Technical Report CS73005-R

Functionally Complete Machines

by

T. C. Wesselkamper

Virginia Polytechnical Institute and State University

Author's address: Dept. of Computer Science, V.P.I. & S.U.,
Blacksburg, Virginia 24061

ABSTRACT

This paper defines a functionally complete machine as a machine which is capable of evaluating every two place function over its data space. Necessary conditions on memory size for completeness are developed. These conditions are applied to System/360 as modelled by the space of bytes, the space of halfwords, and the space of words. Sufficiently large (> 64K bytes) models of System/360 are shown to be complete for the space of bytes. No models of System/360 are complete for the spaces of halfwords or words. The inequalities developed and known examples of universal decision elements suggest structures for complete machines.

1. Preliminaries and Definitions

Computer manufacturers are wont to describe their machines as "general purpose computers." The implication of this description appears to be that an advertised machine can do anything and everything. This corresponds closely to the mathematical notion of functional completeness. This paper investigates the evidence available concerning the validity of such a claim.

A set F of functions defined from a space X to a space Y is complete if every function from X into Y may be defined as a composite of the elements of F. For example if Y is the space $\{0, 1\}$ and if $X = Y \times Y$, then F is complete if each of the sixteen possible functions from X into Y can be defined in terms of the elements of F. Common usage permits a certain amount of imprecision about the domains of the functions of F. Typically both one place and two place functions are included. Thus the functions corresponding to the operations of implication and negation are said to be a complete set of functions for the space $\{0, 1\}$.

Of particular interest is a kind of complete function introduced by Sobocinski. In 1953 he exhibited a four place function defined over $\{0, 1\}$ which has the property that each two place function over $\{0, 1\}$ may be defined by a substitution of variables and constants into a single instance of F. Sobocinski called this a universal decision

element (UDE) and showed that there exist no three place UDEs for the space $\{0, 1\}$. [1] Others have investigated all such four place UDEs. [2]

In this paper we consider completeness to mean the ability to define two place functions over a space. The set of two place functions includes the set of one place functions as a subset. Each function of more than two places may be defined in terms of one and two place functions. [3] We consider here whether stored program machines are functionally complete and conditions, some necessary, some sufficient, for completeness.

2. Completeness via Function Tables

If we discuss the completeness of a machine we must decide upon a space which is to be considered to be a model of the machine's operation. In the case of the System/360 group of machines, there seem to be three reasonable choices: the space of bytes, the space of halfwords, and the space of words.

There are $2^8 = 256$ elements in the space of bytes. Hence there are $256^{256^2} = {}_2 2^{19}$ two place functions over this space. Is it possible to define each of these functions on a System/360 machine? The answer depends on the memory size of the machine.

One completely general way to describe a two place function is by means of its functional table. Evaluation of the function is accomplished by table lookup. For the space of eight bit bytes this table would occupy $(2^8)^2 = 2^{16} = 2^6 K = 64K$ bytes. Additional core space would be required for a table lookup algorithm. Surely any model of System/360 with more than 64K bytes of core is complete over this space of bytes.

While on System/360 the byte is the smallest addressable unit, it is not the normal operational unit. The halfword is an operational unit and the full word is the normal operational unit. The previous table lookup technique works in neither case.

The space of halfwords contains 2^{16} elements and there are $(2^{16})^{2^{32}} = (2^{2^4})^{2^{32}} = 2^{2^{36}}$ two place functions over the space. The function table for a two place function would contain $(2^{16})^2$ halfwords, that is, 2^{33} bytes. Since System/360 employs an addressing technique which uses 24 bit addresses, this table lookup technique would require far more core than is possible for System/360 addressing.

The space of words would require $(2^{32})^2$ words which is 2^{66} bytes for the function table of a two place function. There are $(2^{32})^{2^{64}} = (2^{2^5})^{2^{64}} = 2^{2^{69}}$ two place functions over this space.

3. Complete sets of Algorithms.

The notion of table look-up is not a realistic model of the operation of a machine. If a machine possesses an algorithm such that for each two inputs there is an output, then the machine may be regarded as a two place function. If for each input there is one output, then it may be regarded as a one place function. This in no way describes the behavior of all computer programs.

