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 abstr

- English is important for its semantics—not its synt
- 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 5

PROGRAMS AS DATA—ABSTRACT SYNT

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

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

- Programming languages (even Java and Curl) nee
stract syntaxes defined and function in the language
stract syntax.

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

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

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

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

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

• COBOL: add 3 to x, FORTRAN: x = x + 3, Algo
x + 3; trivial variants

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

• "I need to be at a meeting in Monterrey, Mexico fro
ber 15 to 17" followed by dialog about details.

• "The U.S. wants Iraq to become a non-aggressive, p
democracy"—Plan government policy. [to be done
precise definition of democracy].

INCLUDE LOGICAL SENTENCES IN LISP

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

• Also include a theorem prover-problem solver

• Resolution considered harmful.

• Special provers are just tactics in general provers on subproblems, e.g. Davis-Putnam used when only propositional structure is considered. [confession: I ca yet.

ELEPHANT

- Elephants never forget. An elephant is faithful, 100

- A passenger has a reservation if he has made one ar
been cancelled. Passengers with reservations should
passenger list at flight time.

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

- Alas, Elephant 2000 is not ready to be implemente

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

ALGOL 48

If we introduce time explicitly as distinct from th
counter, Algolic programs can be written as sets of
Here's an Algol 60 program for computing the prod
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 19
programming languages existed.

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

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

PROVING LISP PROGRAMS CORRECT