Automated Reasoning in First-order Real Vector Spaces

by

Carl Kwan

B.Sc., The University of British Columbia, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2019

Abstract

We present a formal first-order theory of arbitrary dimensional real vector spaces in ACL2(r). This includes methods for reasoning about real vectors, metric spaces, continuity, and multivariate convex functions, which, by necessity, involves the formalisation of a selection of classically significant and useful mathematical theorems. Notable formalisations include the first mechanical proof of the Cauchy-Schwarz inequality in a first-order logic and a theorem attributed to Yurii Nesterov for characterising convex functions with Lipschitz continuous gradients.

One motivation for this work is to further contribute to the automated deduction of theorems involving such mathematical objects. Another motivation is the potential applications in the verification of analog circuits, cyberphysical systems, and machine learning algorithms. Indeed, common techniques in these areas involve reasoning about the algebraic properties of higher dimensional structures over the reals or the extremal values, monotonicity, and convexity properties of functions over these structures. These applications, along with Nesterov's theorem, demonstrate that our formalisation serves as a useful foundation in the space of reasoning and verification research.

Preface

This thesis contains the research I conducted while in UBC's Integrated Systems Design Lab under the guidance of Mark Greenstreet. Chapter 4, Chapter 5, and Chapter 6 are based on work previously published as [21]. Similarly, Chapter 7 and Chapter 8 are based on work published as [20]. Both papers were jointly written with Mark Greenstreet.

Table of Contents

Abstract ii

Preface iii

Table of Contents iv

List of Tables vi

List of Figures vii

List of Programs viii

Acknowledgements x

1 Introduction 1

2 Related Work 4

3 Preliminaries 6
- 3.1 Vector and Inner Product Spaces 6
- 3.2 The Cauchy-Schwarz Inequality 7
- 3.3 Metric Spaces 7
- 3.4 Continuity & Differentiability in $\mathbb{R}^n$ 8
- 3.5 Convex Functions & $\mathcal{F}_L^1(\mathbb{R}^n)$ 8
- 3.6 ACL2(r) 10
- 3.7 Non-standard Analysis and ACL2(r) 11

4 Mechanical First-order Real Vector Spaces 13
- 4.1 Vector Space Axioms 15
- 4.2 Inner Product Space Axioms 18

5 Formalising Cauchy-Schwarz 21
5.1 Cauchy-Schwarz I 24
5.2 Cauchy-Schwarz II 24
5.3 Conditions for Equality 28
5.4 Final Form of Cauchy-Schwarz 29

6 Reasoning about Continuity 32
6.1 Metric Space Axioms 32
6.2 Continuity $\mathbb{R}^n \to \mathbb{R}$ 35
6.3 Why are Non-classical Recursive Functions Prohibited? . . . 38

7 Mechanical Multivariate Convex Functions 40
7.1 Example Functions 40
7.2 Reasoning about Convexity 42
7.3 A Useful Lemma 44

8 Formalising Nesterov's Theorem 48
8.1 Approach and Basic Definitions & Lemmas 49
8.2 Instantiating Inequalities 55
8.3 Taking Limits 56
8.4 Nesterov's Final Form 57
8.5 Ambiguities in Nesterov's Statement 59

9 Conclusion and Future Work 61

Bibliography 63

Appendices

A Classical Proofs 68

List of Tables

4.1 Key definitions in Chapter 4. 13
4.2 Key theorems in Chapter 4. 14

5.1 Key definitions in Chapter 5. 21
5.2 Key theorems in Chapter 5. 23

6.1 Key definitions in Chapter 6. 32
6.2 Key theorems in Chapter 6. 33

7.1 Key definitions in Chapter 7. 40
7.2 Key theorems in Chapter 7. 41

8.1 Key theorems in Chapter 8. 49
8.2 Multiple interpretations of Nesterov's statement. 60

List of Figures

1.1 Thesis organisation 3

5.1 Proof structure of Cauchy-Schwarz 22

8.1 Nesterov's proof of Theorem 3 50
8.2 Our proof of Theorem 3 51

List of Programs

2.1 Cauchy-Schwarz in Coq 5
2.2 Cauchy-Schwarz in Mizar 5
2.3 Cauchy-Schwarz in HOL Light 5

4.1 Vector addition. 16
4.2 Scalar multiplication. 16
4.3 Vector subtraction is equivalent to `(vec-+ vec1 (scalar-* -1 vec2))`. 16
4.4 Using the equivalence between `vector--` and `vec---x`. . . . 17
4.5 The dot product. 18
4.6 An example of using commutativity to prove bilinearity of the dot product. 19

5.1 Applying bilinearity of the dot product to prove a simple identity. 25
5.2 Substituting v for $\langle v, v\rangle^{-1}\langle u, v\rangle v$. 26
5.3 Final form of Cauchy-Schwarz I. 26
5.4 Showing Cauchy-Schwarz II. 27
5.5 Linearity follows from equality. 28
5.6 Introducing a Skolem function for linearity. 29
5.7 The conditions for equality for Cauchy-Schwarz I. 29
5.8 The final form of Cauchy-Schwarz. 30

6.1 The Euclidean metric. 34
6.2 The Euclidean norm. 34
6.3 Showing arbitrary entries of a vector are infinitesimal via the maximum element. 37
6.4 A fantastical recognizer for vectors with infinitesimal entries that doesn't exist. 37
6.5 A theorem positing the infinitesimalness of arbitrary entries in $x - y$. 37
6.6 A theorem positing `sum` is continuous. 38

6.7 The function $g(x, y) = xy$ is continuous. 38
6.8 An impossible function in ACL2(r). 38

7.1 $f(x) = x^2$ is convex. 43
7.2 Encapsulating a constant function `cvnf-1` and stating its convexity. 44
7.3 Supressing the definition of `cvfn-1`, stating a special case of distributivity, and stating the convexity of $a \cdot f$. 45
7.4 The string of equalities in Equation 7.3. 46
7.5 Pseudo triangle inequality formalised. 47

8.1 The first inequality follows from Lipschitz continuity. 52
8.2 The second inequality follows from Cauchy-Schwarz. 53
8.3 The desired implication in an expanded form. 53
8.4 Skolem functions that allow us to invoke the `forall` quantifier. 54
8.5 Nest. 0 implies Nest. 4 54
8.6 An "obvious" way to state Nest. 2 implies Nest. 3. 55
8.7 Instantiating two copies of Nest. 2. 56
8.8 Introducing `standard-vecp` into the hypotheses. 57
8.9 Theorem 3. 58
8.10 Nest. 5 in Polish notation. 59

Acknowledgements

I would like to thank Audrey Xu, Jiasi Yu, Kevin Cheang, Adrian She, Max Hui-Bon-Hoa, Chris Chen, Itrat Akhter, Justin Reiher, Yan Peng, Bruce Shepherd, Anne Condon, Warren Hunt, Matt Kaufmann, Ruben Gamboa, Alan Hu, Will Evans, and Mark Greenstreet. Words cannot fully express my appreciation.

Chapter 1

Introduction

Automated reasoning in general and automated theorem proving in particular play a dual role in computer science, treading a line between theory and application. On the application side, theorem provers are a tool for those working in formal methods to reason about and verify complex systems – a more rigourous approach compared to testing. The limitations of testing manifest in a variety of flaws leading to potentially significant human or monetary loss. One high-profile example is the Pentium FDIV bug which has since propelled theorem proving for verification to become a standard in the microprocessor industry [8]. Other examples include the MIM-104 Patriot Missile failure [45], the Therac-25 radiation therapy machine accidents [25], the Vancouver Stock Exchange rounding error [47], and the Meltdown [26] and Spectre [18] vulnerabilities. On the fundamental side, computer proofs provide a rigourous justification for mathematical statements beyond that found in human proofs. Moreover, automated tools have been used to prove novel theorems where human approaches were fallible or otherwise insufficient (at the time). Examples include the four colour theorem [4, 5], the Robbins conjecture [30], a special case of the Erdös discrepancy problem [19], and, more recently, Boolean Pythagorean triples [15] and Schur Number Five [14].

A common theme among these problems is that they all involve discrete mathematical structures which are particularly amenable to computational methods. However, reasoning about continuous structures, such as the reals, poses difficulty as any representation of (some) real numbers inherently involve some form of infinity. The typical computer science solution is to use floating-point numbers, which trades the precision of the reals for a finite but malleable representation. But the speed-ups afforded by techniques for manipulating floating-point numbers can lead to errors that are fundamental to the use of finite representations for infinite domains. In fact, this is exemplified by the Pentium FDIV bug where the look-up table for computing floating-point division was missing entries. On the other hand, real numbers are undoubtedly useful. Calculus and real analysis in general have numerous applications in science beyond just pure mathematics. In formal

methods, recent efforts have been focused on the verification of analog circuits, cyberphysical systems, and optimisation algorithms used in machine learning. The unifying theme in these domains is that they all involve reasoning about functions in real vector spaces – typically about their extremal values, monotonicity, or convexity properties. In particular, these involve not only univariate functions, but multivariate ones, too.

There are, indeed, several well-established approaches to formalising (in the theorem prover sense) the reals (in the univariate sense) but, to the best of our knowledge, only one existing formalisation in a first-order logic. Likewise, we are not aware of any first-order logic formalisation of the reals for higher dimensional structures. First-order logic is of particular interest because it is sufficiently expressive for reasoning about computer systems while maintaining highly automated decision procedures that higher-order logic cannot provide. How can we reason about arbitrary dimensional real vector spaces in a first-order logic?

This thesis presents the first formalisation in a first-order logic, interactive theorem prover of $\mathbb{R}^n$ as an inner-product space and a metric space. We demonstrate that our formulation provides a useful foundation for reasoning about such mathematical structures by mechanically proving interesting theorems about vector spaces and convex functions.

The organisation of this thesis is visualised by Figure 1.1. In particular, we present the first formal first-order theory of arbitrary dimensional real vector, inner product, and metric spaces (Chapters 4 and 6). By necessity, this includes a formal proof of the Cauchy-Schwarz inequality, which is the first of its kind in a first-order logic (Chapter 5). We also present solutions for reasoning in these theories while avoiding fundamental soundness-motivated logical limitations in expressibility. Finally, we apply our theories by reasoning about convex optimisation techniques commonly used in machine learning (Chapter 7). This includes a formal proof for a set of equivalent conditions for inclusion in the class of convex functions with Lipschitz continuous gradients – a theorem that, to the best of our knowledge, has yet to be fully published in a single piece of literature with a correct and unambiguous proof (Chapter 8).

Throughout this work, we use ACL2(r), a highly automated and efficient theorem prover in a first-order logic that supports reasoning about real numbers.

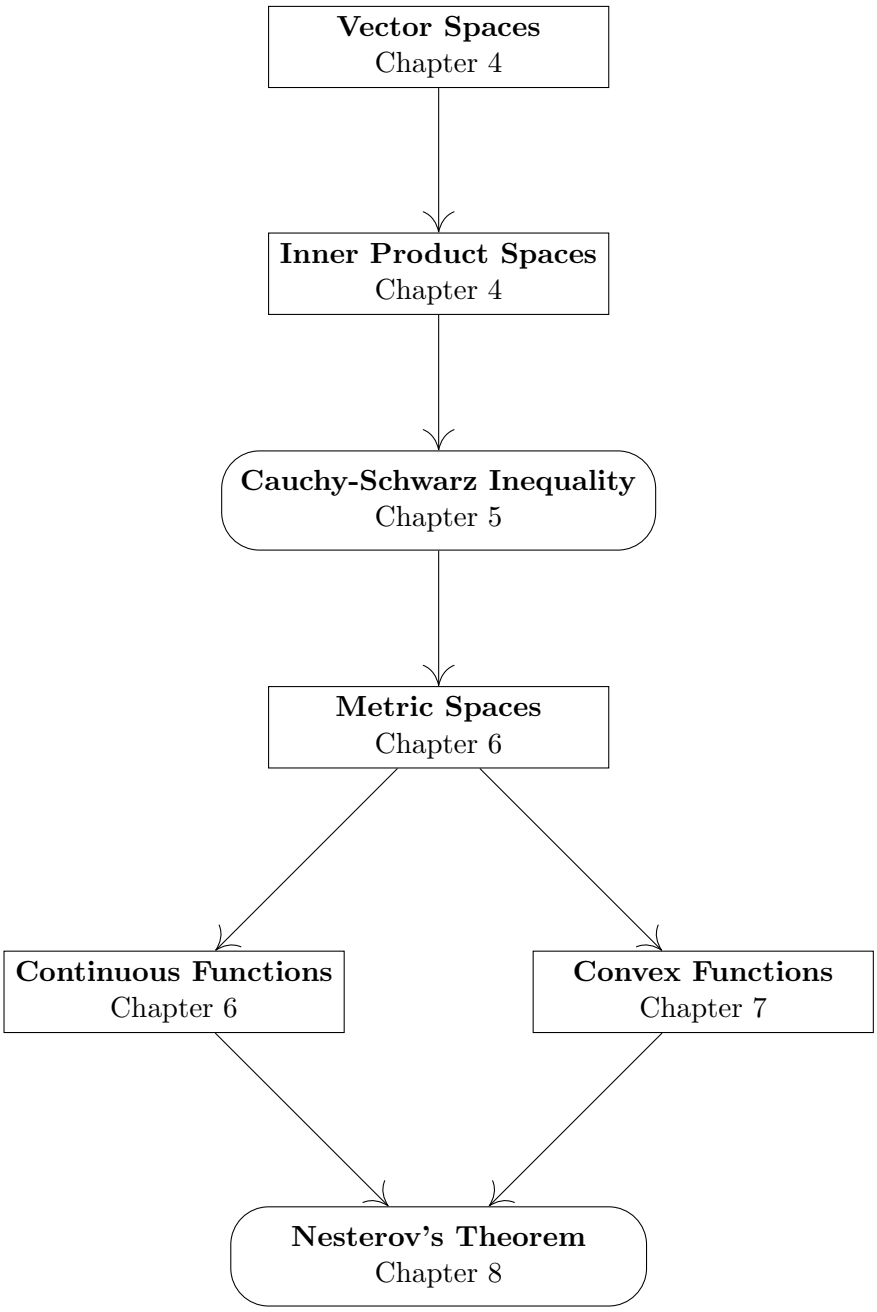

Figure 1.1: Thesis organisation. Major theorems are denoted by round corners.

Chapter 2

Related Work

While this thesis focuses on $\mathbb{R}^n$ from two perspectives, most other formalisations in the literature outline only one view of $\mathbb{R}^n$ – usually either neglecting to address metrics or lacking in the development of vectors. Moreover, to the best of our knowledge, these formalisations of $\mathbb{R}^n$ are verified in a higher-order setting. For example, a theory of complex vectors with a nod towards applications in physics was formalised in HOL Light but does not address metrics [3]. On the other end of the spectrum, there is a formalisation of Euclidean metric spaces and abstract metric spaces in HOL that do not fully include a theory of real vectors [12, 29]. This observation extends to Coq [46].

Within ACL2(r), there has been formalisation for some special cases of n. The work that handled $n = 1$ is indeed fundamental and indispensable [11]. We may view $\mathbb{C} \simeq \mathbb{R}^2$ as a vector space over $\mathbb{R}$ so $n = 2$ is immediate since ACL2(r) supports complex numbers. Moreover, extensions of $\mathbb{C}$ such as the quaternions $\mathbb{H}$ and the octonions $\mathbb{O}$ with far richer mathematical structure than typical vector spaces have recently been formalised, which addresses the cases of $n = 4$ and $n = 8$ [9].

Theorem provers with prior formalisations of Cauchy-Schwarz include HOL Light [13], Isabelle [38], Metamath [32], Coq [28], Mizar [1], PROOFPOWER [40], and PVS [48]. However, it appears most statements of these theorems do not mention the conditions for equality. To the best of our knowledge, only Metamath has a statement concerning the equality case of the Cauchy-Schwarz inequality [31].

We know of no other formalisations of convex functions in the current literature but base our formalisation on reference texts [7, 33]. Most of the mathematics in this thesis can be found in standard texts such as [16, 22, 42, 44]. Non-standard analysis is less standard of a topic but some common references are [24, 27, 41].

