Axiomatic Events in ACL2(r): A Story of `defun`, `defun-std`, and `encapsulate`

Ruben Gamboa John Cowles Nadya Kuzmina
Computer Science Department
University of Wyoming
{ruben,cowles,nadya}@cs.uwyo.edu

November 8, 2004

Abstract

ACL2(r) is a variant of ACL2 that has support for reasoning about the real and complex numbers. It is based on the logic of non-standard analysis, axiomatized by Nelson as an extension of ZF set theory [7, 6]. ACL2(r) is described in [2, 3]. This paper lays out the logical foundations of ACL2(r).

1 Introduction

The logical foundations of ACL2 are presented in [5], where the key question of ACL2 is answered: What can we say about formulas proved by ACL2? In [5] this question is answered as follows:

> Every alleged theorem of an ACL2 session is in fact a theorem first-order derivable from the extension of the built-in logic (with induction) by the axiomatic events of that session.

In the context of an ACL2 "session" this is a very strong statement, because the axiomatic events (e.g., `defun` or `encapsulate` events) in a session may be undone. For example, local definitions in an ACL2 book are not passed onto a session that includes the book, but non-local theorems in the book are. From the perspective of the new session, it follows that the non-local theorems are first-order derivable from the non-local axiomatic events in the book.

ACL2(r) differs from ACL2 in the following ways:

- The Ground Zero theory (GZ) of ACL2(r) contains the predicates `realp` and `complexp` that recognize the real and complex numbers (including the irrationals), respectively. In addition, the arithmetic theory of ACL2 is

modified slightly to admit these new numbers (e.g., the `floor` of a real number is not defined using integer division).

- The Ground Zero theory (GZ) of ACL2(r) also contains the predicate `standard-numberp`, the unary function `standard-part`, and the constant (zero-arity function) `i-large-integer`. `Standard-numberp` is used to recognize a special class of numbers called the standard numbers. All numbers that can be determined uniquely without using the predicates `standard-numberp`, `standard-part`, or `i-large-integer` are standard; thus, 0, 1, π, e, $\sqrt{2}$ and so on are standard. `I-large-integer` is a positive integer that is non-standard; in fact, it is larger than all standard reals, and its multiplicative inverse is smaller than all positive standard reals. Real numbers, such as `i-large-integer`, that are larger in magnitude than all standard reals are called *i-large*, and their multiplicative inverses are called *i-small*. Numbers which are not i-large are called *i-limited*. Two numbers are said to be *i-close* if their difference is i-small. The function `standard-part`, when applied to an i-limited number, returns the unique standard number that is i-close to its argument. These new notions and results are taken directly from non-standard analysis.

- ACL2(r) classifies all function symbols as either *classical* or not. In GZ, the only non-classical symbols are `standard-numberp`, `standard-part`, and `i-large-integer`. When a new function is introduced with `defun`, it is considered to be classical if and only if all the functions used in the body are classical. When a function is introduced with `encapsulate` or `defchoose`, it is considered to be classical. The notion of classical is also extended to terms in the obvious way: a term is classical if and only if it contains only instances of classical functions.

- ACL2(r) allows new classical function symbols to be introduced using `defun-std`. When `defun-std` is used, the body of the definition need *not* be classical. The function introduced is accepted only if the body can be shown to produce standard values when all the arguments to the function are standard. Moreover, the function is defined explicitly by the body only when the arguments are standard. The function is only defined implicitly for other arguments.

- ACL2(r) does not allow the use of recursion to introduce non-classical functions with `defun`, nor does it allow the use of recursion with `defun-std`.

- ACL2(r) allows a theorem to be proved using `defthm-std`, in which case ACL2(r) assumes, in addition to any hypothesis of the theorem, that all free variables in the theorem take standard values. Note: The additional hypothesis are only used during the proof; the theorem that ACL2(r) has actually proved (and entered into ACL2(r)'s theorem database) is the original theorem. `Defthm-std` can only be used when the theorem to be proved contains only classical functions. The use of `defthm-std` is justified by the *transfer principle* of non-standard analysis.

- ACL2(r) limits the use of induction when it is used to prove theorems that use non-classical functions. In these cases, induction can be used only to show that the theorem holds for all standard values of the free variables in the theorem.

In [3] it is argued that the theory of ACL2(r) is consistent with respect to ACL2. The argument is essentially that an ACL2(r) theory can be viewed as a first-order theory inside Internal Set Theory (IST). The consistency of the theory of ACL2(r) follows, since IST is known to be a conservative extension of Zermelo-Fraenkel Set Theory (ZF) [6].

In this paper, we explore how definitional extensions work in the theory of ACL2(r). Our motivation is two-fold. First, we want to make a statement about formulas proved in ACL2(r) that is similar to what is proved in [5] for ACL2. This means that we want the theorems of ACL2(r) to be statements that are first-order derivable from the axiomatic events — which means that we have to state precisely what first-order axioms are introduced by the axiomatic events of ACL2(r). This stands in contrast with the presentation in [3], where it was assumed that ACL2(r) sessions were carried out in the full context of Internal Set Theory, not just first-order theory. Second, we want to extend ACL2(r) to remove some of the limitations encountered in [4], but it would be foolhardy to do so without having a solid foundation for ACL2(r) to start from.

