CS 5000: Lecture 35

Vladimir Kulyukin
Department of Computer Science
Utah State University

Outline

- Construction of the Universal Program $\boldsymbol{U_n}$.

Error: Slide 20 Lecture 32

$\#(\text{IF } X \neq 0 \text{ GOTO } A).$

$\langle a, \langle b, c \rangle \rangle.$

$a = 0.$

$b = \#(A) + 2 = 1 + 2 = 3.$

$c = \#(X) - 1 = 1.$ // this used to say $c = \#(X) = 1$, which is wrong.

$\langle 0, \langle 3,1 \rangle \rangle = 2^0 \left(2 \langle 3,1 \rangle + 1\right) \dot{-} 1 =$

$2^0 \left(2 \left(2^3 (2 \cdot 1 + 1) \dot{-} 1 \right) + 1 \right) \dot{-} 1 =$

$1(2 \cdot 23 + 1) \dot{-} 1 = 47 \dot{-} 1 = 46.$

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

Review: Storage Allocation in $\boldsymbol{U_n}$

Let S be the Godel number of the state.

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

Review: Theorem 3.1 (Ch. 4)

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

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 :

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

Review: U_n Construction

// Z is the source code.

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

// S is the initial state of the program.

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

// K is the instruction counter.

$$K \leftarrow 1$$

Review: U_n Construction

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

$[C]$ can be thought of as CONTINUE

F can be though of as END

Review: U_n Construction

$(Z)_K$ 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;

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

Review: $\boldsymbol{U_n}$ Construction

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

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

// P is the $(r(U)+1)-$st prime.

// We will need P to determine the

// value of the variable used in $(Z)_K$.

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

Review: Example 1

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

Assume, for the sake of example, that $U = \langle b,1 \rangle$.

Then $r(U) = c = 1 = \#(V) - 1$. Thus, $r(U)+1 = 2$, which is the position of X_1 in our sequence of variables. So $V = X_1$. Thus, $p_{r(U)+1} = p_2 = 3$. Then $(S)_{r(U)+1} = (S)_2 = ([0,2,0,1])_2 = 2$. So, $X_1 = 2$.

Review: $\boldsymbol{U_n}$ Construction

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

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

$$K \leftarrow 1$$

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

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

$\boldsymbol{U_n}$ Construction: Part 4

- The next thing that $\boldsymbol{U_n}$ must do is to determine the type of instruction to be executed.
- The following labels are used:
  - N – stands for nothing;
  - A – stands for addition;
  - M – stands for subtraction (minus).

U_n Construction: Part 4

$$\text{IF } l(U) = 0 \text{ GOTO } N$$

If $l(U) = 0$, the instruction to be executed is $V \leftarrow V$. We do not have to do anything.

U_n Construction: Part 4

$$\text{IF } l(U) = 1 \text{ GOTO } A$$

If $l(U) = 1$, the instruction to be executed is $V \leftarrow V + 1$. We do the addition.

$\boldsymbol{U_n}$ Construction: Part 4

- $\boldsymbol{U_n}$ now has to check if the instruction is subtraction.
- However, if we know that the value of the variable is 0, can we subtract 1 from it? No!
- So, before we dispatch to subtraction, we check if the value of the variable is 0. If it is, we do nothing.

Observation

$p_i \mid S$ if and only if $(S)_i \neq 0.$

This is true because if $(S)_i = 0,$

p_i is not a factor of $S.$

For example, $63 = 3^2 \cdot 7 = [0,2,0,1].$

Only 3 and 7 are the factors of 63, because their powers are greater than 0.

U_n Construction: Part 4

// If $\neg(P \mid S)$, the value of the variable

// in the instruction is 0. We do nothing.

IF $\neg(P \mid S)$ GOTO N

U_n Construction: Part 4

$// \text{ If } l(U) = 2, \text{ the instruction type is subtraction.}$

$// \text{ We do the subtraction.}$

$\text{IF } l(U) = 2 \text{ GOTO } M$

U_n Construction So Far

$Z \leftarrow X_{n+1} + 1$ // Get the source number.

$S \leftarrow \prod_{i=1}^{n} (p_{2i})^{X_i}$ // Encode the initial state.

$K \leftarrow 1$ // Initialize instruction counter.

$[\mathrm{C}]$ IF $K = Lt(Z) + 1 \vee K = 0$ GOTO E // Check termination

$U \leftarrow r((Z)_K)$ // Get $\langle b, c \rangle$ of K-th instruction.

$P \leftarrow p_{r(U)+1}$ // Get the prime corresponding to c.

IF $l(U) = 0$ GOTO N // If $V \leftarrow V$, go to NOTHING.

IF $l(U) = 1$ GOTO A // If $V \leftarrow V + 1$, go to ADD.

IF $\neg(P \mid S)$ GOTO N // If $V = 0$, go to NOTHING.

IF $l(U) = 2$ GOTO M // If $V \leftarrow V$-1, go to MINUS.

Review: Coding Conditional Dispatch

IF K-th instruction is of the form IF $V \neq 0$ GOTO L,
then $b = \#(L) + 2$.

U_n Construction: Part 5

IF $l(U) > 2$ and $P \mid S$, then the current instruction is of the form IF $V \neq 0$ GOTO L and $\#(L) = l(U) - 2$.

U_n Construction: Part 5

$l(U) = l(\langle b, c \rangle) = b = l(U) + 2 = \#(L) + 2$. So $\#(L) = b - 2$. Thus, we need to look for the earliest instruction $(Z)_i = \langle a_i, \langle b_i, c_i \rangle \rangle$, such that $a_i = b - 2$, or, equivalently, $a_i + 2 = l((Z)_i) + 2 = b = l(U)$.

U_n Construction: Part 5

$$K \leftarrow \min_{i \le Lt(Z)} \left[l\left((Z)_i \right) + 2 = l(U) \right]$$

GOTO C

If there is no such label, $K = 0$.

Recommended Reading

- Section 4.3.