inheritance of structures that makes a whole hierarchy of interrelated structures available in the library.

```
definition
  let F be 1-sorted;
  struct(addLoopStr) ModuleStr over F
  (# carrier -> set,
     addF -> BinOp of the carrier,
     ZeroF -> Element of the carrier,
     lmult -> Function of [:the carrier of F, the carrier:], the carrier #);
end;
```

Existential cluster

- Existential cluster
  - Mizar language does not allow "empty type" (=does not have any elements)
  - Existence clusters guarantee the presence of elements of the type qualified by attributes.

```
registration
  let n be Nat;
  cluster n-at_most_dimensional subset-closed non empty for set;
end;

:: OK
let x be n-at_most_dimensional subset-closed non empty set;
```

Conditional / functorial cluster

- Conditional / functorial cluster
  - Type conversions are one of the most tedious tasks in formalization. They appear everywhere in the proof.
  - Conditional and functorial clusters are implicit type conversion systems.
  - Once conditional / functorial clusters are regstrated, arguments of constructor(pred, func, attr, mode, struct) are automatically converted to the corresponding types.
  - In type conversion, multiple clusters act successively at once.

Examples of registrations

- Conditional:

```
registration
  let n be Nat;
  cluster n-at_most_dimensional -> finite-membered for set;
end;

:: n-at_most_dimensinal set will be automatically converted to
:: finite-membered set
```

- Functorial (term):

```
let n be Nat;
  let X, Y be n-at_most_dimensional set;
  cluster X \/ Y -> n-at_most_dimensional;
end;

:: Type of X \/ Y will be automatically converted to
:: n-at_most_dimensional (also finite-membered set)
```

Examples of proof (1)

$$X \cap Y \neq \emptyset \leftrightarrow \exists x.\, x \in X \land x \in Y$$

```
theorem Th3:
  X meets Y iff ex x st x in X & x in Y
proof
  hereby
    assume X meets Y;
    then X /\ Y <> {};
    then X /\ Y is not empty by Lm1;
    then consider x such that
A1: x in X /\ Y;
    take x;
    thus x in X & x in Y by A1,Def4;
  end;
  given x such that
A2: x in X & x in Y;
  x in X /\ Y by A2,Def4;
  then X /\ Y <> {} by Def1;
  hence thesis;
end;
```

Examples of proof (2)

```
reserve i,j,k,l,m,n for natural number;

i+k = j+k implies i=j;
proof
  defpred P[natural number] means
  i+$1 = j+$1 implies i=j;
  A1: P[0]
  proof
    assume B0: i+0 = j+0;
    B1: i+0 = i by INDUCT:3;
    B2: j+0 = j by INDUCT:3;
    hence thesis by B0,B1,B2;
  end;
  A2: for k st P[k] holds P[succ k]
  proof
    let l such that C1: P[l];
    assume C2: i+succ l=j+succ l;
    then C3: succ(i+l) = j+succ l by C2,INDUCT:4
    .= succ(j+l) by INDUCT:4;
    hence thesis by C1,INDUCT:2;
  end;
  for k holds P[k] from INDUCT:sch 1(A1,A2);
  hence thesis;
end;
```

Recommended documents

- A. Naumowicz, A. Korniłowicz, A. Grabowski: Mizar Hands-on Tutorial (CICM2016)
  - This presentation is based on this tutorial.
- A.Grabowski, A. Kornilowicz and A. Naumowicz, Mizar in a Nutshell, Journal of Formalized Reasoning 3(2), pp. 153-245, 2010.
  - A de facto manual for the Mizar system.
- F.Wiedijk, Writing a Mizar article in nine easy steps.
  - Very nice tutorial, but the using library version is out of date.
- A. Naumowicz and J. Urban: A Guide to the Mizar Soft Type System (TYPES 2016)