This paper is structured as follows. In section 2 we present the outline of the ACL2(r) story. This story follows the presentation in [5] very closely. In section 3 we show how the `defun` event is used to introduce new classical functions in ACL2(r). This is followed in section 4 by a discussion of the use of `defun` to introduce non-classical functions. Section 5 completes the story for the use of `defun-std` to introduce classical functions using non-classical terms. Details of `encapsulate` are covered in section 6. This paper omits a discussion of `defchoose`; we plan to deal with this in a subsequent version of this paper. We conclude in section 7 with a look towards future enhancements to ACL2(r).

2 Preliminaries

We are concerned in this paper with first-order *theories*: sets of first-order formulas that are closed under logical consequence. In the context of reasoning about ACL2 or ACL2(r), it is sufficient to restrict ourselves to first-order theories with equality and no other predicate symbols. For the remaining of this paper, when we refer to first-order theories it should be understood that we mean first-order theories with equality as their only predicate symbol. We assume that the reader is familiar with the following basic notions: The *language* of a first-order theory is the set of function symbols occurring in its formulas. A theory T_1 *extends* a theory T_2 if every theorem in T_2 is also a theorem in T_1. Moreover, T_1 *conservatively extends* T_2 if every theorem of T_1 in the language of T_2 is also a theorem of T_2.

Since we use the classical notion of logical consequence as our only inference scheme, the theories we consider must include axioms describing any other in-

ference rules, such as induction, or transfer. Now we consider axiom schemas that characterize the derived inference rules of ACL2(r).

The case for induction is straightforward. ACL2(r) contains the binary function symbol $\prec$, which (intuitively) represents a well-founded relation on the ACL2(r) universe[1] The induction axiom schema for classical formulas ϕ is given in [5]. We extend this axiom schema here to include non-classical formulas as well. Recall that in the context of Internal Set Theory induction on non-classical formulas only assures that the formula is true for standard values[2].

Definition. Let ϕ be a formula, let x be a free variable in ϕ, and let y be a variable not occurring in ϕ. Then the *induction axiom for ϕ with respect to x* is given by

$$\begin{array}{ll} (\forall x)(((\forall y \prec x)\phi[x := y]) \Rightarrow \phi) \Rightarrow (\forall x)\phi, & \text{if } \phi \text{ is classical} \\ (\forall x)(((\forall y \prec x)\phi[x := y]) \Rightarrow \phi) \Rightarrow (\forall x)(\mathit{standard}(x) \Rightarrow \phi), & \text{otherwise} \end{array}$$

A first-order theory T is said to be *closed with respect to induction* if it includes every induction axiom in the language of T. □

The transfer principle is also simple. We simply need to add a transfer axiom for every possible classical formula ϕ. Notice that we only add these axioms for classical formulae, since the transfer principle can only be used in these cases.

Definition. Let ϕ be a classical formula with free variables $x_1, \ldots, x_n$ and no other free variables. The *transfer axiom for ϕ* is as follows:

$$(\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} \mathit{standard}(x_i)) \Rightarrow \phi) \Rightarrow (\forall x_1 \ldots x_n)\phi$$

A first-order theory T is said to be *closed with respect to transfer* if it includes every transfer axiom in the language of T. □

There remains an inference rule needed to justify `defun-std`. The formal justification given in [3] for `defun-std` appeals to the Standardization Axiom of Internal Set Theory. This axiom, a weak version of the Specification Axiom of ZF, is as follows. Given a standard set S, and a formula (classical or not) $\phi(x)$ with free variable x

$$(\exists! S')(\mathit{standard}(S') \wedge (\forall x)(\mathit{standard}(x) \Rightarrow (x \in S' \Leftrightarrow (x \in S \wedge \phi(x)))))$$

This axiom can be used to justify the existence of a standard set that is the graph of the function (necessarily standard) defined by `defun-std`. But the problem is that this argument is in the language of set theory, and we are unwilling to change the underlying story of ACL2 in such a drastic way. At a minimum, it would require the existence of a set U containing all the objects in the ACL2 universe, and the other axioms of ACL2 would be relativized to this set. To

[1] A similar definition for the corresponding ACL2 ordering is given in [5].

[2] An alternative view is that the standard natural numbers correspond to the "old" set of natural numbers, and that the non-standard numbers are in fact *new* natural numbers beyond the original number line. In this view, induction on arbitrary formulas works only on the original natural numbers.

avoid this difficulty, we introduce the function symbol f_τ for each possible term τ a priori. These function symbols are disjoint from the set of function symbols that can be introduced by an ACL2(r) user.

Definition. Let L be a language. L *contains all its term functions* if for any term τ in the language of L, L contains a function symbol f_τ with arity n, where n is the number of free variables in τ. □

Definition. Let τ be a term with free variables $x_1, \ldots, x_n$ and no other free variables. The *standardization axiom for* τ is as follows:

$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow standard(\tau))) \Rightarrow$$
$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f_\tau(x_1, \ldots, x_n) = \tau))$$

The function symbols f_τ are said to be *non-visible*. All other function symbols are *visible*. A term or formula is said to be *visible* if it uses only visible function symbols; otherwise, it is said to be *non-visible*. A first-order theory T is said to be *closed with respect to standardization* if it includes all the standardization axioms in the language of T. □ Note: The preceding definition implies that a theory can only be closed with respect to standardization if its language contains all its term functions.

It is possible to start with a theory T that does not contain any standardization axioms and derive a theory T' that extends T and is closed with respect to standardization. The process is simply to introduce the symbols f_τ for each expression τ in T. The resulting theory, say T_1, is *not* closed with respect to standardization, because it does not contain the standardization axioms for formulas that use the function symbols that are in T_1 but not in T. But this process can be iterated to produce the desired theory.