Program 2.1 Cauchy-Schwarz in Coq. Originally formalised by Daniel de Rauglaudre.

```
Notation "'Σ' ( i = b , e ) , g" := (summation b e (λ i, (g)%R)).
Notation "u .[ i ]" := (List.nth (pred i) u 0%R).
Cauchy_Schwarz_inequality
     : ∀ (u v : list R) (n : nat),
       (Σ (k = 1, n), (u.[k] * v.[k])^2
        ≤ Σ (k = 1, n), ((u.[k])^2) * Σ (k = 1, n), ((v.[k])^2))%R
```

Program 2.2 Cauchy-Schwarz in Mizar. Originally formalised by Jaroslaw Kotowicz.

```
theorem :: HERMITAN:45
:: Schwarz inequality
for V be VectSp of F_Complex, v,w be Vector of V
for f be diagReR+0valued hermitan-Form of V
holds |. f.[v,w] .|^2 <= signnorm(f,v) * signnorm(f,w);
```

Program 2.3 Cauchy-Schwarz in HOL Light. Originally formalised by John Harrison.

```
|- !x:real^N y. abs(x dot y) <= norm(x) * norm(y)
```

Chapter 3

Preliminaries

3.1 Vector and Inner Product Spaces

A vector space is a couple (V, F) where V is a set and F a field equipped with scalar multiplication such that, for any $v, u, w \in V$ and any $a, b \in F$,

$$v + (u + w) = (v + u) + w \quad \textit{(associativity)} \tag{3.1}$$
$$v + u = u + v \quad \textit{(commutativity)} \tag{3.2}$$
$$\exists\ 0 \in V,\ v + 0 = v \quad \textit{(additive identity)} \tag{3.3}$$
$$\exists\ -v \in V,\ v + (-v) = 0 \quad \textit{(additive inverse)} \tag{3.4}$$
$$a(bv) = (ab)v \quad \textit{(compatibility)} \tag{3.5}$$
$$1v = v \quad \textit{(scalar identity)} \tag{3.6}$$
$$a(v + u) = av + au \quad \textit{(distributivity of vector addition)} \tag{3.7}$$
$$(a + b)v = av + bv \quad \textit{(distributivity of field addition).} \tag{3.8}$$

An inner product space is a triple $(V, F, \langle -, - \rangle)$ where (V, F) is a vector space and $\langle -, - \rangle : V \times V \to F$ is a function called an inner product, i.e. a function satisfying

$$\langle au + v, w \rangle = a\langle u, w \rangle + \langle v, w \rangle \quad \textit{(linearity in the first coordinate)} \tag{3.9}$$
$$\langle u, v \rangle = \overline{\langle v, u \rangle} \text{ when } F = \mathbb{C} \quad \textit{(conjugate symmetry)} \tag{3.10}$$
$$\langle u, u \rangle \geq 0 \text{ with equality iff } u = 0 \quad \textit{(positive definiteness)} \tag{3.11}$$

for any $u, v, w \in V$ and $a \in F$. If F is not complex, then we may simply replace conjugate symmetry with symmetry, i.e. $\langle u, v \rangle = \langle v, u \rangle$. Indeed, for the purposes of this thesis, we take $\mathbb{R}^n$ over $\mathbb{R}$ to be our vector space and endow it with the dot product

$$\langle (u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n) \rangle = u_1 v_1 + u_2 v_2 + \cdots u_n v_n \tag{3.12}$$

to obtain an inner product space. In particular, we note that one may obtain different inner product spaces by replacing any entry of the triple $(\mathbb{R}^n, \mathbb{R}, \langle -, - \rangle)$ with an object of the respective type, but leave such a generalisation for future work.

3.2 The Cauchy-Schwarz Inequality

Cauchy-Schwarz allows us to relate the magnitude of vectors to their inner product. Let $\|\cdot\|$ be the norm induced by $\langle -, - \rangle$.

Theorem 1. Formally, the Cauchy-Schwarz inequality states

$$|\langle u, v\rangle|^2 \leq \langle u, u\rangle \cdot \langle v, v\rangle \qquad \text{Cauchy-Schwarz I}$$

or, equivalently,

$$|\langle u, v\rangle| \leq \|u\|\|v\| \qquad \text{Cauchy-Schwarz II}$$

for any vectors u, v. Moreover, equality holds iff u, v are linearly dependent.

Proof (sketch). If $v = 0$, then the claims are immediate. Suppose $v \neq 0$ and let a be a field element. Observe

$$0 \leq \|u - av\|^2 = \langle u - av, u - av\rangle = \|u\|^2 - 2a\langle u, v\rangle + a^2\|v\|^2. \tag{3.13}$$

Setting $a = \|v\|^{-2}\langle u, v\rangle$ and rearranging produces Cauchy-Schwarz I. Take square roots and we get Cauchy-Schwarz II. Note that $0 \leq \|u - av\|^2$ is the only step with an inequality so there is equality iff $u = av$. □

More details will be provided as we discuss the formal proof – especially the steps that involve "rearranging".

3.3 Metric Spaces

Observe by setting $d(x, y) = \|x - y\|$, a *metric* is induced. Metric spaces are topological spaces equipped with a metric function which is a rigorous approach to defining the intuitive notion of distance between two vectors. Formally, a metric space is a couple (M, d) where M is a set and $d : M \times M \to \mathbb{R}$ a function satisfying

$$d(x, y) = 0 \iff x = y \qquad \textit{(definitness)} \tag{3.14}$$

$$d(x, y) = d(y, x) \qquad \textit{(symmetry)} \tag{3.15}$$

$$d(x, y) \leq d(x, z) + d(z, y) \qquad \textit{(triangle inequality).} \tag{3.16}$$

From this it follows that

$$d(x, y) \geq 0 \tag{3.17}$$

because

$$0 = d(x, x) \leq d(x, y) + d(y, x) = 2d(x, y). \tag{3.18}$$

A function $f : M \to M'$ between metric spaces is *continuous* if for every $x \in M$ and for any $\epsilon > 0$, there is a $\delta > 0$ such that

$$d_M(x, y) < \delta \implies d_{M'}(f(x), f(y)) < \epsilon \tag{3.19}$$

for any $y \in E$.

3.4 Continuity & Differentiability in $\mathbb{R}^n$

When $M = M' = \mathbb{R}$, a univariate function $f : \mathbb{R} \to \mathbb{R}$ is continuous if for any $x \in \mathbb{R}$ and $\epsilon > 0$, there is a $\delta > 0$ such that for any $y \in \mathbb{R}$, if $|x - y| < \delta$, then $|f(x) - f(y)| < \epsilon$. Moreover, the *derivative* of f is defined to be

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \tag{3.20}$$

if such a form exists.

For a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$, continuity is defined similarly: we call f continuous if for any $x \in \mathbb{R}^n$ and $\epsilon > 0$, there is a $\delta > 0$ such that for any $y \in \mathbb{R}^n$, if $\|x - y\| < \delta$, then $|f(x) - f(y)| < \epsilon$. Likewise, if it exists, there is a derivative for multivariate functions defined similarly to the univariate case:

$$\langle f'(x), y \rangle = \lim_{h \to 0} \frac{f\,(x + hy) - f(x)}{h}. \tag{3.21}$$

We call $f' : \mathbb{R}^n \to \mathbb{R}^n$ the *gradient* of f and it satisfies

$$f'(x) = (f_1'(x_1), f_2'(x_2), \ldots, f_n'(x_n)) \tag{3.22}$$

where $f_i'(x)$ is the univariate derivative of f with respect to the i-th component x_i of x.

3.5 Convex Functions & $\mathcal{F}_L^1(\mathbb{R}^n)$

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if for any $x, y \in \mathbb{R}^n$ and $\alpha \in [0, 1]$,

$$\alpha f(x) + (1 - \alpha) f(y) \geq f(\alpha x + (1 - \alpha) y). \tag{3.23}$$

Equivalently, if f is differentiable once with gradient f', then it is convex if

$$f(y) \geq f(x) + \langle f'(x), y - x \rangle. \tag{3.24}$$

Write $\mathcal{F}(\mathbb{R}^n)$ to denote the class of convex functions from $\mathbb{R}^n$ to $\mathbb{R}$. Examples of convex functions include $f(x) = x^2$, $\|\cdot\|_2$, and $\|\cdot\|_2^2$. Moreover, the class of convex functions is closed under certain operations.

Theorem 2. If $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R} \to \mathbb{R}$ are convex with h monotonically increasing, then

1. $a \cdot f$ is convex for any real $a \geq 0$,
2. $f + g$ is convex,
3. $h \circ f$ is convex.

The proof of these claims follow directly from the definitions which we defer to Appendix A.

Often, convex optimisation algorithms require f to be both convex and sufficiently "smooth". Here, we take "smooth" to be stronger than continuous but not necessarily differentiable. In particular, we say that $f : \mathbb{R}^n \to \mathbb{R}$ is *Lipschitz continuous* if for any $x, y \in \mathbb{R}^n$ there is some $L > 0$ such that

$$\|f(x) - f(y)\| \leq L\|x - y\|. \tag{3.25}$$

Informally, we have the following chain of inclusions for classes of functions:

$$\text{Differentiable} \subset \text{Lipschitz Continuous} \subset \text{Continuous}.$$

We write $\mathcal{F}^1(\mathbb{R}^n)$ for the class of once continuously differentiable convex functions on $\mathbb{R}^n$ and $\mathcal{F}_L^1(\mathbb{R}^n)$ for functions in $\mathcal{F}^1(\mathbb{R}^n)$ with Lipschitz continuous gradient with constant L. Functions in the class $\mathcal{F}_L^1(\mathbb{R}^n)$ have many useful properties for optimisation. A significant result of this thesis is proving a theorem that gives six "equivalent" ways of showing that a convex function is in $\mathcal{F}_L^1(\mathbb{R}^n)$:

Theorem 3. Let $f \in \mathcal{F}^1(\mathbb{R}^n)$, $x, y \in \mathbb{R}^n$ and $\alpha \in [0, 1]$. The following

conditions are equivalent to $f \in \mathcal{F}_L^1(\mathbb{R}^n)$:

$$f(y) \leq f(x) + \langle f'(x), y - x \rangle + \frac{L}{2}\|x - y\|^2 \qquad \text{Nest. 1}$$

$$f(x) + \langle f'(x), y - x \rangle + \frac{1}{2L}\|f'(x) - f'(y)\|^2 \leq f(y) \qquad \text{Nest. 2}$$

$$\frac{1}{L}\|f'(x) - f'(y)\|^2 \leq \langle f'(x) - f'(y), x - y \rangle \qquad \text{Nest. 3}$$

$$\langle f'(x) - f'(y), x - y \rangle \leq L\|x - y\|^2 \qquad \text{Nest. 4}$$

$$f(\alpha x + (1-\alpha)y) + \frac{\alpha(1-\alpha)}{2L}\|f'(x) - f'(y)\|^2 \leq \alpha f(x) + (1-\alpha)f(y) \qquad \text{Nest. 5}$$

$$\alpha f(x) + (1-\alpha)f(y) \leq f(\alpha x + (1-\alpha)y) + \alpha(1-\alpha)\frac{L}{2}\|x - y\|^2. \qquad \text{Nest. 6}$$

The first proof of this theorem is attributed to Yurii Nesterov and can be found in his classic 2004 text, *Introductory Lectures on Convex Optimization* [33, Thm. 2.1.5]. We defer a (better) hand proof of this theorem to Appendix A.

3.6 ACL2(r)

ACL2 (A Computational Logic for Applicative Common Lisp) is a strict subset of Common Lisp and an automated theorem prover in a total, first-order logic commonly used to model and verify computer systems and mathematics. By total, we mean all functions map all objects in the logic. By first-order, we mean quantifiers can only predicate over individuals (in contrast to sets and sets of sets, etc. in higher-order logic). In particular, functions cannot be naturally quantified over. The return on this restriction is that first-order theorem proving is highly developed, semi-decidable, and allows for truly automated reasoning. ACL2, in particular, is primarily based on *term-rewriting*, which, simply put, is a set of rules for replacing one logical expression with an another equivalent expression. In addition to these facts, ACL2 is mostly quantifier-free which allows it to be a highly automated and efficient tool for reasoning about computer hardware and software systems appealing to commercial applications in the verification space. ACL2 is sometimes referred to as an *industrial-strength* theorem prover.

Some commonly used ACL2 functions in this thesis are:

- `defun` - defines a function symbol

- `defthm` - name and prove a theorem
- `defun-sk` - define a Skolem function, i.e. a function with an outermost quantifier
- `encapsulate` - hide some events and constrain some functions
- `car` - returns the head of a list
- `cdr` - returns a list without its head
- `len` - returns the length of a list.

Despite the first-order nature of ACL2, there is a mechanism, called *encapsulation*, for proving theorems about functions. An encapsulation hides certain events (which include theorems) and/or function definitions from the global logic; such events and definitions are called *local*. Local theorems can then be proven about local functions. Furthermore, there are *global theorems* whose proofs only rely on local theorems, and not on the function definitions themselves. Functional instantiation allows these global theorems to be deduced for other functions that satisfy the local theorems. Local functions that are originally used to satisfy the local theorems within the encapsulation are called *witnesses*. This provides a psuedo-higher-order ability for reasoning about functions in general. Details are given in the ACL2 documentation [2]. In this thesis, encapsulations are used to prove theorems about continuous functions and convex functions in Chapters 6 to 8.

3.7 Non-standard Analysis and ACL2(r)

Classical real analysis is well known for its epsilon-delta approach to mathematical theory-building. For example, we say that function $f : \mathbb{R} \to \mathbb{R}$ is continuous at $x \in \mathbb{R}$ iff

$$\forall \epsilon > 0,\ \exists \delta > 0 :\ \forall y \in \mathbb{R},\ |y - x| < \delta \implies |f(y) - f(x)| < \epsilon. \qquad 3.26$$

This classical approach makes extensive use of nested quantifiers and support for quantifiers in ACL2 is limited. In fact, proofs involving terms with quantifiers often involve recursive witness functions that enumerate all possible values for the quantified term. Of course, we cannot enumerate all of the real numbers. Instead of using epsilon-delta style reasoning, ACL2(r) extends ACL2 by supporting real numbers via a formalisation of non-standard

analysis – a more algebraic yet isomorphic approach to the theory of real analysis. Since quantifiers are unwieldy, this alternative approach is more amenable to automated deduction.

Non-standard analysis introduces an extension of $\mathbb{R}$ called the *hyperreals* ${}^*\mathbb{R} \supset \mathbb{R}$ which include numbers larger in magnitude than any finite real and the reciprocals of such numbers. These large hyperreals are aptly named *infinite* and their reciprocals are named *infinitesimal*. If ω is an infinite hyperreal, then it follows that $|1/\omega| < x$ for any positive finite real x. Also, 0 is an infinitesimal.

Any finite hyperreal is the sum of a real number and an infinitesimal. The real part of a hyperreal can be obtained through the *standard-part* function $\text{st} : {}^*\mathbb{R} \to \mathbb{R}$.

To state a function $f : \mathbb{R}^n \to \mathbb{R}$ is continuous in the language of non-standard analysis amounts to: if $d(x, y)$ is an infinitesimal for a standard x, then so is $|f(x) - f(y)|$.

Some commonly used ACL2(r) functions in this thesis are:

- `realp` - the recognizer for a real number
- `real-listp` - the recognizer for lists of real numbers
- `i-small` - the recognizer for infinitesimals
- `i-large` - the recognizer for infinite hyperreals
- `i-limited` - the recognizer for reals that are not `i-large`
- `st` - returns the standard-part of a real.

Chapter 4

Mechanical First-order Real Vector Spaces

$\mathbb{R}^n$ is ubiquitous across much of mathematics. Indeed, as a geometric structure, we view $\mathbb{R}^n$ as a place in which to perform analysis; as an algebraic object, $\mathbb{R}^n$ is a direct sum of subspaces. Under different lenses we view $\mathbb{R}^n$ as different structures and it thus inherits the properties of the related structures. It is a metric space, a Hilbert space, a topological space, a group, etc.

In this chapter, we present $\mathbb{R}^n$ formalised from two perspectives. First, we consider $\mathbb{R}^n$ as a vector space. Then, by endowing $\mathbb{R}^n$ with the dot product, we obtain an inner product space. Inner product spaces are vector spaces equipped with an inner product which induces – among other notions – a rigorous definition of the angle between two vectors and the size of a vector. Notably, by playing with the definitions of the inner product, vector addition, scalar multiplication, etc. we reap dividends in the later chapters by way of short and elegant proofs. For this reason, and since this is the first appearance of ACL2 code, we discuss and showcase some selected definitions to exposit our approach to reasoning about and formalising $\mathbb{R}^n$.

