A Brief Introduction to Nonstandard Analysis and ACL2(r)

The Foundation for Taylor's Theorem with Remainder

Brittany E. Middleton
The University of Texas at Austin
Computer Sciences Department
Taylor Hall 2.124
Austin, TX 78712-1188
brittany@math.utexas.edu
www.cs.utexas.edu/users/brittany

May 21, 2002

Abstract

`ACL2(r)`, authored by Dr. Ruben A. Gamboa, is an extention of ACL2, to allow reasoning about the real and complex irrational numbers. The modifications are based on nonstandard analysis. Jun Sawada, of IBM Research Labs, required the support of `ACL2(r)` in his proof efforts to verify that the approximation used in the square root function of the IBM Power4 processor had the accuracy required. To verify the accuracy of the square root function, Taylor's Theorem was chosen. Taylor's Theorem with Remainder is an approximation tool commonly used in mathematics that provides a means for estimating *f(x)* for an arbitrary x in the interval [a, b] from the values of f and its derivitives at a. This thesis serves as an introduction to the work completed in the proof of Taylor's Theorem with Remainder, as well as a general, concise introduction of nonstandard analysis in `ACL2(r)`. The introduction is presented as a guide through the foundational lemmas needed in the proof of Taylor's Theorem with Remainder.

Advisor: Dr. J Strother Moore
Department of Computer Sciences
University of Texas at Austin

Second Reader: Dr. Ruben A. Gamboa
Department of Computer Sciences
University of Wyoming

Computer Science Honors Advisor: Dr. Mohamed G. Gouda
Department of Computer Sciences
University of Texas at Austin

To Dr. Robert S. Boyer, who supported my mathematical endeavors in the face of my struggle with epilepsy, with his patience, guidance and genius. To Dr. J Strother Moore and Dr. Ruben A. Gamboa for continuing these efforts.

There is nothing in pursuit of mathematics apart from proximity to God.
-Author Unknown

1 Introduction

More often than not, hardware products are produced and released without verification of correctness. "Intel Corp. last week took a $475 dollar million write-off to cover costs associated with the divide bug in the Pentium microprocessor's floating-point unit."[7]. Even with extensive testing, errors in these products often do not surface until after the product is in use. With hardware verification, these errors can be reduced, if not eliminated. The product is more robust, easier to maintain, and as a result, yields a higher lever of customer satisfaction. "The dream is that we should be able to write down mathematically precise conjectures about special-purpose hardware designs, microprocessors, microcode programs, compilers, algorithms, etc., and then to prove them with a mechanized reasoning system." [2, p. 9]. ACL2 is a prominent reasoning tool used in industry to realize this dream. Components of the following devices have been verified with `ACL2`: AMD K5, AMD Athlon, Motorola CAP digital signal processor, Rockwell Collins JEM1, Rockwell Collins CAPS family of avionic microprocessors, and IBM 4758 cryptographic co-processor. These particular proofs were done before chip fabrication.

At times, verification of correctness in industry requires the reasoning about the real number system. `ACL2(r)`, an extension of `ACL2`, authored by Dr. Ruben A. Gamboa, provides a tool for reasoning about the real numbers. Jun Sawada, of IBM Research Labs, needed to verify that the square root function of the IBM Power4 processor had the accuracy required, requiring the use of `ACL2(r)`. In particular, Taylor's Theorem with Remainder was used to estimate the value of the square root function and place a bound on the error in the approximation. Therefore, Taylor's Theorem with Remainder was formalized in `ACL2(r)`, as well as the various foundational lemmas required by the formalization. Jun Sawada then used this work by instantiating Taylor's Theorem with Remainder with the square root function yielding the desired results.

In the beginning of these efforts, a brief, concise introduction to `ACL2(r)` was lacking. The goal of this paper is to provide such an introduction, while referencing a concrete example of `ACL2(r)` used in industry. The paper is organized as follows. An introduction to the notions of nonstandard analysis is given, followed by a description of one approach used to define irrational functions in `ACL2(r)`. Two basic lemmas in `ACL2(r)` needed in the proofs of the various foundational lemmas that support the proof of Taylor's Theorem

with Remainder are presented. The foundational lemmas are then presented with their corresponding proofs in `ACL2(r)`. New user information regarding various tools and lemmas needed in nonstandard analysis is interspersed within this section. A brief outline of Taylor's Theorem with Remainder is presented. Finally, there are brief comments provided in regards to beginning the use of the `ACL2(r)` system.

2 A Brief Introduction to Nonstandard Analysis

What *is* nonstandard analysis? In short, nonstandard analysis, a construction of analysis, constructs a new notion of numbers, including the notions of infinitesimal and infinitely large. In regards to this construction, do not confuse the phrase 'new notion of numbers' with 'new numbers.' Nonstandard analysis simply uses different notions of numbers to reason about concepts in analysis. Keisler offers an intuitive introduction to a freshman level calculus course using nonstandard analysis. The Keisler text will serve as a basis for this brief introduction to nonstandard analysis [1]. "The calculus was originally developed using the intuitive concept of an infinitesimal, or an infinitely small number. But for the past one hundred years infinitesimals have been banished from the calculus course for reasons of mathematical rigor." The approach of nonstandard analysis and the concept of the infinitesimal, however, has its advantages. The most important advantage is that it is closer to the intuition which initially led to the development of calculus. As a result, the concepts of the derivative and integral are quite intuitive. The purpose of this information is not to defend this position, but rather to offer this alternative information in regards to standard analysis. For a history of nonstandard analysis, refer to [5, Ch. 1].

Why was nonstandard analysis chosen as the basis for reasoning about the real numbers in `ACL2(r)`? A typical proof in standard analysis uses quantifiers extensively, whereas nonstandard analysis does not rely as heavily on quantifiers. For example, consider the following definition of a limit in standard analysis. Let $p_1, p_2, p_3, ...,$ be a sequence of points in the metric space E. A point p in metric space E is called a limit of the sequence $p_1, p_2, p_3, ...,$ if, given any real number $\epsilon > 0$, there exists a positive integer N, such that distance between $p, p_n < \epsilon$ whenever $n > N$ [3, p. 92]. This

definition relies on the use of the quantifiers *for all* and *there exists*. Although ACL2 supports first-order quantifiers *for all* and *there exists*, with use of the defun-sk event, ACL2 currently does not reason about quantifiers well. Proof support for quantification is therefore limited in ACL2.

There are other reasons for the choice of nonstandard analysis, noted in [4, p. 3]. For example, if we consider the axiom of a least upper bound, namely, that every bounded set of real numbers has a least upper bound, one must reason about the an infinite set. ACL2 does not have a first-class notion of the concept *infinite set*. It is these reasons, among others, that led to the choice of nonstandard analysis as the framework on which to build ACL2(r).

2.1 The Number Line in Nonstandard Analysis: The Inclusion of Infinitesimals and Infinitely Large Numbers

This paper assumes the reader has knowledge of the construction of the real number line, familiar from previous courses in elementary algebra. The number line in nonstandard analysis is commonly referred to as the hyperreal line, denoted commonly as $\Re*$. The hyperreal number line is constructed such that it includes all the previously known real numbers, with also numbers that are infintesimal and numbers that are infinitely large. The hyperreal numbers infinitely close to zero are called infinitesimal numbers, or simply infinitesimals. Intuitively, imagine that the previously known real number line is stretched and new numbers fill the gaps. Infinitesimals are the new numbers between zero and the rest of the real numbers. For technical reasons, we consider zero to be infinitesimal as well. Infinitely large numbers are the reciprocals of nonzero infinitesimals. Zero is the only previously known real number that is an infinitesimal.

