BEYOND LISP

John McCarthy, Stanford University

Stanford June 22, 2005

http://www-formal.stanford.edu/jmc/

jmc@cs.stanford.edu

MAIN POINTS

- Lisp programs are Lisp data—abstract syntax.

Programming languages need functions for their abstract syntax.

- English is important for its semantics—not its syntax
- The largest piece of cake—Kleene μ operator
- An elephant never forgets and is faithful.
- Resolution considered harmful.
- Special provers are just strategies—Davis-Putnam
- Programs as logical formulas—Algol 48 and Algol 50

PROGRAMS AS DATA—ABSTRACT SYNTAX

- Sums may be represented as (+ x y), x + y, $3^x 7^y$ as text.

- $issum(exp)$, $s1(exp)$, $s2(exp)$, $mksum(exp1, exp2)$

- Programming languages (even Java and Curl) need their abstract syntaxes defined and function in the language for the abstract syntax.

- Lisp is close to its abstract syntax, but needs it anyway.

- Input syntax, print syntax, compute with it syntax. $x+3$, $x+3$, (+ x 3). *I should have given the list structure.*

- $issubroutine(exp)$, $body(exp)$, $isjavaprogram(exp)$

- http://www-formal.stanford.edu/jmc/towards.html (1962)

ENGLISH AS A PROGRAMMING LANGUAGE—IT'S THE SEMANTICS THAT'S IMPORTANT

- COBOL: add 3 to x, FORTRAN: x = x + 3, Algol 60: x := x + 3; trivial variants

- "the largest piece of cake", Kleene μ operator

- "I need to be at a meeting in Monterrey, Mexico from November 15 to 17" followed by dialog about details.

- "The U.S. wants Iraq to become a non-aggressive, prosperous, democracy"—Plan government policy. [to be done without a precise definition of democracy].

INCLUDE LOGICAL SENTENCES IN LISP

• (assert ‘(on block1 ,x)), (assert ’((forall boy)((exists girl)((loves girl (only boy))))))

• Also include a theorem prover-problem solver

• Resolution considered harmful.

• Special provers are just tactics in general provers to be used on subproblems, e.g. Davis-Putnam used when only the outer propositional structure is considered. [confession: I can’t do this yet.

ELEPHANT

- Elephants never forget. An elephant is faithful, 100 percent.

- A passenger has a reservation if he has made one and it hasn't been cancelled. Passengers with reservations should be on the passenger list at flight time.

- A necessary condition for program correctness is that a program fulfill its promises.

- Alas, Elephant 2000 is not ready to be implemented.

- http://www-formal.stanford.edu/jmc/elephant.html

ALGOL 48

If we introduce time explicitly as distinct from the program counter, Algolic programs can be written as sets of equations. Here's an Algol 60 program for computing the product mn of two natural numbers.

$$
\begin{array}{rcl}
start: & & \\
i & := & n; \\
p & := & 0; \\
loop: & & if\ i = 0\ then\ go\ to\ done; \\
i & := & i - 1; \\
goto & & loop; \\
done: & &
\end{array}
$$

Here's what mathematicians might have written in 1948, before programming languages existed.

$$
\begin{aligned}
pc(0) &= 0; \\
i(t+1) &= \text{if } pc(t) = 0 \text{ then } n \\
&\text{else if } pc(t) = 4 \text{ then } i(t) - 1 \text{ else } i(t); \\
p(t+1) &= \text{if } pc(t) = 1 \text{ then } 0 \\
&\text{else if } pc(t) = 5 \text{ then } p(t) + m \text{ else } p(t) \\
pc(t+1) &= \text{if } pc(t) = 2 \wedge i(t) = 0 \text{ then } 6 \\
&\text{else if } pc(t) = 5 \text{ then } 2 \text{ else } pc(t) + 1;
\end{aligned}
\tag{1}
$$

The proof that $\exists t.(t \geq 0 \wedge pc(t) = 6 \wedge p(t) = mn)$ follows from the sentences expressing the program and the laws of arithmetic, i.e. no theory of program correctness is needed. However, the proof ideas are essentially the same as those used to prove that an algolic program terminates and that the outputs have the correct relation to the inputs. Amir Pnueli and Nissim Francez had this idea before I did, but they mistakenly abandoned it for temporal logic.

PROVING LISP PROGRAMS CORRECT