

CS 5000: Lecture 34

Vladimir Kulyukin

Department of Computer Science

Utah State University

Outline

- Universal Programs

Review: Church's Thesis

Any algorithm for computing on numbers
can be carried out by a program in L (C++, Java, Python, etc.)

Review: Church's Thesis

Any algorithm for computing on numbers
can be carried out by a standard formalization
of the informal notion of algorithm.

Why is it a Thesis?

- Why is ***Church's Thesis*** a ***thesis***?
- There is no general definition of algorithm separate from a standard formalization of algorithm (a particular programming language or some other formalism).
- The claim that a standard programming language provides a satisfactory characterization of computation remains empirical and cannot be proved.
- Church's Thesis cannot be proved as a theorem.

Universal Programs

Universality

- Coding programs by numbers gives us a mathematical theory of compilation.
- Compiled programs must be executed or interpreted.
- Universality gives us a mathematical theory of program execution and interpretation, i.e., a mathematical theory of operating systems and interpreters.

Universal Programs: Step 1

- Suppose we have a program P that takes n arguments.
- Suppose that $\#(P) = y$.
- We would like to construct another program that can take $\#(P)$ and n argument values x_1, \dots, x_n and execute it on those values.

Universal Programs: Step 2

- But why stop with just one program?
- We can attempt to construct a program that can execute/interpret **any** program of ***n*** arguments.
- Thus, we can have programs that can execute any program of 1 argument, any program of 2 arguments, any program of 3 arguments, etc.

Universal Programs: Step 2

- For each $n > 0$, such a program is denoted by U_n and is called *universal*.
- Why is it *universal*? Because it can run *any* program of n arguments.
- We have a sequence of universal programs: $U_1, U_2, U_3, U_4, \dots$

Modern Operating Systems

- Modern operating systems and virtual machines are universal programs.
- Why? They can execute any program that accepts finite numbers of arguments.
- The number of arguments is always finite because of the hardware limitations.

Definition

For each $n > 0$,

$$\Phi^{(n)}(x_1, \dots, x_n, y) = \Psi_P^{(n)}(x_1, \dots, x_n),$$

where $\#(P) = y$.

Universal Programs

For each $n > 0$,

$$\Psi_{U_n}^{(n+1)}(x_1, \dots, x_n, x_{n+1}) = \Phi^{(n)}(x_1, \dots, x_n, x_{n+1}).$$

Universal Programs: Example

U_1 is a program that computes any partially computable function of 1 variable.

Let $f(x)$ be computed by a program P whose number is y , then

$$(\forall x)f(x) = \Phi^{(1)}(x, y) = \Psi_{U_1}^{(2)}(x, y).$$

What Does U_n Do?

- U_n is given the number of a program.
- From that number, U_n extracts instructions one by one and executes them.
- U_n keeps track of the state of the program.
- U_n runs the instruction counter.

Review: Mapping Variables and Labels

- The variables are arranged as follows:
 - $Y X_1 Z_1 X_2 Z_2 X_3 Z_3 \dots$
- $\#(Y)=1, \#(X_1)=2, \#(X_3)=6$.

Encoding the Program's State

The state of the program is encoded as a Godel number.

The positions of the input variables in the Godel number are even. The positions of the internal variables are odd.

State : $\{Y = 0, X_1 = 2, Z_1 = 0, X_2 = 1, Z_2 = 0\}$.

Godel number : $[0,2,0,1,0] = [0,2,0,1] = 3^2 \cdot 7 = 63$.

Storage Allocation in U_n

Let S be the Godel number of the state.

Let K be the number of the instruction to be executed.

Review: Computing $\#(P)$

If P is a program that consists of I_1, I_2, \dots, I_k .

$$\#(P) = [\#(I_1), \#(I_2), \dots, \#(I_k)] - 1.$$

Theorem 3.1 (Ch. 4)

For each $n > 0$, the function $\Phi^{(n)}(x_1, \dots, x_n, y)$ is partially computable.