What *properties* hold for the hyperreal numbers? Naturally, we would like to see the same properties hold for the hyperreal numbers that hold for the previously known real number numbers. In fact, the previous properties that hold for the known real numbers also hold for the hyperreals as well. For example, the same algebraic properties hold. An example of such an algebraic property is as follows. If a, b are hyperreal numbers, so are $a + b$, ab, $a - b$. If a is a hyperreal number and a is not equal to zero, $1/a$ is a hyperreal number. Furthermore, the following properties also hold for the

hyperreal numbers, a, b, c, z, e.

DEFINITION. Commutativity.
$a + b = b + a$ and $ab = ba$

DEFINITION. Associativity.
$(a + b) + c = a + (b + c)$ and $(ab)c = a(bc)$

DEFINITION. Distributivity.
$a(b + c) = (ab) + (ac)$

DEFINITION. Closure.
$a + b$ and ab are both in $\Re*$

DEFINITION. Existence of Identity Element.
There exist elements z and e in $\Re*$ such that $a + z = a$ and $ae = a$

DEFINITION. Existence of Inverse Elements.
There exist elements $-a$ and a^{-1}, with $a \neq 0$, for every a such that $a+(-a) = z$ and $a(a^{-1}) = e$

This concept of transferring properties of the previously known real numbers to the hyperreal numbers is known as the *Transfer Principle.* Intuitively, it states that if a property is held by the previously known real number line, then this property extends to the hyperreal number line as well. The transfer priciple is beyond the scope of this thesis, a principle that requires an extensive digression into mathematical logic.

One relationship between the hyperreals and the previously known real numbers is denoted by the notion of the *standard part.*

DEFINITION. The *standard* part of a, where a is a hyperreal number, is the real number infinitely close to a. The *standard* part of a is denoted by $st(a)$.

This definition is only true when a is not infinitely large. Following from this definition, we have that if b is a hyperreal number,

(i) $st(b)$ is a real number.

(ii) $b = st(b) + \epsilon$, where ϵ is an infinitesimal.
(iii) If b is real, then $b = st(b)$.

In computing standard parts, a helpful tool is used. Before introducing this tool, we must first define the terms finite, positive infinite, and negative infinite. Let b be a hyperreal number. Then b is classified as

(i) *Finite* if b is between two real numbers.
(ii) *Positively infinite* if b is greater than every real number.
(iii) *Negatively infinite* if b is less than every real number.

The computation tool for the hyperreals can now be stated below.

DEFINITION. If a, b are finite hyperreals and n is a standard number, then

(i) $st(-a) = -st(a)$.
(ii) $st(a + b) = st(a) + st(b)$.
(iii) $st(a - b) = st(a) - st(b)$.
(iv) $st(ab) = st(a)st(b)$.
(v) If $st(b) \neq 0$, then $st(a/b) = st(a)/st(b)$.
(vi) $st(a^n) = (st(a))^n$.
(vii) If $a \geq 0$, then $st(\sqrt[n]{a}) = \sqrt[n]{(st(a))}$.
(viii) If $a \leq b$,then $st(a) \leq st(b)$.

In conclusion, the set of standard numbers is simply the previously known set of real numbers. That is, the set of standard numbers is denoted

$$\{x | x \epsilon \Re\}$$

The set of nonstandard numbers is the previously known set of real numbers, with the infinitesimals and infinitely large numbers included. That is, the set of nonstandard number is denoted

$$\{x | x \epsilon \, (\Re * - \Re)\}$$

One relationship between hyperreal numbers, *infinitely close*, is defined below.

DEFINITION. Two hyperreal numbers b and c are said to be infinitely close

to each other, if their difference $b - c$ is an infinitesimal.

The theorem below states reflexivity, symmetry, and transitivity in regards to the infinitely close notion.

THEOREM.
(i) a is infinitely close to a.
(ii) If a is infinitely close to b, then b is infinitely close to a.
(iii) If a is infinitely close to b and b is infinitely close to c, then a is infinitely close to c.

2.2 The Derivative

Before moving to the definition of the derivative, the definition of the slope in nonstandard analysis must be discussed. The definition of slope is as follows.

DEFINITION. S is said to be the slope of f at a if

$$S = st\left(\frac{f(a+\Delta x) - f(a)}{\Delta x}\right)$$

Intuitively, the slope, when it exists, is infinitely close to the ratio of the change in $f(x)$ to an infinitely small change in x. We can now define the derivative in nonstandard analysis.

DEFINITION. Let f be a real function of one variable. The derivative of f, denoted f' is the new function f' whose values at x is the slope of f at x. That is,

$$f'(x) = st\left(\frac{f(x+\Delta x) - f(x)}{\Delta x}\right)$$

whenever the slope exists. Intuitively, for any point a, the slope of f at a is the same as the derivative of f at a.

There are several foundational lemmas in regards to differentiation. These lemmas are known commonly as the Sum Rule, Product Rule, Constant Rule, Chain Rule, and Power Rule, which will be stated in section 4. The reason behind presenting the foundational lemmas in this paper is that the differences between nonstandard analysis and standard analysis lie mostly at the foundational level.

2.3 Continuity

"Intuitively, a curve $y = f(x)$ is continuous if it forms an unbroken line, that is , whenever x_1 is close to x_2, $f(x_1)$ is close to $f(x_2)$. To make this intuitive idea into a mathematical definition, we substitute *infinitely close* for *close*." [1, p. 145].

DEFINITION. f is said to be continuous at a point c if
(i) f is defined at c.
(ii) whenever x is infinitely close to c, $f(x)$ is infinitely close to $f(c)$.

There are several foundational lemmas in regards to continuity.

THEOREM. Suppose functions f and g are continuous at c.
(i) For any constant k, the function $k * f(x)$ is continuous at c.
(ii) $f(x) + g(x)$ is continuous at c.
(iii) $f(x) * g(x)$ is continuous at c.
(iv) $(f(x))^n$, where n is a natural number, is continuous at c.

3 Nonstandard Analysis Terminology Represented in `ACL2(r)`

In section 2, a brief, general introduction to nonstandard analysis was given. In this section, some of those notions will be discussed as they appear in `ACL2(r)`. The notion of a number as an infinitesimal is formalized with the `ACL2(r)` function `i-small`, that is, `(i-small x)` is the `ACL2(r)` way to assert that `x` is infinitesimal. A number x is i-small if and only if x is an infinitesimal. The notion of infinitely large is formalized with the `ACL2(r)` function `i-large`, that is, `(i-large x)` is the `ACL2(r)` way to assert that `x` is infinitely large. A number x is i-large if and only if x is not zero and $1/x$ is an infinitesimal. A number x is i-limited if and only if x is not i-large. That is, a number x is i-limited if and only if it is not infinitely large, denoted by `(i-limited x)`. Standard numbers are not infinitely large, as discussed earlier. Therefore, standard numbers are i-limited. The notion of infinitely close is formalized with the `ACL2(r)` function `i-close`, denoted by `(i-close x y)` is the `ACL2(r)` way to assert that `x` and `y` are i-close. Numbers x, y are i-close if and only if $x - y$ is an infinitesimal.