Recommended links

- Mizar Homepage
- Mizar System
- Formalized Mathematics
- XML-ized presentation of Mizar articles
- Bibliography of Mizar Project
- Mizar mode for Emacs
- Mizar extension for VSCode
- emwiki: A browse and search system for the MML
- MizAR: parallelized AI/ATP, verification, and presentation service for Mizar

Formalization of multivariable calculus

- Focus on formalization of implicit function theorem

Motivation for formalizing multivariable calculus

- Differential geomerty
  - Undergraduate level, but have not been formalized for a long time.
  - There is a gap between a hand-written proof and a formalized proof because geometry tends to be intuitive.
  - We have started the formalization project of differential geometry since around 2017.
- Numerical analysis (of partial differential equations)
  - One of the most industrially utilized fields

Motivation for formalizing implicit function theorem

- The most fundamental tool for formalizing differential manifolds
    - Inverse function theorem is its corollary
- Status of formalization in other languages
    - Isabelle:
        - first-order differentiable version only?
    - Lean:
        - first-order differentiable version
        - They started formalization of differential geometry in around 2019(?), but they have already formalized differential manifolds, tangent bundles and Whitney's embedding theorem. They announced that Smale's sphere eversion theorem has been formalized today!

Implicit function theorem

Implicit function theorem (C^1 version)
$f : X \times Y \to Z$ be continuously Fréchet differentiable. If $(a, b) \in X \times Y$, $f(a, b) = c$, and $y \mapsto \frac{\partial f}{\partial y}(a, b)(0, y)$ is a Banach space isomorphism from Y onto Z, then there exist neighbourhoods U of a and V of b and a Fréchet differentiable function $g : U \to V$ such that $f(x, g(x)) = c$ and $f(x, y) = c$ if and only if $y = g(x)$, for all $(x, y) \in U \times V$.

(By using Fréchet derivative on Banach space, $f(x, y)$ can be treated as a two-variable function.)

Proof outline (1) (existence)

It does not lose generality to fix $b = 0$ and $c = 0$. (consider $f_1(x, y) = f(x, y + b) - c$.)

Consider

$$\Phi(x, y) = y - \left(\frac{\partial f}{\partial y}(a, 0)\right)^{-1} \cdot f(x, y).$$

Then $\frac{\partial \Phi}{\partial y}(a, 0) = 0$. So we can assume $\left|\frac{\partial \Phi}{\partial y}(x, y)\right| < \frac{1}{2}$ where $(x, y) \in U \times V$.

From mean value theorem, $|\Phi(x, y') - \Phi(x, y'')| < \frac{1}{2}|y' - y''|$ for all $x \in U$ and $y', y'' \in V$.

Therefore

$$\Phi_x : y \mapsto \Phi(x, y)$$

is contraction mapping. From Banach fixed-point theorem, Φ_x determines a unique continuous function $y = g(x)$ in $(x, y) \in U \times V$.

Statement of Banach fixed-point theorem

Banach fixed-point theorem

Let X be Banach space and $f : X \to X$ be a contraction mapping.
(i.e. $0 < \exists k < 1$ such that $|f(x) - f(x')| < k\,|x - x'|$ for all $x, x' \in E$)
Then there exists a unique fixed point $a \in X$ such that $f(a) = a$

Corollary

Let X be topological space, Y be Banach space, and $f : X \times Y \to Y$ be a function.
Assume $f_x : y \mapsto f(x, y)$ is continuous for fixed x.
If there exists $0 < k < 1$ independent of x such that f_x is contraction mapping,
then fixed point of f_x determines a unique continuous function $g : x \mapsto y$.