The standardization axioms in a theory can be used to generate a partial orderings on the formulas τ of the theory. We say that $\tau_1 \ll \tau_2$ if f_{τ_1} appears in τ_2, and the implied ordering is the transitive closure of $\ll$. In general, the relation $\ll$ is not an ordering. However, the process suggested earlier that generates the standardization axioms for a given theory can be modified so that it generates a valid partial ordering $\ll$. For the remainder of this paper, when we say that a theory T is closed under standardization, we are also asserting that the ordering implied by the standardization axioms of T is a valid partial ordering.

Definition. A first-order theory T is said to be *closed* if it is closed with respect to induction, transfer, and standardization. The *closure* of the theory T is the theory resulting from extending T by the induction, transfer, and standardization axioms in the language of T. □

The basic story that we lay out in this paper is as follows. We start with a closed theory GZ of ACL2(r). Then we show that the axiomatic events of ACL2(r) — `defun`, `defun-std`, and `encapsulate` — conservatively extend a closed theory T into a closed theory T'. Moreover, in a closed theory the

derived inference rules of ACL2(r) — `defthm-std`, the non-standard principle of induction — are simply first-order consequences of the axioms.

Why is the Ground Zero theory of ACL2(r) closed? The answer comes from [6] and [3]. In [6] it is shown that Internal Set Theory is a conservative extension of ZF set theory. Moreover, in IST the predicate *standard* and derived predicates such as *standard-part* are given interpretations. IST restricts the ways in which non-classical terms (i.e., those defined in terms of *standard*) can be manipulated. For instance, it restricts the use of induction for non-classical formulas. But IST provides inference rules that justify the use of non-classical formulas, such as idealization, transfer, and standardization. As shown in [3] these have direct counterparts in the closed theory GZ.

In the remaining sections we discuss why axiomatic events in ACL2(r) extend a closed theory T conservatively into a closed theory T'.

3 `Defun`: Classical Functions

The story of `defun` is told definitively in [5]. There it is shown that `defun` events result in conservative extensions of an ACL2 theory. In this section, we modify the argument in [5] to ACL2(r) theories. Our intent has been to stick as closely as possible to the arguments in [5].

Definition. A *classical definitional axiom* D over a theory T is a finite conjunction of equations of the following form

$$f(x_1, \ldots, x_n) = term$$

where the function symbols f in the left-hand side of this axioms are distinct classical function symbols disjoint from the function symbols in T, $term$ is a classical term in the union of the language of T with the set of left-hand side function symbols of D, the variables x_i are distinct and these are the only variables free in $term$. □

Other than the restriction that $term$ be a classical term, this is entirely equivalent to the definition of a definitional axiom in [5]. Therefore any function that can be introduced into an ACL2 session using a definitional axiom of ACL2 can also be introduced into ACL2(r) using the classical definitional axiom.

For the remainder of this section fix a closed theory T and a definitional axiom D over T. Let F be the set of function symbols introduced by D, i.e., those in the left-hand side of equations of D.

Following [5] we define the canonical interpreter for D as follows: Suppose D contains the equation

$$f(x_1, \ldots, x_n) = term$$

and let d be a variable not in $term$. Replace this equation with the following

$$f'(d, x_1, \ldots, x_n) = if\ zp(d)\ then\ NIL\ else\ term_{d-1,1}$$

The formula $u_{d,b}$ is defined as follows:

- if u is a constant or variable, then $u_{d,b} = cons(u, NIL)$
- else if u is *if* t_0 *then* t_1 *else* t_2, $b = 1$, and $(t_0)_{d,0} = NIL$, then $u_{d,b} = NIL$
- else if u is *if* t_0 *then* t_1 *else* t_2, $b = 1$, and $car((t_0)_{d,0}) \neq NIL$, then $u_{d,b} = (t_1)_{d,1}$
- else if u is *if* t_0 *then* t_1 *else* t_2 and $b = 1$, then $u_{d,b} = (t_2)_{d,1}$
- else if u is $f(t_1, \ldots, t_n)$ and at least one of $(t_i)_{d,0} = NIL$, then $u_{d,b} = NIL$
- else if u is $f(t_1, \ldots, t_n)$ where $f \notin F$, then $u_{d,b} = cons(f(car((t_1)_{d,0}), \ldots, car((t_n)_{d,0})), NIL)$
- else u must be $f(t_1, \ldots, t_n)$ for some $f \in F$, and $u_{d,b} = f'(d, car((t_1)_{d,0}), \ldots, car((t_n)_{d,0}))$

The symbols f' can be thought of as new function symbols, not in the language of T or F. However, we can define them instead as expressions in the language of T. The idea is as follows. It is clear that the functions f' terminate: the variable d serves to limit the number of times a term involving f' is "opened" and all other branches through the definition of $u_{d,b}$ dive into a subterm of u. A computation of f' can be thought of as a sequence of equalities, e.g., the sequence produced by expanding the leftmost term into its definition: $f'(d, t_1, \ldots, t_n) = u$ if and only if there is a sequence of terms starting with $f'(d, t_1, \ldots, t_n)$ and ending with u such that each element (other than the first) of the sequence follows from the previous one by the expansion of its leftmost term. This condition can be stated as a first-order formula in the language of T; in other words, the f' are first-order definable in T. From now on, when we say f' what we mean is this first-order definition in the language of T, so that in fact the f' are *not* new function symbols.