A relationship between the reals and hyperreals is denoted in ACL2(r) as follows. There exists an ACL2(r) function that maps i-limited numbers to standard numbers in the following way. For a given i-limited number x, the standard part of this number is defined to be the unique real number i-close to x. The function is denoted (standard-part x).

The theorem in the above section about the reflexivity, symmetry, and transitiviy of infinitely close numbers is represented in ACL2(r) as follows.

Reflexivity.

```
(defthm i-close-reflexive
  (implies (acl2-numberp x)
           (i-close x x)))
```

Symmetry.

```
(defthm i-close-symmetric
  (implies (i-close x y)
           (i-close y x))
  :hints (("Goal''"
           :use ((:theorem
                   (equal (+ (- x) y) (- (+ x (- y))))))
           :in-theory
              (disable DISTRIBUTIVITY-OF-MINUS-OVER-+))))
```

Transitivity.

```
(defthm i-close-transitive
  (implies (and (i-close x y)
                (i-close y z))
           (i-close x z))
  :hints (("Goal"
           :use ((:instance standard-part-of-plus
                            (x (- x y))
                            (y (- y z))))
                 :in-theory (disable standard-part-of-plus
                                     i-close-reflexive
                                     i-close-symmetric))))
```

3.1 Defining Generic Functions in ACL2(r): One Approach

In beginning the proof of the foundational lemmas, generic hyperreal functions must be defined. In order to define generic functions in ACL2(r), we used encapsulation of local functions. Encapsulation can be used to constrain some functions or to hide events. The functions dc-fn1 and dc-fn2 are hyperreal functions defined in ACL2(r), which take one argument and return one value. dc-fn-domain-p is used to define the domain over which the function is both defined and differentiable.

```
(encapsulate
```

;; Generic functions, dc-fn1 and dc-fn2.

```
((dc-fn1 (x) t)
 (dc-fn2 (x) t)
 (dc-fn1-deriv (x) t)
 (dc-fn2-deriv (x) t)
 (dc-fn-domain-p (x) t))
```

;; The function dc-fn-domain-p recognizes a standard interval of reals.

```
(local (defun dc-fn-domain-p (x) (realp x))))

(defthm dc-fn-domain-standard
  (implies (dc-fn-domain-p x)
           (dc-fn-domain-p (standard-part x))))

(defthm dc-fn-domain-real
  (implies (dc-fn-domain-p x)
           (realp x)))

(defthm dc-fn-domain-is-interval
  (implies (and (dc-fn-domain-p l)
                (dc-fn-domain-p h)
                (realp x)
                (<= l x)
                (<= x h))
           (dc-fn-domain-p x)))
```

;; fn1 and fn-1-deriv are standard real-valued functions,
;; and fn1-deriv is the derivative of fn1

```
(local (defun dc-fn1 (x) x))
(local (defun dc-fn1-deriv (x) (declare (ignore x)) 1))
```

For the purposes of ACL2(r), we must know that the function returns real values for real arguments and standard values for standard arguments. This basically was a result of bridging the gap between ACL2 and ACL2(r). The reason for this is beyond the scope of this thesis. The theorem dc-fn-domain-standard states that if x is a member of the domain upon which f_1 is defined and differentiable, then the standard part of x is in the domain as well. The theorem dc-fn-domain-real states that if x is a member of the domain upon which f_1 is defined and differentiable, then x is a hyperreal number.

```
(defthm dc-fn1-standard
  (implies (and (dc-fn-domain-p x)
                (standard-numberp x))
           (standard-numberp (dc-fn1 x))))

(defthm dc-fn1-deriv-standard
  (implies (and (dc-fn-domain-p x)
                (standard-numberp x))
           (standard-numberp (dc-fn1-deriv x))))

(defthm dc-fn1-real
  (implies (dc-fn-domain-p x)
           (realp (dc-fn1 x))))

(defthm dc-fn1-deriv-real
  (implies (dc-fn-domain-p x)
           (realp (dc-fn1-deriv x))))
```

;; similar definitions and constraints for fn2...

3.2 The Derivative in ACL2(r)

We can now define the derivative in ACL2(r), which states that if x is a standard number, and x, y are in the interval upon which the function and

its derivative are defined, that is, `dc-fn-domain`, and if x is infinitely close to y, with x, y not equal, then the slope of the function at that point is infinitely close to the derivative of f_1 at x.

```
(defthm dc-fn1-derivative
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (i-close x y)
                (not (= x y)))
           (i-close (/ (- (dc-fn1 x) (dc-fn1 y)) (- x y))
                    (dc-fn1-deriv x))))
```

3.3 Continuity in ACL2(r)

We can now define the concept of continuity in `ACL2(r)` as follows, stating that if x is a standard number, x, y are members of the domain upon which f_1 is defined and continuous, x, y are i-close, then $f_1(x)$ is i-close to $f_1(y)$.

```
(defthm cc-fn1-continuous
  (implies (and (standard-numberp x)
                (cc-fn-domain-p x)
                (i-close x y)
                (cc-fn-domain-p y))
           (i-close (cc-fn1 x) (cc-fn1 y))))
```

3.4 Two Basic Example Theorems in ACL2(r)

The following two important theorems are used in later proofs regarding continuity and differentiation. If x_1, x_2 are i-close, and y_1, y_2 are i-close, then the sum of x_1 and y_1 is i-close to the sum of x_2 and y_2. That is,

```
(defthm close-plus
  (implies (and (i-close x1 x2)
                (i-close y1 y2))
           (i-close (+ x1 y1) (+ x2 y2)))
  :hints (...))
```

If x_1, x_2 are i-close, and y_1, y_2 are i-close, and both x_1, y_1 are i-limited, then the product of x_1 and y_1 is i-close to the product of x_2 and y_2. That is,

```
(defthm close-times
  (implies (and (i-close x1 x2)
                (i-close y1 y2)
                (i-limited x1)
                (i-limited y1))
           (i-close (* x1 y1) (* x2 y2)))
  :hints (...))
```

A good, basic exercise in nonstandard analysis, as well as an exercise in `ACL2(r)`, is for the reader is to prove, by hand, the lemma `close-times`, then complete the proof in `ACL2(r)`. To begin the hand proof, one could assume that x_1, y_1 are not limited, that is, that x_1, y_1 are infinitely large, and show that the product of x_1, y_1 is not infinitely close to the product of x_2, y_2. The `ACL2(r)` proof of `close-times` is discussed in section 4.2.

4 Differentiation

At this point, the reader has been introduced to many concepts in nonstandard analysis, as well as their formalization in `ACL2(r)`, without discussion of the hints and tools needed in `ACL2(r)` to complete the proof. It is now that we introduce a subset of the hints, tools and lemmas needed to complete these proofs. The generic functions defined in section 3.1 can now be used here in the proof of the foundational lemmas.