Proof outline (2) (g(x) is C^1 function)

g(x) is C^1 function: If f is C^1 function, then g is also C^1 function and $g'(x) = -Q^{-1}(x) \circ P(x)$ is satisfied where

$$P(x) = \frac{\partial f}{\partial x}(x, g(x)),\ Q(x) = \frac{\partial f}{\partial y}(x, g(x))$$

Outline of proof: Since f is differentiable, $0 = \Delta f = P \cdot dx + Q \cdot dy + \alpha(|dx| + |\Delta y|)$ where α depends on dx and Δy. Then $\Delta y = (-Q^{-1} \circ P) \cdot dx - (Q^{-1} \cdot \alpha)(|dx| + |\Delta y|)$. We can take β that satisifies $\Delta y = (-Q^{-1} \circ P) \cdot dx + \beta|dx|$ and $|\beta| \leq k|Q^{-1}||\alpha|$. Then $dx \to 0$ means $\beta \to 0$ and $g'(x) = -Q^{-1}(x) \circ P(x)$ is proved. Since $P(x)$ and $Q(x)$ is continuous, if inverse and composition operation preserve continuity, $g'(x)$ is also continuous.

Proof outline (3) (g(x) is C^n function)

g(x) is C^n function: If $f(x, y)$ is C^n function, then the implicit function g is C^n function.
Outline of proof: $g'(x) = -Q^{-1}(x) \circ P(x)$ then it is sufficient to prove inverse and composition also preserve C^n differentiability. It is concluded by using mathematical induction.

What should we prepare for our formalization?

- Fréchet derivative
  - Higher order derivatives
  - Partial derivatives
  - Derivatives of composition and inverse of functions
- Mean value theorem
- Banach fixed point theorem
- Jacobian matrix (for n-dimensional case)

Higher-order derivatives of single-variable functions

Formalization of f' on Z

```
definition
 let f be PartFunc of REAL,REAL;
 let Z be Subset of REAL;
 func diff(f,Z) -> Functional_Sequence of REAL,REAL means
:: TAYLOR_1:def 5
 it.0 = f|Z & for i be Nat holds it.(i+1) = (it.i) `| Z;
end;
```

Taylor expansion of a single-variable function

If $x \in (x_0 - r, x_0 + r)$ and $f(x)$ is n-order differentiable,
then there exists $0 < s < 1$ and

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \frac{f(x_0 + s(x - x_0))}{(n+1)!}(x - x_0)^{n+1}$$

is satisfied.

```
theorem :: TAYLOR_1:33
  for n be Nat,f be PartFunc of REAL,REAL,x0,r be Real st
  ].x0-r,x0+r.[ c= dom f & 0 < r & f is_differentiable_on n+1,].x0-r,x0+r.[
  for x be Real st x in ].x0-r,x0+r.[
  ex s be Real st 0 < s & s < 1
   & f.x  = Partial_Sums(Taylor(f,].x0-r,x0+r.[,x0,x)).n
          + (diff(f,].x0-r,x0+r.[).(n+1)).(x0+s*(x-x0)) * (x-x0) |^ (n+1) / ((n+1)!);
```

Cumbersome points of multivariable functions

nth-order partial derivative usually defined as follows:

"Let f be a function defined by the open set U of $\mathbb{R}^n$. (omitted) For a positive integer r, all partial differential coefficients of f up to the rth order

$$\frac{\partial^{\alpha_1+\alpha_2+\cdots+\alpha_n} f}{(\partial x^1)^{\alpha_1} \cdots (\partial x^n)^{\alpha_n}}, \alpha_i \geq 0, \alpha_1 + \alpha_2 + \cdots + \alpha_n \leq r$$

exist and are continuous on U, f is called a C^r-class function on U."

First-order derivatives → vectors

Second-order derivatives → matrices

nth-order derivatives → ???

Higher order derivatives in Banach spaces

If we regard $\mathbb{R}^n$,$\mathbb{R}^m$ as Banach spaces and formalize Fréchet derivative, it becomes analogous to the case of a single-variable function. A derivative is an element of the Banach space $\mathcal{L}(\mathbb{R}^n, \mathbb{R}^m)$.

