Computer Science 513

Priority Methods and a Solution for Post's Problem

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #20

Goals for the Final Lectures

Goals for the Final Lectures

- Introduce the idea of a ***priority method***
- Describe a specific priority method, and use it to solve Post's problem.

Reference: Joseph R. Shoenfield, *Degrees of Unsolvability*

Expectations for Students

Expectations for Students

- Unfortunately, the proof of the technical result presented in this lecture is long, with lots of details.
- Students ***will not*** be asked to recall the details of this proof. There might not even be enough time for all of these details to be presented in class (if the course had not been flipped).

Expectations for Students

- On the other hand,
  - the ***result*** being established, and its significance,
  - the new thing being introduced to establish it — a ***priority method*** — and
  - the overall organization of the argument

  are things that *can* be reasonably understood, and that students might later be asked about.

How Post's Problem Will Be Solved

Post's Problem will be solved by describing a construction for a sequence $A_0, A_1, A_2, A_3, \ldots$ of ***sets*** — so that $A_i \subseteq \mathbb{N}$ for $i \in \mathbb{N}$ — with the following properties:

(a) This sequence has a ***limit***. This will be called the set $A \subseteq \mathbb{N}$.

(b) A is not ***computable***, because no $\mathcal{S}$-program computes the characteristic function of A.

(c) This above *sequence* is ***computable***. This — and the fact that $A_i \subseteq A_{i+1}$ for all $i \in \mathbb{N}$ — makes it reasonably easy to prove that A is ***computably enumerable***.

(d) The ***leap*** A'' of A is $\emptyset^{(1)}$-computable. It follows from this that A cannot be Turing-complete for the set Σ_1. Thus this set satisfies all the properties needed needed to solve Post's problem.

Objectives To Be Attempted

The sequence $A_0, A_1, A_2, A_3, \ldots$ will be constructed using repeated attempts to accomplish the following ***objectives:***

I_i for $i \in \mathbb{N}$: Confirm that the $\mathcal{S}$-program $\mathcal{P}$ such that $\#(\mathcal{P}) = i$ ***does not compute*** the characteristic function p_A of A, where A is the limit of the above sequence.

$J_{i,j}$ for $i, j \in \mathbb{N}$: ***Decide whether*** the $\mathcal{S}$ program $\mathcal{Q}$ with an oracle, such that $\#(\mathcal{Q}) = i$, halts when executed on input j using an oracle for A. That is, decide whether $j \in W_i^A$.

Priorities of Objectives

Each objective has an associated ***priority***, which is an element of $\mathbb{N}$:

- For $i \in \mathbb{N}$, the priority of I_i is $\#(I_i) = 2i$ — which is always *even*.
- For $i, j \in \mathbb{N}$, the priority of $J_{i,j}$ is $\#(J_{i,j}) = 2\langle i, j \rangle + 1$ — which is always *odd*.

Every natural number $n \in \mathbb{N}$ is the priority of exactly one of the above objectives.

In the process to be described, a finite number of these objectives will sometimes "conflict." The objective with the lowest priority will always win, when this happens.

Additional Sequences

The following will also be maintained:

- After each step $s \in \mathbb{N}$, some finite subset of the objectives $\{I_i \mid i \in \mathbb{N}\} \cup \{J_{i,j} \mid i, j \in \mathbb{N}\}$ will have been "achieved"; **Achieved(s)** $\subseteq \mathbb{N}$ is the set of ***priorities*** of the objectives that are achieved immediately after step s.
- For $x, y, s \in \mathbb{N}$, $\boldsymbol{B_s} \subseteq \mathbb{N}$ and, in particular, $\langle x, y \rangle \in B_s$ if — on completion of step s — it has been confirmed that the $\mathcal{S}$-program $\mathcal{Q}$ with an oracle for A_s, such that $\#(\mathcal{Q}) = x$, halts when executed on the input y.

Additional Sequences

- **Neg-Required** : $\mathbb{N}^2 \to \mathcal{P}(\mathbb{N})$ is a partial function from pairs of natural numbers to ***finite*** subsets of $\mathbb{N}$ satisfying the following properties.
  - For $n, s \in \mathbb{N}$, Neg-Required(s, n) is defined if and only if $n \in$ Achieved(s) and $n = \#(J_{i,j})$ for some $i, j \in \mathbb{N}$.
  - If Neg-Required(s, n) is defined then Neg-Required(s, n) is the set of numbers $x \in \mathbb{N}$ such that $x \notin A_s$ and an oracle query was used to ***confirm*** that $x \notin A_s$, when the execution of the $\mathcal{S}$-program $\mathcal{Q}$, such that $\#(\mathcal{Q}) = i$, was executed on input j with an oracle for A_s.
  - It will be confirmed that Neg-Required(s, n) is a ***finite*** subset of $\mathbb{N}$, whenever this set is defined, as this construction is described.

Additional Sequences and Initialization

- For $s \in \mathbb{N}$ let $\text{NDef}_s \subseteq \mathbb{N}$ be the set of numbers $n \in \mathbb{N}$ such that Neg-Required(s, n) is defined. Then, as noted above,

$$\text{NDef}_s = \text{Achieved}(s) \cap \{\#(J_{i,j} \mid i, j \in \mathbb{N}\}$$

  for all $s \in \mathbb{N}$. While this is easily computed from Achieved(s) it will be useful to have a name for this when presenting the process that follows.

Initialization: Set $A_{-1} = \text{Achieved}(-1) = B_{-1} = \text{NDef}_{-1} = \emptyset$ in order to simplify the description of how A_0, Achieved(0), B_0 and NDef_0 will be defined.

