McCarthy 91 function

The McCarthy 91 function is a recursive function, defined by computer scientist John McCarthy as a test case for formal verification within computer science.

The McCarthy 91 function is defined as

\[
M(n) = \begin{cases} 
  n - 10, & \text{if } n > 100 \\
  M(M(n + 11)), & \text{if } n \leq 100
\end{cases}
\]

The results of evaluating the function are given by \( M(n) = 91 \) for all integer arguments \( n \leq 101 \), and \( M(n) = n - 10 \) for \( n > 101 \).

History

The 91 function was introduced in papers published by Zohar Manna, Amir Pnueli and John McCarthy in 1970. These papers represented early developments towards the application of formal methods to program verification. The 91 function was chosen for having a complex recursion pattern (contrasted with simple patterns, such as defining \( f(n) \) by means of \( f(n - 1) \)). The example was popularized by Manna's book, *Mathematical Theory of Computation* (1974). As the field of Formal Methods advanced, this example appeared repetitively in the research literature. In particular, it is viewed as a "challenge problem" for automated program verification.

Often, it is easier to reason about non-recursive computation. As one of the examples used to demonstrate such reasoning, Manna's book includes a non-recursive algorithm that simulates the original (recursive) 91 function. Many of the papers that report an "automated verification" (or termination proof) of the 91 function only handle the non-recursive version.

A formal derivation of the non-recursive version from the recursive one was given in a 1980 article by Mitchell Wand, based on the use of continuations.

Examples

Example A:

\[
M(99) = M(M(110)) \quad \text{since } 99 \leq 100 \\
= M(110) \quad \text{since } 110 > 100 \\
= M(M(111)) \quad \text{since } 100 \leq 100 \\
= M(101) \quad \text{since } 111 > 100 \\
= 91 \quad \text{since } 101 > 100
\]

Example B:

\[
M(87) = M(M(88)) \\
= M(M(M(109))) \\
= M(M(99)) \\
= M(M(M(110))) \\
= M(M(100)) \\
= M(M(M(111))) \\
= M(M(101)) \\
= M(91) \\
= M(M(102)) \\
= M(92) \\
= M(M(103))
\]
= M(93)
..... Pattern continues
= M(99)
(same as example A)
= 91

Code
Here is how John McCarthy may have written this function in Lisp, the language he invented:

(defun mc91 (n)
  (cond ((<= n 100) (mc91 (mc91 (+ n 11))))
        (t (- n 10))))

Here is an implementation of the non-recursive algorithm in C:

int mccarthy(int n)
{
  for (int c = 1; c != 0; ) {
    if (n > 100) {
      n = n - 10;
      c--;
    } else {
      n = n + 11;
      c++;
    }
  }
  return n;
}

Proof
Here is a proof that the function behaves as expected.

For 90 ≤ n < 101,

\[ M(n) = M(M(n + 11)) \]
\[ = M(n + 11 - 10), \text{ where } n + 11 \geq 101 \text{ since } n \geq 90 \]
\[ = M(n + 1) \]

So \( M(n) = 91 \) for 90 ≤ n < 101.

We can use this as a base case for induction on blocks of 11 numbers, like so:

Assume that \( M(n) = 91 \) for \( a \leq n < a + 11 \).

Then, for any \( n \) such that \( a - 11 \leq n < a \),

\[ M(n) = M(M(n + 11)) \]
\[ = M(91), \text{ by hypothesis, since } a \leq n + 11 < a + 11 \]
\[ = 91, \text{ by the base case.} \]

Now by induction \( M(n) = 91 \) for any \( n \) in such a block. There are no holes between the blocks, so \( M(n) = 91 \) for \( n < 101 \). We can also add \( n = 101 \) as a special case.
**Knuth's generalization**

Donald Knuth generalized the 91 function to include additional parameters. John Cowles developed a formal proof that Knuth's generalized function was total, using the ACL2 theorem prover.

**References**


**References**

Article Sources and Contributors


License

Creative Commons Attribution-Share Alike 3.0 Unported
http://creativecommons.org/licenses/by-sa/3.0/