In the higher-order derivative, they constitute "nested" Banch spaces as follows:

$$\mathbb{R}^m, \mathcal{L}(\mathbb{R}^n, \mathbb{R}^m), \mathcal{L}(\mathbb{R}^n, \mathcal{L}(\mathbb{R}^n, \mathbb{R}^m)), \mathcal{L}(\mathbb{R}^n, \mathcal{L}(\mathbb{R}^n, \mathcal{L}(\mathbb{R}^n, \mathbb{R}^m))) \cdots$$

It becomes more complicated when higher-order partial derivatives are introduced.

($\mathcal{L}(S, T)$: the normed linear space of bounded linear maps from S to T.)

Linear operators between normed spaces

Let S, T be a normed linear space, then a linear map $L : S \to T$ are formalized as follows:

```
definition
 let S,T be non empty addMagma;
 let L be Function of S,T;
 attr L is additive means :: VECTSP_1:def 20
 for x,y being Element of S holds L.(x+y) = L.x + L.y;
 attr L is homogeneous means :: LOPBAN_1:def 5
 for x being VECTOR of S,r being Real holds L.(r*x) = r*L.x;
end;

definition
 let S,T be RealLinearSpace;
 mode LinearOperator of S,T is additive homogeneous Function of S,T;
end;
```

Lipschitz continuity

Boundedness of a linear map L is formalized with ***Lipschitzian*** attribute.

```
definition
 let S,T be RealNormSpace;
 let f be LinearOperator of S,T;
 attr f is Lipschitzian means :: LOPBAN_1:def 8
 ex K being Real st 0 <= K & for x being VECTOR of S holds ||. L.x .|| <= K * ||. x .||;
end;
```

where

$$\|L\| = \sup\{\|L(x)\| : x \in S \;\&\; \|x\| \leq 1\}$$

Definition of Fréchet derivative

Definition

f has Fréchet derivative at $x_0 \in S$:
There exist $L \in \mathcal{L}(S,T)$, $R : S \to T$ where $\lim_{||h||\neq 0, ||h||\to 0} \frac{||R.h||}{||h||} = 0$, and
$\mathcal{N}$: neighborhood of $x_0 \in S$ and $\forall x \in \mathcal{N}$, $f(x) - f(x_0) = L(x - x_0) + R(x - x_0)$.

Usually it is expressed as

$$\frac{d}{dx} f(x_0) \in \mathcal{L}(S,T)$$

Fromalization of Fréchet derivative (1)

```
definition
  let S,T;
  let IT be PartFunc of S,T;
  attr R is RestFunc-like means :: NDIFF_1:def 5
  R is total
  & for h st h is non-zero holds (||.h.||")(#)(R/*h) is convergent
  & lim ((||.h.||")(#)(R/*h)) = 0.T;
end;

theorem :: NDIFF_1:23
 for R be PartFunc of S,T st R is total
 holds
 R is RestFunc-like
   iff
 for r be Real st r > 0 ex d be Real
 st d > 0
  & for z be Point of S st z <> 0.S & ||.z.|| < d
    holds ( ||.z.||" * ||. R/.z .||) < r;
```

Fromalization of Fréchet derivative (2)

```
definition
  let S,T;
  let f be PartFunc of S,T;
  let x0 be Point of S;
  pred f is_differentiable_in x0 means  :: NDIFF_1:def 6
  ex N being Neighbourhood of x0 st N c= dom f & ex L,R
  st for x be Point of S st x in N holds f/.x - f/.x0 = L. (x-x0) + R/.(x-x0);
end;

definition
  let S,T;
  let f be PartFunc of S,T;
  let x0 be Point of S;
  assume
  f is_differentiable_in x0;
  func diff(f,x0) -> Point of R_NormSpace_of_BoundedLinearOperators(S,T) means
  :: NDIFF_1:def 7
  ex N being Neighbourhood of x0 st N c= dom f & ex R st for x be Point of
  S st x in N holds f/.x-f/.x0 = it.(x-x0) + R/.(x-x0);
end;
```

