UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA TEÓRICA

JOAO LUIZ GRAVE GROSS
RODRIGO LEITE

TURMA B – G16

Trabalho sobre o problema do "Castor Ocupado"

Trabalho da Disciplina de Teoria da Computação N

Prof. Dr. Tiarajú Asmuz Diverio

Porto Alegre, 29 de junho de 2011.

SUMÁRIO

1 HISTÓRICO E CARACTERIZAÇÃO DO PROBLEMA

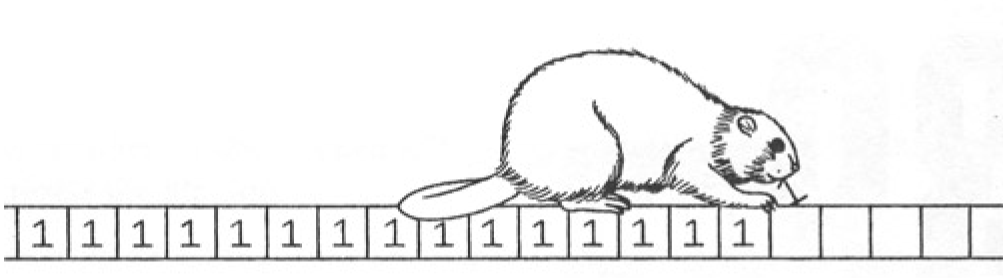

Figura 1: Castor ocupado colocando mais um “1” na fita na máquina de Turing [AKD]

Por volta de 1960, mais precisamente em 1962, Tibor Rado, professor na Universidade do Estado de Ohio, EUA, propôs uma função que descrevia que dado um número finito de símbolos e estados, a mesma deveria selecionar aquelas máquinas de Turing nas quais eventualmente houvesse uma parada, ao rodarem com uma fita em branco de entrada. Dentre estes programas, aquele que apresentasse o máximo número de símbolos não brancos que restassem na fita ao parar era o escolhido. Concomitantemente, a função procurava o máximo número de transições antes de parar.

Este problema foi depois denominado “Problema do Castor Ocupado” ou, em inglês, “Busy Beaver's Problem” e sua abordagem restringiu-se a denotar o máximo número finito de 1's que uma máquina de Turing qualquer com n-estados, sobre um alfabeto de apenas dois símbolos, $\Sigma = \{0, 1\}$, pode gravar em uma fita inicialmente preenchida apenas com 0's, até atingir um estado de parada.

A função para o problema é bem definida, mas rapidamente torna-se não-computável, mesmo para pequenos números de estados, visto que sua taxa de crescimento é absurdamente elevada. Para programas com 1, 2, 3 ou 4 estados o número máximo de 1's escritos na fita é conhecido, porém para uma quantidade de estados maior do que 5 a solução ainda é apenas aproximada ou inexistente. Por exemplo, considerando B(n) a função do castor ocupado dados n estados, sendo $n \in N^*$, e como resultado dessa função a quantidade de 1's escritos na fita após a execução do programa em Turing para a dada quantidade de estados, temos os seguintes resultados:

Máquina de Turing para a função castor ocupado com 1 estado:

- a0 -> p1d (quando o valor na fita for 0 e estiver no estado 'a', vai para o estado 'p' - parada -, gravando '1' na fita e movendo a cabeça da fita para a direita – 'd')

Aplicação da função do castor ocupado para n = 1:

- BB(1) = 1 (apenas um '1' gravado na fita)

Máquina de Turing para a função castor ocupado com 2 estados:

- a0 -> b1d; a1 -> b1e; b0 -> a1e; b1 -> p1d

Aplicação da função do castor ocupado para n = 2:

- BB(2) = 4

Figura 2: mudanças na fita (de cima para baixo) para para a função do castor ocupado com 2 estados

Máquina de Turing para a função castor ocupado com 3 estados:

- a0 -> b1d; a1 -> p1d; b0 -> c0d; b1 -> b1d; c0 -> c1e; c1 -> a1e

Aplicação da função do castor ocupado para n = 3:

- BB(3) = 6;

