Contributions to the Theory of Tail Recursive Functions

John Cowles

Ruben Gamboa

University of Wyoming

{cowles,ruben}@cs.uwyo.edu

SUMMARY
Part 1

Tail recursive definitional axioms have desirable properties:

- always consistent to add a tail recursive definitional axiom

  P. Manolios and J S. Moore. Partial functions in ACL2, **J. Automated Reasoning** 31 (2003), 107–127.

- existence of **unique** total function satisfying a tail recursive definitional axiom ensures the recursion always halts

- neither true about arbitrary recursive definitional axioms.

What is tail recursion?

A function is **tail recursive** if its definition is tail recursive.

The definition of a function `f` is **tail recursive** provided

- the *body* of the definition contains at least one recursive call to `f`
- each such recursive call to `f` is tail recursive.

Here is what it means for a recursive call to be tail recursive in a definition:

```
(defun f (x1 ... xn)
  body)
```

Assume *body* contains no macros or lambda applications:

- expand all macros in *body*
- reduce the lambda applications by β-reduction.

Think of the expanded *body* as an **expression tree**.

A recursive call of `f` in *body* is **tail recursive** just in case

1. the call to `f` is not on the test branch of any `if`.
2. On any branch containing the call to `f`, only `if` may appear above the call to `f`.

Example 1

```
(defun f (x)
  (if (f x)
      x
      x))
```

The recursive call is **not** tail recursive.

The call to `f` is on the test branch of `if`.

Example 2

```
(defun f (x)
  (if (zp x)
      1
      (* x
         (f (- x 1)))))
```

The recursive call is **not** tail recursive.

∗ appears above `f` in the expression tree

Example 3

```
(defun M91 (x)
  (declare
   (xargs :guard
          (integerp x)))
  (if (> x 100)
      (- x 10)
      (M91
       (M91 (+ x 11)))))
```

There are two recursive calls to `M91` in this *body*.

- The outer call in `(M91 (M91 (+ x 11)))` is tail recursive.

- The inner call `(M91 (+ x 11))` is **not** tail recursive.
  - ◇ The outer call to `M91` appears above this inner call in the expression tree.

Example 4

```
(defun 3x+1 (x)
  (declare
   (xargs :guard (natp x)))
  (if (<= x 1)
      x
      (if (evenp x)
          (3x+1 (/ x 2))
          (3x+1
           (+ (* 3 x) 1)))))
```

The two calls to `3x+1` in this *body* are both tail recursive.

Tail Recursive Functions

Let `test`, `base`, and `step` be unary functions.

Consider the following proposed tail recursive definition.

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

This recursive call to `f` is simple and explicitly given.

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

Possible to be explicit and very precise about the meanings of the following:

- A total function satisfies the defining tail recursion axiom for this definition.

- The tail recursion in this definition terminates for a given input.

- The tail recursion in this definition satisfies a measure conjecture.

Possible to state these concepts in ACL2.

Therefore proofs of the **theorems** given later can be mechanically verified using ACL2.

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

A total ACL2 function `f` **satisfies the defining tail recursion axiom** for this definition provided the following is true about every `x`.

```
(equal (f x)
       (if (test x)
           (base x)
           (f (step x))))
```

Pete and J's `defpun` paper shows that **there is always at least one total ACL2 function satisfying the defining tail recursion axiom for any such tail recursive definition**.

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

The tail recursion in this definition **terminates for a given** `x` provided the following holds

$$\exists \mathtt{n}(\mathtt{test}(\mathtt{step}^{\mathtt{n}}\,\mathtt{x})).$$

The tail recursion in this definition **always halts** provided the tail recursion terminates for all `x`.

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

The tail recursion in this definition **satisfies a measure conjecture** provided there is a well-founded binary relation `rel`, on the set of objects recognized by some predicate `mp`, and a measure `m` satisfying

```
 (and (mp (m x))
      (implies (not (test x))
               (rel (m (step x))
                    (m x))))
```

The binary relation `rel` is **well-founded** on the set of objects recognized by `mp` iff there is a `rel`-order-preserving function `fn` that embeds objects recognized by `mp` into ACL2's ordinals:

```
(and (implies (mp x)(O-p (fn x)))
     (implies (and (mp x)
                   (mp y)
                   (rel x y))
              (O< (fn x)(fn y))))
```

In ACL2 Version 2.9,

- `O-p` recognizes the ordinals up to epsilon-0

- `O<` is the well-founded less-than relation on those ordinals

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

Theorem 1 *The following are equivalent for any function with a tail recursive definition like that for* `f`*.*

1. *The recursion satisfies a nonnegative-integer-valued measure conjecture.*

2. *The recursion satisfies a measure conjecture.*

3. *The recursive defining axiom is satisfied by an unique total function.*

4. *The recursion always halts.*

3. *The recursive defining axiom is satisfied by an unique total function.*
4. *The recursion always halts.*

The equivalence *3* $\Leftrightarrow$ *4* suggests one way to show the famous "$3x + 1$" function always terminates on all natural number inputs:

It is sufficient to show the defining axiom

```
(equal (3x+1 x)
       (if (<= x 1)
           x
           (3x+1 (if (evenp x)
                     (/ x 2)
                     (+ (* 3 x) 1)))))
```

is satisfied by only one total function on the nonnegative integers.

The termination of this function on all nonnegative integer inputs remains an open problem.