There are several foundational lemmas in nonstandard analysis about the differentiation of functions. These lemmas are known as the Sum Rule, Product Rule, Constant Rule, Chain Rule, and Power Rule for differentiation. Below is the statement of each foundational lemma, along with its corresponding proof in `ACL2(r)`.

4.1 The Sum Rule

$$[f(x) + g(x)]' = f'(x) + g'(x)$$

The function `dc-fn1+fn2` is introduced, the statement of the sum of two generic functions `dc-fn1` and `dc-fn2`, followed by the statement of the derivative of the sum of these functions. That is,

```
(defun dc-fn1+fn2 (x)
  (+ (dc-fn1 x) (dc-fn2 x)))

(defun dc-fn1+fn2-deriv (i n a x)
  (+ (dc-fn1-deriv x) (dc-fn2-deriv x)))
```

Suppose x is a standard number and y is i-close to x. From the theorem `dc-fn1-derivative` it follows that $\frac{(f_1(x)-f_1(y))}{(x-y)}$ is i-close to ${f_1}'(x)$. Similarly, $\frac{(f_2(x)-f_2(y))}{(x-y)}$ is i-close to ${f_2}'(x)$. Adding these two and simplifying yields the desired result. The key lemma is that when x_1 is i-close to x_2 and y_1 is i-close to y_2, $x_1 + y_1$ is i-close to $x_2 + y_2$, shown again below.

```
(defthm close-plus
  (implies (and (i-close x1 x2)
                (i-close y1 y2))
           (i-close (+ x1 y1) (+ x2 y2)))
  :hints (("Goal" :in-theory (enable i-close))))
```

The `in-theory` hint allows the rules in the `ACL2(r)` database regarding `i-close` to be enabled. *Why would something be disabled?* A rewrite rule fires whenever an instance of the conclusion is encountered. A `:use` hint adds, as a hypothesis, an instance of the lemma used. If the lemma is an enabled rewrite rule, it will "see" the added hypothesis and rewrite it to T. Intuitively, this results in the hypotheses being *cancelled out*, and `ACL2(r)` is unable to use this information to prove the theorem. For non-recursive functions such as `i-close`, it is necessary to disable definitions to prevent `ACL2(r)` from *opening them up*, missing out on all the lemmas we may know about that particular function. With the `close-plus` lemma now in the `ACL2(r)` database, `ACL2(r)` can prove the main result, the Sum Rule.

```
(defthm dc-fn1+fn2-derivative
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (i-close x y)
                (not (= x y)))
```

```
          (i-close (/ (- (dc-fn1+fn2 x) (dc-fn1+fn2 y))
                      (- x y))
                   (dc-fn1+fn2-deriv x)))
  :hints (("Goal"
           :in-theory (disable close-plus)
           :use (dc-fn1-derivative
                 dc-fn2-derivative
                (:instance close-plus
                           (x1 (/ (- (dc-fn1 x) (dc-fn1 y))
                                  (- x y)))
                           (x2 (dc-fn1-deriv x))
                           (y1 (/ (- (dc-fn2 x) (dc-fn2 y))
                                  (- x y)))
                           (y2 (dc-fn2-deriv x)))))))
```

4.2 The Product Rule

$$[f(x)g(x)]' = f(x)g'(x) + f'(x)g(x)$$

First the function `dc-fn1*fn2` is introduced, which is the statement of the product of generic functions `dc-fn1` and `dc-fn2`, followed by the statement of the derivative of `dc-fn1*fn2`. That is,

```
(defun dc-fn1*fn2 (x)
  (* (dc-fn1 x) (dc-fn2 x)))

(defun dc-fn1*fn2-deriv (x)
  (+ (* (dc-fn1 x) (dc-fn2-deriv x))
     (* (dc-fn1-deriv x) (dc-fn2 x))))
```

There are several lemmas needed in this proof, most of which are shown below. Presentation of these lemmas review more important concepts in nonstandard analysis, as well as expose the new user to simple examples of `ACL2(r)` tools used in the proofs. The lemma `i-limited-plus-lemma` states that if x,y are standard numbers, and *x-eps*, *y-eps* are i-small, then the sum of x, *x-eps*, y, *y-eps* is i-limited. This lemma uses the lemma `standard+small->i-limited`, not shown, which states that the sum of a

standard number and an i-small number are added, the result is i-limited. In this lemma, a hint is given to `ACL2(r)` to complete the proof. That is, `ACL2(r)` is told to use an instance of `standard+small->i-limited`, by instantiating `x`, and `eps` in `standard+small->i-limited`, with `(+ x y)` and `(+ x-eps y-eps)`, respectively. The `instance hint` simply means that `ACL2(r)` is told to use the lemma `standard+small->i-limited` with these new values. The new rule after instantiation will be used in the proof of `i-limited-plus-lemma`.

```
(defthm i-limited-plus-lemma
   (implies
       (and (standard-numberp x)
            (i-small x-eps)
            (standard-numberp y)
            (i-small y-eps))
       (i-limited (+ x
                     x-eps
                     y
                     y-eps)))
   :hints
        (("Goal"
            :use
           ((:instance
                standard+small->i-limited
                         (x (+ x y))
                         (eps (+ x-eps y-eps))))
            :in-theory
           (disable
                standard+small->i-limited)))))
```

The lemma `i-limited-plus` states that if x,y are i-limited, then their sum is i-limited. This lemma uses `i-limited-plus-lemma` by instantiating the variables `x`, `x-eps`, `y`, `y-eps`, with `(standard-part x)`, `(non-standard -part x)`, `(standard-part y)`, and `(nonstandard-part y)`, respectively.

```
(defthm i-limited-plus
  (implies (and (i-limited x)
                (i-limited y))
           (i-limited (+ x y)))
```

```
  :hints (("Goal"
           :use ((:instance
                     i-limited-plus-lemma
                        (x (standard-part x))
                        (x-eps (non-standard-part x))
                        (y (standard-part y))
                        (y-eps (non-standard-part y))))
           :in-theory (disable i-limited-plus-lemma))))
```

The lemma `small*limited->small` states that if x is i-small and y is i-limited, then the product of x,y is i-small. Here, the necessary fact that i-small numbers are also i-limited is given explicitly by the hint `:in-theory (enablesmall-are-limited)`.

```
(defthm small*limited->small
  (implies (and (i-small x)
                (i-limited y))
           (i-small (* x y)))
  :hints (("Goal" :in-theory (enable small-are-limited))))
```

Next, the lemma `i-close-limited` is shown, which is needed later for the proof of the lemma `close-times`. It states that if x is i-limited and x,y are i-close, then y is i-limited. The lemma `i-close-limited` clearly relies on `i-limited-plus`.

```
(defthm i-close-limited
  (implies (and (i-limited x)
                (i-close x y))
           (i-limited y))
  :hints (("Goal"
           :use ((:instance i-limited-plus
                            (x (- x y))
                            (y (- x))))
           :in-theory (disable i-large
                               i-small
                               i-limited-plus))))
```