Definition	ACL2 Name	Program
Vector Addition	`vec-+`	4.1
Scalar Multiplication	`scalar-*`	4.2
Vector Subtraction (I)	`vec--`	4.3
Vector Subtraction (II)	`vec--x`	
Dot Product	`dot`	4.5

Table 4.1: Key definitions in this chapter.

Theorem	Statement	ACL2 Name	Program
Vector Space Axioms	See Section 3.1		
Closure of Vector Addition	$u, v \in \mathbb{R}^n \implies u + v \in \mathbb{R}^n$	`vec-+-closure-1` `vec-+-closure-2`	4.1
Equivalence of `vec--` and `vec--x`	$u - v = u -_{\texttt{x}} v$	`vec---equivalence`	4.4
Anticommutativity of Vector Subtraction (`vec--x`)	$u -_{\texttt{x}} v = -(v -_{\texttt{x}} u)$	`vec---x-anticommutativity`	4.4
Anticommutativity of Vector Subtraction (`vec--`)	$u - v = -(v - u)$	`vec---anticommutativity`	4.4
Inner Product Space Axioms	See Section 3.1		
Commutativity of the Dot Product	$\langle u, v \rangle = \langle v, u \rangle$	`dot-commutativity`	4.6
Linearity of the Dot Product in the First Coordinate (I)	$\langle au, v \rangle = a\langle u, v \rangle$	`dot-linear-first-coordinate-1`	4.6
Linearity of the Dot Product in the Second Coordinate (I)	$\langle u, av \rangle = a\langle u, v \rangle$	`dot-linear-second-coordinate-1`	4.6

Table 4.2: Key theorems in this chapter.

4.1 Vector Space Axioms

Program 4.1 shows our definition of vector addition, `vec-+`. Unsurprisingly, vectors in $\mathbb{R}^n$ are represented by lists of n real numbers. For operations between such elements to be well defined, the two vectors must be of the same length, and closure forces the function to return a vector of the same dimension. With these considerations, `vec-+` is defined via a straightforward recursion. In ACL2, all functions, including `vec-+`, are defined for all arguments. Thus `vec-+` is defined for vectors of mismatched length or even values that are not vectors. For example, `(vec-+ ''hello world'' 42)` returns `nil`. The guard for a function describes the *intended* usage. When a function is defined in ACL2, checks can be made to ensure that terms in the function body satisfy their respective guards. This is a generalisation of type checking that allows many simple programming errors to be caught and corrected before attempting subtle proofs. Since all functions, including `vec-+`, are total, theorems about these functions, such as `vec-+-closure-1` (also in Program 4.1), must hold for all values of function parameters. Thus, it is common, as seen in `vec-+-closure-1`, to restate the guard as a hypothesis of the theorem. ACL2 also uses guard information to optimise the execution of functions in contexts where the guard has been proven to hold. Note, in Program 4.1, `define` is a wrapper for `defun` that simplifies many common aspects of function definition in ACL2, such as guards. The `more-returns` section of a `define` is a convenient way of introducing theorems about the function. Likewise, there are similar theorems positing associativity, commutativity, etc. that are readily proven in ACL2 without further user assistance when provided the appropriate induction scheme.

Scalar multiplication `scalar-*` is defined similarly in Program 4.2 with the usual theorems about closure, compatibility, etc. proven via induction.

Ideally, vector subtraction would be defined as a macro as in Program 4.3 involving `vec-+` and `scalar-*` as it is with subtraction for reals in ACL2. Happenstantially, however, proving theorems regarding a function equivalent to `vec--` (which we call `vec--x`) and then proving the theorems for `vec--` via the equivalence is more amenable to verification in ACL2 than immediately proving theorems about `vec--`. In fact, the definition for `vec--x` is similar to `vec-+` but with `-` appearing wherever `+` previously appeared. Hence, the theorems involving closure, identity, inverses, anticommutativity, etc. are almost immediate. An example of this can be seen in Program 4.4. Upon execution of the desired properties for `vec--`, the theorem positing the equivalence of `vec--` to `vec--x` is disabled so as to not pollute the space of rules which the rewriter uses.

Program 4.1 Vector addition.

```
(define vec-+ ((vec1 real-listp) (vec2 real-listp))
 :guard (and (real-listp vec1)
             (real-listp vec2)
             (= (len vec1) (len vec2)))
 :returns (vec real-listp)
  (b* (((unless (and (consp vec1) (consp vec2))) nil)
       ((cons hd1 tl1) vec1)
       ((cons hd2 tl2) vec2)
       ((unless (and (realp hd1) (realp hd2))) nil))
      (cons (+ hd1 hd2) (vec-+ tl1 tl2)))
 ///
 (more-returns
  (vec (real-listp vec) :name vec-+-closure-1)
  (vec (implies (and (real-listp vec1)
                     (real-listp vec2)
                     (= (len vec1) (len vec2)))
                (and (= (len vec) (len vec1))
                     (= (len vec) (len vec2))))
       :name vec-+-closure-2) ...))
```

Program 4.2 Scalar multiplication.

```
(define scalar-* ((a realp) (vec real-listp))
 :returns (retvec real-listp)
 (b* (((unless (consp vec)) nil)
      ((cons hd tl) vec)
      ((unless (and (realp a) (realp hd))) nil))
     (cons (* a hd) (scalar-* a tl)))
 ///
 (more-returns ...))
```

Program 4.3 Vector subtraction is equivalent to `(vec-+ vec1 (scalar-* -1 vec2))`.

```
(defmacro vec-- (vec1 vec2)
 (list 'vec-+ vec1 (list 'scalar-* -1 vec2)))
```

Program 4.4 Using the equivalence between `vector--` and `vec---x`.

```
(defthm vec---equivalence
 (implies (and (real-listp vec1)
               (real-listp vec2)
               (= (len vec1) (len vec2)))
          (equal (vec-- vec1 vec2) (vec--x vec1 vec2)))
 :hints (("GOAL" :in-theory (enable vec--x vec-+ scalar-*)
                 :induct (and (nth i vec1) (nth i vec2)))))
...
(defthm vec--x-anticommutativity
        (= (vec--x vec1 vec2) (scalar-* -1 (vec--x vec2 vec1)))
 :hints (("GOAL" :in-theory (enable vec--x scalar-*))))

(defthm vec---anticommutativity
 (implies (and (real-listp vec1) (real-listp vec2)
               (= (len vec2) (len vec1)))
          (= (vec-- vec1 vec2) (scalar-* -1 (vec-- vec2 vec1))))
 :hints (("GOAL" :use ((:instance vec---equivalence)
                       (:instance vec---equivalence
                                  (vec1 vec2) (vec2 vec1))
                       (:instance vec--x-anticommutativity)))))
```

4.2 Inner Product Space Axioms

The definition of the dot product (Program 4.5) is rather similar to `vec-+` in that it is a simple recursive definition. The particular differences amount to returning a real instead of a vector. Proofs of return values, commutativity, etc. pass with a simple unwinding of the definition.

Program 4.5 The dot product.

```
(define dot ((vec1 real-listp) (vec2 real-listp))
 :guard (and (real-listp vec1) (real-listp vec2)
             (= (len vec2) (len vec1)))
 :returns (r realp)
 (cond ((or (not (real-listp vec1)) (not (real-listp vec2))) 0)
       ((not (= (len vec2) (len vec1))) 0)
       ((or (null vec1) (null vec2)) 0)
       (t (+ (* (car vec1) (car vec2))
             (dot (cdr vec1) (cdr vec2))))))
```

Aside from the usual suspects, one troublesome property is bilinearity of the dot product. While linearity of the first coordinate passes inductively without any hints, linearity for the second coordinate does not execute so easily. Providing an explicit induction scheme via a hint would likely produce the desired proof; however, it was simpler to apply commutativity and use linearity of the first coordinate to exhibit the same result, e.g. given

$$\langle au, v\rangle = a\langle u, v\rangle, \tag{4.1}$$

$$\langle u, v\rangle = \langle v, u\rangle, \tag{4.2}$$

we have

$$\langle u, av\rangle = \langle av, u\rangle = a\langle v, u\rangle = a\langle u, v\rangle. \tag{4.3}$$

Program 4.6 shows the ACL2 version of this proof.

Our reliance on proving theorems about algebraic structures via their algebraic properties instead of via induction is well exemplified here. This is especially important for the following formalisation of metric spaces. Because non-classical recursive function are not permitted in ACL2(r), suppressing the definition of recursive functions on vectors, say `dot`, within a `define` facilitates the reasoning of infinitesimals in the space of $\mathbb{R}^n$. In particular, since $\langle -, -\rangle : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function, we may connect the notion of infinitesimal values of $\langle -, -\rangle$ with the entries of the vectors

Program 4.6 An example of using commutativity to prove bilinearity of the dot product.

```
(defthm dot-commutativity
 (implies (and (real-listp vec1) (real-listp vec2)
               (= (len vec2) (len vec1)))
          (= (dot vec1 vec2) (dot vec2 vec1)))
 :hints (("GOAL" :in-theory (enable dot))))

(defthm dot-linear-first-coordinate-1
 (implies (and (real-listp vec1) (real-listp vec2)
               (= (len vec2) (len vec1)) (realp a))
          (= (dot (scalar-* a vec1) vec2)
             (* a (dot vec1 vec2))))
 :hints (("GOAL" :in-theory (enable dot scalar-*))))
...
(defthm dot-linear-second-coordinate-1
 (implies (and (real-listp vec1) (real-listp vec2)
               (= (len vec2) (len vec1)) (realp a))
          (= (dot vec1 (scalar-* a vec2))
             (* a (dot vec1 vec2))))
 :hints (("GOAL" :do-not-induct t
                 :use ((:instance scalar-*-closure (vec vec2))
                       (:instance dot-commutativity
                                  (vec2 (scalar-* a vec2)))))))
```

on which $\langle -, - \rangle$ is evaluated without unravelling the recursive definition of `dot`.

However, as can be seen in Chapter 5, algebraic proofs of theorems involving functions on vectors can require significant guidance by the user in what would have otherwise been a simple application of associativity, commutativity, identity, etc. On the other hand, applying combinations of algebraic rules in search of a satisfying expression can be viewed as a constraint satisfaction problem. We leave this as future work and discuss it further in Chapter 9.

Chapter 5

Formalising Cauchy-Schwarz

The Cauchy-Schwarz inequality is considered to be one of the most important inequalities in mathematics. It plays a role in areas as diverse as functional analysis, real analysis, probability theory, linear algebra and combinatorics to name a few. Cauchy-Schwarz even made an appearance on an online list of "The Hundred Greatest Theorems" [17] and the subsequent formalised version of the list "Formalizing 100 Theorems" [48]. In this chapter, we present the first formal proof of the Cauchy-Schwarz inequality in a theorem prover for first-order logic including both forms (squared and norm versions) and the conditions for equality. Such a formalisation suggests ACL2 can be used in the various areas of mathematics in which the inequality appears. In fact, Cauchy-Schwarz is used in Chapter 6 to prove the triangle inequality for ACL2(r) real metric spaces and in the proof of Nesterov's Theorem (Theorem 3) in Chapter 8.

We begin by outlining some of the key lemmas in ACL2(r) that result in the Cauchy-Schwarz inequality. Much of the proof is user guided via the algebraic properties of norms, the dot product, etc. that were proven in Chapter 4. Since this is the first significant formal proof of the thesis, surplus details are provided.

Definition	ACL2 Name	Program
Zero Vector Recognizer	`zvecp`	
Euclidean Norm	`eu-norm`	
Euclidean Norm Squared	`norm^2`	
Square Roots	`acl2-sqrt`	ACL2(r) Books
$\exists a : u = av$	`linear-dependence-nz`	5.6

Table 5.1: Key definitions in this chapter.

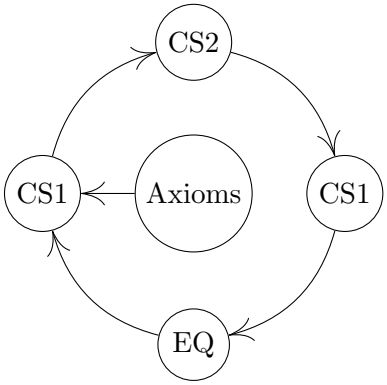

Figure 5.1: Structure of the proof for Cauchy-Schwarz. CS1, CS2, and EQ denotes Cauchy-Schwarz I, Cauchy-Schwarz II, and the conditions for equality, respectively.

Theorem	Statement	ACL2 Name	Program
Cauchy-Schwarz I	$\lvert\langle u, v\rangle\rvert^2 \leq \langle u, u\rangle \cdot \langle v, v\rangle$	`cauchy-schwarz-1`	5.3
Relationship between Norms and Inner Product (I)	$\langle u, u\rangle = \lVert u\rVert^2$	`norm-inner-product-equivalence`	
Relationship between Norms and Inner Product (II)	$\sqrt{\langle u, u\rangle} = \lVert u\rVert$	`lemma-16`	5.4
Cauchy-Schwarz II	$\lvert\langle u, v\rangle\rvert \leq \lVert u\rVert \cdot \lVert v\rVert$	`cauchy-schwarz-2`	5.4
Conditions for Equality (Cauchy-Schwarz I)	$\lvert\langle u, v\rangle\rvert^2 = \langle u, u\rangle \cdot \langle v, v\rangle$ $\Updownarrow$ $(u = 0) \vee (v = 0) \vee (\exists a : u = av)$	`cauchy-schwarz-3`	5.7
Conditions for Equality (Cauchy-Schwarz II)	$\lvert\langle u, v\rangle\rvert = \lVert u\rVert \cdot \lVert v\rVert$ $\Updownarrow$ $(u = 0) \vee (v = 0) \vee (\exists a : u = av)$	`cauchy-schwarz-4`	5.8

Table 5.2: Key theorems in this chapter.

5.1 Cauchy-Schwarz I

Recall that Cauchy-Schwarz I states

$$|\langle u, v\rangle|^2 \leq \langle u, u\rangle \cdot \langle v, v\rangle. \qquad \text{Cauchy-Schwarz I}$$

Note the case where $v = 0$ is trivial. Suppose $v \neq 0$. To see Cauchy-Schwarz I, first we prove

$$\|u - v\|^2 = \langle u, u\rangle - 2\langle u, v\rangle + \langle v, v\rangle \tag{5.1}$$

by applying multiple instances of the bilinearity of $\langle -, -\rangle$. This can be seen in Program 5.1. Since $\|u - v\| \geq 0$, replacing v with $\frac{\langle u,v\rangle}{\langle v,v\rangle}v$ in Equation 5.1 above produces

$$0 \leq \left\| u - \frac{\langle u, v\rangle}{\langle v, v\rangle}v \right\|^2 = \langle u, u\rangle - 2\left\langle u, \frac{\langle u, v\rangle}{\langle v, v\rangle}v \right\rangle + \left\langle \frac{\langle u, v\rangle}{\langle v, v\rangle}v, \frac{\langle u, v\rangle}{\langle v, v\rangle}v \right\rangle \tag{5.2}$$

which can be seen in Program 5.2. The inequality reduces to

$$0 \leq \langle u, u\rangle - 2\frac{\langle u, v\rangle}{\langle v, v\rangle}\langle u, v\rangle + \frac{\langle u, v\rangle^2}{\langle v, v\rangle^2}\langle v, v\rangle \tag{5.3}$$

Factoring out $\langle u, v\rangle$ and rearranging via the algebraic rules produces Cauchy-Schwarz I. This first version of Cauchy-Schwarz can be seen in Program 5.3.

Despite claiming $v \neq 0$ throughout the proof and, indeed, the case-split in Program 5.3 corroborating this fact, note that the preceding lemmas in Programs 5.1 and 5.2 do not need such a hypothesis. This is particularly concerning for `lemma-7` in Program 5.2 since we factor out $\langle u, v\rangle/\langle v, v\rangle$ which would reduce to $0/0$ if $v = 0$. However, in ACL2, all functions are total and expressions representing, say, $x/0$ become 0 so the algebra remains valid.

5.2 Cauchy-Schwarz II

To see Cauchy-Schwarz II (i.e. $|\langle u, v\rangle| \leq \|u\|\|v\|$) from Cauchy-Schwarz I, we simply take square roots (defined as `acl2-sqrt` in ACL2) and show the equivalence between the dot products and the square of the norms. In particular, we use the fact that $\sqrt{\langle u, u\rangle} = \|u\|$ which follows simply from the definitions. Which form one desires is a matter of preference but for ACL2 to automatically recognize their relationship in the final statement of Cauchy-Schwarz II (`cauchy-schwarz-2`), the fact must be stated explicitly in a preceding lemma (`lemma-16`). This part can be seen in Program 5.4.