Theorem 3.1 (Ch. 4): A Perspective

- The significance of Theorem 3.1 (***Universality Theorem***) should not be underestimated.
- This theorem gives a mathematical proof of the feasibility of operating systems and interpreters.
- Modern operating systems and interpreters, e.g. Windows, Linux, Java Virtual Machine, owe their existence to the Universality Theorem.

Theorem 3.1: Proof

To prove Theorem 3.1, we will show how, for each $n > 0$, we can construct U_n that computes the following function :

$$Y = \Phi^{(n)}(X_1, \dots, X_n, X_{n+1}).$$

U_n Construction: Part 1

1. Extract the source code.
2. Initialize the state.
3. Initialize the instruction counter.

U_n Construction: Part 1 Summary

// Z is the source code.

$$Z \leftarrow X_{n+1} + 1$$

// S is the initial state of the program.

$$S \leftarrow \prod_{i=1}^n (p_{2i})^{X_i}$$

// K is the instruction counter.

$$K \leftarrow 1$$

U_n Construction: Part 2

- The next thing that U_n must do is check for the termination condition.
- The termination condition occurs when the instruction counter is $I+1$ or 0, where I is the number of instructions in the program.
- We can easily find out the value of I , because we know how to compute the length of a number.

U_n Construction: Part 2

$[C]$ IF $K = Lt(Z)+1 \vee K = 0$ GOTO E

$[C]$ can be thought of as CONTINUE

F can be thought of as END

U_n Construction: Part 3

- The next thing that U_n must do is decode the instruction to be executed.
- **Remember that the source code has no macros!** All macros are assumed to have been expanded.
- Recall that an instruction is defined by three components:
 - the label,
 - the statement,
 - the variable.

U_n Construction: Part 3

$(Z)_K$ is the current instruction's number (prime power);

$$(Z)_K = \langle a, \langle b, c \rangle \rangle;$$

$$a = \#(L);$$

b is the type of instruction;

$$c = \#(V) - 1.$$

U_n Construction: Part 3

$$U \leftarrow r((Z)_K)$$

$$U = \langle b, c \rangle$$

U_n Construction: Part 3

$$r((Z)_K) = U = \langle b, c \rangle;$$

$$c = \#(V) - 1 = r(U);$$

$$\#(V) = r(U) + 1;$$

What is $p_{r(U)+1}$? $p_{r(U)+1}$ is a prime P such that

$$P^X \mid S, \text{ where } (S)_{r(U)+1} = X.$$

Example 1

Assume that $S = [0,2,0,1] = 3^2 \cdot 7 = 63$.

Assume that $U = \langle b,1 \rangle$. Then

$$r(U) = c = 1 = \#(V) - 1.$$

$r(U) + 1 = 2$, which is the # of X_1 .

$$V = X_1.$$

$$p_{r(U)+1} = p_2 = 3.$$

$(S)_{r(U)+1} = (S)_2 = 2$. So, $X_1 = 2$ and

$$3^2 \mid 63.$$

.

Example 2

Assume that $S = [0,2,0,1] = 3^2 \cdot 7 = 63$.

Let $U = \langle b,3 \rangle$. Then $r(U) = c = 3 = \#(V) - 1$.

So $r(U) + 1 = 4$, which is the # of X_2 , so $V = X_2$.

$p_{r(U)+1} = p_4 = 7$.

$(S)_{r(U)+1} = (S)_4 = 1$. So, $X_2 = 1$.

.

U_n Construction: Part 3

$$U \leftarrow r\left(\left(Z\right)_K\right)$$

$$P \leftarrow p_{r(U)+1}$$

U_n Construction: So Far

$$Z \leftarrow X_{n+1} + 1$$

$$S \leftarrow \prod_{i=1}^n (p_{2i})^{X_i}$$

$$K \leftarrow 1$$

$$U \leftarrow r((Z)_K)$$

$$P \leftarrow p_{r(U)+1}$$

Recommended Reading

- Section 4.3.