The theorem `close-times-h` states that if x_1, x_2 are i-close and y is i-limited, then the product of x_1, y is i-close to x_2, y. The lemma close-times, discussed above, plays a role in the proof of the product rule twice, as will be explained below.

```
(defthm close-times-h
  (implies (and (i-close x1 x2)
                (i-limited y))
           (i-close (* x1 y) (* x2 y)))
  :hints (("Goal" :in-theory (enable i-close))
          ("Goal''" :use ((:instance small*limited->small
                                     (x (- x1 x2))
                                     (y y)))
           :in-theory (disable small*limited->small))))
```

ACL2(r) can now prove close-times, which states that if x_1, x_2 are i-close and y_1, y_2 are i-close, and x_1, y_1 are i-limited, then the product of x_1, y_1 is i-close to x_2, y_2, using an instance of the theorems close-times-h, i-close-transitive, and i-close-limited.

```
 (defthm close-times
   (implies (and (i-close x1 x2)
                 (i-close y1 y2)
                 (i-limited x1)
                 (i-limited y1))
            (i-close (* x1 y1) (* x2 y2)))
   :hints (("Goal"
            :use ((:instance close-times-h
                             (x1 x1)
                             (x2 x2)
                             (y y1))
                  (:instance close-times-h
                             (x1 y1)
                             (x2 y2)
                             (y x2))
                  (:instance i-close-transitive
                             (x (* x1 y1))
                             (y (* x2 y1))
                             (z (* x2 y2)))
                  (:instance i-close-limited
                             (x x1)
                             (y x2)))
            :in-theory (disable close-times-h
                                i-close-transitive
```

```
                                        i-close-limited))))
```

This next lemma begins the general proof of the product rule, as seen in nonstandard analysis. The fact that standards are limited is used to signify to `ACL2(r)` that `x` is limited. From the following set of hypotheses and the definition of the derivative of f_2, that is, f_2', $\frac{(f_2(x)-f_2(y))}{(x-y)}$ is i-close to $f_2{}'$, the first lemma can be proved. The set of hypotheses, used throughout the rest of the proof of the product rule, are x is a standard number, x, y are members of the domain upon which f_1 and f_2 are defined and differentiable, x is i-close to y, $x \neq y$, and $f_2(x)$ is i-close to $f_2(y)$. Since $\frac{(f_2(x)-f_2(y))}{(x-y)}$ is i-close to $f_2{}'$, $f_1(x)\frac{(f_2(x)-f_2(y))}{(x-y)}$ is i-close to $f_1(x)f_2{}'$. `Close- times` is used, with the instantiation listed below. This is possible because $f_1(x)$ is i-limited. Recall that standards are i-limited and x is a standard number in the hypothesis. $f_1(x)$ will return standard values for standard arguments.

At this time, it is important to point out that there is a *very* common mistake made by new users. Assume that one has proven a theorem previously which is now applicable in a current proof effort and is currently available in the `ACL2(r)` database. One cannot assume that `ACL2(r)` knows to use the theorem in the current proof effort. One may, at times, need to explicitly tell the theorem prover to use a certain theorem. This is evident in the need for the `:use` hint for `dc-fn1-derivative` and `dc-fn2-derivative` in `dc-fn1*fn2-derivative-lemma-1`, below.

```
(defthm dc-fn1*fn2-derivative-lemma-1
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (i-close x y)
                (not (= x y))
                (i-close (dc-fn2 x) (dc-fn2 y)))
           (i-close (* (dc-fn1 x)
                       (/ (- (dc-fn2 x) (dc-fn2 y)) (- x y)))
                    (* (dc-fn1 x)
                       (dc-fn2-deriv x))))
  :hints (("Goal"
           :in-theory (enable-disable
                               (standards-are-limited)
                               (close-times))
```

```
          :use (dc-fn1-derivative
                dc-fn2-derivative
                (:instance close-times
                     (x1 (dc-fn1 x))
                     (x2 (dc-fn1 x))
                     (y1 (dc-fn2-deriv x))
                     (y2 (/ (- (dc-fn2 x)
                               (dc-fn2 y))
                            (- x y)))))))
```

Since $\frac{(f_1(x)-f_1(y))}{(x-y)}$ is i-close to $f_1{}'$, $f_2(y)\frac{(f_1(x)-f_1(y))}{(x-y)}$ is i-close to $f_2(y)f_1{}'$. The lemma `close-times` is used, with the instantiation listed below. This is possible because $f_2(y)$ is i-limited.

```
(defthm dc-fn1*fn2-derivative-lemma-2
  (implies
     (and (standard-numberp x)
          (dc-fn-domain-p x)
          (dc-fn-domain-p y)
          (i-close x y)
          (not (= x y))
          (i-close (dc-fn2 x)
                   (dc-fn2 y)))
     (i-close (* (dc-fn2 y)
                 (/ (- (dc-fn1 x)
                       (dc-fn1 y))
                    (- x y)))
              (* (dc-fn2 y)
                 (dc-fn1-deriv x))))
  :hints (("Goal"
           :in-theory
                (enable-disable
                   (standards-are-limited)
                   (close-times))
           :use (dc-fn1-derivative
                 dc-fn2-derivative
                 (:instance i-close-limited-lemma
                      (a (dc-fn2 x))
                      (b (dc-fn2 y)))
```

```
(:instance close-times
     (x1 (dc-fn2 y))
     (x2 (dc-fn2 y))
     (y1 (dc-fn1-deriv x))
     (y2 (/ (- (dc-fn1 x)
               (dc-fn1 y))
            (- x y)))))))
```

In the final lemma needed for the proof, the lemma `close-plus` is used, as shown above, as well as `dc-fn1*fn2-derivative-lemma-1` and `dc-fn1*fn2-derivative-lemma-2` above. It is known now from these two lemmas that $f_1(x)\frac{(f_2(x)-f_2(y))}{(x-y)}$ is i-close to $f_1(x){f_2}'$ and that $f_2(x)\frac{(f_1(x)-f_1(y))}{(x-y)}$ is i-close to $f_2(x){f_1}'$. That

$$f_1(x)\frac{(f_2(x)-f_2(y))}{(x-y)}+f_2(x)\frac{(f_1(x)-f_1(y))}{(x-y)}$$

is i-close to $f_1(x){f_2}' + f_2(x){f_1}'$ follows from the lemma `close-plus`. Furthermore, because x is i-close to y, it is known that

$$f_1(x)\frac{(f_2(x)-f_2(y))}{(x-y)}+f_2(y)\frac{(f_1(x)-f_1(y))}{(x-y)}$$

is i-close to $f_1(x){f_2}' + f_2(x){f_1}'$. The result is `dc-fn1*fn2-derivative-lemma-3`.

```
(defthm dc-fn1*fn2-derivative-lemma-3
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (i-close x y)
                (not (= x y))
                (i-close (dc-fn2 x) (dc-fn2 y)))
           (i-close (+ (* (dc-fn1 x)
                          (/ (- (dc-fn2 x) (dc-fn2 y))
                             (- x y)))
                       (* (dc-fn2 y)
                          (/ (- (dc-fn1 x) (dc-fn1 y))
                             (- x y))))
                    (+ (* (dc-fn1 x)
```

```
                        (dc-fn2-deriv x))
                     (* (dc-fn2 x)
                        (dc-fn1-deriv x)))))
  :hints (("Goal"
           :in-theory (enable-disable
                                (standards-are-limited)
                                (close-plus))
           :use (dc-fn1-derivative
                 dc-fn2-derivative
                 dc-fn1*fn2-derivative-lemma-1
                 dc-fn1*fn2-derivative-lemma-2
                (:instance close-plus
                     (x1 (* (dc-fn1 x)
                            (/ (- (dc-fn2 x) (dc-fn2 y))
                               (- x y))))
                     (x2 (* (dc-fn1 x)
                            (dc-fn2-deriv x)))
                     (y1 (* (dc-fn2 y)
                            (/ (- (dc-fn1 x) (dc-fn1 y))
                               (- x y))))
                     (y2 (* (dc-fn2 x)
                            (dc-fn1-deriv x))))))))
```