Fréchet derivative function

$$f`|X : X \ni x \mapsto \frac{d}{dx} f(x) \in \mathcal{L}(S,T)$$

```
definition
  let X,S,T;
  let f be PartFunc of S,T;
  pred f is_differentiable_on X means  :: NDIFF_1:def 8
  X c=dom f & for x be Point of S st x in X holds f|X is_differentiable_in x;
end;

definition
  let S,T,X;
  let f be PartFunc of S,T;
  assume
  f is_differentiable_on X;
  func f`|X -> PartFunc of S,R_NormSpace_of_BoundedLinearOperators(S,T) means
  :: NDIFF_1:def 9
  dom it = X & for x be Point of S st x in X holds it/.x = diff(f,x);
end;
```

Inductive definition of higher-order derivatives

Second order derivative function:

$$(f'|X)'|X : x \in X \mapsto \frac{d}{dx}(f'|X)(x) \in \mathcal{L}(S, \mathcal{L}(S, T))$$

Third order derivative function:

$$((f'|X)'|X)'|X : x \in X \mapsto \frac{d}{dx}((f'|X)'|X)(x) \in \mathcal{L}(S, \mathcal{L}(S, \mathcal{L}(S, T)))$$

By repeating the same operation, higher-order derivatives are defined.

Although $\mathcal{L}(S, \mathcal{L}(S, T))$ is isormorphic to $\mathcal{L}^2(S \times S, T)$, these should be distinguished in strict formalization. (We also formalized this isomorphism.)

Formalization of a sequence of derivative spaces

diff_SP is a sequence of derivative spaces:

$$T, \mathcal{L}(S,T), \mathcal{L}(S,\mathcal{L}(S,T)), \mathcal{L}(S,\mathcal{L}(S,\mathcal{L}(S,T))) \cdots$$

```
definition let S,T be RealNormSpace;
  func diff_SP(S,T) -> Function means :: NDIFF_6:def 2
  dom it = NAT & it.0 = T & for i be Nat holds
  it.(i+1) = R_NormSpace_of_BoundedLinearOperators(S,modetrans(it.i));
end;

definition
  let S,T be RealNormSpace,i be Nat;
  func diff_SP(i,S,T) -> RealNormSpace equals
  :: NDIFF_6:def 3
  diff_SP(S,T).i;
end;
```

modetrans is just a type conversion functor from set to RealNormSpace.

Sequence of higher-order derivatives

diff is a sequence of derivatives:

$$\frac{df}{dx}, \frac{d^2 f}{dx^2}, \frac{d^3 f}{dx^3}, \ldots$$

```
definition let S,T be RealNormSpace,f be PartFunc of S,T,
    Z be Subset of S;
 func diff(f,Z) -> Function means  :: NDIFF_6:def 5
 dom it = NAT & it.0 = f|Z & for i be Nat holds it.(i+1) = modetrans(it.i,S,diff_SP(i,S,T)) `| Z;
end;

definition
 let S,T be RealNormSpace,f be PartFunc of S,T,
  Z be Subset of S,i be Nat;
 func diff(f,i,Z) -> PartFunc of S,diff_SP(i,S,T) equals  :: NDIFF_6:def 6
 diff(f,Z).i;
end;
```

Higher-order differentiability

Define nth-order differentiability by a predicate ***is_partial_differentiable_on n,Z*** using a sequence of higher-order derivatives.

```
definition
  let S,T be RealNormSpace,f be PartFunc of S,T;
  let Z be Subset of S,n be Nat;
  pred f is_differentiable_on n,Z means  :: NDIFF_6:def 7
  Z c= dom f & for i be Nat st i <= n-1 holds modetrans(diff(f,Z).i,S,diff_SP(i,S,T)) is_differentiable_on Z;
end;
```

Coordinate system (for partial derivative)

RealNormSpace-Sequence defines a finite sequence of norm spaces. A finite sequence (***FinSequence***) is a map which domain is equal to $\mathrm{Seg}\, n = \{i \in \mathbb{N} : i \in i \leq n\}$.

