A Datatype Manager for ACL2

A Work in Progress

J Strother Moore
February, 2011
The Idea

Make it easy for the user to declare and reason about a large number of list-structured datatypes

This is a work in progress
Aside

This work is morphing from a “datatype manager” to a “rule manager” for ACL2.
Type-Set

ACL2 keeps track of the type of a value by associating a bit-mask with it.

This mask is called a type set.

Each bit represents a set of ACL2 objects and the mask represents the set obtained by unioning together those sets.
ACL2 has fourteen disjoint primitive types
zero
positive integers
positive ratio
negative integers
negative ratio
complex rational
nil
t
other symbols
proper cons
improper cons
string
character
other
Supoose $x$ is known to be either a natural number or nil.

$$x . \ #B00000001000011$$

- zero
- positive integer
- nil

Consider $(\text{if } x \alpha \beta)$
Type sets go all the way back to the earliest Boyer-Moore prover and have been extraordinarily effective at keeping track of what we know.
Suppose \( x \) is known to be an integer and \( y \) is known to be a positive rational.

\[
\text{(if (equal x y)}
\quad \text{(and (integerp y) (< 0 y))}
\quad \text{(symbolp x))}
\]
Pros and Cons

Type-sets are great for encoding implications (and other propositional relations) between types

(IMPLIES (NATP X) (RATIONALP X))

(IMPLIES (AND (TRUE-LISTP X) (NOT (CONSP X))) (SYMBOLP X))
\texttt{(IMPLIES (BOOLEANP X) (NOT (STRINGP X)))}

Unfortunately, type-sets don’t code any structural properties of lists (other than \texttt{true-listp})
Basic Approach

Let’s keep type-set but use lemmas to extend the behavior of the system for composite types

My current work is entirely focused on propositional relationships between structural types
Related ACL2 Work

The ACL2 Sedan supports

```
(defun foo (oneof nil (cons all foo))
```

```
(defun
  (sexpr
    (oneof symbol
      (cons symbol sexpr-list))
  (sexpr-list
    (oneof nil
      (cons sexpr sexpr-list)))
```

13
(defunc read-file (fname dir)
  :input-contract (and (stringp fname)
                        (dirp dir))
  :output-contract (filep (read-file fname dir))
  ...)

The Language

\[ <\text{litconst}> := T \]
\[ | \ \text{NIL} \]
\[ | <\text{keyword}^*> \]
\[ | <\text{number}> \]
\[ | <\text{string}> \]
\[ | <\text{char}> \]

* Note: All keywords except :OR, :AND, :NOT, and :REC.
\[ \text{<type>} := \text{<litconst>} \]
| \text{<recog>} 
| (QUOTE \text{<any>}) 
| ? 
| (\text{<type> . <type>}) 
| (:AND \text{<type>} \ldots \text{<type>}) 
| (:OR \text{<type>} \ldots \text{<type>}) 
| (:NOT \text{<type>}) 
| (:REC \text{<type>}) 
| (:REC \text{name} \text{<type>}) 
| *
Examples

Booleans:
(:OR T NIL)

One of the symbols MON, WED, or FRI:
(:OR 'MON 'WED 'FRI)

True lists:
(:REC (:OR NIL (? . *))")
True list of NATPs:
( :REC ( :OR NIL (NATP . *)) )

Same as above, except named:

( :REC NAT-LISTP
  ( :OR NIL (NATP . *)) )

( :REC NAT-LISTP
  ( :OR NIL (NATP . NAT-LISTP)) )
Three field record:
(STRINGP BIT32P (:OR T NIL 'X))
Deficiency

Right now I do not support constructors or accessors!

Reasons:

(1) I am focused on the relationships between types

(2) Many Lisp programs are not disciplined in their use of constructors and accessors
Demo

This work is morphing from a “datatypemanager” to a “rule manager” for ACL2. This will become clearer toward the end of the demo.