With the above theorems in the `ACL2(r)` database, the main theorem is proved. This is done with some algebraic manipulation by the theorem prover, with the main result, the Product Rule, that is, `dc-fn1*fn2--derivative`.

```
(defthm dc-fn1*fn2-derivative
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (i-close x y)
                (not (= x y)))
           (i-close (/ (- (dc-fn1*fn2 x) (dc-fn1*fn2 y))
                       (- x y))
                    (dc-fn1*fn2-deriv x)))
           :hints (("Goal"
                    :use (dc-fn1*fn2-derivative-lemma-3))))
```

4.3 The Constant Rule

$$[kf(x)]' = kf'(x)$$

First the function `dc-k*fn1` is introduced, which is the statement of a hyperreal constant multiplied by a generic function in `ACL2(r)`, followed by the statement of the derivative. That is,

```
(defun dc-k*fn1 (x)
  (* (dc-k) (dc-fn1 x)))

(defun dc-k*fn1-deriv (x)
  (* (dc-k) (dc-fn1-deriv x)))
```

At this point, functional instantiation failed, the reasoning explained in a different paper, Taylor's Theorem with Remainder. Therefore the lemma `close-times-k` was proven which states if x_1, x_2 are i-close, k is i-limited, and k is a hyperreal number, then the product of x_1, k is i-close to the product of x_2, k.

```
(defthm close-times-k
  (implies (and (i-close x1 x2)
                (i-limited k)
                (realp k))
           (i-close (* x1 k) (* x2 k)))
  :hints (("Goal"
           :in-theory (enable i-close))
          ("Goal''"
           :use ((:instance small*limited->small
                            (x (- x1 x2))
                            (y k)))
           :in-theory (disable small*limited->small))))
```

The main result can now be proved, the Constant Rule, as shown below, `dc-k*fn1-derivative`.

```
(defthm dc-k*fn1-derivative
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
```

```
                (dc-fn-domain-p y)
                (i-close x y)
                (not (= x y)))
           (i-close (/ (- (dc-k*fn1 x) (dc-k*fn1 y))
                       (- x y))
                    (dc-k*fn1-deriv x)))
  :hints (("Goal"
           :in-theory (disable close-times-k)
           :use (dc-fn1-derivative
                 dc-fn2-derivative
                (:instance close-times-k
                     (x1 (/ (- (dc-fn1 x) (dc-fn1 y))
                            (- x y)))
                     (x2 (dc-fn1-deriv x))
                     (k  (dc-k)))))))
```

4.4 The Chain Rule

$$[f \circ g(x)]' = f'(g(x))g'(x)$$

The reader now has enough background to prove the Chain Rule at this point. Where does one start? To be consistent with the ideas presented earlier, one might begin by defining a function that states the composition of functions as well as a function that states the derivative of the composition of functions. As before, the function `dc-fn1-o-fn2` is introduced, which is the statement of the composition of functions in `ACL2(r)`, followed by the statement of its derivative. That is,

```
(defun dc-fn1-o-fn2 (x)
  (dc-fn1 (dc-fn2 x)))

(defun dc-fn1-o-fn2-deriv (x)
  (* (dc-fn1-deriv (dc-fn2 x))
     (dc-fn2-deriv x)))
```

Next, we need to consider the hand proof of the chain rule. There are two cases. The first case is the case in which $f_2(x) = f_2(y)$, the second case is when $f_2(x) \neq f_2(y)$. The proof of the first case is as follows. Using the

lemma `close-times`, one can directly show that $\frac{(f_2(x)-f_2(y))}{(x-y)} f_1'(f_2(x))$ is i-close to $f_2(x)f_1'(f_2(x))$, which simplifies with algebraic manipulation to the result of case one of the chain rule, namely `dc-fn1-o-fn2-derivative-case-1`. `ACL2(r)` performs this algebraic manipulation without assistance from the user.

```
(defthm dc-fn1-o-fn2-derivative-case-1
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (dc-fn-domain-p (dc-fn2 x))
                (i-close x y)
                (= (dc-fn2 x) (dc-fn2 y))
                (not (= x y)))
           (i-close (/ (- (dc-fn1-o-fn2 x) (dc-fn1-o-fn2 y))
                       (- x y))
                    (dc-fn1-o-fn2-deriv x)))
  :hints (("Goal"
           :in-theory (enable-disable
                              (standards-are-limited)
                              (close-times))
           :use (dc-fn2-derivative
                (:instance close-times
                     (x1 (/ (- (dc-fn2 x) (dc-fn2 y))
                            (- x y)))
                     (x2 (dc-fn2-deriv x))
                     (y1 (dc-fn1-deriv (dc-fn2 x)))
                     (y2 (dc-fn1-deriv (dc-fn2 x)))))))))
```

The second case also uses the lemma `close-times`, with instantiations shown below. However, to simplify algebraically to the result needed, `ACL2(r)` needed some help with algebra, that is, `alg-help` and `alg-help1`. The lemma `alg-help` simply states that if a, b, c are hyperreal numbers, with $b, c \neq 0$, then $\frac{a}{b} = \frac{a}{c}\frac{c}{b}$. The lemma `alg-help1` states that if x, y are hyperreal numbers, then the term $x \neq y$ is equivalent to the statement $x - y = 0$.

```
(defthm alg-help
  (implies (and (realp a)
```

```
              (realp b)
              (realp c)
              (not (= c 0))
              (not (= b 0)))
         (equal (/ a b)
                (* (/ a c) (/ c b)))))


(defthm alg-help1
  (implies (and (realp x)
                (realp y))
           (equal (not (= x y))
                  (not (= (- x y) 0)))))
```

