Lecture #20: Priority Methods and a Solution for Post's Problem

Lecture Presentation

Topic To Be Discussed

The lecture notes introduced a ***priority method*** and its use to solve Post's problem.

This was one of *two* priority methods that were presented, to do this, at almost exactly the same time. The other priority method, to do this, will be described.

Objectives of Another Process

The objective of the new priority method will be to construct a *pair* of sequences

$$A_0, A_1, A_2, A_3, \ldots \qquad \text{and} \qquad B_0, B_1, B_2, B_3, \ldots \tag{1}$$

which satisfy the following properties.

(a) $A_i \subseteq A_{i+1}$ and $B_i \subseteq B_{i+1}$ for every number $i \in \mathbb{N}$, so that these sequences both have limits. Let A be the limit of the sequence $A_0, A_1, A_2, A_3, \ldots$ and let B be the limit of the sequence $B_0, B_1, B_2, B_3, \ldots$.

(b) The sequences $A_0, A_1, A_2, A_3, \ldots$ and $B_0, B_1, B_2, B_3, \ldots$ are both ***computable***.

(c) $A \not\leq_T B$ and $B \not\leq_T A$.

Note that B ***does not*** correspond to the set B used in the construction that is described in the lecture notes concerning a solution for Post's problem.

Details of the Process

If $S \subseteq \mathbb{N}$ and $n, k \in \mathbb{N}$ then the sets W_n^S and $W_{n,k}^S$, described below, will be used to describe the process that will be follows.

- W_n^S is the domain of the partial function $g : \mathbb{N} \to \mathbb{N}$ that is c_S-computed by an $\mathcal{S}$-program $\mathcal{P}$ such that $\#(\mathcal{P}) = n$, where $c_S : \mathbb{N} \to \{0, 1\}$ is the characteristic function of the set S.
- $W_{n,k}^S$ is the set of numbers $x \in \mathbb{N}$ such that the $\mathcal{S}$-program $\#(P)$ with an oracle, such that $\#(\mathcal{P}) = n$, halts when executed on input x, with an oracle for S, ***after taking at most k steps***.

Thus $W_{n,k}^S \subseteq W_{n,k+1}^S \subseteq W_n^S$ for all $S \subseteq \mathbb{N}$ and $k, n \in \mathbb{N}$.

The process makes use of a pair of vertical lists of the natural numbers, called the ***A*-List** and the ***B*-List**, the smaller numbers above larger ones, as follows.

***A*-List**	***B*-List**
0	0
1	1
2	2
3	3
4	4
$\vdots$	$\vdots$

At each point in this process a finite number of the numbers in the list are marked with either "$+$" or "$-$", but not both at once:

***A*-List**		***B*-List**	
0	$+$	0	
1		1	$-$
2		2	
3		3	
4		4	
$\vdots$		$\vdots$	

Indeed, this process proceeds in a series of ***stages***, numbered by positive integers, with symbols in the A-List possibly being marked with $+$ and symbols in the B-List possibly being marked with $-$ in odd-numbered rounds, and with symbols in the B-List possibly being marked with $+$ and symbols in the A-List possibly being marked with $-$ in (positive) even-numbered rounds. Only a finite number of symbols can be marked in this way in each round.

$A_0 = B_0 = \emptyset$ and, for each positive integer i, A_i is the set of numbers in the A-list marked with "$+$" after round $2i - 1$, and B_i is the set of numbers in the B-list marked after round $2i$. Thus, if the lists include markers as shown at the bottom of the previous page after round 1, and no integers greater than 4 are marked on either list, then $A_1 = \{0\}$.

Once the process has been described, it will be clear that no number that is marked with "$+$" will ever lose that marker, on either list — so that it will be true that $A_i \subseteq A_{i+1}$ and $B_i \subseteq B_{i+1}$ for all $i \in \mathbb{N}$, as desired.

On the other hand, it *will* be possible that a number marked with "$-$" on a list will have this marker replaced with "$+$" at a later point.

A number on either list is said to be ***vacant*** if it is *not* marked with "$+$". Thus, if no numbers greater than 4 are marked on either of the A-list or B-list, above, then every number on the A-list except 0 is vacant, and all numbers on the B-list are vacant.

The numbers on the A-list are also marked with "movable markers" $\boxed{0}_A, \boxed{1}_A, \boxed{2}_A, \ldots$ and the numbers on the B-list are also marked with "movable markers" $\boxed{0}_B, \boxed{1}_B, \boxed{2}_B, \ldots$ — with a finite (initial) number of each of these used at each point in the process. For example, the lists might be as follows.

***A*-List**			***B*-List**	
$\boxed{0}_A$	0	$+$	0	
$\boxed{1}_A$	1		1	$-$
	2		$\boxed{0}_B$ 2	
	3		3	
	4		4	
	$\vdots$		$\vdots$	

A number on either list is said to be ***free*** if it is *not* marked with any marker and, furthermore, there is no number below it on this list that is marked with any marker, either. For example, if the lists are as shown above and no number greater than 4 has a marker on either list, then every number greater than or equal to 2 is free on the A-list, and every number greater than or equal to 3 is free on the B-list.