To see the other direction, we simply square both sides and rearrange to obtain the desired form.

Program 5.1 Applying bilinearity of the dot product to prove a simple identity.

```
;; < u - v , u - v > = < u , u > - < u , v > - < v , u > + < v , v >
(defthm lemma-3
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (norm^2 (vec-- u v))
                 (+ (dot u u)
                    (- (dot u v))
                    (- (dot v u))
                    (dot v v))))
 :hints (("GOAL" :use ((:instance scalar-*-closure (a -1) (vec v))
                       (:instance dot-linear-second-coordinate-2
                                  (vec1 v)
                                  (vec2 u)
                                  (vec3 (scalar-* -1 v)))
                       (:instance dot-linear-second-coordinate-2
                                  (vec1 u)
                                  (vec2 u)
                                  (vec3 (scalar-* -1 v)))))))

;; < u - v, u - v > = < u, u > - 2 < u , v > + < v, v >
(defthm lemma-4
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (norm^2 (vec-- u v))
                 (+ (dot u u) (- (* 2 (dot u v))) (dot v v))))
 :hints (("GOAL" :use ((:instance dot-commutativity
                                  (vec1 u) (vec2 v))))))
```

Program 5.2 Substituting v for $\langle v, v\rangle^{-1}\langle u, v\rangle v$.

```
;; 0 <= < u, u > - 2 < u , v > + < v, v >
(local (defthm lemma-6
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (<= 0 (norm^2 (vec-- u v)))
                 (<= 0
                     (+ (dot u u)
                        (- (* 2 (dot u v)))
                        (dot v v)))))))

;; let v = (scalar-* (* (/ (dot v v)) (dot u v)) v)
(local (defthm lemma-7
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (<= 0
                     (norm^2 (vec-- u
                                    (scalar-* (* (/ (dot v v))
                                                 (dot u v))
                                              v))))
                 (<= 0
                     (+ (dot u u)
                        (- (* 2 (dot u
                                     (scalar-* (* (/ (dot v v))
                                                  (dot u v))
                                               v))))
                        (dot (scalar-* (* (/ (dot v v))
                                          (dot u v))
                                       v)
                             (scalar-* (* (/ (dot v v))
                                          (dot u v))
                                       v))))))
 :hints (("GOAL" :use (...
                       (:instance lemma-6
                                  (v (scalar-* (* (/ (dot v v))
                                                  (dot u v)) v)))))))))
```

Program 5.3 Final form of Cauchy-Schwarz I.

```
(defthm cauchy-schwarz-1
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
                 (<= (* (dot u v) (dot u v))
                     (* (dot u u) (dot v v))))
 :hints (("GOAL" ... :cases ((zvecp v)(not (zvecp v))))))
```

Program 5.4 Showing Cauchy-Schwarz II.

```
(local (defthm lemma-16
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (and (equal (acl2-sqrt (* (dot u v) (dot u v)))
                      (abs (dot u v)))
               (equal (acl2-sqrt (dot u u)) (eu-norm u))
               (equal (acl2-sqrt (dot v v)) (eu-norm v))
               (equal (acl2-sqrt (* (dot u u) (dot v v)))
                      (* (eu-norm u) (eu-norm v)))))
 :hints (("GOAL" :use ((:instance norm-inner-product-equivalence
                                  (vec u))
                       (:instance norm-inner-product-equivalence
                                  (vec v)))))))

(defthm cauchy-schwarz-2
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (<= (abs (dot u v))
              (* (eu-norm u) (eu-norm v))))
 :hints (("GOAL" :use ((:instance cauchy-schwarz-1)
                       (:instance norm-inner-product-equivalence
                                  (vec v))
                       (:instance norm-inner-product-equivalence
                                  (vec u)) ...) ...)))
```

5.3 Conditions for Equality

Suppose $u, v \neq 0$ and

$$\langle u, v\rangle^2 = \langle u, u\rangle\langle v, v\rangle. \tag{5.4}$$

Then we simply reverse all the equalities used to prove Cauchy-Schwarz I from the axioms (see Equation 5.2) until we return to

$$0 = \left\| u - \frac{\langle u, v\rangle}{\langle v, v\rangle} v \right\|^2. \tag{5.5}$$

Since $\|\cdot\|^2$ is positive definite, we must have

$$u = \frac{\langle u, v\rangle}{\langle v, v\rangle} v. \tag{5.6}$$

This can be seen in Program 5.5. The cases for $u = 0$ or $v = 0$ are immediate.

To express linearity, we introduce a Skolem function so that the final form of the conditions for equality are in the greatest generality. Skolem functions have bodies with an outermost quantifier and are equivalent to certain predicate formulas. For ACL2, this means that the logic can express statements with quantifiers while maintaining fast and automated decision procedures. The definition of the Skolem function is in Program 5.6.

We also attempted a proof where the value of the Skolem constant was explicitly computed – simply find the first non-zero element of v and divide the corresponding element of u by the v element. The proof for the Skolem function approach was much simpler because the witness value comes already endowed with the assertion that it has the properties we need for subsequent reasoning. The final result can be seen in Program 5.7.

The conditions for equality for Cauchy-Schwarz II follow similarly.

Program 5.5 Linearity follows from equality.

```
;; equality implies the two vectors are linearly dependent
(local (defthm lemma-19
 (implies (and (real-listp u) (real-listp v)
               (= (len u) (len v)) (not (zvecp v))
               (= (* (dot u v) (dot u v)) (* (dot u u) (dot v v))))
          (equal u (scalar-* (* (/ (dot v v)) (dot u v)) v))) ...))
```

Program 5.6 Introducing a Skolem function for linearity.

```
(defun-sk linear-dependence-nz (u v)
 (exists a (equal u (scalar-* a v))))
```

Program 5.7 The conditions for equality for Cauchy-Schwarz I.

```
;; conditions for CS1 equality
(defthm cauchy-schwarz-3
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (= (* (dot u v) (dot u v)) (* (dot u u) (dot v v)))
                 (or (zvecp u)
                     (zvecp v)
                     (linear-dependence-nz u v)))) ...)
```

5.4 Final Form of Cauchy-Schwarz

Program 5.8 displays the final form of the Cauchy-Schwarz. The theorems `cauchy-schwarz-1` and `cauchy-schwarz-2` are equivalent to Cauchy-Schwarz I and Cauchy-Schwarz II, respectively. Likewise, the theorems `cauchy-schwarz-3` and `cauchy-schwarz-4` are the conditions for equality for Cauchy-Schwarz I and Cauchy-Schwarz II, respectively.

Here we would like to note the choice of classical proof on which we base this formalisation. In particular, we would like to compare its flavour to other proofs of Cauchy-Schwarz. Indeed, there are geometric proofs, analytical proofs, combinatorial proofs, inductive proofs, etc. whereas we followed an algebraic approach. Considering ACL2(r)'s strengths with regards to induction, the choice may seem odd. There are several potential inductive candidates we considered before proceeding at the onset of this endevour. However, most of these candidates inducted over the dimension of $\mathbb{R}^n$ and required reasoning over the real entries of vectors. We suspect unwinding the vectors and guiding ACL2(r) through such a proof would be more onerous than the one outlined in this paper. Moreover, our formalisation of inner product spaces already provided the exact tools necessary for the chosen proof of Cauchy-Schwarz (i.e. vectors, vector-vector operations, scalar-vector operations, inner products, etc.) without resorting to reasoning over individual reals. The precision of this approach is arguably more elegant.

Finally, while this formalisation of the Cauchy-Schwarz inequality is not

Program 5.8 The final form of Cauchy-Schwarz.

```
(defthm cauchy-schwarz-1
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
                (<= (* (dot u v) (dot u v))
                    (* (dot u u) (dot v v)))) ...)

(defthm cauchy-schwarz-2
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (<= (abs (dot u v))
              (* (eu-norm u) (eu-norm v)))) ...)

(defthm cauchy-schwarz-3
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (= (* (dot u v) (dot u v))
                    (* (dot u u) (dot v v)))
                 (or (zvecp u)
                     (zvecp v)
                     (linear-dependence-nz u v)))) ...)

(defthm cauchy-schwarz-4
 (implies (and (real-listp u) (real-listp v) (= (len u) (len v)))
          (equal (=  (abs (dot u v)) (* (eu-norm u) (eu-norm v)))
                 (or (zvecp u)
                     (zvecp v)
                     (linear-dependence-nz u v)))) ...)
```

a particularly deep proof, it has historic significance by merit of its first-order context. There is no profound mathematical insight to be gleaned,[1] but being the first proof in a first-order logic holds some inherent value by itself. Furthermore, various applications may be introduced as a result of the inequality. The appearance of Cauchy-Schwarz in functional analysis, real analysis, probability theory, combinatorics, etc. speaks to its utility. In this thesis, we will use it in Chapter 6 to prove the triangle inequality for real metric spaces and in Chapter 8 to prove Theorem 3.

[1] Or, at least, none that we found obvious.

Chapter 6

Reasoning about Continuity

Continuity is a central idea in calculus and analysis in general, and hardly needs justification for its utility. Unfortunately, the dense and infinite nature of the reals introduces difficulty for machine reasoning. Non-standard analysis provides a more algebraic approach to analysis by favouring the use of infinitesimals in lieu of the classical epsilon-delta approach. One consequence of this is that quantifiers are relegated to a minor role in ACL2 - thus further facilitating the automated nature of the logic.

Formalising $\mathbb{R}^n$ for arbitrary n, however, introduces certain difficulties. The fundamental differences between the rational and irrational induces a subtle schism between ACL2 and ACL2(r) wherein the notions formalised in ACL2 (which we call *classical*) are far more well-behaved than those unique to ACL2(r) (which we call *non-classical*). The arbitrariness of n suggests the necessity of defining operations recursively – yet, as explained in Section 6.3, non-classical recursive functions are not permitted in ACL2(r).

First, we briefly outline our approach to formalising a theory of real metric spaces. Next, we describe our method for reasoning about continuity. Finally, we conclude with a short discussion about non-classical recursive functions.

6.1 Metric Space Axioms

Observe

$$d_2(x, y) = \|x - y\|_2 = \sqrt{\langle x - y, x - y \rangle}. \tag{6.1}$$

Definition	ACL2 Name	Program
Euclidean Metric	`eu-metric`	6.1
Euclidean Norm	`eu-norm`	6.2
Euclidean Metric Squared	`metric^2`	
Maximum Norm	`max-abs-reals`	6.3

Table 6.1: Key definitions in this chapter.

Theorem	Statement	ACL2 Name	Program
Metric Space Axioms	See Section 3.3		
Norm Infinitesimal implies Maximum Infinitesimal	$\|x\|$ infinitesimal $\Downarrow$ $\max_i \|x_i\|$ infinitesimal	`eu-norm-i-small-implies-max-abs-reals-i-small`	6.3
Norm Infinitesimal implies Entries Infinitesimal	$\|x\|$ infinitesimal $\Downarrow$ $\|x_i\|$ infinitesimal	`eu-norm-i-small-implies-elements-i-small`	6.3
Metric Infinitesimal implies Difference of Entries Infinitesimal	$\|x - y\|$ infinitesimal $\Downarrow$ $\|x_i - y_i\|$ infinitesimal	`eu-metric-i-small-implies-difference-of-entries-i-small`	6.5
$f(x) = x_1 + x_2 + x_3$ is Uniformly Continuous	$\|x - y\|$ infinitesimal $\Downarrow$ $\|f(x) - f(y)\|$ infinitesimal	`sum-is-continuous`	6.6
$g(x) = x_1 x_2$ is Continuous	For standard x, $\|x - y\|$ infinitesimal $\Downarrow$ $\|g(x) - g(y)\|$ infinitesimal	`prod-is-continuous`	6.7
Continuity / Uniform Continuity of Other Various Functions	See Section 6.2		

Table 6.2: Key theorems in this chapter.

Proving theorems regarding metrics reduces to proving theorems about the norm from which the metric is induced. Likewise, proving theorems involving norms can be reduced to proving properties of the inner product underlying the norm. The process of formalisation, then, should ideally define the metric via the norm, and the norm should be defined via the inner product. This is useful for proving properties such as positive-definiteness of both the metric and the norm, e.g. if

$$\langle u, u \rangle \geq 0 \tag{6.2}$$

with equality iff $u = 0$, then the same applies for

$$\|x\|_2 = \sqrt{\langle x, x \rangle} \geq 0 \tag{6.3}$$

and

$$d_2(x, y) = \|x - y\|_2 = \sqrt{\langle x - y, x - y \rangle} \geq 0. \tag{6.4}$$

However, as exemplified by `vec--` and `vec--x`, the obvious sequence is not necessarily the easiest. Indeed, not only is it simpler to prove theorems on functions equivalent to the desired functions, we also prove properties of similar functions not equivalent to the desired functions but such that if the properties hold for the similar functions, then they also hold for the desired functions.

For example, suppose we wish to define commutativity for the Euclidean metric `eu-metric`. The elegant form of `eu-metric` can be seen in Program 6.1 and the elegant form of the Euclidean norm `eu-norm` can be seen in Program 6.2.

Program 6.1 The Euclidean metric.

```
(defun eu-metric (u v)
 (eu-norm (vec-- u v)))
```

Program 6.2 The Euclidean norm.

```
(defun eu-norm (u)
 (acl2-sqrt (dot u u)))
```

Now recall `(vec-- x y)` is a macro for `(vec-+ x (scalar-* -1 y))` and, together with an instance of `acl2-sqrt`, passing the proofs for commutativity require a non-trivial amount of provided hints and user guidance.

Instead, define a recursive function `metric^2` which is the square of the norm of the difference of two vectors (which is equivalent to the square of the Euclidean metric, i.e. $\|x - y\|_2^2$). Moreover, proving those equivalences simply amounts to unwinding the definitions. Having established

$$\|x - y\|_2^2 = \|y - x\|_2^2, \tag{6.5}$$

it follows that

$$d_2(x, y) = \sqrt{\|x - y\|_2^2} = \sqrt{\|y - x\|_2^2} = d_2(y, x). \tag{6.6}$$

Hence, commutativity for `eu-metric` is proven.

The triangle inequality follows from the Cauchy-Schwarz inequality with a bit of algebraic coaxing.

6.2 Continuity $\mathbb{R}^n \to \mathbb{R}$

To showcase continuity, let us begin with an enlightening example. Recall the non-standard analysis definition of continuity for a function $f : \mathbb{R}^n \to \mathbb{R}$ stated in the language of non-standard analysis: if $d_2(x, y)$ is an infinitesimal for a standard x, then so is $f(x) - f(y)$. Take $f(x) = \sum_{i=1}^n x_i$. It is clear that f is continuous in our usual theory of classical real analysis. However, we must translate into the language of infinitesimals. By hypothesis,

$$d_2(x, y) = \|x - y\|_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \tag{6.7}$$

is an infinitesimal. We would like to show that

$$f(x) - f(y) = \sum_{i=1}^n x_i - \sum_{i=1}^n y_i = \sum_{i=1}^n (x_i - y_i) \tag{6.8}$$

is also an infinitesimal. Indeed, by Equation 6.7 we see that each $x_i - y_i$ must be an infinitesimal since otherwise $d_2(x, y)$ wouldn't be an infinitesimal. Because the RHS of Equation 6.8 is a finite sum of infinitesimals, so is $f(x) - f(y)$ as desired.

Often, the line of reasoning for proving such a multivariate function to be continuous can be visualised as follows:

$d_2(x, y)$ infinitesimal $\longrightarrow$ $|x_i - y_i|$ infinitesimal $\longrightarrow$ $|f(x) - f(y)|$ infinitesimal.

Showing the second arrow depends on the definition of the function (and is, of course, false for discontinuous functions). This leaves the first arrow and two motivating questions:

1. How do we make ACL2(r) recognize $x_i - y_i$ are infinitesimals from $d_2(x, y)$ being infinitesimal?
2. How do we state "all $x_i - y_i$ are infinitesimals"?