As in [5], we are interested only in definitions D so that for each formula

$$f(x_1, \ldots, x_n) = term$$

in D, it is a theorem of T that $(\forall x_1 \ldots x_n)(\exists d)(f'(d, x_1, \ldots, x_n) \neq NIL)$. Such definitions are called *interpreter admissible*, and [5] shows that definitions which satisfy the measure-oriented admissibility criterion of Nqthm and ACL2 are also interpreter admissible.

Definition. Let T be a closed theory and D an interpreter admissible definitional axiom over T. Then T^D is the extension of T by the universal closures of the following equations, one for each f defined in D:

$$f(x_1, \ldots, x_n) = \begin{cases} car(f'(d, x_1, \ldots, x_n)), & \text{where } d \text{ is the least such that} \\ & car(f'(d, x_1, \ldots, x_n)) \neq NIL \\ NIL, & \text{if there is no such } d \end{cases}$$

The theory T_D is the extension of T^D by all induction axioms in the language of T^D, i.e., T_D is the inductive closure of T^D. □

Observation. The theory T^D is a conservative extension of T. This follows because the new functions $f \in F$ are explicitly defined using only terms in the language of T (i.e., without recursion). Recall, in particular, that the f' are first-order definable in T. □

What remains to be seen, however, is that T_D is also a conservative extension of T. To demonstrate this, we prove that the induction axioms in the language of T^D are first-order derivable in T^D from the induction axioms in the language of T. Consequently, T_D is the same theory as T^D.

Lemma. Let T_2 be the extension of T_1 formed by explicit definitions of new function symbols in F. Then for every term τ in the language of T_2, there is a term τ' in the language of T_1 such that $\tau = \tau'$ is a theorem of T_2. Moreover, τ' is classical if τ is classical.

Proof. We proceed by induction on the terms τ of T_2. If τ is a variable or constant symbol, then τ is already in the language of T_1 (since we are extending the language of T_1 only by introducing new function symbols), and $\tau = \tau$ is certainly a theorem of T_2. Otherwise, τ is of the form $f(\tau_1, \ldots, \tau_n)$ for some terms τ_i. Using the induction hypotheses, there are terms τ_1', $\ldots$, τ_n' in the language of T_1 such that $\tau_i = \tau_i'$ is a theorem of T_2 for each i. Moreover, if τ is classical, each of the τ_i are classical, and so are each of the τ_i'. If $f \notin F$, then f must be in the language of T_1, in which case letting $\tau' = f(\tau_1', \ldots, \tau_n')$ it follows that $\tau = \tau'$ is a theorem of T_2, and clearly τ' is classical if τ is classical. Otherwise, f is one of the functions explicitly defined to extend T_1. I.e., there is a term τ_f in the language of T_2 such that $(\forall x_1 \ldots x_n)(f(x_1, \ldots, x_n) = \tau_f)$ is an axiom of T_2, and moreover the x_i are the only variables free in τ_f. But then letting $\tau' = \tau_f[x_i := \tau_i']$, we can conclude in T_2 that $\tau = \tau'$, and τ' is in the language of T_1. Moreover, if τ is classical f is a classical function, which means that τ_f is classical. So τ' is also classical. □

Lemma. Let T_2 be the extension of T_1 formed by explicit definitions of new function symbols in F. Then for every formula ϕ in the language of T_2, there is a formula ϕ' in the language of T_1 such that $\phi \Leftrightarrow \phi'$ is a theorem of T_2. Moreover, ϕ' is classical if ϕ is classical.

Proof. This is an simple extension of the previous lemma, proved using induction on the logical structure of ϕ. □

Theorem. Let T_1 be a theory that is closed with respect to induction and let T_2 be the extension of T_1 formed by explicit definitions of new function symbols in F. Then T_2 is closed with respect to induction.

Proof. We prove this by showing that each induction axiom over the language of T_2 is a theorem of T_2. Let ϕ be an induction axiom over the language of T_2. Recall that there are two types of induction axioms, depending on whether the underlying formula is classical or not. We consider each case separately. Suppose ϕ takes the following form, where ψ is classical:

$$(\forall x)(((\forall y \prec x)\psi[x := y]) \Rightarrow \psi) \Rightarrow (\forall x)\psi$$

From the previous lemma, we can find ψ' in the language of T_1 such that $\psi \Leftrightarrow \psi'$ is a theorem of T_2 and ψ' is also classical. Therefore, the following is an induction

axiom in T_1 and hence a theorem of T_2:

$$(\forall x)(((\forall y \prec x)\psi'[x := y]) \Rightarrow \psi') \Rightarrow (\forall x)\psi'$$

Since $\psi \Leftrightarrow \psi'$ is also a theorem of T_2, it trivially follows that

$$(\forall x)(((\forall y \prec x)\psi[x := y]) \Rightarrow \psi) \Rightarrow (\forall x)\psi$$

is a theorem of T_2.