Initially, no number on either list is marked with either "$+$" or "$-$", so that $A_0 = B_0 = \emptyset$ (as noted above). 0 is marked with $\boxed{0}_A$ on the A-List, and 0 is marked with $\boxed{0}_B$ on the B-List. The markers $\boxed{i}_A$ and $\boxed{i}_B$ are not used for any positive integer i, so lists are initially as follows.

***A*-List**		***B*-List**	
$\boxed{0}_A$	0	$\boxed{0}_B$	0
	1		1
	2		2
	3		3
	4		4
	$\vdots$		$\vdots$

For each number $n \in \mathbb{N}$, ***stage* $2n+1$** is as follows.

(a) Assign marker $\boxed{n+1}_A$ to the smallest free integer in the A-list.

(b) For $0 \leq i \leq n+1$, let $x_i \in \mathbb{N}$ be the number in the A-list with marker $\boxed{i}_A$; if x_i is vacant, check whether $x_i \in W^{B_n}_{i,n+1}$.

(c) If there is no number $k \in \mathbb{N}$ such $0 \leq k \leq n+1$, x_k is vacant (that is, not marked with "$+$") in the A-list, and $x_k \in W^{B_n}_{k,n+1}$, then proceed to stage $2n+2$ without making any changes — so that $A_{n+1} = A_n$. Otherwise set j to be the ***smallest*** integer k that satisfies these properties and continue as follows.

(d) Mark the number x_j labelled by $\boxed{j}_A$ in the A-list with "$+$", so that it is no longer vacant, and so that $A_{n+1} = A_n \cup \{x_j\}$.

Mark any number in $\overline{B_n}$, whose membership in B_n was queried using the oracle, during the computation of oracle program with number j on input j, with "$-$" in the B-list. (***Note*** that this could not have included any numbers currently marked with "$+$".)

(e) Let $m \in \mathbb{N}$ be the smallest number on the B-list that is currently free. For $i = j, j+1, \ldots, n$, move marker $\boxed{i}_B$ from its current position down to the number $m + i - j$.

Stage $2n+2$ is almost the same, except that the roles of A and B have been exchanged. In particular, for each number $n \in \mathbb{N}$, ***stage*** $\boldsymbol{2n+2}$ is as follows.

(a) Assign marker $\boxed{n+1}_B$ to the smallest free integer in the B-list.

(b) For $0 \leq i \leq n+1$, let $x_i \in \mathbb{N}$ be the number in the B-list with marker $\boxed{i}_B$; if x_i is vacant, check whether $x_i \in W_{i,n+1}^{A_{n+1}}$.

(c) If there is no number $k \in \mathbb{N}$ such $0 \leq k \leq n+1$, x_k is vacant (that is, not marked with "+") in the B-list, and $k \in W_{k,n+1}^{A_{n+1}}$, then proceed to stage $2n+3$ without making any changes — so that $B_{n+1} = B_n$. Otherwise set j to be the ***smallest*** integer k that satisfies these properties and continue as follows.

(d) Mark the number labelled by $\boxed{j}_B$ in the B-list with "+", so that it is no longer vacant. Mark any number in $\overline{A_{n+1}}$, whose membership in A_{n+1} was queried using the oracle, during the computation of oracle program with number j on input j, with "−" in the A-list. (***Note*** that this could not have included any numbers currently marked with "+".)

(e) Let $m \in \mathbb{N}$ be the smallest number on the A-list that is currently free. For $i = j+1, j+2, \ldots, n$, move marker $\boxed{i}_A$ from its current position down to the number $m+i-j$ - 1.

Application to an Example

Suppose that a process begins as follows.[1] Recall that the lists are initially as shown below.

***A*-List**	***B*-List**
$\boxed{0}_A$ 0	$\boxed{0}_B$ 0
1	1
2	2
3	3
4	4
⋮	⋮

[1] Things cannot actually work this way if the $\mathcal{S}$-programs with oracles and encoded are listed in the usual way. Details about their computations have been changed to make it easier to provide an example where the A-list and B-list are modified in interesting ways, right away.

Stage 1 begins by placing marker $\boxed{1}_A$:

***A*-List**		***B*-List**	
$\boxed{0}_A$	0	$\boxed{0}_B$	0
	1		1
	2		2
	3		3
	4		4
	$\vdots$		$\vdots$

Suppose, now, that

- The execution of W_0 on input 0, using an oracle for $B_0 = \emptyset$, halts after a single step — during which the oracle is consulted to confirm that $2 \notin B_0$.
- The execution of W_1 on input 1, using an oracle for $B_0 = \emptyset$, halts after a single step — during which the oracle is consulted to confirm that $3 \notin B_0$.

The A-list and the B-list are then updated as follows. First, markers "$-$" and "$+$" are adjusted in these lists as follows.

***A*-List**		***B*-List**	
$\boxed{0}_A$	0	$\boxed{0}_B$	0
	1		1
	2		2
	3		3
	4		4
	$\vdots$		$\vdots$

Then markers $\boxed{i}_B$ for $i \in \mathbb{N}$ are moved, as needed.

***A*-List**		***B*-List**	
$\boxed{0}_A$	0	$\boxed{0}_B$	0
	1		1
	2		2
	3		3
	4		4
	$\vdots$		$\vdots$