To answer the first question, observe for any vector z and $i \leq n$,

$$\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} \geq \max_i |z_i| \geq |z_i|. \tag{6.9}$$

Setting $z = x - y$, we see that if the norm is an infinitesimal, then so must each entry of the vector. By introducing an ACL2(r) function, say `max-abs-reals`, equivalent to $\max_i$ and reasoning over the arbitrariness of i instead of over the length of x and y, we may exhibit the infinitesimalness of any entry in $x - y$ as seen in Program 6.3. Note that the converse of `eu-norm-i-small-implies-elements-i-small` in Program 6.3, e.g. a theorem of the form

```
(implies (i-small (nth i vec)) (i-small (eu-norm vec)))
```

is not particularly useful since this reverses the line of reasoning above for showing multivariate functions are continuous. Moreover, since `eu-norm` is defined recursively on the entries of a vector which, in this case, are all infinitesimal, proving such a theorem would encounter the issues outlined in Section 6.3.

To address the second question, one could imagine a recognizer for vectors with infinitesimal entries – such a recognizer is depicted in Program 6.4. However, this recognizer would be recursive on the entries of the vector with each recursive step invoking the non-classical recognizer `i-small` for infinitesimal reals. Because non-classical recursive functions are forbidden, so is the suggested recognizer. We considered using a Skolem function, but to remain consistent with `eu-norm-i-small-implies-elements-i-small` in Program 6.3, we opted for a theorem positing the condition for an arbitrary index i as seen in Program 6.5 instead.

To see `eu-metric-i-small-implies-difference-of-entries-i-small` in action, consider once again the example of $f(x) = \sum_{i=1}^{n} x_i$. If $n = 3$ and `sum` is the ACL2(r) function equivalent to f, then Program 6.6 proves continuity for `sum` – in fact, it is *uniformly continuous* since we do not hypothesise x is standard.

Other examples of functions with proofs of continuity in ACL2(r) include the Euclidean norm (and its square), the dot product with one coordinate fixed, and the function $g(x, y) = xy$.

Program 6.3 Showing arbitrary entries of a vector are infinitesimal via the maximum element.

```
(define max-abs-reals ((vec real-listp))
 ...
 (b* (((unless (consp vec)) 0)
       ((cons hd tl) vec)
       ((unless (realp hd)) 0))
      (max (abs hd) (max-abs-reals tl)))...)

(defthm eu-norm-i-small-implies-max-abs-reals-i-small
 (implies (and (real-listp vec) (i-small (eu-norm vec)))
          (i-small (max-abs-reals vec))))

(defthm eu-norm-i-small-implies-elements-i-small
 (implies (and (real-listp vec) (i-small (eu-norm vec))
               (natp i) (< i (len vec)))
          (i-small (nth i vec))))
```

Program 6.4 A fantastical recognizer for vectors with infinitesimal entries that doesn't exist.

```
(defun i-small-vecp (x)
 (cond ((null x) t)
       ((not (real-listp x)) nil)
       (t (and (i-small (car x)) (i-small-vecp (cdr x))))))
```

Program 6.5 A theorem positing the infinitesimalness of arbitrary entries in $x - y$.

```
(defthm eu-metric-i-small-implies-difference-of-entries-i-small
 (implies (and (real-listp x) (real-listp y) (= (len y) (len x))
               (i-small (eu-metric x y)) (natp i) (< i (len x)))
          (i-small (- (nth i x) (nth i y)))) ... )
```

Program 6.6 A theorem positing `sum` is continuous.

```
(defthm sum-is-continuous
 (implies (and (real-listp x) (real-listp y)
               (= (len x) 3) (= (len y) (len x))
               (i-small (eu-metric x y)))
          (i-small (- (sum x) (sum y))))...)
```

Program 6.7 The function $g(x, y) = xy$ is continuous.

```
(defun prod (x)
 (* (nth 0 x) (nth 1 x)))
...
(defthm prod-is-continuous
 (implies (and (real-listp x) (real-listp y)
               (= (len x) 2) (= (len y) (len x))
               (i-limited (eu-norm x)) (i-limited (eu-norm y))
               (i-small (eu-metric x y)))
          (i-small (- (prod x) (prod y))))...)
```

6.3 Why are Non-classical Recursive Functions Prohibited?

This chapter is significant because it provides a method for reasoning about continuity in a logic where non-classical recursive functions are prohibited. So why are they prohibited? Simply, the introduction of such functions will also introduce inconsistency into the logic of ACL2(r). In particular, it is possible to define a function that violates the rules of non-standard analysis. The following example comes from Ruben Gamboa in a personal communication on March 1, 2018 [10].

Program 6.8 An impossible function in ACL2(r).

```
(defun f (n)
 (cond ((zp n) 0)
       ((standardp n) n)
       (t (f (-1 n)))))
```

For example, consider the hypothetical function defined in Program 6.8.

Suppose `f` terminates. If `n` is standard, then `f` returns it without issue. If `n` is infinite, then `f` returns the largest standard number. However, this is impossible since such a number does not exist and, if `n` is infinite, so is `(-1 n)` which means `f` should not have terminated anyways. Moreover, this applies if `n` is any non-standard hyperreal since if $n = \text{st}(n) + \epsilon$ is equivalent to `n` and $\epsilon > 0$ is an infinitesimal, then $n - 1 = \text{st}(n-1) + \epsilon$ is not standard either.

The apparent issue with `f` is that its measure is non-standard. If the measure of a function can be proven to be standard, then a recursive non-classical function could be conceded. However, in practice, such a proof would likely be involved. Moreover, there are many non-obvious results in analysis and proving the measure of a non-classical recursive function to be standard does not preclude the existence of other even more subtle logical issues. A more detailed exploration of these issues is beyond the scope of this thesis but the interested reader can find a collection of counterexamples in analysis in [6].

Chapter 7

Mechanical Multivariate Convex Functions

Convex optimisation is a branch of applied mathematics that finds widespread use in financial modelling, operations research, machine learning, and many other fields. Algorithms for convex optimisation often have many parameters that can be tuned to improve performance. However, a choice of parameter values that produces good performance on a set of test cases may suffer from poor convergence or non-convergence in other cases. Hand written proofs for convergence properties often include simplifying assumptions to make the reasoning tractable. This motivates using machine generated and/or verified proofs for the convergence and performance of these algorithms. Once an initial proof has been completed, the hope is that simplifying assumptions can be incrementally relaxed or removed to justify progressively more aggressive implementations. These observations motivate our exploration of convex functions within ACL2(r).

In this chapter, we present example proofs of convexity for some simple functions and some basic theorems of convex optimisation. Moreover, we use these results to prove a major theorem in the next chapter.

7.1 Example Functions

In this section, we provide some selected examples of formalised theorems involving convex functions. The formalised proofs follow almost directly those of the informal proofs. For the sake of exposition, the proof for the first theorem is outlined but the rest are omitted for the sake of brevity.

Definition	ACL2 Name	Program
Square Function	`square-fn`	
Convex Function (Witness)	`cvfn-1`	7.2

Table 7.1: Key definitions in this chapter.

Theorem	Statement	ACL2 Name	Program
$f(x) = x^2$ is Convex	$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$	`square-fn-is-convex`	7.1
$\|\cdot\|$ is Convex.	$\|\alpha x + (1-\alpha)y\| \leq \alpha\|x\| + (1-\alpha)\|y\|$		
$\|\cdot\|^2$ is Convex	$\|\alpha x + (1-\alpha)y\|^2 \leq \alpha\|x\|^2 + (1-\alpha)\|y\|^2$		
Closure of Convexity under (Non-negative) Scalar Multiplication	f convex, $a \geq 0 \implies a \cdot f$ convex	`a-*-cvfn-1-convex`	7.3
Closure of Convexity under Addition	f, g convex $\implies f + g$ convex		
Closure of Convexity under Composition	$f : \mathbb{R}^n \to \mathbb{R}, g : \mathbb{R} \to \mathbb{R}$ convex with g non-decreasing $\Downarrow$ $g \circ f$ convex		
Pseudo triangle inequality	$\alpha(1-\alpha)\|x-y\|^2 \leq \alpha\|x\|^2 + (1-\alpha)\|y\|^2$	`pseudo-triangle-inequality`	7.5

Table 7.2: Key theorems in this chapter.

The first is a simple theorem positing the convexity of the square function. Recall from Equation 3.23 that a function f is convex if for $\alpha \in [0, 1]$,

$$\alpha f(x) + (1 - \alpha)f(x) \geq f(\alpha x + (1 - \alpha)y) \tag{3.23}$$

Then, the function $f(x) = x^2$ is convex because

$$\begin{aligned} \alpha x^2 + (1 - \alpha)y^2 - (\alpha x + (1 - \alpha)y)^2 &= \alpha(1 - \alpha)(x^2 - 2xy + y^2) \\ &= \alpha(1 - \alpha)(x - y)^2 \\ &\geq 0. \end{aligned} \tag{7.1}$$

We first define a square function: `(defun square-fn (x) (* (realfix x) (realfix x)))`. The chain of equalities in Equation 7.1 is immediately recognized by ACL2. The inequality in Equation 7.1 also passes without issue. The convexity of `square-fn` then follows from a simple application of the two lemmas. This entire proof is in Program 7.1.

Other functions for which convexity is exhibited include the Euclidean norm $\| \cdot \|_2$ and its square $\| \cdot \|_2^2$.

7.2 Reasoning about Convexity

Also formalised is a proof of each of the statements in Theorem 2. Here we outline the proof of the convexity of $a \cdot f$ given that $a \geq 0$ and f is convex. The rest are similar. Moreover, the approach we take resembles our approach to formalising Theorem 3 albeit much simpler. In particular, to reason about functions, we use the technique of encapsulation to first prove the desired theorem for a witness function. Functional instantiation then provides a method for reasoning about functions in general. Our witness, `cvfn-1`, is a constant function and so is clearly convex. This can be seen in Program 7.2.

Explicitly, $a \cdot f$ is convex because

$$af(\alpha x + (1-\alpha)y) \leq a(\alpha f(x) + (1-\alpha)f(y)) = \alpha(af(x)) + (1-\alpha)\,(af(y))\,. \tag{7.2}$$

In particular, we invoke convexity and need to distribute a. Moreover, this line of reasoning is not dependent on the definition of f so we may disable the definition of `cvfn-1` in Program 7.2. This can be seen in Program 7.3. We omit the formal proofs for the other claims of Theorem 2.

The encapsulation in Programs 7.2 and 7.3 deserves further elucidation. In ACL2, encapsulation is used to hide events from the "outside" logical world and (sometimes) introduce constrained functions. Local theorems

Program 7.1 $f(x) = x^2$ is convex.

```
(encapsulate
 nil

;; ax^2 + (1-a)y^2 - (ax + (1-a)y)^2 = a(1-a)(x-y)^2
(local (defthm lemma-1
 (implies (and (realp x) (realp y) (realp a) (<= 0 a) (<= a 1))
          (equal (- (+ (* a (square-fn x))
                       (* (- 1 a) (square-fn y)))
                    (square-fn (+ (* a x) (* (- 1 a) y))))
                 (* a (- 1 a) (square-fn (- x y)))))))

(defthm square-fn-positivity (<= 0 (square-fn x)))

;; replace a with a(1-a) and x with x-y to obtain the desired
;; inequality
(local (defthm lemma-2
 (implies (and (realp a) (<= 0 a))
          (<= 0 (* a (square-fn x))))))

(defthm square-fn-is-convex
 (implies (and (realp x) (realp y) (realp a) (<= 0 a) (<= a 1))
          (<= (square-fn (+ (* a x) (* (- 1 a) y)))
              (+ (* a (square-fn x)) (* (- 1 a) (square-fn y)))))
 :hints (("GOAL" :use ((:instance lemma-2 (a (* a (- 1 a)))
                                          (x (- x y))))))))
```

Program 7.2 Encapsulating a constant function `cvnf-1` and stating its convexity. This encapsulation also originally contained proofs for the other parts of Theorem 2 which are ommited here.

```
(encapsulate
 (((cvfn-1 *) => *)...)

 (local (defun cvfn-1 (x) (declare (ignore x)) 1337))
 ...
 (defthm cvfn-1-convex
  (implies (and (real-listp x) (real-listp y) (= (len y) (len x))
                (realp a) (<= 0 a) (<= a 1))
           (<= (cvfn-1 (vec-+ (scalar-* a x) (scalar-* (- 1 a) y)))
               (+ (* a (cvfn-1 x)) (* (- 1 a) (cvfn-1 y)))))) ...)
```

(e.g. `lemma-1` in Program 7.3) are erased or hidden whereas non-local theorems (e.g. `a-*-cvfn-1-convex`) are exported to the logical world. Similarly, the local definition of a constrained function introduced by an encapsulation (e.g. `cvfn-1` in Program 7.2) is erased. All that is known about the constrained function is exported non-local theorems. Following the encapsulation, one can prove other theorems about the constrained functions but the proofs can only depend on the theorems exported from the encapsulation. Note that a constrained function is non-executable – ACL2 can reason about it using the rules created from the proven theorems but the function itself cannot be evaluated. The upshot is that the user can define some other function and prove theorems about it equivalent to the theorems exported from the encapsulation where the constrained function was defined via functional instantiation.

7.3 A Useful Lemma

Nesterov's original proof of Theorem 3 in *Introductory Lectures in Convex Optimization* uses a particular inequality twice. However, its proof is not given and neither could we find a proof in the existing literature. For completeness, we include a machine (and handwritten) proof here and, for lack of a better name, we call it the *pseudo triangle inequality*.

Lemma 4. Let $\alpha \in [0,1]$ and $x, y \in \mathbb{R}^n$. Then

$$\alpha(1-\alpha)\|x-y\|^2 \leq \alpha\|x\|^2 + (1-\alpha)\|y\|^2. \qquad \text{Pseudo triangle inequality}$$

Program 7.3 Supressing the definition of `cvfn-1`, stating a special case of distributivity, and stating the convexity of $a \cdot f$.

```
(local (in-theory (disable (:d cvfn-1) (:e cvfn-1) (:t cvfn-1))))

(encapsulate
 nil

  ;;  factor out alpha
  (local (defthm lemma-1
   (implies (and (real-listp x) (real-listp y) (= (len y) (len x))
                 (realp a) (<= 0 a) (<= a 1)
                 (realp alpha) (<= 0 alpha))
            (= (+ (* a (* alpha (cvfn-1 x)))
                  (* (- 1 a) (* alpha (cvfn-1 y))))
               (* alpha
                  (+ (* a (cvfn-1 x))
                     (* (- 1 a) (cvfn-1 y))))))))

  (defthm a-*-cvfn-1-convex
   (implies (and (real-listp x) (real-listp y) (= (len y) (len x))
                 (realp a) (<= 0 a) (<= a 1)
                 (realp alpha) (<= 0 alpha))
            (<= (* alpha (cvfn-1 (vec-+ (scalar-* a x)
                                        (scalar-* (- 1 a) y))))
                (+ (* a (* alpha (cvfn-1 x)))
                   (* (- 1 a) (* alpha (cvfn-1 y))))))
   :hints (("GOAL" :in-theory (disable distributivity)
                   :use ((:instance cvfn-1-convex)
                         (:instance lemma-1))))))
```

Proof. Observe,

$$\begin{aligned}
0 &\leq (\alpha\|x\| - (1-\alpha)\|y\|)^2 \\
&= \alpha^2\|x\|^2 + (1-\alpha)^2\|y\|^2 - 2\alpha(1-\alpha)\|x\|\|y\| \\
&= (\alpha - \alpha(1-\alpha))\|x\|^2 + ((1-\alpha) - \alpha(1-\alpha))\|y\|^2 - 2\alpha(1-\alpha)\|x\|\|y\| \\
&= \alpha\|x\|^2 + (1-\alpha)\|y\|^2 - \alpha(1-\alpha)\left(\|x\|^2 + \|y\|^2 - 2\|x\|\|y\|\right) \\
&= \alpha\|x\|^2 + (1-\alpha)\|y\|^2 - \alpha(1-\alpha)\left(\|x\| + \|y\|\right)^2
\end{aligned} \tag{7.3}$$

so

$$\alpha(1-\alpha)\|x-y\|^2 \leq \alpha(1-\alpha)\left(\|x\| + \|y\|\right)^2 \leq \alpha\|x\|^2 + (1-\alpha)\|y\|^2 \tag{7.4}$$

where the first inequality follows from the (actual) triangle inequality. □

The machine proof follows exactly the handwritten proof. Notably, the string of equalities in Equation 7.3 immediately passes in ACL2(r) which is a testament to ACL2's affinity for truly automated deduction. We see this in Program 7.4.