A direct product $\prod_{1 \leq i \leq n} G_i$ of a finite sequence of normed space $G = [G_1, G_2, \cdots, G_n]$ is generated by a functor ***product***. We regard the direct product as a coordinate system.

```
definition
  let G be RealLinearSpace-Sequence;
  func product G -> non empty strict RLSStruct equals :: PRVECT_2:def 9
  RLSStruct(# product(carr G),zeros G,[:addop G:],[:multop G:] #);
  coherence;
end;

definition
  let G be RealNormSpace-Sequence;
  func product G -> strict non empty NORMSTR means  :: PRVECT_2:def 13
  the RLSStruct of it = product(G qua RealLinearSpace-Sequence)
  & the normF of it = productnorm G;
end;
```

Extract a coordinate system component

Extract the i-th element of the vector $x = (x_1, x_2, \cdots, x_n)$

$$\mathrm{proj}(i) : \prod_{1 \leq k \leq n} G_k \ni (x_1, x_2, \cdots, x_n) \mapsto x_i \in G_i$$

```
definition
  let G be RealNormSpace-Sequence;
  let i be Element of dom G;
  func proj i -> Function of product G,G.i means :: NDIFF_5:def 3
  for x be Element of product carr G holds it.x = x.i;
end;
```

Substitute a coordinate system component

Substitute the i-th element of the vector $x = (x_1, x_2, \cdots, x_n)$

$$\mathrm{reproj}(i, x) : G_i \ni r \mapsto (x_1, x_2, \cdots, x_{i-1}, r, x_{i+1}, \cdots, x_n) \in \prod_{1 \leq k \leq n} G_k$$

```
definition
  let G be RealNormSpace-Sequence,i be Element of dom G,
  x be Element of product G;
  func reproj(i,x) -> Function of G.i,product G means :: NDIFF_5:def 4
  for r be Element of G.i holds it.r = x +* (i,r);
end;
```

First-order partial differentiability

Let $f : \prod_i G_i \to F$ be a function, then f ***is_partial_differentiable_in*** x, i is a partial differentiability of the ith-component at $x = (x_1, x_2, \cdots, x_n) \in \prod_i G_i$.

```
definition
  let G be RealNormSpace-Sequence,F be RealNormSpace,let i be set,
  f be PartFunc of product G,F;
  let x being Element of product G;
  pred f is_partial_differentiable_in x,i means :: NDIFF_5:def 6
  f * reproj(In(i,dom G),x) is_differentiable_in proj(In(i,dom G)).x;
end;
```

First-order partial derivatives

partdiff(f, x, i) is a partial derivative function of the ith-component.

```
definition
  let G be RealNormSpace-Sequence,F be RealNormSpace;
  let i be set,f be PartFunc of product G,F;
  let x be Point of product G;
  func partdiff(f,x,i)
  -> Point of R_NormSpace_of_BoundedLinearOperators(G.In(i,dom G),F)
  equals :: NDIFF_5:def 7
  diff(f * reproj(In(i,dom G),x),proj(In(i,dom G)).x);
end;
```

Higher order partial derivative spaces

Partdiff_SP is a sequence of a n-th partial derivative spaces:

Let T be a normed space, $S = (X, Y, Z)$, $I = (2, 1, 2, 1)$.
Then ***Partdiff_SP(***S***,***T***,***I***)*** satisfies

$$\mathcal{L}(Y, T), \mathcal{L}(X, \mathcal{L}(Y, T)), \mathcal{L}(Y, \mathcal{L}(X, \mathcal{L}(Y, T))), \mathcal{L}(X, \mathcal{L}(Y, \mathcal{L}(X, \mathcal{L}(Y, T)))).$$