A similar argument suffices to show that ϕ is a theorem of T_2 when ϕ is of the form

$$(\forall x)(((\forall y \prec x)\psi[x := y]) \Rightarrow \psi) \Rightarrow (\forall x)(standard(x) \Rightarrow \psi)$$

for a non-classical ψ. So we conclude that T_2 contains all induction axioms over its language; i.e., it is closed with respect to induction. □

Applying this theorem to T^D as defined above, we find that T^D is closed with respect to induction; i.e., $T^D = T_D$. So T_D is a conservative extension of T. The theory T_D is discussed extensively in [5]. The following is an important theorem proved in that paper:

Lemma. Let D be interpreter admissible over the theory T. Then each formula of D is a theorem of T_D. Moreover, T_D is a subtheory of the inductive closure of the extension of T by D. □

To complete the story, we must show that T^D is also closed with respect to transfer and standardization.

Theorem. Let T_1 be a theory that is closed with respect to transfer and let T_2 be the extension of T_1 formed by explicit definitions of new function symbols in F. Then T_2 is closed with respect to transfer.

Proof. Let ϕ be a transfer axiom over the language of T_2. Then ϕ has the form

$$(\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow \psi) \Rightarrow (\forall x_1 \ldots x_n)\psi$$

for some classical formula ψ in the language of T_2. There is a formula ψ' in the language of T_1 such that $\psi \Leftrightarrow \psi'$ is a theorem of T_2, and moreover ψ' is classical. That means that the transfer axiom for ψ' is a theorem of T_1:

$$(\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow \psi') \Rightarrow (\forall x_1 \ldots x_n)\psi'$$

But then ϕ is provable in T_2. □

Matters are not as straightforward in the case of standardization. The problem is that the standardization axiom requires a classical function symbol f_τ for each term τ in the language of T_2. But since the language of T_2 extends the language of T_1 (by the new function symbols in F), this means that we need some new symbols f_τ.

Theorem. Let T_1 be a theory that is closed with respect to standardization and let T_2 be the extension of T_1 formed by explicit definitions of new function

symbols in F. Then there is a conservative extension ${}^*T_2$ of T_2 that is closed with respect to standardization. We call ${}^*T_2$ the *standardization closure of* T_2 with respect to T_1.

Proof. Since T_2 is an extension of T_1, it contains all the standardization axioms for terms τ in the language of T_1. We will now extend T_2 by introducing a new function symbol f_τ for each term τ in the language of T_2 that is not in the language of T_1. Let τ be such a term, and let $x_1, \ldots, x_n$ be the free variables in τ. By a previous lemma, there is a term τ' in the language of T_1 such that T_2 proves $\tau = \tau'$. Since T_1 is closed with respect to standardization, the standardization axiom for τ' is a theorem of T_1 and therefore of T_2:

$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow standard(\tau'))) \Rightarrow$$
$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f_{\tau'}(x_1, \ldots, x_n) = \tau'))$$

Now we extend T_2 by defining $f_\tau(x_1, \ldots, x_n) = f_{\tau'}(x_1, \ldots, x_n)$; call the resulting theory ${}^1T_2$. It immediately follows that the following is a theorem of ${}^1T_2$:

$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow standard(\tau))) \Rightarrow$$
$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f_\tau(x_1, \ldots, x_n) = \tau))$$

And f_τ is classical, since it is defined in terms of a classical function, i.e., $f_{\tau'}$. So the standardization axiom for τ is a theorem of ${}^1T_2$. Notice that this is a conservative extension, since the new function f_τ is defined explicitly in terms of $f_{\tau'}$.

Then we extend ${}^1T_2$ by considering a τ_2 that is in the language of ${}^1T_2$ but not in the language of T_1. Since the set of (finite) terms τ over a countable language is countable, we can arrange the order in which the terms are considered such that after a countable number of extensions the resulting theory is closed with respect to standardization; i.e., given any term τ in the language, we are guaranteed to add the standardization axiom for τ after a finite number of extensions. Call the resulting theory ${}^*T_2$. It is a conservative extension of T_2 that satisfies the requirements of the theorem. □

Definition. Let T be a closed theory and D an interpreter admissible definitional axiom over T. The *closed extension of* T *by* D is the theory ${}^*T^D$ which is formed as follows: (1) Extend T by D, (2) take the standardization closure of the resulting theory, (3) take the inductive closure of the resulting theory, and finally (4) take the transfer closure of the resulting theory. □

When a definitional axiom D is introduced into the ACL2(r) theory T, the ACL2(r) theory of the session is extended to T' the closed extension of T by D. As in [5], T' is a subtheory of the (standardization, inductive, and transfer) closure of T_D. Moreover, it is clear that the ACL2(r) event of `defthm-std` is

justified in T', since T' is closed with respect to transfer. Similarly, the classical and non-classical induction principles of ACL2(r) follow from the fact that T' is closed with respect to induction. In section 5, we explore the justification for `defun-std`.

4 `Defun`: Non-Classical Functions

ACL2(r) allows the user to define non-classical functions using `defun`, simply by providing a non-classical definition in the body. However, the introduction of such functions is limited in ACL2(r) only to non-recursive functions. That is, only non-recursive definitions can be used to introduce non-classical functions in ACL2(r). This is formalized in the following definition.

Definition. A *non-classical definitional axiom* D over a theory T is an equation of the following form

$$f(x_1, \ldots, x_n) = term$$

where the function symbol f is a non-classical function symbol not in the language of T, *term* is a non-classical term in the language of T (hence not including f), the variables x_i are distinct and these are the only variables free in *term*. □

Consider the extension T' of a closed theory T by adding the following non-classical definitional axiom D:

$$f(x_1, \ldots, x_n) = term$$

Since f is introduced by an explicit definition, T' is a conservative extension over T. Moreover, by the lemmas shown in the previous section, the induction, transfer, and standardization axioms of f are theorems of the new theory T', because they are equivalent in T' to the comparable axioms for *term* in T. This extension T' is the result of encountering such a `defun` event in an ACL2(r) session.