Program 7.4 The string of equalities in Equation 7.3. We temporarily replace $\|x\|, \|y\|$ with reals $x,\ y$, respectively.

```
(local (defthm lemma-4
 (implies (and (realp a) (realp x) (realp y))
          (equal (* (- (* a x) (* (- 1 a) y))
                    (- (* a x) (* (- 1 a) y)))
                 (+ (* a (* x x))
                    (* (- 1 a) (* y y))
                    (- (* a (- 1 a) (* (+ x y) (+ x y)))))))))
```

The rest of the proof follows simply from the positivity of squares, the triangle inequality, and replacing $x,\ y$ with $\|x\|,\ \|y\|$, respectively. The final theorem is in Program 7.5.

Program 7.5 Pseudo triangle inequality formalised.

```
(defthm pseudo-triangle-inequality
 (implies (and (real-listp x) (real-listp y) (= (len y) (len x))
               (realp a) (<= 0 a) (<= a 1))
          (<= (* a (- 1 a) (metric^2 x y))
              (+ (* a (norm^2 x))
                 (* (- 1 a) (norm^2 y))))) ...)
```

Chapter 8

Formalising Nesterov's Theorem

While convexity and differentiability can open up a slew of optimisation techniques, requiring moderate smoothness conditions on the derivatives of convex functions can lead to extreme speed-ups in the convergence rates of algorithms (e.g. certain stochastic gradient descent methods [43]). In particular, Lipschitz continuity is one such condition. Recall the notation of Section 3.5: $\mathcal{F}^1(\mathbb{R}^n)$ refers to the class of once continuously differentiable convex functions on $\mathbb{R}^n$ and $\mathcal{F}^1_L(\mathbb{R}^n)$ refers to the class of functions in $\mathcal{F}^1(\mathbb{R}^n)$ with Lipschitz continuous derivatives (or gradients) with constant L. Then Nesterov's Theorem (i.e. Theorem 3 in Section 3.5) provides six equivalent conditions for showing that a function with a continuous first derivative (i.e. a function in $\mathcal{F}^1(\mathbb{R}^n)$) also has a Lipschitz continuous derivative (ie. is in $\mathcal{F}^1_L(\mathbb{R})$).

Theorem 3. Let $f \in \mathcal{F}^1(\mathbb{R}^n)$, $x, y \in \mathbb{R}^n$ and $\alpha \in [0, 1]$. The following conditions are equivalent to $f \in \mathcal{F}^1_L(\mathbb{R}^n)$:

$$f(y) \leq f(x) + \langle f'(x), y - x\rangle + \frac{L}{2}\|x - y\|^2 \qquad \text{Nest. 1}$$

$$f(x) + \langle f'(x), y - x\rangle + \frac{1}{2L}\|f'(x) - f'(y)\|^2 \leq f(y) \qquad \text{Nest. 2}$$

$$\frac{1}{L}\|f'(x) - f'(y)\|^2 \leq \langle f'(x) - f'(y), x - y\rangle \qquad \text{Nest. 3}$$

$$\langle f'(x) - f'(y), x - y\rangle \leq L\|x - y\|^2 \qquad \text{Nest. 4}$$

$$f(\alpha x + (1-\alpha)y) + \frac{\alpha(1-\alpha)}{2L}\|f'(x) - f'(y)\|^2 \leq \alpha f(x) + (1-\alpha)f(y) \qquad \text{Nest. 5}$$

$$\alpha f(x) + (1-\alpha)f(y) \leq f(\alpha x + (1-\alpha)y) + \alpha(1-\alpha)\frac{L}{2}\|x - y\|^2. \qquad \text{Nest. 6}$$

There are several motivations to prove Theorem 3 in ACL2(r). First, such a formalisation provides an unambiguous statement of the theorem. We

Theorem	Statement	ACL2 Name	Program
Nesterov's Theorem	See Theorem 3	`nesterov`	8.9

Table 8.1: Key theorems in this chapter.

discuss the issues of Nesterov's original theorem statement in Section 8.5. Secondly, this enables the use of Theorem 3 for further reasoning about convex functions and optimisation algorithms. Indeed, Theorem 3 has applications in the convergence proofs for many gradient descent algorithms.

This chapter discusses our formalisation of Theorem 3. First, we look at the differences between our approach and Nesterov's approach to proving Theorem 3. Second, we outline a few basic definitions and lemmas useful for the proof. Then, we outline some of the challenges we encountered during our formalisation of Theorem 3. Several of these issues involve the proofs of the remaining lemmas, which all require some user intervention beyond simple algebraic manipulation. We discuss two such instances; the others are omitted because we solve them similarly. Finally, we discuss the final form of Theorem 3 and the various considerations regarding it and alternative approaches.

8.1 Approach and Basic Definitions & Lemmas

In *Introductory Lectures on Convex Optimization*, Nesterov provided a proof that followed the structure visualised by Figure 8.1. His proof uses techniques that are not amenable to proofs in ACL2(r). In particular, integration is used multiple times to show some inequalities but integration in ACL2(r) is dependent on the function that is being integrated. This places extra obligations on the user. Our alternate approach shown in Figure 8.2 requires fewer instances of integration than Figure 8.1. Moreover, Figure 8.2 has fewer implications to prove in general and is visually more pleasing when compared to Figure 8.1[2]. The primary difference in our approach is that we prove Nest. 4 from a straightforward application of Cauchy-Schwarz and omit Nest. 1 implies Nest. 4.

[2] In addition to differences in visual symmetry, the more imaginative reader may notice that Figure 8.1 resembles a pet fish whereas Figure 8.2 resembles a rabbit. However, closer inspection reveals that the long "ears" in Figure 8.2 are more characteristic of hares than rabbits [23] and hares are superior to pet fish. Snowshoe hares found in Yukon, Canada have been observed to eat, among other carrion, the carcasses of birds, Canadian Lynx (their primary predator), and, indeed, fish [34, 37]. The superiority of hares over fish serves as further evidence for the advantages in our approach over Nesterov's.

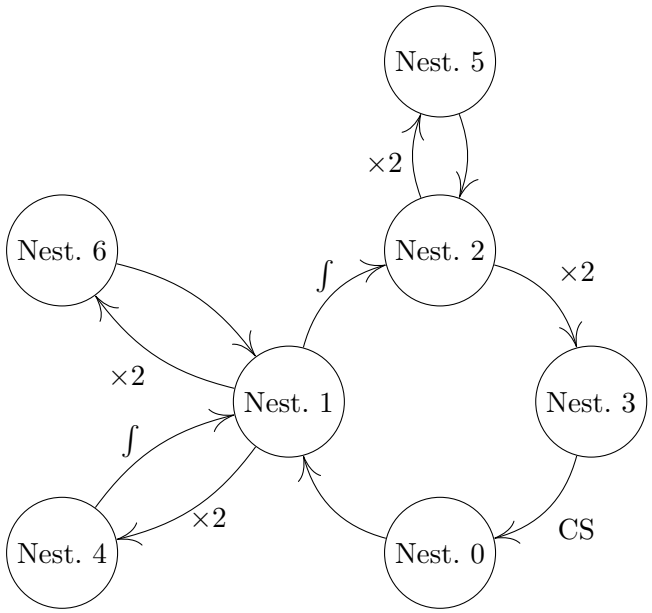

Figure 8.1: Nesterov's proof of Theorem 3. Here Nest. 0 is Lipschitz continuity. Applying Cauchy-Schwarz is denoted by CS. Integration is denoted by $\int$. Instantiating inequalities twice is denoted by $\times 2$.

Stating a theorem about functions in ACL2 is an unnatural endeavour because ACL2 is a theorem prover for first-order logic so we cannot predicate over sets in general. The natural solution is to leverage encapsulation and functional instantiation to obtain pseudo-higher order behaviour. However, this means that the desired function for which the user wishes to apply Theorem 3 must pass the theorems within the encapsulation. To formalise the theorem in its greatest generality, it is necessary to suppress the definition of the witness function in the encapsulation and instead prove the theorems based on the properties of the function. To use functional instantiation, these properties must be proven for the desired function. Thus, we aim to minimize the number of properties that the user must show, and derive as much as possible within the encapsulation. In our case, the user obligations are the theorems and identities involving the continuity, derivative, and integral of the encapsulated functions as well as any forms explicitly involving the dimension of the space.

The encapsulated functions include the multivariate function of inter-

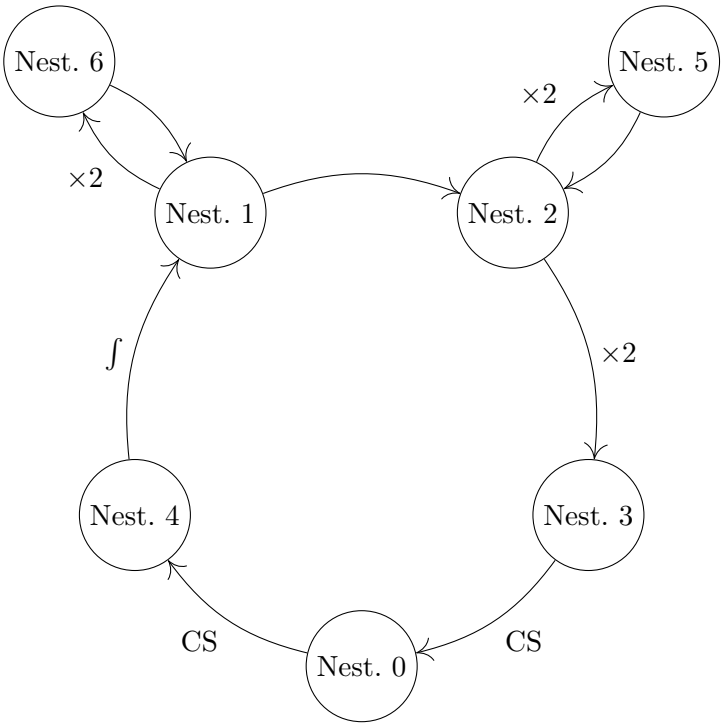

Figure 8.2: Another proof of Theorem 3. Here Nest. 0 is Lipschitz continuity. Applying Cauchy-Schwarz is denoted by CS. Integration is denoted by $\int$. Instantiating inequalities twice is denoted by $\times 2$.

est `mvfn` and its derivative `nabla-mvfn`, the function that evaluates to the Lipschitz constant `L`, a helper function `phi` based on `mvfn` and its derivative `nabla-phi`, the recognizer for vectors with standard real entries `standard-vecp`, and the function, `DIM`, that evaluates to the dimension n of the vector space. The helper function `phi` as originally defined by Nesterov is used to better elucidate the proof of a lemma [33, Lemma 1.2.3].

The prohibition of non-classical recursive functions and the necessity of a recognizer for vectors with standard entries forces the recognizer to be defined with a specific n within the encapsulation. An encapsulation requires nevertheless a witness for the convex function of interest (which in this case happens to be the function $f(x) = 42$) so we set $n = 2$. The recognizer `standard-vecp` simply checks whether a vector of dimension 2 has standards in both its coordinates.

A recurring and particularly useful theorem is the Cauchy-Schwarz inequality. It was formalised in Chapter 5 and the particular version we use is Cauchy-Schwarz II, or `cauchy-schwarz-2` in Program 5.8.

There are several proofs of implications in Figure 8.2 that follow almost immediately from the mentioned definitions and lemmas. We outline one of them: Nest. 0 implies Nest. 4. The proof mimics the chain of inequalities

$$L\|x-y\|^2 \geq \|f'(x)-f'(y)\| \cdot \|x-y\| \geq \langle f'(x)-f'(y), x-y\rangle \qquad 8.1$$

where the first inequality follows from Lipschitz continuity and the second inequality follows from Cauchy-Schwarz II. Indeed, applying Lipschitz continuity gives us the first inequality in Equation 8.1 as seen in Program 8.1. Applying `cauchy-schwarz-2` from Program 5.8 gives the second inequality

Program 8.1 The first inequality follows from Lipschitz continuity.

```
;; ||f'(x) - f'(y)|| <= L||x - y|| implies
;; ||f'(x) - f'(y)|| ||x - y|| <= L||x - y||^2
(local (defthm lemma-2
 (implies (and (real-listp x) (real-listp y)
               (= (len y) (len x)) (= (len x) (DIM))
               (<= (eu-norm (vec-- (nabla-mvfn x) (nabla-mvfn y)))
                   (* (L) (eu-norm (vec-- x y)))))
          (<= (* (eu-norm (vec-- (nabla-mvfn x) (nabla-mvfn y)))
                 (eu-norm (vec-- x y)))
              (* (L) (eu-norm (vec-- x y))
                     (eu-norm (vec-- x y)))))...))
```

of Equation 8.1 under absolute values as seen in Program 8.2. Eliminating absolute values gives the desired inequality (in an "expanded" form) as seen in Program 8.3. To state the implication in its full generality and for reasons to appear in the next section, we use Skolem functions to replace the inequalities in the theorem. The function definitions can be seen in Program 8.4. Here, `(defun-sk ineq-0 (L) ...)` introduces two new functions: `ineq-0` and `ineq-0-witness`. These are constrained functions similar to those defined by encapsulations (see Section 7.2). The rules introduced for these functions state that:

1. If there is a counterexample to the `forall` term, then `ineq-0-witness` returns such a counterexample; otherwise, `ineq-0-witness` just returns an arbitrary `x`, `y` pair.

Program 8.2 The second inequality follows from Cauchy-Schwarz.

```
;; |<f'(x) - f'(y), x - y>| <= L||x - y||^2
(local (defthm lemma-3
 (implies (and (real-listp x) (real-listp y)
               (= (len y) (len x)) (= (len x) (DIM))
               (<= (eu-norm (vec-- (nabla-mvfn x) (nabla-mvfn y)))
                   (* (L) (eu-norm (vec-- x y)))))
          (<= (abs (dot (vec-- (nabla-mvfn x) (nabla-mvfn y))
                        (vec-- x y)))
              (* (L) (eu-norm (vec-- x y))
                     (eu-norm (vec-- x y)))))
 :hints (("GOAL" :use ((:instance lemma-2)
                       ...
                       (:instance cauchy-schwarz-2
                                  (u (vec-- (nabla-mvfn x)
                                            (nabla-mvfn y)))
                                  (v (vec-- x y))))))))
```

Program 8.3 The desired implication in an expanded form.

```
;; <f'(x) - f'(y), x - y> <= L ||x - y||^2
(defthm ineq-0-implies-ineq-4-expanded
 (implies (and (real-listp x) (real-listp y)
               (= (len y) (len x)) (= (len x) (DIM))
               (<= (eu-metric (nabla-mvfn x) (nabla-mvfn y))
                   (* (L) (eu-metric x y))))
          (<= (dot (vec-- (nabla-mvfn x) (nabla-mvfn y)) (vec-- x y))
              (* (L) (metric^2 x y))))
 :hints (("GOAL" :use (...(:instance lemma-3)))))
```

2. The function `ineq-0` returns the result of applying the test for the inequality to the values of `x`, `y` returned by `ineq-0-witness`.

Note that if the `forall` claim holds, then it holds for the arbitrary `x` and `y` returned by `ineq-0-witness`, and `ineq-0` returns `t`. Conversely, if the `forall` claim does not hold, then `ineq-0-witness` returns a counterexample, and `ineq-0` returns `nil`. Because these are constrained functions and, in particular, not exacutable, the user doesn't need to write code that tests such a witness. When proving `ineq-0-implies-ineq-4`, ACL2 uses the hypothesis of `ineq-0` to infer that no such counterexample exists, and uses that to prove that `ineq-4` must hold for any `x` and `y`.

The theorem then becomes of the form seen in Program 8.5 where the `hypotheses` function constrains any witness of a counterexample to must consist of two real vectors of length `(DIM)`. All implications in Figure 8.2 are of this form.

The other implications that follow mainly from the definitions are Nest. 4 implies Nest. 1 and Nest. 1 implies Nest. 2.

Program 8.4 Skolem functions that allow us to invoke the `forall` quantifier.

```
;; Lipschitz continuity ||f'(x) - f'(y)|| <= L ||x - y||
(defun-sk ineq-0 (L)
 (forall (x y)
  (<= (eu-metric (nabla-mvfn x) (nabla-mvfn y))
      (* L (eu-metric x y))))...)
...
;; <f'(x) - f'(y), x - y> <= L ||x - y||^2
(defun-sk ineq-4 (L)
 (forall (x y)
  (<= (dot (vec-- (nabla-mvfn x) (nabla-mvfn y)) (vec-- x y))
      (* L (metric^2 x y))))...)
```

Program 8.5 Nest. 0 implies Nest. 4

```
(defthm ineq-0-implies-ineq-4
  (implies (and (hypotheses (ineq-4-witness (L)) (DIM))
                (ineq-0 (L)))
           (ineq-4 (L)))...)
```

8.2 Instantiating Inequalities

The proof of Nest. 2 implies Nest. 3 amounts to adding two copies of Nest. 2 with x, y swapped. This induces issues with the proof of the implication. The natural form of the lemma would involve Nest. 2 among the hypotheses as in Program 8.6.

