Programs as Agents in First-Order Logic

Fangzhen Lin

The Voice: If you build it, he (they) will come.

— *Field of Dreams (1989)*

Current Research Interests

Broadly speaking, logic-based AI. Specifically:

- Answer Set Programming: a constraint-based problem solving paradigm using nonmonotonic logic programs, with real world applications in product design, bioinformatics, and robotics.
- High level robot design based on a formal theory of actions.
- Game theory and social choice theory:
  - A formulation of HK Legislative Council GC election.
  - Computer-aided theorem discovery in game theory.
  - Iterative game theory (iterative Prisoner's Dilemma).
- Computer programs as agents in first-order logic.

Computer Programs as Agents in First-Order Logic

- Programs are some of the most complex man-made systems.
- To understand these systems, we propose to treat them as agents with knowledge in first-order logic.
- As a first step, we will construct a translator from programming languages like C and Java to first-order logic.

Examples

The following program changes the value of X given the values of X and Y:

```
X = X+Y;
X = X+Y
```

If we use X and Y to denote the input values of X and Y, respectively, and X' and Y' the output values, then we have (X_1 and Y_1 are intermediate variables):

$$
\begin{aligned}
X_1 &= X + Y, \\
Y_1 &= Y, \\
X' &= X_1 + Y_1, \\
Y' &= Y_1.
\end{aligned}
$$

How about real programs, especially those with loops?

Examples

Consider the following while loop

```
while X < M do { X = f(X) }
```

What does it output? No effect on M, but for X, it depends on f:

$$
\begin{aligned}
&M' = M,\\
&X \geq M \to X' = X,\\
&X < M \to X' = X(N),\\
&X(0) = X,\\
&\forall n. X(n+1) = f(X(n)),\\
&X(N) \geq M,\\
&\forall n. n < N \to X(n) < M.
\end{aligned}
$$

N denotes the number of iterations that the loop runs until termination. $X(n)$ is the value of X after the nth iteration.

Properties during execution - use V^L to denote the value of V at label L:

```
1:    while I < N do
2:      if X < A(I) then
3:         X = A(I);
4:      I = I+1
```

The axioms for the body of the loop are (we ignore $A(x)$ and N as they do not change):

$$
\begin{aligned}
&X^4 = X^2 \wedge I^4 = I^2 + 1,\\
&X^2 = \text{ if } X < A(I) \text{ then } X^3 \text{ else } X,\\
&I^2 = \text{ if } X < A(I) \text{ then } I^3 \text{ else } I,\\
&X^3 = A(I) \wedge I^3 = I.
\end{aligned}
$$

Thus the axioms for the program are:

$$
\begin{aligned}
& X^1(0) = X \wedge I^1(0) = I, \\
& X^1(n+1) = X^4(n), \\
& I^1(n+1) = I^4(n), \\
& X^4(n) = X^2(n), \\
& I^4(n) = I^2(n) + 1, \\
& X^2(n) = \text{ if } X^1(n) < A(I^1(n)) \text{ then } X^3(n) \text{ else } X^1(n), \\
& I^2(n) = \text{ if } X^1(n) < A(I^1(n)) \text{ then } I^3(n) \text{ else } I^1(n), \\
& X^3(n) = A(I^1(n)), \\
& I^3(n) = I^1(n), \\
& X^1 = X^4(M) \wedge I^1 = I^4(M), \\
& n < M \rightarrow I^1(n) < N, \\
& \neg I^1(M) < N.
\end{aligned}
$$

Remarks

Where do we stand now:

- a core procedural programming language with loops and functions.
- a prototype system for translating these programs to first-order theories;
- A simplifier for program verification.
- Some simple heuristics for proving correctness of a program in integer domain using mathematica - integer division, least common multiple, largest common factor, ...
- pointers and struct data structure for linked lists: manually prove the correctness of an algorithm for in-place reversing of a list, and that of Schorr-Waite graph marking algorithm.
- Working on threads and concurrency.