5 `Defun-std`

We now turn our attention to `defun-std`, which allows the introduction of a classical symbol from a non-classical body. Before such a definition is accepted, ACL2(r) checks that the body produces standard outputs when it is given standard inputs. This is meant to ensure the existence of the classical function introduced by this event.

Definition. A *classical definitional axiom D from a non-classical term* over a theory T is an equation of the following form

$$(\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f(x_1, \ldots, x_n) = term)$$

where the classical function symbol f is not in the language of T, *term* is a possibly non-classical term in the language of T such that *term* is provably (in

T) standard whenever all the x_i are standard, the variables x_i are distinct, and these are the only variables free in *term*. □

Consider a closed theory T and the following classical definitional axiom D from a non-classical term over T:

$$(\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f(x_1, \ldots, x_n) = term)$$

We will now show how to construct a theory T' that is a closed, conservative extension of T such that D is a theorem of T'. Since T is closed, the following is a theorem of T.

$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow standard(term))) \Rightarrow$$
$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f_{term}(x_1, \ldots, x_n) = term))$$

Note that x_1, ..., x_n are precisely the free variables of *term*. Moreover, notice that the hypothesis in this theorem can be discharged from the restrictions imposed on *term*, namely that it return standard values for standard values of its parameters. Now consider the following equation:

$$f(x_1, \ldots, x_n) = f_{term}(x_1, \ldots, x_n)$$

Since f_{term} is a classical function in the language of T, this equation actually comprises a classical definitional axiom D' over T. Therefore, the theory T can be extended conservatively into a closed theory T' such that D' is a theorem of T'. But then D is necessarily a theorem of T'.

For technical reasons we prefer to introduce the function f using an axiom over the visible language of T[3]. So consider again the following definitional axiom

$$(\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f(x_1, \ldots, x_n) = term)$$

We can safely assume that *term* is a term over the visible language of T. Otherwise, *term* must use a function symbol f_τ. We can remove f_τ simply by using `defun` to introduce the new (visible) function f' such that f' is equal to f_τ. Let T'' be the closure with respect to transfer of the extension of T by this visible definitional axiom. We claim that this theory is precisely the theory T' defined above. The reason is that since f_τ and f are classical, we can use the transfer axiom to prove that f is equal to f_τ from the definitional axiom given above.

[3]This is what is actually done in the implementation of ACL2(r). The non-visible function symbols are never used directly in the implementation.

6 `Encapsulate` and Functional Instantiation

In this section, we consider how ACL2(r) works with `encapsulate` events and subsequent functional instantiations.

The story of `encapsulate` itself is a simple one. Essentially, an encapsulate event lets the user introduce as an axiom a theorem about a given function without introducing a definitional axiom for the function. A careful argument given in [5] shows that this can be done conservatively. The gist of this argument is that the theory introduced by an encapsulate event is a subtheory of the one that would result if the functions introduced by that encapsulate were simply defined explicitly. Since we already know that explicitly defining a function results in a conservative extension, the (weaker) theory resulting from an encapsulate event is necessarily conservative.

The difficulty, however, lies with the correctness of functional instantiation, which can be used by ACL2 and ACL2(r) to prove theorems. In this paper we will discuss only simple cases of functional instantiations as defined below[4].

Definition. Let T be a theory. A *simple functional substitution* is a function over the function symbols of (the language of) T that preserves arity and classicalness. I.e., it maps classical function symbols to classical function symbols, non-classical function symbols to non-classical function symbols, unary function symbols to unary function symbols, binary function symbols to binary function symbols, etc. Moreover, a simple functional substitution is required to map each function in the Ground Zero theory of ACL2(r) to itself. A simple functional substitution that maps each non-visible symbol of (the language of) T to itself and each visible symbol of T to a (possibly different) visible symbol of T is called a *visible simple functional substitution.* If X is a formula of T and fs is a simple functional substitution, the formula $X\backslash fs$ is the formula that results by substituting each functional instance in X with the function to which fs maps it. □

To see the validity of functional substitution as a proof rule, we can proceed as follows. Suppose that ϕ is a theorem of some closed theory T in ACL2(r), and let fs be a functional substitution over this theory. We know there is a proof of ϕ in T. Suppose that $A\backslash fs$ is a theorem of T for each axiom A used in this proof of ϕ. Then it follows that $\phi\backslash fs$ is a theorem of T.

The trick is to show that $A\backslash fs$ is a theorem of T for every axiom used in the proof. The reason this is difficult is that the axioms of T include induction, transfer, and standardization axioms. In ACL2(r) these axioms are never explicit; rather they are used implicitly in the implementation. So as a matter of practicality, we would like to avoid considering $A\backslash fs$ for any induction, transfer, or standardization axiom A.

To make this notion explicit, [5] introduces the notion of a labeled formula. For our purposes, we can think informally of the labeled formulas of a theory T as the set of axioms directly introduced by the user during the course of an

[4] More complex cases, e.g., involving lambda expressions, can also be handled, but this requires a change to ACL2(r).

ACL2(r) session that defined T. I.e., this includes the axioms that define or constrain new function symbols, but it excludes all the induction, transfer, and standardization axioms added automatically by ACL2(r) on the user's behalf. Observe that all labeled formulas are in the visible language of T. With this notion we are ready to prove the validity of simple functional instantiation.