Formalization of higher order partial derivative spaces

```
definition
  let S be RealNormSpace-Sequence,T be RealNormSpace,
  I be non empty FinSequence of dom S;
  func Partdiff_SP(S,T,I) -> RealNormSpace-Sequence means  :: NDIFF14:def 3
    dom it = dom I
  & it.1 = R_NormSpace_of_BoundedLinearOperators(S.(In(I.1,dom S)),T)
  & for i be Nat st 1<=i & i < len I holds
    it.(i+1) = R_NormSpace_of_BoundedLinearOperators(S.(In(I.(i+1),dom S)),modetrans(it.i));
end;

definition
  let S be RealNormSpace-Sequence, T be RealNormSpace,
      I be non empty FinSequence of dom S, i be Nat;
  assume 1 <= i & i <= len I;
  func Partdiff_SP(S,T,I,i) -> RealNormSpace equals :: NDIFF14:def 4
  Partdiff_SP(S,T,I).i;
end;

definition
  let S be RealNormSpace-Sequence, T be RealNormSpace,
      I be non empty FinSequence of dom S;
  func PartdiffSP(S,T,I) -> RealNormSpace equals :: NDIFF14:def 5
  (Partdiff_SP(S,T,I)).(len I);
end;
```

Higher-order partial derivatives

Let T be a normed space, $S = (X, Y, Z)$, $I = (2, 1, 2, 1)$.

$$f : X \times Y \times Z \ni (x, y, z) \mapsto f(x, y, z) \in T$$

PartDiffSeq(f,Z,I) is defined as:

$$\left(\frac{\partial}{\partial_Y} f, \frac{\partial^2 f}{\partial_x \partial_y}, \frac{\partial^3 f}{\partial_y \partial_x \partial_y}, \frac{\partial^4 f}{\partial_x \partial_y \partial_x \partial_y} \right).$$

Formalization of higher-order partial derivatives

Partial differentiability

```
definition
  let S be RealNormSpace-Sequence, T be RealNormSpace, Z be Subset of product S,
      I be non empty FinSequence of dom S,
      f be PartFunc of product S,T;
  pred f is_partial_differentiable_on Z,I means :: NDIFF14:def 8
    f is_partial_differentiable_on Z,I.1
  & for i be Nat st 1 <= i & i < len I
    holds PartDiffSeq(f,Z,I,i) is_partial_differentiable_on Z,I.(i+1);
end;
```

Partial derivatives

```
definition
  let S be RealNormSpace-Sequence,T be RealNormSpace,
  Z be Subset of product S, I be non empty FinSequence of (dom S),
  f be PartFunc of product S,T;
  func f `partial| (Z,I) ->
  PartFunc of product S,Partdiff_SP(S,T,I,len I) equals :: NDIFF14:def 9
  PartDiffSeq (f,Z,I,len I);
end;
```

Bounded bilinear functions

To define the n-order composite derivative, we use the derivative of the bounded bilinear map. A type of bilinear map $E \times F \to G$ is formalized:

```
definition
  let E,F,G be RealNormSpace;
  mode BilinearOperator of E,F,G is Bilinear Function of [:E,F:],G;
end;
```

Boundedness (continuity) is also characterized as follows.

```
definition
  let E,F,G be RealNormSpace;
  let f be BilinearOperator of E,F,G;
  attr f is Lipschitzian means
  :: LOPBAN_9:def 3
  ex K being Real st 0 <= K &
  for x being VECTOR of E,y being VECTOR of F
  holds ||. f.(x,y) .|| <= K * ||. x .|| * ||. y .||;
end;
```