With these two results in the `ACL2(r)` database, and the `:use` hint `close-times`, case two can be proven, that is, `dc-fn1-o-fn2-derivative-case-2`. Since we are using the composition of functions, `x`, `y` in `dc-fn1-derivative` are instantiated with `(dc-fn2 x)` and `(dc-fn2 y)`, respectively. The exercise of the algebraic manipulations needed after instantiation of `close-times` with the shown values is left for the reader. This exercise is helpful in understanding the manipulation of formulas while maintaining certain relationships between values, such as the relationship i-close. The exercise is very similar to the detailed example of the Product Rule above.

```
(defthm dc-fn1-o-fn2-derivative-case-2
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (dc-fn-domain-p (dc-fn2 x))
                (dc-fn-domain-p (dc-fn2 y))
                (i-close x y)
                (not (= x y))
                (not (= (dc-fn2 x) (dc-fn2 y))))
           (i-close (/ (- (dc-fn1-o-fn2 x) (dc-fn1-o-fn2 y))
                       (- x y))
                    (dc-fn1-o-fn2-deriv x)))
  :hints (("Goal"
           :in-theory (enable-disable
```

```
                        (standards-are-limited)
                        (close-times
                         alg-help
                         dc-fn1-derivative))
           :use (dc-fn2-derivative
                 alg-help1
                 (:instance dc-fn1-derivative
                      (x (dc-fn2 x))
                      (y (dc-fn2 y)))
                 (:instance close-times
                      (x1 (dc-fn1-deriv (dc-fn2 x)))
                      (x2 (/ (- (dc-fn1-o-fn2 x)
                                (dc-fn1-o-fn2 y))
                             (- (dc-fn2 x)
                                (dc-fn2 y))))
                      (y1 (dc-fn2-deriv x))
                      (y2 (/ (- (dc-fn2 x)
                                (dc-fn2 y))
                             (- x y))))
                 (:instance alg-help
                      (a (- (dc-fn1-o-fn2 x)
                            (dc-fn1-o-fn2 y)))
                      (b (- x y))
                      (c (- (dc-fn2 x)
                            (dc-fn2 y)))))))
```

At this point, the main result can be proven, by combining both cases as shown below. The result is the Chain Rule in `ACL2(r)`, formalized in the theorem `dc-fn1-o-fn2-derivative`.

```
(defthm dc-fn1-o-fn2-derivative
  (implies (and (standard-numberp x)
                (dc-fn-domain-p x)
                (dc-fn-domain-p y)
                (dc-fn-domain-p (dc-fn2 x))
                (dc-fn-domain-p (dc-fn2 y))
                (i-close x y)
                (not (= x y)))
```

```
          (i-close (/ (- (dc-fn1-o-fn2 x) (dc-fn1-o-fn2 y))
                      (- x y))
                   (dc-fn1-o-fn2-deriv x)))
  :hints (("Goal"
           :use ((:instance dc-fn1-o-fn2-derivative-case-1)
                 (:instance dc-fn1-o-fn2-derivative-case-2))
           :in-theory nil)))
```

5 Continuity

The following work in regards to continuity presented here was not used directly for Taylor's Theorem. However, it presents important concepts in nonstandard analysis, as well as the presentation of their corresponding proof in `ACL2(r)`.

Redundantly, generic, continuous functions are defined, as well as their definition in regards to continuity in `ACL2(r)`.

```
((encapsulate

((cc-fn1 (x) t)
 (cc-fn2 (x) t)
 (cc-fn-domain-p (x) t))

;; Our witness continuous function is the identity function.

 (local (defun cc-fn1 (x) x))
 (local (defun cc-fn2 (x) x))
 (local (defun cc-fn-domain-p (x) (realp x)))

;; The function returns standard values for
;; standard arguments.

(defthm cc-fn1-standard
  (implies (and (cc-fn-domain-p x)
                (standard-numberp x))
           (standard-numberp (cc-fn1 x)))
  :rule-classes (:rewrite :type-prescription))
```

```
(defthm cc-fn2-standard
  (implies (and (cc-fn-domain-p x)
                (standard-numberp x))
           (standard-numberp (cc-fn2 x)))
  :rule-classes (:rewrite :type-prescription))

(defthm cc-fn-domain-standard
  (implies (cc-fn-domain-p x)
           (cc-fn-domain-p (standard-part x)))
  :rule-classes (:rewrite :type-prescription))

;; For real arguments, the function returns real values.

(defthm cc-fn-domain-real
  (implies (cc-fn-domain-p x)
           (realp x)))

(defthm cc-fn-domain-is-interval
  (implies (and (cc-fn-domain-p l)
                (cc-fn-domain-p h)
                (realp x)
                (<= l x)
                (<= x h))
           (cc-fn-domain-p x)))

(defthm cc-fn1-real
  (implies (cc-fn-domain-p x)
           (realp (cc-fn1 x)))
  :rule-classes (:rewrite :type-prescription))

(defthm cc-fn2-real
  (implies (cc-fn-domain-p x)
           (realp (cc-fn2 x)))
 :rule-classes (:rewrite :type-prescription))

 ;; If x is a standard real and y is a real close to x,
 ;; then fn1(x) is close to fn1(y).
```

```
(defthm cc-fn1-continuous
  (implies (and (standard-numberp x)
                (cc-fn-domain-p x)
                (i-close x y)
                (cc-fn-domain-p y))
           (i-close (cc-fn1 x) (cc-fn1 y))))

(defthm cc-fn2-continuous
  (implies (and (standard-numberp x)
                (cc-fn-domain-p x)
                (i-close x y)
                (cc-fn-domain-p y))
           (i-close (cc-fn2 x) (cc-fn2 y))))
```

The definition of the sum of two generic, continuous functions in `ACL2(r)`, that is, the sum of two continuous function is also continuous. The sum of continuous functions, `cc-fn1` and `cc-fn2` is defined by function `cc-fn1+fn2`, that is,

```
(defun cc-fn1+fn2 (x)
  (+ (cc-fn1 x) (cc-fn2 x)))
```

With the assumptions x is a standard number, x, y are members of the domain upon which f_1 and f_2 are defined and continuous, and x is i-close to y, then it follows that the sum of the functions applied to x is i-close to the sum of the functions applied to y. The lemma `close-plus` is again needed in this proof. Although it is not used explicitly with a hint, `ACL2(r)` does use it in the proof of `cc-fn1+fn2-continuous`, shown below.

```
(defthm cc-fn1+fn2-continuous
  (implies (and (standard-numberp x)
                (cc-fn-domain-p x)
                (i-close x y)
                (cc-fn-domain-p y))
           (i-close (cc-fn1+fn2 x) (cc-fn1+fn2 y)))
  :hints (("Goal"
           :in-theory
                (enable standards-are-limited))))
```

Analogously, the proof for the product of two functions is shown below. The lemma `close-times` is used by `ACL2(r)`, without the user providing the hint that `close-times` is needed.

```
(defun cc-fn1*fn2 (x)
  (* (cc-fn1 x) (cc-fn2 x)))

(defthm fn1*fn2-continuous
  (implies (and (standard-numberp x)
                (cc-fn-domain-p x)
                (i-close x y)
                (cc-fn-domain-p y))
           (i-close (cc-fn1*fn2 x) (cc-fn1*fn2 y)))
  :hints (("Goal"
           :in-theory
                (enable standards-are-limited))))
```

Intially, there was an attempt to functionally instantiate the above product rule for continuity as follows. `cc-fn1` would be instantiated with constant, hyperreal value, and `cc-fn2` would again be a generic continuous function. However, an obstacle was encountered, the same of which was mentioned in the proof of the Constant Rule in regards to differentiation. Again, this obstacle is discussed in a different paper, Taylor's Theorem with Remainder. Therefore, the lemma `close-times-k` was needed stating that if x_1, x_2 are i-close, k is a hyperreal and i-limited, then the product of k, x_1 is i-close to the product of k, x_2.

```
(defun k*cc-fn1 (k x)
  (* k (cc-fn1 x)))

(defthm close-times-k
  (implies (and (i-close x1 x2)
                (realp k)
                (i-limited k))
           (i-close (* k x1) (* k x2)))
  :hints (("Goal"
           :in-theory (enable i-close))
          ("Goal''"
```

```
          :use ((:instance small*limited->small
                    (x (- x1 x2))
                    (y k)))
          :in-theory
               (disable small*limited->small)))))
```

The main result can be proved, the product of hyperreal number k and $f_1(x)$ is i-close to the product of k and $f_1(y)$.

```
(defthm k*cc-fn1-continuous
  (implies (and (standard-numberp x)
                (cc-fn-domain-p x)
                (i-close x y)
                (cc-fn-domain-p y)
                (realp k)
                (i-limited k))
           (i-close (k*cc-fn1 k x) (k*cc-fn1 k y)))
  :hints (("Goal"
           :in-theory (enable standards-are-limited)))))
```

6 Taylor's Theorem with Remainder

The point of including this section of the paper is to provide the hand proof of Taylor's Theorem with Remainder, giving the reader insight as to why the foundational lemmas are needed in the proof.