How much of **Theorem 1** holds for recursive definitions that may **not** be tail recursive?

Proposition 1 *The following are equivalent for any function with a recursive definition.*

1. *The recursion satisfies a nonnegative-integer-valued measure conjecture.*

2. *The recursion satisfies a measure conjecture.*

4. *The recursion always halts.*

Proposition 2 *The following implications hold for any function with a recursive definition.*

Each of these

1. *The recursion satisfies a nonnegative-integer-valued measure conjecture.*

2. *The recursion satisfies a measure conjecture.*

4. *The recursion always halts.*

implies

3. *The recursive defining axiom is satisfied by an unique total function.*

Proposition 3 *The following implications* **could fail** *for any function with a recursive definition.*

3. *The recursive defining axiom is satisfied by an unique total function.*

implies each of these

1. *The recursion satisfies a nonnegative-integer-valued measure conjecture.*

2. *The recursion satisfies a measure conjecture.*

4. *The recursion always halts.*

Counter Example

The equation

```
(equal (f x)
       (if (f x)
           x
           x))
```

is satisfied by only one total function, namely the **identity function**,

but the recursion suggested by the equation does not terminate nor satisfy any measure conjecture.

SUMMARY
Part 2

```
(equal (f x)
       (if (test x)
           (base x)
           (f (step x))))
```

Theorem 2 *Let* `a` *and* `b` *be constants. Suppose that the only constraint on the function* `f` *that mentions* `f` *is the defining tail recursive axiom for* `f`*. If ACL2 can prove* `(equal (f a) b)`*, then ACL2 can also prove, that the recursion for* `f` *terminates on input* `a`*.*

This **Meta Theorem** has application to Tail Recursive Interpreters.

SUMMARY
Part 3

Obtain result about Knuth's generalization of McCarthy's 91 Function as a corollary of more general results about reflexive tail recursive functions.

Reflexive Tail Recursion:

```
(defun f (x)
  (if (test x)
      (base x)
      (f (step x))))
```

`(step x)` mentions `f`

Nested recursive calls are sometimes called **reflexive**.

ACL2 can verify the following two theorems.

Theorem 3 *Let* `c` *be a positive integer and let* `test`, `base`, *and* `step` *be total functions such that*

- ```
  (implies (test (base x))
           (test x))
  ```

- `base` *and* `step` *commute:*

  ```
  (equal (base (step x))
         (step (base x)))
  ```

- *either the recursion with respect to* $\mathtt{base}^{(-\mathtt{c}\ 1)} \circ \mathtt{step}$ *and* `test` *always halts OR it never halts when* `x` *satisfies* `(not (test x))`*:*

  $$[\forall \mathtt{x} \exists \mathtt{n} (\mathtt{test}([\mathtt{base}^{(-\mathtt{c}\ 1)} \circ \mathtt{step}]^{\mathtt{n}}\, \mathtt{x}))]$$

  OR

  $$[\forall \mathtt{x} \forall \mathtt{n} ((\mathtt{not}(\mathtt{test}\, \mathtt{x})) \Rightarrow (\mathtt{not}(\mathtt{test}([\mathtt{base}^{(-\mathtt{c}\ 1)} \circ \mathtt{step}]^{\mathtt{n}}\, \mathtt{x}))))]$$
  ```

Theorem 3 continued

Then there is a total function `f` *that satisfies both the reflexive tail recursive equation*

$$\texttt{(equal (f x)}$$
$$\texttt{(if (test x)}$$
$$\texttt{(base x)}$$
$$\texttt{(f}^{\mathtt{c}}\texttt{ (step x))))}$$

and the simpler tail recursive equation

$$\texttt{(equal (f x)}$$
$$\texttt{(if (test x)}$$
$$\texttt{(base x)}$$
$$\texttt{(f (base}^{(-\mathtt{c}\ 1)}\texttt{ (step x))))}$$

Theorem 4 *Let* `c` *be a positive integer and let* `f`, `test`, `base`, *and* `step` *be total functions such that*

- `f` *is reflexive tail recursive:*

```
(equal (f x)
       (if (test x)
           (base x)
           (f^c (step x))))
```

- ```
  (implies (test (base x))
           (test x))
  ```

- `base` *and* `step` *commute:*

```
(equal (base (step x))
       (step (base x)))
```

- *recursion with respect to* `step` *and* `test` *always halts:*
  $\forall x \exists n (\mathrm{test}(\mathrm{step}^n\, x))$
```

Theorem 4 continued

Then `f` *also satisfies the simpler tail recursive equation*

$$
\texttt{(equal (f x)}
$$

```
(equal (f x)
       (if (test x)
           (base x)
           (f (base^(-c 1) (step x)))))
```

Corollary 1 (Knuth) *Let* `c` *be a positive integer and let* $\mathtt{a}, \mathtt{b} > 0, \mathtt{d}$ *be real numbers.*

1. *There is a total function on the reals satisfying the reflexive tail recursive equation*

```
(equal (K x)
       (if (> x a)
           (- x b)
           (K^c (+ x d))))
```

2. *If* `(< (* (- c 1) b) d)` *then there is an unique function on the reals satisfying the above reflexive tail recursive equation for* `K`*.*

Corollary 2 *There is an unique function on the reals satisfying the reflexive tail recursive equation for McCarthy's 91 function,*

```
(equal (M91 x)
       (if (> x 100)
           (- x 10)
           (M91 (M91 (+ x 11)))))
```