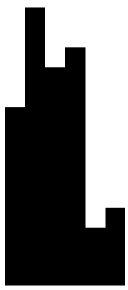

Figura 3: mudanças na fita (de cima para baixo) para para a função do castor ocupado com 3 estados

Máquina de Turing para a função castor ocupado com 4 estados:

- a0 -> b1d; a1 -> b1e; b0 -> a1e; b1 -> c0e; c0 -> p1d; c1 -> d1e; d0 -> d1d; d1 -> a0d

Aplicação da função do castor ocupado para n = 4:

- BB(4) = 13;

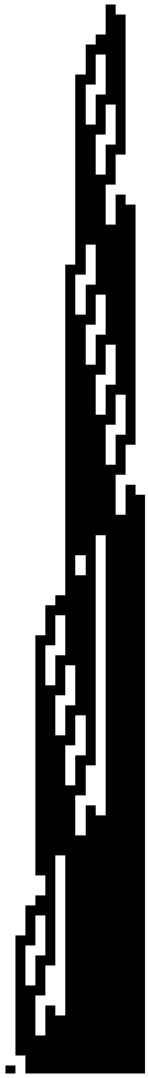

Figura 4: mudanças na fita (de cima para baixo) para para a função do castor ocupado com 4 estados

Aplicação da função do castor ocupado para n = 5:

- BB(5) ≥ 4098 (o valor exato ainda não foi encontrado);

Aplicação da função do castor ocupado para n = 6:

- BB(6) ≥ $4.6 * 10^{1439}$ (o valor exato dificilmente será conhecido) [CGJ]

Percebe-se que a função cresce muito rapidamente, a propósito. A função do castor ocupado cresce mais rápido do que qualquer função computável. [DJM] Só para se ter uma ideia das dimensões dos valores obtidos em cada aplicação do problema do castor ocupado, se nós fossemos usar um átomo para cada 1 colocado na fita por este problema para n = 6, nós teríamos preenchido todo o universo. É nesta velocidade que a função do castor ocupado cresce.

2 PROVA

Suponha que BB(n), a função do castor ocupado aplicada para n estados, seja computável. Suponha também que nos é dada uma máquina de Turing M, e alguém nos pergunta se M roda indefinidamente ou para em algum ponto no tempo (problema da parada). Nós rodamos M para a função BB(n) + 1 passo. Se ela parar, com essa quantidade de passos, nós teremos uma resposta. Se neste ponto M ainda não parou, ela ultrapassou a barreira do problema do castor ocupado e nunca vai parar (pela definição de BB(n)) e nossa resposta será de que M nunca irá parar. Logo, se BB(n) + 1 passo rodou e não parou, nós geramos uma contradição, pois pela definição de BB(n), rodando apenas BB(n) a máquina M já deveria ter parado. Assim BB(n) é não-solucionável.

Outra forma de demonstrar que este problema é não-solucionável é aplicando redução. Queremos saber se BB(n) para para qualquer $n \in N^*$, ou seja, reduzimos o problema do castor ocupado ao problema da parada. Sabemos por demonstração [TAD] que o problema da parada é não-solucionável, logo o problema do castor ocupado também é não-solucionável.

Já o complemento da função do castor ocupado também é não-computável, visto que o problema do castor ocupado é não-computável e o complemento de qualquer problema não-computável é não-computável. [TAD]

REFERÊNCIAS

[TAD] Diverio, Tiarajú Asmuz. Teoria da computação: máquinas universais e computabilidade / Tiarajú Asmuz Diverio, Paulo Blauth Menezes. – Porto Alegre: Instituto de Informática da UFRGS : Bookman, c2011. 3ª edição.

[AKD] Dewdney, A. K. The (new) Turing omnibus: 66 excursions in computer science. Publisher: Hnery Holt, 1993. 480 p.

[CGJ] Chaitin, G. J. "Computing the Busy Beaver Function." §4.4 in Open Problems in Communication and Computation (Ed. T. M. Cover and B. Gopinath). New York: Springer-Verlag, pp. 108-112, 1987.

[DJM] Darakhshan J. Mir. Foundations of Computer Science - Lecture 4: Second Half. Department of Computer Science. Rutgers, The State University of New Jersey, 2009.