Given a function f with n continuous derivatives on the interval $[a, b]$ and its $(n+1)$st derivative defined on (a, b), Taylor's formula with remainder provides a means for estimating $f(x)$ for an arbitrary $x \in [a, b]$ from the values of f and its derivatives at a. Specifically,

$$f(x) = f(a) + \sum_{i=1}^{n} \frac{f^{(i)}(a) \cdot (x-a)^i}{i!} + \frac{f^{(n+1)}(\beta)}{(n+1)!}(x-a)^{n+1},$$

where β is some point in the interval (a, b).

The proof, presented in [6] among others, follows the proof of the mean value theorem. First, a special function F is constructed, and then Rolle's lemma is applied to F to find a β for which $F'(\beta) = 0$. Rolle's Lemma states: If a function f is differentiable on interval [a,b], and if $f(a) = f(b)$, then there is some β, a member of [a,b], such that $f(\beta) = 0$.

Rolle's Theorem had been formalized in `ACL2(r)` prior to this research. Taylor's formula follows from solving $F'(\beta) = 0$ for $f(x)$.

The function F is defined differently for each point x in $[a, b]$. In the following discussion, let x be a specific, fixed point in $[a, b]$. Define F over the interval $[a, x]$ as follows:

$$F(t) = f(x) - f(t) - \sum_{i=1}^{n} \frac{f^{(i)}(t) \cdot (x-t)^i}{i!} + \frac{(x-t)^{n+1}}{(n+1)!} A,$$

where A is a constant that does not depend on t. The specific value of A is chosen to satisfy the criteria for Rolle's lemma. Specifically, before Rolle's lemma can be applied to F on $[a, x]$, we must show that F is differentiable, and that $F(a) = F(x) = 0$. Clearly $F(x) = 0$, since all the $(x-t)^i$ terms vanish, leaving only $f(x) - f(t)$ with $t = x$. To ensure $F(a) = 0$, it is only necessary to set $F(a) = 0$ in the expression above and solve for A. This gives

$$A = -\left\{ f(x) - f(a) - \sum_{i=1}^{n} \frac{f^{(i)}(a) \cdot (x-a)^i}{i!} \right\} \frac{(n+1)!}{(x-a)^{n+1}}.$$

Moreover, F is clearly differentiable since it is the sum of differentiable terms. Using Rolle's lemma, we can conclude that there is some $\beta \in (a, x)$ such that $F'(\beta) = 0$. But observe that the terms in $F'(t)$ neatly cancel out. That is, the derivative of $\sum_{i=1}^{n} \frac{f^{(i)}(t) \cdot (x-t)^i}{i!}$ simplifies to $-f'(t) + \frac{f^{(n+1)}(t)(x-t)^n}{n!}$. This means that $F'(\beta) = 0 = -\frac{f^{(n+1)}(\beta)(x-\beta)^n}{n!} - \frac{(x-\beta)^n}{n!} A$. Since $x \neq \beta$, the terms $\frac{(x-\beta)^n}{n!}$ can be factored and eliminated, leaving

$$\begin{aligned} 0 &= -f^{(n+1)}(\beta) - A \\ &= -f^{(n+1)}(\beta) + \left\{ f(x) - f(a) - \sum_{i=1}^{n} \frac{f^{(i)}(a) \cdot (x-a)^i}{i!} \right\} \frac{(n+1)!}{(x-a)^{n+1}}. \end{aligned}$$

Solving for $f(x)$ in the formula above results in Taylor's formula with remainder:

$$f(x) = f(a) + \sum_{i=1}^{n} \frac{f^{(i)}(a) \cdot (x-a)^i}{i!} + \frac{f^{(n+1)}(\beta)}{(n+1)!} (x-a)^{n+1}$$

.

The reader now has a helpful background, a general introduction to the paper, Taylor's Theorem with Remainder.

7 Further Information about the ACL2(r) System

To begin using ACL2(r), one needs some basic notions in nonstandard analysis, some of which have been presented here. A more comprehensive guide can be found in the Keisler text [1]. Another helpful guide written by Joel Tropp, is *Infinitesimals: History and Application* [5]. A complete resource for ACL2(r) is Dr. Ruben Gamboa's dissertation, *Mechanically Verifying Real-Valued Algorithms in ACL2* and the article *Nonstandard Analysis in* ACL2(r) [4] both currently available at *http://www.gamboas.org/-ruben/research*. An essential resource for any work with ACL2 is *Computer-Aided Reasoning: An Approach* by Kaufmann, Manolios, and Moore, currently available at *http://www.cs.utexas.edu/users/ moore/publications/acl2-books/OrderingInformation.html*.

The top-level book of theorems that should be included in ACL2(r) is nsa.lisp. If it is not already included, realp.lisp must be included as well. There are many theorems in nonstandard analysis proven is these books, such theorems about continuity, derivatives, trigonometric functions, expt, nsa, sqrt, binomial theorem, to indicate a few.

The home page of ACL2 is available at *http://www.cs.utexas.edu/users/-moore/acl2*, and current research involving ACL2 can be found at *http://-www.cs.utexas.edu/users/moore/acl2/admin/atp*.

Further examples of ACL2 use can be found in *Computer-Aided Reasoning: ACL2 Case Studies* by Kaufmann, Manolios, and Moore (eds.), which can also be found at *http://www.cs.utexas.edu/ users/moore/ publications/-acl2-books/OrderingInfor mation.html*.

References

[1] H.J. Keisler. Elementary Calculus. Prindle, Weber and Schmidt, Boston, 1976.

[2] M. Kaufmann, P. Manolios, and J S. Mooore. Computer-Aided Reasoning: An Approach. Kluwer Academic Press, 2000.

[3] M. Rosenlicht. Introduction to Analysis, Dover Publications, Inc., 1999.

[4] R. Gamboa and M. Kaufmann. Nonstandard Analysis in ACL2. Journey of Automated Reasoning, 27(4):323-351, November 2001.

[5] J. Tropp. Infinitesimals: History and Application. University of Texas, Undergraduate Honors Thesis, Department of Mathematics, 1999.

[6] W. Fulks. Advanced Calculus: An Introduction to Analysis. John Wiley & Sons, third edition, 1978.

[7] EE Times, May, 1987.