Throughout this paper we use the function log to denote the logarithm to the base 2.

If a machine has word length k bits and has M machine instructions, then there are $(2^k)^{2^{2k}}$ two place functions over the space of words and to each of these functions there must correspond an evaluation algorithm consisting of (not necessarily) distinct) machine instructions.

There is one trivial program consisting of no instruction; there are M programs of one instruction; there are M^2 programs of two instructions; and, in the general case, there are M^n programs of n instructions. If a set of algorithms is to evaluate all two place functions there must be at least as many algorithms as functions. The set of all algorithms of n or fewer expressions contains

$$1 + M + M^2 + \ldots + M^n = \frac{M^{n+1} - 1}{M - 1} \quad \text{elements.}$$

Hence for completeness n must satisfy the inequality

$$\frac{M^{n+1} - 1}{M - 1} \geq (2^k)2^{2k} . \qquad (1)$$

The left side of inequality (1) is strictly greater than M^n for $M > 1$ and so if (1) is a necessary condition for completeness then the inequality

$$M^n > (2^k)2^{2k} \qquad (2)$$

is a necessary condition for completeness.

Now suppose that, as in the case of System/360, there exists an integer j such that $k = 2^j$. The last inequality becomes:

$$M^n > (2^{2^j})2^{2k} = 2^{2^j+2k} . \qquad (3)$$

Since the function log is a monotone function, taking the logarithm of the inequality twice results, successively, in:

$$n \log M > 2^{j+2k} \log 2 \quad 2^{j+2k}$$

and

$$\log n + \log \log M > j + 2k \log 2 = j + 2k.$$

This last inequality may be rewritten:

$$\log n > j + 2^{j+1} - \log \log M. \qquad (4)$$

For System/360 the possible values of k are 8, 16, and 32, and so the possible values of j are 3, 4, and 5.

Machine instructions for System/360 are 16, 32, or 48 bits long. In each case the first 8 bits denote the op code of the instruction. There are approximately 55 instructions of 8 bits (type RR), 70 instructions of 32 bits (types RX, RS, and SI), and 20 instructions of 48 bits (type SS). [4]

If the host machine is so large that each possible 24 bit address is an actual address, then there are:

$$k = 55 \cdot 2^8 + 70 \cdot 2^{24} + 20 \cdot 2^{40}$$

possible machine instructions, and

$$\log \log k = 5.47 \text{ approximately.}$$

Small changes in k have little effect on the value of log log k. The average value of the length of a machine instruction 28.138 bits or about 3.5 bytes.

For the possible values of j noted above we have from (4) the inequalities:

$$j = 3: \log n > 19 - \log \log k \approx 13.53;$$

$$j = 4: \log n > 36 - \log \log k \approx 30.53;$$

$$j = 5: \log n > 69 - \log \log k \approx 63.53. \qquad (5)$$

To be functionally complete for each of the three spaces a machine must evaluate programs of length greater than or equal to n instructions where n is the smallest integer which satisfies, respectively, the three inequalities of (5).

If a program has n instructions and the expected value of the length of each instruction is 3.5 bytes, then the expected size S of the program is $S = 3.5 \cdot n$, where S is measured in bytes. Hence to be functionally complete over the space of bytes, halfwords, and words the number of bytes in core must at least satisfy the inequalities:

bytes: $\log S > 13.53 + \log 3.5 \approx 15.34;$

halfwords: $\log S > 30.53 + \log 3.5 \approx 32.34;$

words: $\log S > 63.53 + \log 3.5 \approx 65.34.$ (6)

In the case of the space of bytes this means that there be in excess of 2^{15} = 32K bytes, which is comparable to the result based upon evaluation using table lookup.

In the case of the spaces of halfwords and of full words, there is clearly no way to store such a program on System/360 equipment. System/360 is not complete for these two spaces.