The following technical lemma is proved in [5].

Lemma. Suppose that ϕ is a theorem of a given first-order theory T and that fs is a simple functional substitution whose domain is disjoint from the set of function symbols of T. Then $\phi\backslash fs$ is a theorem of T. □

This lemma makes a deceptively simple claim: If a theorem involves function symbols that are not mentioned in the axioms of the theory in which it is proved, then the meaning of those function symbols is irrelevant, so the functions they represent can be replaced with different functions. We use this lemma to prove the following theorem.

Theorem. Let T be a closed first-order theory, let fs be a visible simple functional substitution over the language of T, and let ϕ be a theorem of T such that ϕ uses only visible function symbols. Moreover, suppose that $A\backslash fs$ is a theorem of T for each labeled formula A in T. Then $\phi\backslash fs$ is a theorem of T.

Proof. Since ϕ is a theorem of T, there is some proof of ϕ in T. Fix one such proof. Let P be the conjunction of the axioms used in this proof of ϕ. Then $P \Rightarrow \phi$ is a theorem of a subtheory of T that does not contain any axioms about the function symbols in ϕ, e.g., the Ground Zero theory GZ.

We construct a simple functional substitution fs' that is an extension of fs as follows. Consider all standardization axioms A used in the fixed proof of T. Let $\tau_1, \ldots, \tau_m$ be the terms standardized by these axioms, such that τ_j is not less than τ_i according to the partial ordering implied by standardization when $i < j$. Then let $fs_0 = fs$, and define fs_i as the extension of fs_{i-1} that maps f_{τ_i} to $f_{\tau_i\backslash fs_{i-1}}$. The functional substitution fs' is equal to fs_m, i.e., the final extension.

Then the preceding lemma assures us that $(P \Rightarrow \phi)\backslash fs'$ is also a theorem of this subtheory. But that means that $(P\backslash fs' \Rightarrow \phi\backslash fs)$ is a theorem of this subtheory and hence also of T. We will complete the proof by showing that $P\backslash fs'$ is a theorem of A.

Consider each conjunct A of P, i.e., each axiom used in the fixed proof of ϕ. If A is a labeled formula of T, then by hypothesis $A\backslash fs$ is a theorem of T. Since labeled formulas are in the visible language of T, it follows that $A\backslash fs'$ is equal to $A\backslash fs$ and we're done.

If A is either an induction or a transfer axiom, then $A\backslash fs'$ is also an induction or transfer axiom. The reason is that $A\backslash fs'$ preserves the structure of A, changing only the function symbols. Since fs' also preserves classicalness, the formula $A\backslash fs'$ will be of the right type (e.g., classical or non-classical induction axiom as appropriate). And since T is closed, it follows that $A\backslash fs'$ is a theorem of T.

Finally, suppose A is a standardization axiom. Then A has the form

$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow standard(\tau))) \Rightarrow$$
$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f_\tau(x_1, \ldots, x_n) = \tau))$$

where τ is a term with free variables $x_1, \ldots x_n$ and f_τ is classical. So $A\backslash fs'$ has the following form

$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow standard(\tau\backslash fs'))) \Rightarrow$$
$$((\forall x_1 \ldots x_n)((\bigwedge_{i=1}^{n} standard(x_i)) \Rightarrow f_\tau(x_1, \ldots, x_n)\backslash fs' = \tau\backslash fs'))$$

There must be an i such that τ is equal to τ_i, one of the formulas used in the construction of fs'. Then $f_\tau(x_1, \ldots, x_n)\backslash fs'$ is equal to $f_{\tau_i\backslash fs_{i-1}}(x_1, \ldots, x_n)$. Because the τ_i are ordered according to the implied ordering imposed by standardization, τ_i can not contain any of the f_{τ_j} for $j \geq i$. What this means is that $f_{\tau_i\backslash fs_{i-1}}(x_1, \ldots, x_n)$ is equal to $f_{\tau_i\backslash fs'}(x_1, \ldots, x_n)$. Therefore, $A\backslash fs'$ has the form of a standardization axiom, and since T is closed $A\backslash fs'$ is a theorem of T. □

7 Looking to the Future

In this paper we have laid the logical foundations of ACL2(r). The net result is that we can state formally what it means to be a theorem of an ACL2(r) session.

In doing so, we were partly motivated by a desire to enhance ACL2(r). Some of its limitations were shown in [4], where restrictions placed by ACL2(r) on the use of functional instantiation made reasoning about differentiable functions very tedious. Since then, we have been considering some enhancements to ACL2(r) to make it a more practical theorem prover over the reals. In this section, we will outline some of the enhancements we have in mind.

First of all, ACL2(r) knows about the non-standard numbers, both reals and complex. Pointedly, the presentation in this paper did not single out the numbers as the only possible non-standard objects in the ACL2(r) universe. So it should be straightforward to include other non-standard objects in ACL2(r). It suffices to show that there is a model of the Ground Zero theory of ACL2(r) that contains non-standard objects. Such a model can be constructed by embedding an ACL2(r) universe in Internal Set Theory.