Program 8.6 An "obvious" way to state Nest. 2 implies Nest. 3.

```
(defthm ineq-2-implies-ineq-3
  (implies (and (real-listp x) (real-listp y)
                (= (len y) (len x)) (= (len x) (DIM))
                (ineq-2 (L)))
           (ineq-3 (L)))...)
```

However, to instantiate a copy of Nest. 2 with swapped x, y in such a form would be equivalent to

$$\forall x, y, (P(x, y) \implies P(y, x)) \tag{8.2}$$

where P is a predicate (in this case equivalent to Nest. 2), which is not necessarily true. The form we wish to have is

$$(\forall x, y, P(x, y)) \implies (\forall x, y, P(y, x)). \tag{8.3}$$

In order to instantiate another copy of Nest. 2 within the implication requires quantifiers within the theorem statement. The usual approach involves using Skolem functions to introduce quantified variables. With this method, we can instantiate the two copies with swapped variables as in Program 8.7.

We also considered simply including two copies of the inequality with swapped variables among the hypotheses. This has two advantages. Firstly, with such a form, the lemma becomes stronger because the hypothesis $P(x, y) \wedge P(y, x)$ is weaker than $\forall x, y, P(x, y)$. Secondly, the lemma is slightly easier to pass in ACL2(r). However, the primary drawback is that this form is inconsistent with the other lemmas and the final form of Nesterov's theorem becomes less elegant (e.g. showing Nest. 1 implies Nest. 2 would also require two copies of Nest. 1).

The lemmas Nest. 1 implies Nest. 6 and Nest. 2 implies Nest. 5 also requires instantiating multiple copies of the antecedent inequalities (albeit with different vectors).

Program 8.7 Instantiating two copies of Nest. 2 with swapped variables.

```
(defthm ineq-2-expanded-v1
  (implies (ineq-2 (L))
           (and (<= (+ (mvfn x)
                       (dot (nabla-mvfn x) (vec-- y x))
                       (* (/ (* 2 (L)))
                          (metric^2 (nabla-mvfn x) (nabla-mvfn y))))
                    (mvfn y))
                (<= (+ (mvfn y)
                       (dot (nabla-mvfn y) (vec-- x y))
                       (* (/ (* 2 (L)))
                          (metric^2 (nabla-mvfn y) (nabla-mvfn x))))
                    (mvfn x))))...)
```

8.3 Taking Limits

In the language of non-standard analysis, limits amount to taking standard-parts. For example, $\lim_{x \to a} f(x)$ is equivalent to $\mathrm{st}(f(x))$ when $x - a$ is an infinitesimal. However, for products, say, xy, the identity $\mathrm{st}(xy) = \mathrm{st}(x)\,\mathrm{st}(y)$ only holds when x, y are both finite reals. In the proof of Nest. 6 implies Nest. 1, there is a step that requires taking the limit of $(1 - \alpha)\|y - x\|^2$ as $\alpha \to 0$. Now, if $\alpha > 0$ is an infinitesimal,

$$\mathrm{st}((1 - \alpha)\|y - x\|^2) = \mathrm{st}(1 - \alpha)\,\mathrm{st}(\|y - x\|^2) = \|y - x\|^2 \tag{8.4}$$

is easy to satisfy when x, y are vectors with standard real components. Moreover, requiring variables to be standard is consistent with some instances of single variable theorems (e.g. the product of continuous functions is continuous). It then remains to state such a hypothesis using, say, a recognizer `standard-vecp` as in Program 8.8. The natural approach to defining `standard-vecp` would be to simply recurse on the length of a vector applying `standardp` to each entry. However, `standardp` is non-classical and this definition encounters a common issue throughout our ACL2(r) formalisation in that such a definition would be a non-classical recursive function which is forbidden in ACL2(r) (see Section 6.3). Because `standard-vecp` is dependent on the dimension, our solution is to encapsulate the function and prove the necessary theorems involving it (e.g. `metric^2` is `standardp` on `standard-vecp` values). For the case $n = 2$, we simply check the length of the vector is two and that both entries are standard reals. A final note regarding the lemma is the hypothesis $\alpha > 0$ replacing the weaker $\alpha \geq 0$

Program 8.8 Introducing `standard-vecp` into the hypotheses.

```
(defthm ineq-6-implies-ineq-1-expanded
 (implies (and (real-listp x) (real-listp y)
               (= (len y) (len x)) (= (len x) (DIM))
               (realp alpha) (i-small alpha)
               (< 0 alpha) (<= alpha 1)
               (standard-vecp x) (standard-vecp y)
               (<= (+ (* alpha (mvfn y))
                      (* (- 1 alpha) (mvfn x)))
                   (+ (mvfn (vec-+ (scalar-* alpha y)
                                   (scalar-* (- 1 alpha) x)))
                      (* (/ (L) 2)
                         alpha
                         (- 1 alpha)
                         (metric^2 y x)))))
          (<= (mvfn y)
              (+ (mvfn x)
                 (dot (nabla-mvfn x) (vec-- y x))
                 (* (/ (L) 2) (metric^2 y x))))))...)
```

since part of the proof depends on dividing by α.

The other lemma that requires similar hypotheses is the implication Nest. 5 implies Nest. 2.

8.4 Nesterov's Final Form

Finally, we discuss our final form of Theorem 3 as well as several alternatives and their considerations. The final form can be seen in Program 8.9. The function `hypotheses` ensures that we are dealing with real vectors of the correct dimension. The function `st-hypotheses` ensure that the vectors have standard entries due to the necessity of taking limits. The function `alpha-hypotheses` is the same as `hypotheses` but includes the hypothesis that $\alpha \in [0, 1]$. The function `alpha->-0-hypotheses` ensures that $\alpha > 0$ for the case of taking limits.

In Section 8.1, we saw, for each lemma, the basic structure involving Skolem functions. In Section 8.2, we cited elegance and ease of instantiation as reasons for using Skolem functions. Because stating even the shorter inequalities in Polish notation would quickly become awkward and unintelligible (e.g. Program 8.10), it became desirable for us to define the inequalities

Program 8.9 Theorem 3.

```
(defthm nesterov
 ;; theorem statement
 (implies (and (hypotheses (ineq-0-witness (L)) (DIM))
                   (hypotheses (ineq-1-witness (L)) (DIM))
                   (hypotheses (ineq-2-witness (L)) (DIM))
                   (hypotheses (ineq-3-witness (L)) (DIM))
                   (hypotheses (ineq-4-witness (L)) (DIM))
                   (st-hypotheses (ineq-1-witness (L)))
                   (st-hypotheses (ineq-2-witness (L)))
                   (alpha-hypotheses (ineq-5-witness (L)) (DIM))
                   (alpha-hypotheses (ineq-6-witness (L)) (DIM))
                   (alpha->-0-hypotheses (ineq-5 (L)))
                   (alpha->-0-hypotheses (ineq-6 (L)))
                   (or (ineq-0 (L)) (ineq-1 (L)) (ineq-2 (L))
                       (ineq-3 (L)) (ineq-4 (L)) (ineq-5 (L))
                       (ineq-6 (L))))
          (and (ineq-0 (L)) (ineq-1 (L)) (ineq-2 (L)) (ineq-3 (L))
               (ineq-4 (L)) (ineq-5 (L)) (ineq-6 (L))))
 ;; hints, etc. for ACL2(r)
 ...)
```

in a clean and clear manner. One simple approach would be to define the inequalities as ACL2 functions or macros. Unfortunately, during the course of our formalisation, we found that the rewriter would be tempted to "simplify" or otherwise change the form of the inequality via arithmetic rules. This made applications of certain theorems more involved and arduous than necessary. Therefore, it would be necessary to disable the function definitions anyways. In addition to permitting instantiations of inequalities with different vectors within a single theorem statement, Skolem functions would allow us to suppress or "hide" the explicit inequality thus providing a clear, concise, and compact package.

Program 8.10 Nest. 5 in Polish notation.

```
(<= (+ (mvfn (vec-+ (scalar-* alpha y) (scalar-* (- 1 alpha) x)))
       (* (/ (* 2 (L)))
          (* alpha (- 1 alpha) (metric^2 (nabla-mvfn y)
                                         (nabla-mvfn x)))))
    (+ (* alpha (mvfn y)) (* (- 1 alpha) (mvfn x))))
```

On the other hand, this form has the unfortunate drawback of making the proof of the theorem slightly more involved. By introducing Skolem functions we also introduce the necessity of witness functions; proving the lemmas in terms of the witness functions may occasionally become onerous. For example, to state the hypotheses that the entries of the witness functions are real vectors of the same dimension, we would like to define a `hypotheses` function. However, explicitly exhibiting the witness functions within the definition of `hypotheses` leads to a signature mismatch. We instead pass the witnesses for each Skolem function to the appropriate `hypotheses` function.

8.5 Ambiguities in Nesterov's Statement

Theorem 3 as stated by Nesterov in *Introductory Lectures on Convex Optimization* begins with

> "All the conditions below, holding for all $x, y \in \mathbb{R}^n$ and α from $[0, 1]$, are equivalent to inclusion $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^n) \ldots$"

and then proceeds to list inequalities Nest. 1 through Nest. 6 (Nesterov writes $\mathcal{F}_L^{k,p}(\mathbb{R}^n)$ for convex k times differentiable functions on $\mathbb{R}^n$ with Lipschitz continuous p-th derivative with constant L). What does Nesterov

Interpretation		Correctness
$\forall f : \mathbb{R}^n \to \mathbb{R}$,	$f \in \mathcal{F}_L^{1,1} \iff \text{N1} \iff \cdots \iff \text{N6}$	False
$\forall f \in C$,	$f \in \mathcal{F}_L^{1,1} \iff \text{N1} \iff \cdots \iff \text{N6}$	False
$\forall f \in C^1$,	$f \in \mathcal{F}_L^{1,1} \iff \text{N1} \iff \cdots \iff \text{N6}$	False
$\forall f \in C^{1,1}$,	$f \in \mathcal{F}_L^{1,1} \iff \text{N1} \iff \cdots \iff \text{N6}$	False
$\forall f \in \mathcal{C}_L^{1,1}$,	$f \in \mathcal{F}_L^{1,1} \iff \text{N1} \iff \cdots \iff \text{N6}$	Almost True
$\forall f \in \mathcal{F}^{1,1}$,	$f \in \mathcal{F}_L^{1,1} \iff \text{N1} \iff \cdots \iff \text{N6}$	True

Table 8.2: Multiple interpretations of Nesterov's statement. Here, we write $\mathcal{F}_L^{1,1}$ for $\mathcal{F}_L^{1,1}(\mathbb{R}^n)$, $C_L^{1,1}$ for continuous functions on $\mathbb{R}^n$ with indices retaining their previous meaning, N1 for Nest. 1, etc.

mean? At first glance, it appears that he means if any function satisfies any of the six inequalities, then it is convex with Lipschitz continuous gradient and satisfies any of the other inequalities. However, looking at any of Nest. 1 to Nest. 6 makes it clear this is not the case. In fact, there are multiple interpretations of the statement, yet only one is true. Some examples are seen in Table 8.2.

The theorem actually requires the assumption $f \in \mathcal{F}^1(\mathbb{R}^n)$, but this hypothesis is not explicitly stated in the theorem statement in Nesterov's original text. Instead, the assumption is implicit in the preceding text. On the other hand, Nest. 5 implies that f is convex. Nest. 2 through Nest. 6 implicitly have an existential quantification of L. Furthermore, note the placement of the quantifiers (as discussed in Section 8.2). Each N1 is of the form

$$\forall x, y,\ \texttt{hypotheses}(x, y) \implies \texttt{ineq-1}(x, y) \tag{8.5}$$

and similarly for N2, N3, etc. Thus, N1 $\iff$ N2 means

$$\begin{array}{c} (\forall x, y,\ \texttt{hypotheses}(x, y) \implies \texttt{ineq-1}(x, y)) \\ \Updownarrow \\ (\forall x, y,\ \texttt{hypotheses}(x, y) \implies \texttt{ineq-2}(x, y)). \end{array} \tag{8.6}$$

and, in particular, Nesterov's theorem is not

$$\forall x, y,\ \texttt{hypotheses}(x, y) \implies (\texttt{ineq-1}(x, y) \iff \texttt{ineq-2}(x, y)). \tag{8.7}$$

By stating and proving the theorem in ACL2(r), these ambiguities are avoided.

Chapter 9

Conclusion and Future Work

The various structures of $\mathbb{R}^n$ are very rich in mathematical theory and hold applications in various areas of science. In this thesis, we presented a formalisation of the space from two perspectives. This choice of perspectives is arguably among the most fundamental. It is the vector space structure of $\mathbb{R}^n$ that provides the necessary operations between its elements. Indeed, one would be hard-pressed to find any view of $\mathbb{R}^n$ that does not assume any operations on the domain. Moreover, inner products are the path to calculus: inner products lead to norms; norms lead to metrics; and metrics lead to real analysis. The formalisation of $\mathbb{R}^n$ as a metric space is among the last steps before multivariate calculus which is in and of itself full of applications and left as future work.

During the course of formalisation, emphasis was placed on using algebraic methods to prove theorems that would have otherwise been proved via induction. For example, compare the various flavours Cauchy-Schwarz proofs; there are geometric proofs, analytical proofs, combinatorial proofs, inductive proofs, etc. [49] whereas we followed an algebraic approach. Considering ACL2(r)'s strengths with regards to induction, the choice may seem odd. Indeed, there are several potential inductive candidates we considered before proceeding at the onset of this endevour. However, most of these candidates inducted over the dimension of $\mathbb{R}^n$ and required reasoning over the real entries of vectors. We suspect unwinding the vectors and guiding ACL2(r) through such a proof would be more onerous than the one outlined in this paper. Moreover, our formalisation of inner product spaces already provided the exact tools necessary for the chosen proof of Cauchy-Schwarz (i.e. vectors, vector-vector operations, scalar-vector operations, inner products, etc.) without resorting to reasoning over individual reals. The precision of this approach is arguably more elegant. Not only does the formalisation of Cauchy-Schwarz suggest various applications to areas in which it appears (e.g. in functional analysis, probability theory, combinatorics, etc.), but the ACL2(r) proof is a demonstration of how the inner product space properties can be used to prove useful and interesting mathematical theorems with a mechanical theorem prover.

Unfortunately, algebraic approaches require significant guidance from the user. The solution may be to leverage automated tools that perform better at symbolic manipulation. To this end, the properties of the inner product space axioms might be amenable to certification by a SMT solver via `smtlink` [35, 36]. The challenge here is that SMT solvers do not perform induction – we need to leave that for ACL2. On the other hand, we might be able to treat operations on vectors as uninterpreted functions with constraints corresponding to the requirements for a function to be an inner product, a norm, etc.

One more reason to take an algebraic approach is to explicitly avoid an inductive one. While ACL2 relies heavily on and excels at induction, this recursive strategy can fail for vectors in a non-standard setting due to soundness motivated limitations outlined in Section 6.3. We avoid recursion by instead reasoning about the essential properties of the data structure; for example, the maximum element in a vector of real numbers. The upshot is that this approach provides a viable method for reasoning about continuity which serves to justify our formalisation of metric spaces (with respect to the Euclidean metric). Moreover, there is still potential to further extend $\mathbb{R}^n$ as a metric space. The notions of continuity are independent of the metric used and d_2 may be replaced with any metric on $\mathbb{R}^n$. By way of encapsulation, pseudo-higher-order techniques may be employed to easily formalise various real metric spaces – especially if we consider the metrics induced by other p-norms. Among the extensions of $\mathbb{R}^n$ as a metric space is proving its completeness. Addressing Cauchy sequences traditionally follows from an application of Bolzano-Weierstrass which has yet to be formalised [44]. Stating completeness in terms of infinitesimals and ACL2(r) is a farther but tantalizing prospect. Upon doing so, we would have a formalisation of $\mathbb{R}^n$ as a Hilbert space [39].