The development of inequality (4) makes it clear that it represents a necessary condition for completeness and not a sufficient condition. Among other things, its sufficiency would require that of the programs of length n statements or fewer, no two programs define the same function. This is surely false since the one statement programs corresponding to the Assembly language instructions

OR n, n $(0 \leq n \leq 15);$

and

NR n, n $(0 \leq n \leq 15);$

represent thirty-two examples of programs which evaluate the identity mapping.

4. Universal Decision Elements.

If F is a UDE for a space of n points, then a machine which evaluates F is complete for the space. We can show that for any n-valued space X there is an $(n^2 + 2)$-place UDE.

Let $N = n^2$ and define:

$$(x, y; z_1, z_2, \ldots, z_N) = z_i, \text{ where } i = n(x - 1) + y. \tag{7}$$

Now if fxy is any two place function over the space X, fxy is given by:

$$fxy = (x, y; f11, f12, \ldots, fnn),$$

that is, by the substitution:

$$z_i = fjk, \text{ where } j = \left(\left[(i-1)/n \right] + 1 \right), \text{ and}$$

$$k = (((i - 1) \bmod n) + 1), \text{ and}$$

$\left[x\right]$, represents the greatest integer in x.

J. C. Muzio has conjectured that there is a UDE of $n^2 + 1$ inputs for the n-valued space X. [5]

The UDE defined in (6) is a glorified table look-up technique. Programming for such a computer would certainly differ from present programming. However, experience with small values of n suggests that when UDE's exist they exist in abundance. In the case n = 2, Eric Foxley showed that there are 263 four place UDEs. [2] All of this suggests that there may be complete machines which evaluate functions in familiar ways over the space of k bit words and have memory size in the neighborhood of $2^{2k} + 2$ words. The numbers involved here suggest a configuration consisting of 2^{2k} words of core, each with a double word address, together with a double word next-instruction register.

In Section 3 we obtained the necessary condition:

$$M^n > (2^k)2^{2k} \tag{2}$$

Again, taking the logarithm twice yields:

$$\log n + \log \log M > 2k + \log k \tag{7}$$

If each instruction is I words long then $M = 2^{kI}$, and (7) gives:

$$\log n + \log I + \log k > 2k + \log k, \text{ or}$$

$$\log n + \log I > 2k, \text{ which is to say,}$$

$$nI > 2^{2k} . \qquad (9)$$

Clearly nI is the total number of words required to store n instructions of I words per instruction. Thus the stored algorithm method of function evaluation require that there be strictly more memory for program storage than can be addressed using double word addresses.

The derivation of sufficient conditions for functional completeness would require some knowledge about the distinctness of the algorithms defined by different sequences of machine instructions. This appears to be a very difficult question. Of course, increasing memory to that memory addresses include an additional bit means doubling the size of memory. Conditions for complete sets of functions have been given by I. Rosenberg. [6]

In conclusion, a machine with a k bit word is functionally complete by virtue of the table lookup technique provided that it has 2^{2k+1} words of memory. In virtue of this sufficient condition existing machines are complete over spaces corresponding to values of k such as k = 6 or k = 8. For completeness over spaces corresponding to larger values of k, the size of memory would have to be vastly increased.

References

1. Boleslaw Sobocinski, "On a Universal Decision Element", The Journal of Computing Systems, v. 1, no. 2(1953), pp. 71-80.

2. Eric Foxley, "Determination of the Set of All Four-variable Formulae Corresponding to Universal Decision Elements Using a Logical Computer", Zeitsch. f. Math. Logik und Grundlagen d. Math., Bd 10, (1964), pp. 302-314.

3. J. B. Rosser and A. R. Turquette, Many-valued Logics, (Amsterdam: North Holland Publishing Company, 1952), pp. 19-26.

4. IBM, IBM System/360 Reference Data, GX20-1703-9, n.d., (This is the "Green Card.")

5. J. C. Muzio, and D. M. Miller, "Decomposition of Ternary Switching Functions", Proc. 1973 International Symposium on Multiple Valued Logic, (Toronto, 1973)

6. Ivo Rosenberg, "La Structure des Fonctions de Plusiers Variables Sur un Ensemble Fini", C. R. Acad. Sc. Paris, t. 260 (April, 1965), pp. 3817-9.