Differentiation of bounded bilinear functions

Bounded linear functions and bounded bilinear functions are continuously differentiable at arbitrary ranks.

```
theorem :: NDIFF12:20
  for L be Lipschitzian LinearOperator of E,F
  holds for i be Nat holds
    diff(L,i,[#]E) is_differentiable_on [#]E & diff(L,i,[#]E) `| [#]E is_continuous_on [#]E;

theorem :: NDIFF12:21
  for B be Lipschitzian BilinearOperator of E,F,G
  holds
  for i be Nat
  holds diff(B,i,[#][:E,F:]) is_differentiable_on [#][:E,F:]
   & diff(B,i,[#][:E,F:]) `| [#][:E,F:] is_continuous_on [#][:E,F:];
```

Higher order differentiability of bounded bilinear functions

$w : E \times F \to S$ is C^n function if $u : E \to S$ and $v : F \to S$ is C^n function and $w = u \times v$.

```
theorem :: NDIFF13:61
  for S,E,F be RealNormSpace, u be PartFunc of S,E, v be PartFunc of S,F,
      w be PartFunc of S,[:E,F:], Z be Subset of S, i be Nat
  st w = <:u,v:>
   & u is_differentiable_on i+1,Z & diff(u,i+1,Z) is_continuous_on Z
   & v is_differentiable_on i+1,Z & diff(v,i+1,Z) is_continuous_on Z
 holds
   w is_differentiable_on i+1,Z & diff(w,i+1,Z) is_continuous_on Z;
```

Partial derivative and total derivative

nth-order continuous total differentiability is equvalent to nth-order continuous partial differentiability for all variables.

```
theorem :: NDIFF14:36
  for G being RealNormSpace-Sequence,
      X being non empty open Subset of product G,
      F being RealNormSpace,
      f being PartFunc of product G,F
  for n being Nat st 1 <= n
  holds ( f is_differentiable_on n,X & diff(f,n,X) is_continuous_on X )
    iff
    for I being non empty FinSequence of (dom G)
    st len I <= n holds f is_partial_differentiable_on X,I & f `partial| (X,I) is_continuous_on X;
```

Jabobian matrix

Jacobian matrix is used for formalizing n-dimensional Euclidean space version.

The existence of the inverse function is verified by $\det M \neq 0$.

```
definition
  let m,n be non zero Nat;
  let f be PartFunc of REAL m,REAL n;
  let x be Element of REAL m;

  func Jacobian(f,x) -> Matrix of m,n,F_Real
  means
  for i,j be Nat st i in Seg m & j in Seg n
  holds it * (i,j) = partdiff(f,x,i,j);
end;
```

Formalization of Banach fixed point theorem

Basic version

```
theorem :: NDIFF_8:28
  for X be RealBanachSpace,
      S be non empty Subset of X,
      f be PartFunc of X,X
  st S is closed & dom f = S & rng f c= S
   & ex k be Real
     st 0 < k < 1
      & for x,y be Point of X st x in S & y in S
        holds ||. f/.x - f/.y .|| <= k * ||.x-y.||
  holds
    (ex x0 be Point of X st x0 in S & f.x0 = x0)
   &(for x0,y0 be Point of X st x0 in S & y0 in S & f.x0 = x0 & f.y0 = y0
     holds x0 = y0);
```

Extended version1

Extended version2

Implicit function theorem

- Fréchet derivative version
  - Existence: NDIFF_8:36
  - 1st-order differentiability: NDIFF_9:21
- Euclidean space version
  - Existence and 1st-order differentiability: NDIFF11:33
- nth-order differentiability: NDIFF13:65

Inverse function theorem

- Fréchet derivative version
  - Existence and 1st-order differentiability: NDIFF10:17
- Euclidean space version
  - Existence and 1st-order differentiability: NDIFF11:34
- nth-order differentiability: NDIFF13:67

Conclusions and future works

- Current status
  - Existence and nth-order differentiability of implicit and inverse function theorems have been formalized.
  - n-dimensional cases using Jacobian matrix were also formalized.
- Future works
  - differential manifold - Stokes' theorem
  - calculus of variations - Isoperimetric theorem
  - numerical analysis - Newton's method

Thank you for your attention

Please feel free to ask any questions