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 $\mu$ 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
• Sums may be represented as \((+ \times y), x + y, 3^x7^y\) and so on.

• \textit{issum}(\textit{exp}), \textit{s1}(\textit{exp}), \textit{s2}(\textit{exp}), \textit{mksum}(\textit{exp}1, \textit{exp}2)

• Programming languages (even Java and Curl) need abstract syntaxes defined and function in the language itself through abstract syntax.

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

• Input syntax, print syntax, compute with it syntax. \((+ \times 3)\). \textit{I should have given the list structure.}

• \textit{issubroutine}(\textit{exp}), \textit{body}(\textit{exp}), \textit{isjavaprogram}(\textit{exp})

• \texttt{http://www-formal.stanford.edu/jmc/towards.html}
ENGLISH AS A PROGRAMMING LANGUAGE—IT’S SEMANTICS THAT’S IMPORTANT

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

• “the largest piece of cake”, Kleene $\mu$ 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, peaceful democracy”—Plan government policy. [to be done with precise definition of democracy].
• (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 that focus on subproblems, e.g. Davis-Putnam used when only propositional structure is considered. [confession: I can't say I'm sure yet.]
Elephants never forget. An elephant is faithful, 100

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

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

Alas, Elephant 2000 is not ready to be implemented.

http://www-formal.stanford.edu/jmc/elephant.html
If we introduce time explicitly as distinct from the counter, Algolic programs can be written as sets of

Here’s an Algol 60 program for computing the product of two natural numbers.

```
start:
    i := n;
    p := 0;
loop:    if i = 0 then go to done;
    i := i - 1;
    goto loop;
done:
```
Here’s what mathematicians might have written in 1948, as programming languages existed.

\[
\begin{align*}
pc(0) &= 0; \\
i(t + 1) &= \text{if } pc(t) = 4 \text{ then } i(t) - 1 \text{ else } i(t); \\
p(t + 1) &= \text{if } pc(t) = 5 \text{ then } p(t) + m \text{ else } p(t) \\
pc(t + 1) &= \text{if } pc(t) = 5 \text{ then } 2 \text{ else } pc(t) + 1;
\end{align*}
\]
The proof that $\exists t. (t \geq 0 \land pc(t) = 6 \land p(t) = mn)$ follows the sentences expressing the program and the laws of a program, i.e. no theory of program correctness is needed. How proof ideas are essentially the same as those used to show an algolic program terminates and that the outputs have a correct relation to the inputs. Amir Pnueli and Nissim had this idea before I did, but they mistakenly abandoned temporal logic.
PROVING LISP PROGRAMS CORRECT