To serve as proof of feasibility for these constructions, we also presented a set of theorems for reasoning about convex functions in ACL2(r). Our theory of convex functions also justifies the utility of our formalisation of the proceeding inner product and metric space theories by serving as an automated tool for reasoning in an application-heavy area of mathematics. Our particular interest in this work is the potential applications to verifying, among other areas, optimisation algorithms used in machine learning. To this end, we chose a theorem of Nesterov's to serve as an example of the analytical reasoning possible in our formalisation. The natural next step would be to develop a proper theory of multivariate calculus to further automate the reasoning of optimisation algorithms.

Bibliography

[1] *Formalizing 100 Theorems in Mizar*. Online, `https://mizar.org/100`.

[2] ACL2: *Encapsulate*. Available at `https://www.cs.utexas.edu/users/moore/acl2/manuals/current/manual/?topic=ACL2___ENCAPSULATE`.

[3] Sanaz Khan Afshar, Vincent Aravantinos, Osman Hasan & Sofiène Tahar (2014): *Formalization of Complex Vectors in Higher-Order Logic*. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka & Josef Urban, editors: *Intelligent Computer Mathematics*, Springer International Publishing, Cham, pp. 123–137, doi:10.1023/A:1012692601098.

[4] K. Appel & W. Haken (1977): *Every planar map is four colorable. Part I: Discharging*. *Illinois J. Math.* 21(3), pp. 429–490, doi:10.1215/ijm/1256049011.

[5] K. Appel, W. Haken & J. Koch (1977): *Every planar map is four colorable. Part II: Reducibility*. *Illinois J. Math.* 21(3), pp. 491–567, doi:10.1215/ijm/1256049012.

[6] John M. H. Olmsted Bernard R. Gelbaum (2003): *Counterexamples in Analysis. Dover Books on Mathematics* , Dover Publications.

[7] Stephen Boyd & Lieven Vandenberghe (2004): *Convex Optimization*. Cambridge University Press, doi:10.1017/CBO9780511804441.

[8] T. Coe, T. Mathisen, C. Moler & V. Pratt (1995): *Computational aspects of the Pentium affair*. *IEEE Computational Science and Engineering* 2(1), pp. 18–30, doi:10.1109/99.372929.

[9] John Cowles & Ruben Gamboa (2017): *The Cayley-Dickson Construction in ACL2*. In Anna Slobodova & Warren Hunt, Jr., editors: Proceedings 14th International Workshop on the *ACL2 Theorem Prover and its Applications*, Austin, Texas, USA, May 22-23, 2017, *Electronic*

Proceedings in Theoretical Computer Science 249, Open Publishing Association, pp. 18–29, doi:10.4204/EPTCS.249.2.

[10] Ruben Gamboa: personal communication via the ACL2 help list on 2018-03-01.

[11] Ruben A. Gamboa & Matt Kaufmann (2001): *Nonstandard Analysis in ACL2*. *Journal of Automated Reasoning* 27(4), pp. 323–351, doi:10.1023/A:1011908113514.

[12] John Harrison (2005): *A HOL Theory of Euclidean Space*. In Joe Hurd & Tom Melham, editors: *Theorem Proving in Higher Order Logics*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 114–129, doi:10.1007/11541868-8.

[13] John Harrison (2013): *The HOL Light Theory of Euclidean Space*. *Journal of Automated Reasoning* 50(2), pp. 173–190, doi:10.1007/s10817-012-9250-9.

[14] Marijn Heule (2018): *Schur Number Five*. Available at `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16952`.

[15] Marijn J. H. Heule, Oliver Kullmann & Victor W. Marek (2016): *Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer*. In Nadia Creignou & Daniel Le Berre, editors: *Theory and Applications of Satisfiability Testing – SAT 2016*, Springer International Publishing, Cham, pp. 228–245.

[16] Nathan Jacobson (1985): *Basic Algebra I*, 2nd edition. Dover Publications.

[17] Nathan Kahl: *The Hundred Greatest Theorems*. Online. Available at `http://pirate.shu.edu/~kahlnath/Top100.html`. Originally published by Paul and Jack Abad (1999).

[18] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz & Yuval Yarom (2019): *Spectre Attacks: Exploiting Speculative Execution*. In: *40th IEEE Symposium on Security and Privacy (S&P'19)*.

[19] Boris Konev & Alexei Lisitsa (2014): *A SAT Attack on the Erdős Discrepancy Conjecture*. In Carsten Sinz & Uwe Egly, editors: *Theory*

and Applications of Satisfiability Testing – SAT 2014, Springer International Publishing, Cham, pp. 219–226.

[20] Carl Kwan & Mark R. Greenstreet (2018): *Convex Functions in ACL2(r).* In: Proceedings 15th International Workshop on the *ACL2 Theorem Prover and its Applications,* Austin, Texas, USA, November 5-6, 2018, Electronic Proceedings in Theoretical Computer Science 280, Open Publishing Association, pp. 128-142.

[21] Carl Kwan & Mark R. Greenstreet (2018): *Real Vector Spaces and the Cauchy-Schwarz Inequality in ACL2(r).* In: Proceedings 15th International Workshop on the *ACL2 Theorem Prover and its Applications,* Austin, Texas, USA, November 5-6, 2018, Electronic Proceedings in Theoretical Computer Science 280, Open Publishing Association, pp. 111-127.

[22] Serge Lang (2002): *Algebra*, 3rd edition. *Graduate Texts in Mathematics 211* , Springer-Verlag New York, doi:10.1007/978-1-4613-0041-0.

[23] Liz Langley (2014): *What's the Difference Between Rabbits and Hares?* Available at `https://news.nationalgeographic.com/news/2014/12/141219-rabbits-hares-animals-science-mating-courtship/`. Online.

[24] C. Ward Henson (Eds.) Leif O. Arkeryd, Nigel J. Cutland (1997): *Nonstandard Analysis: Theory and Applications*, 1st edition. *Nato Science Series C: 493* , Springer Netherlands, doi:10.1007/978-94-011-5544-1.

[25] N. G. Leveson & C. S. Turner (1993): *An Investigation of the Therac-25 Accidents. Computer* 26(7), pp. 18–41, doi:10.1109/MC.1993.274940.

[26] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom & Mike Hamburg (2018): *Meltdown: Reading Kernel Memory from User Space.* In: *27th USENIX Security Symposium (USENIX Security 18).*

[27] Peter A. Loeb & Manfred P. H. Wolff (2015): *Nonstandard Analysis for the Working Mathematician*, 2nd edition. Springer Netherlands, doi:10.1007/978-94-017-7327-0.

[28] Jean-Marie Madiot: *Formalizing 100 theorems in Coq.* Online, `https://madiot.fr/coq100`.

[29] Marco Maggesi (2018): *A Formalization of Metric Spaces in HOL Light*. *Journal of Automated Reasoning* 60(2), pp. 237–254, doi:10.1007/s10817-017-9412-x.

[30] William Mccune (1997): *Solution of the Robbins Problem*. *Journal of Automated Reasoning* 19(3), pp. 263–276, doi:10.1023/A:1005843212881.

[31] Norman Mcgill (2001): *Metamath Proof Explorer, Theorem bcseqi*. Online, `https://us.metamath.org/mpeuni/bcseqi.html`.

[32] Norman Mcgill (2008): *Metamath Proof Explorer, Theorem sii*. Online, `https://us.metamath.org/mpeuni/sii.html`.

[33] Yurii Nesterov (2004): *Introductory Lectures on Convex Optimization*, 1st edition. *Applied Optimization 87* , Springer US, doi:10.1007/978-1-4419-8853-9.

[34] Michael JL Peers, Yasmine N Majchrzak, Sean M Konkolics, Rudy Boonstra & Stan Boutin (2018): *Scavenging By Snowshoe Hares (Lepus americanus) In Yukon, Canada*. *Northwestern Naturalist* 99(3), pp. 232 – 235 – 4, doi:10.1898/NWN18-05.1.

[35] Yan Peng & Mark Greenstreet (2015): *Integrating SMT with Theorem Proving for Analog/Mixed-Signal Circuit Verification*. In Klaus Havelund, Gerard Holzmann & Rajeev Joshi, editors: *NASA Formal Methods*, Springer International Publishing, Cham, pp. 310–326, doi:10.1007/978-3-319-17524-9-22.

[36] Yan Peng & Mark R. Greenstreet (2015): *Extending ACL2 with SMT Solvers*. In: *Proceedings Thirteenth International Workshop on the ACL2 Theorem Prover and Its Applications, Austin, Texas, USA, 1-2 October 2015.*, pp. 61–77, doi:10.4204/EPTCS.192.6.

[37] Adam Popescu (2019): *Hares are cannibals and eat meat, surprising photos reveal*. Available at `https://www.nationalgeographic.com/animals/2019/01/snowshoe-hares-carnivores-cannibals-photos-yukon/`. Online.

[38] Benjamin Porter (2006): *Cauchy's Mean Theorem and the Cauchy-Schwarz Inequality*. *Archive of Formal Proofs*. `http://isa-afp.org/entries/Cauchy.html`, Formal proof development.

[39] Frigyes Riesz & Bela Sz.-Nagy (1990): *Functional Analysis.* Dover Publications.

[40] Roger Jones Rob Arthan (2017): *43 famous theorems in ProofPower.* Online, `https://www.rbjones.com/rbjpub/pp/rda001.html`.

[41] Abraham Robinson (1966): *Non-Standard Analysis.* North-Holland Publishing Company.

[42] Steven Roman (2008): *Advanced Linear Algebra*, 3rd edition. *Graduate Texts in Mathematics 135* , Springer-Verlag New York, doi:10.1007/978-0-387-72831-5.

[43] Nicolas L. Roux, Mark Schmidt & Francis R. Bach (2012): *A Stochastic Gradient Method with an Exponential Convergence _Rate for Finite Training Sets.* In F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, editors: *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 2663–2671.

[44] Walter Rudin (1976): *Principles of Mathematical Analysis*, 3rd edition. *International Series in Pure and Applied Mathematics* , McGraw-Hill.

[45] Robert Skeel (1992): *Roundoff error and the Patriot missile. SIAM News* 25(4), p. 11.

[46] Jasper Stein (2001): *Documentation for the formalization of Linerar Agebra.* Online, `http://www.cs.ru.nl/~jasper`.

[47] H. D. Vinod & B. D. McCullough (1999): *The Numerical Reliability of Econometric Software. Journal of Economic Literature* 37(2), pp. 633–665.

[48] Freek Wiedijk: *Formalizing 100 Theorems.* Online, `http://www.cs.ru.nl/~freek/100`.

[49] Hui-Hua Wu & Shanhe Wu (2009): *Various proofs of the Cauchy-Schwarz inequality. Octogon Mathematical Magazine* 17(1), pp. 221–229.

Appendix A

Classical Proofs

Theorem 2. If $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R} \to \mathbb{R}$ are convex with h monotonically increasing, then

1. $a \cdot f$ is convex for any real $a \geq 0$,
2. $f + g$ is convex,
3. $h \circ f$ is convex.

Proof. Let $\alpha \in [0, 1]$. If f is convex, then so is af for $a \geq 0$ since

$$af(\alpha x+(1-\alpha)y) \leq a(\alpha f(x)+(1-\alpha)f(y)) = \alpha(af(x))+(1-\alpha)(af(y)). \quad \text{A.1}$$

If f and g are both convex, then

$$\begin{aligned}(f+g)(\alpha x+(1-\alpha)y) &= f(\alpha x+(1-\alpha)y)+g(\alpha x+(1-\alpha)y) \\ &\leq \alpha f(x)+(1-\alpha)f(y)+\alpha g(x)+(1-\alpha)g(y) \quad \text{A.2} \\ &= \alpha(f+g)(x)+(1-\alpha)(f+g)(y).\end{aligned}$$

If f is convex and h is convex and monotonically increasing, then

$$h\circ f(\alpha x+(1-\alpha)y) \leq h(\alpha f(x)+(1-\alpha)f(y)) \leq \alpha h\circ f(x)+(1-\alpha)h\circ f(y) \quad \text{A.3}$$

where the first inequality follows from the convexity of f and monoticity of h and the second inequality follows from the convexity of h. □

Theorem 3. Let $f \in \mathcal{F}^1(\mathbb{R}^n)$, $x, y \in \mathbb{R}^n$ and $\alpha \in [0, 1]$. The following

conditions are equivalent to $f \in \mathcal{F}_L^1(\mathbb{R}^n)$:

$$f(y) \leq f(x) + \langle f'(x), y - x\rangle + \frac{L}{2}\|x - y\|^2 \qquad \text{Nest. 1}$$

$$f(x) + \langle f'(x), y - x\rangle + \frac{1}{2L}\|f'(x) - f'(y)\|^2 \leq f(y) \qquad \text{Nest. 2}$$

$$\frac{1}{L}\|f'(x) - f'(y)\|^2 \leq \langle f'(x) - f'(y), x - y\rangle \qquad \text{Nest. 3}$$

$$\langle f'(x) - f'(y), x - y\rangle \leq L\|x - y\|^2 \qquad \text{Nest. 4}$$

$$f(\alpha x + (1-\alpha)y) + \frac{\alpha(1-\alpha)}{2L}\|f'(x) - f'(y)\|^2 \leq \alpha f(x) + (1-\alpha)f(y) \qquad \text{Nest. 5}$$

$$\alpha f(x) + (1-\alpha)f(y) \leq f(\alpha x + (1-\alpha)y) + \alpha(1-\alpha)\frac{L}{2}\|x - y\|^2. \qquad \text{Nest. 6}$$

Proof. We first show Nest. 1 implies Nest. 6. By Nest. 1, we have

$$f(x) \leq f(z) + \langle f'(z), x - z\rangle + \frac{L}{2}\|z - x\|^2 \qquad \text{A.4}$$

and

$$f(y) \leq f(z) + \langle f'(z), y - z\rangle + \frac{L}{2}\|z - y\|^2. \qquad \text{A.5}$$

Add Equation A.4 multiplied by α to Equation A.5 multiplied by $1 - \alpha$ and simplify to obtain

$$\begin{aligned}\alpha f(x) + (1-\alpha)f(y) \geq\ & f(z) + \alpha\langle f'(z), x - z\rangle + (1-\alpha)\langle f'(z), y - z\rangle \\ & + \frac{\alpha(1-\alpha)}{2L}\|f'(y) - f'(z)\|^2\end{aligned} \qquad \text{A.6}$$

Set $z = \alpha x + (1-\alpha)y$ and apply Lemma 4 to obtain Nest. 6. The proof for Nest. 2 implies Nest. 5 is similar.

To see Nest. 1 from Nest. 6, swap x and y in Nest. 6 and rearrange to obtain

$$f(y) \geq f(x) + \frac{f(x + \alpha(y - x)) - f(x)}{\alpha} + \frac{1-\alpha}{2L}\|f'(x) - f'(y)\|^2. \qquad \text{A.7}$$

Take $\alpha \to 0$ to obtain Nest. 1. The proof of Nest. 5 implies Nest. 2 is similar.

For Nest. 0 implies Nest. 4, apply Cauchy-Schwarz.

We now prove Nest. 4 implies Nest. 1. Observe

$$\begin{aligned} f(y) - f(x) - \langle f'(x), y - x\rangle &= \int_0^1 \langle f'(x + \tau(y-x)) - f'(x), y - x\rangle d\tau \\ &\leq L\|y - x\|^2 \int_0^1 \tau d\tau \\ &= \frac{L}{2}\|y - x\|^2 \end{aligned} \tag{A.8}$$

which gives us Nest. 1.

Now we show Nest. 1 implies Nest. 2 Assume Nest. 1. Let $\varphi(x) = f(x) - \langle f'(x_0), x\rangle$ for some arbitrary but fixed x_0. Observe φ is continuous, convex, and differentiable and $\varphi(x_0) \leq \varphi(x)$ for any x. Moreover, $\varphi'(x) = f'(x) - f'(x_0)$. Then

$$\begin{aligned} \varphi\left(y - \frac{1}{L}\varphi'(x)\right) &\leq \varphi(y) + \langle \varphi'(y), y - \frac{1}{L}\varphi'(y) - y\rangle + \frac{1}{2L}\|\varphi'(y)\|^2 \\ &= \varphi(y) - \frac{1}{2L}\|\varphi'(y)\|^2 \end{aligned} \tag{A.9}$$

gives us Nest. 2.

Add two copies of Nest. 2 with swapped variables to obtain Nest. 3.

Apply Cauchy-Schwarz to Nest. 3 to obtain Lipschitz continuity which brings us back to Nest. 0. □