March 6, 1966

ARTIFICIAL INTELLIGENCE, NON-NUMERIC COMPUTATION AND
MATHEMATICAL THEORY OF COMPUTATION PH.D. QUALIFYING EXAMINATION

Answer all of the following three questions:

1. A. You have been given two papers. Both address themselves to present-day limitations and inadequacies in the concepts and techniques used in heuristic programming.

   What are the main issues being treated? What is your analysis and opinion of the contentions of the authors?

   B. Newell explicitly brings up the issue of representation. What is the problem of representation? Why is it important? Concrete examples will help your argument. What is being done to attack the problem of representation and how do you think this attack should proceed.

2. Write the simplest program in LISP m-expressions that you can for determining whether 3-dimensional tic-tac-toe (in a cube 4 x 4 x 4) is a win, draw, or loss for the first player. Explain carefully the role of each auxiliary function, giving examples when this will be helpful.

3. What is meant by an unsolvable class of problems? Give an example of an unsolvable class of problems. Outline a proof that it is unsolvable.

Answer any three of the following six questions:

4. Write the statements that the 3-dimensional tic-tac-toe is a win, loss, or draw for the first player as sentences of first order logic. Give the intuitive meaning of any predicate or function letters that you use.

5. Answer part (a) and either (b) or (c):

   (a) What is "hash addressing"?

   (b) How might it be used in the implementation of a list processing language?

   (c) What kinds of facilities for using hash addressing capabilities at the source language level might be made available in a symbolic computation language?
6. Let the concatenation \( x*y \) of two lists \( x \) and \( y \) be defined by

\[
x*y = \text{if } n \times \text{then } y \text{ else } ax. \quad [dx*y]
\]

Let the length of the list \( x \) be defined by

\[
l[x] = \text{if } n \times \text{then } 0 \text{ else } [l[dx]]'
\]

Let addition be defined by

\[
m + n = \text{if } n = 0 \text{ then } m \text{ else } m' + n
\]

Prove that

\[
l[x*y] = l[x] + l[y]
\]

Hint: an alternative form of the recursive formula for addition

\[
\text{plus } [m;n] = \text{if } m = 0 \text{ then } n \text{ else } [\text{plus } [m';n]]'
\]

may be useful. If you use it, prove it equivalent to the usual definition.

7. What has been achieved so far in the heuristic program area in making programs learn from their experience? What are the limitations of the methods used? What do you think should be done next? What will be the limitations if any of the programs you advocate?

8. Theorem-proving by Computer

A. Each of the following examples has been discussed in connection with more than one computer program. Discuss at least two approaches to theorem-proving by computer, using in your discussion at least one of the examples.

\[
\neg (P \lor Q) \supset \neg P
\]

\[
(\exists x) (\exists y) (\forall z) (Fxy \supset FyzFzz) (FxyGxy \supset GxzGzz)
\]

In a group, the existence of a right inverse follows from on the remaining axioms.

B. State the completeness theorem for the first-order predicate calculus and Church's Theorem and discuss their relevance to theorem-proving by computer.