Second, we would like to make it easier to prove that certain terms are standard. Consider the object $f(t_1, \ldots, t_n)$. If f is classical and all the t_i are standard, it necessarily follows that $f(t_1, \ldots, t_n)$ is standard. However, this fact

can not be proved directly in the current version of ACL2(r). Making such a conclusion is justified by the theory we outlined here. Suppose, for a moment, that $f(t_1, \ldots, t_n)$ is in fact not standard. Then we have that

$$(\forall y)(standard(y) \Rightarrow y \neq f(t_1, \ldots, t_n))$$

But since $y \neq f(t_1, \ldots, t_n)$ is a classical formula, we can use this theorem and the corresponding transfer axiom to prove

$$(\forall y) y \neq f(t_1, \ldots, t_n)$$

This is an obvious contradiction. It follows, therefore, that $f(t_1, \ldots, t_n)$ must be standard.

Third, we want to make a finer distinction among the non-classical functions. There is a significant difference between the following non-classical functions:

- $f(x) = \mathit{if}\ \mathit{standard}(x)\ \mathit{then}\ 1\ \mathit{else}\ 0$
- $g(x) = x + \epsilon$

where ϵ is a fixed non-standard number. Informally, functions like g have close cousins in the classical world, e.g., the function $h(x) = x + 1$. On the other hand, functions like f simply have no close relative in the classical sense. In fact, functions like f violate the rules of classical functions, e.g., the class $\{x \in Nat \mid f(x) = 0\}$ is not a set. Functions like g are called *internal*. Non-classical functions that are not internal are called *external*. Syntactically, the forbidden predicate `standard` and functions `standard-numberp` and `standard-part` are still off-limits to internal expressions, but `i-large-integer` is not. In addition, we would like to allow `encapsulate` to introduce all three types of functions: classical, internal, and external.

The reason for doing this is that it is convenient sometimes to reason about all internal functions, for example. Consider the definition of derivative. We are all familiar with the usual limit-based definition used in classical analysis. A simpler but non-classical definition is used in [2]. In [4], it was necessary to reason about expressions such as the following: $\frac{d(x^n)}{dx}$. Using the chain rule, it is easy to prove the familiar result by induction on n. But since the definition of derivative we used is non-classical, we were only able to conclude that the $\frac{d(x^n)}{dx} = n \cdot x^{n-1}$ for *standard* values of n.

To prove the result for all values of n requires a different approach. The first step is to introduce the classical notion of derivative and to prove that it is equivalent to the one used in [4]. Now we can use induction on the classical version of derivative, and the result will follow. It turns out that there are many slightly different notions of derivative. Some of these are equivalent to each other. Others are equivalent only at standard points, and still others are equivalent only for classical or even internal functions. Some of these definitions are proved equivalent (under the appropriate circumstances) in [1]. But to do so in ACL2(r) we need to reason about arbitrary internal functions, or arbitrary

external functions. Hence, we plan to allow such functions to be introduced with `encapsulate`.

Finally, we want to allow the introduction of non-classical recursive functions. This is especially important in the context of ACL2(r), since recursion is often used to reason about bounded quantifiers. For example, consider the following theorem: "The sum of a standard number of standard numbers is standard." Not only can we not prove this in ACL2(r), we can not even state it. The reason is that the concept of a "list of standard numbers" is recursive. I.e., we would like to write something like the following:

```
(defun standard-list-p (lst)
  (if (endp lst)
      t
    (if (not (standard-p (length lst)))
        nil
      (and (standard-p (car list))
           (standard-list-p (cdr lst))))))

(defthm standard-sumlist
  (implies (standard-list-p lst)
           (standard-p (sumlist lst))))
```

However, the first definition is inadmissible. Introducing non-standard recursive functions presents a major challenge, however. For example, the following function illustrates the risks involved:

```
(defun standard-floor (n)
  (if (or (zp n) (standard-p n))
      (nfix n)
    (standard-floor (1- n))))
```

A naive admission of this function would uncover a standard integer such that its successor is non-standard — but no such number exists. We believe that a modification of interpreter admissibility can be used to accept such functions. A function is said to be interpreter admissible if we can prove that for every input, there is some integer d such that the canonical interpreter for the function terminates after d steps. For non-classical functions, we would further require that d be a standard integer. We are presently pursuing this and other possibilities.

Acknowledgments

The authors would like to thank Matt Kaufmann for his patience, promptness, and thoroughness in answering all questions related to [5]. It is simply stating the facts to say that this paper could not have been written without his help. We would also like to thank him for fruitful discussions that led to this paper in the first place.

References

[1] A.M. Ballantyne and W. W. Bledsoe. Automatic proofs of theorems in analysis using non-standard techniques. *Journal of the Association for Computing Machinery (JACM)*, 24(3):353–371, 1977.

[2] R. Gamboa. *Mechanically Verifying Real-Valued Algorithms in ACL2*. PhD thesis, The University of Texas at Austin, 1999.

[3] R. Gamboa and M. Kaufmann. Nonstandard analysis in ACL2. *Journal of Automated Reasoning*, 27(4):323–351, November 2001.

[4] R. Gamboa and B. Middleton. Taylor's formula with remainder. In *Proc of the Third International Workshop of the ACL2 Theorem Prover and its Applications (ACL2-2002)*, 2002.

[5] M. Kaufmann and J S. Moore. Structured theory development for a mechanized logic. *Journal of Automated Reasoning*, 26(2):161–203, 2001.

[6] E. Nelson. Internal set theory: A new approach to nonstandard analysis. *Bulletin of the American Mathematical Society*, 83:1165–1198, 1977.

[7] A. Robinson. *Non-Standard Analysis*. Princeton University Press, 1996.