Since $\text{NDef}_{-1} = \emptyset$ it is trivially true that Neg-Required$(-1, n)$ is a finite set whenever this set is defined, because no such set *is* defined.

Details of the Process

- For $s \in \mathbb{N}$ the objective is being considered is I_i, for $i \in \mathbb{N}$, if $\ell(s)$ is even and $\#(I_i) = \ell(s)$, and the priority being considered is $J_{i,j}$, for $i, j \in \mathbb{N}$, if $\ell(s)$ is odd and $\#(J_{i,j}) = \ell(s)$ — so that every objective is considered infinitely often, for $s = 0, 1, 2, \ldots$
- If $s \geq 0$ and $\ell(s) \in \text{Achieved}(s-1)$ (so the objective to be considered next has currently been achieved) then we can set $A_s = A_{s-1}$, $\text{Achieved}(s) = \text{Achieved}(s-1)$, $B_s = B_{s-1}$, and $\text{Neg-Required}(s, m) = \text{Neg-Required}(s-1, m)$ for all $m \in \mathbb{N}$, so that $\text{NDef}_s = \text{NDef}_{s-1}$ as well.
- Otherwise, the details of the process depend on which objective is being considered. These details are presented next.

Details of the Process: Step s when $\ell(s) = n = \#(J_{i,j})$

Suppose that $\ell(s) = n$ where $n = \#(J_{i,j})$ for $i, j \in \mathbb{N}$, and that $n \notin$ Achieved(s). Recall that, for this objective, we are trying to discover whether the $\mathcal{S}$-program $\mathcal{Q}$ with an oracle for A, such that $\#(\mathcal{Q}) = i$, halts when executed on input j.

1. Set $A_s = A_{s-1}$.
2. Check whether the $\mathcal{S}$-program $\mathcal{Q}$ ***with an oracle for A_s***, such that $\#(\mathcal{Q}) = i$, halts when executed on input j ***after taking at most s steps***.

   If this condition is ***not*** satisfied then we should set Achieved(s) = Achieved($s-1$), $B_s = B_{s-1}$, and Neg-Required(s, m) = Neg-Required($s-1, m$) for all $m \in \mathbb{N}$, so that $\text{NDef}_{s-1} = \text{NDef}_s$ as well. There is nothing more to do in this case.

Details of the Process: Step s when $\ell(s) = n = \#(J_{i,j})$

3. If the above condition ***is*** satisfied then set

$$\text{Achieved}(s) = \text{Achieved}(s-1) \cup \{n\},$$

$$B_s = \{B_{s-1}\} \cup \{\langle i, j \rangle\},$$

set Neg-Required(s, n) to be the set of numbers $x \in \mathbb{N}$ such that $x \notin A_s$ and the oracle is used to ask whether $x \in A_s$ during the computation of the above $\mathcal{S}$-program $\mathcal{Q}$ on input j (so that the answer is "No" for these values). Set Neg-Required$(s, m) =$ Neg-Required$(s-1, m)$ for all $m \in \mathbb{N}$ such that $m \neq n$.

Then $\text{NDef}_s = \text{NDef}_{s-1} \cup \{n\}$.

Details of the Process: Step s When $\ell(s) = n = \#(I_i)$

Now suppose that $\ell(s) = n$ where $n = \#(I_i)$ for some number $i \in \mathbb{N}$ and that $n \notin$ Achieved(s). Recall that, for this objective, we are trying to show that $\mathcal{P}$ ***does not compute*** the characteristic function of A, where $\#(\mathcal{P}) = i$.

1. Check whether there exists an integer $h \in \mathbb{N}$ that satisfies the following properties.
   (a) $0 \leq h \leq s$ and $\ell(h) = i$
   (b) $h \notin$ Neg-Required$(s - 1, m)$, for each number $m \in \mathbb{N}$ such that $0 \leq m < n$ and $m \in \text{NDef}_{s-1}$ — so that no achievements of objectives with priorities less than n require that $h \notin A$
   (c) The $\mathcal{S}$ program $\mathcal{P}$, such that $\#(\mathcal{P}) = i$, halts when executed on the input h ***using at most s steps*** — and, furthermore, the output returned by this execution of the algorithm is ***not*** equal to 1.

Details of the Process: Step s When $\ell(s) = n = \#(I_i)$

If the above condition is ***not*** satisfied then we should set $A_s = A_{s-1}$, Achieved(s) = Achieved$(s-1)$, $B_s = B_{s-1}$, and Neg-Required(s, m) = Neg-Required$(s-1, m)$ for all $m \in \mathbb{N}$, so that $\text{NDef}_s = \text{NDef}_s$ as well; there is nothing more to do in this case.

Otherwise we should proceed as follows.

2. Set k to be the ***smallest*** integer $h \in \mathbb{N}$ that satisfied the above properties (a), (b) and (c), and set $A_s = A_{s-1} \cup \{k\}$.

Details of the Process: Step s When $\ell(s) = n = \#(I_i)$

3. Let

$$\text{Failed}(s) = \{x \in \text{NDef}_{s-1} \mid x > n \text{ and } k \in \text{Neg-Required}(s-1, x)\}.$$

4. Set

$$\text{Achieved}(s) = (\text{Achieved}(s-1) \cup \{n\}) \setminus \text{Failed}(s)$$

— so every element $x \in \text{Failed}(s)$ is ***removed*** from A_{s-1} when A_s is produced.

Details of the Process: Step s When $\ell(s) = n = \#(I_i)$

5. Set $B_s = B_{s-1} \setminus \{\langle i, j\rangle \mid \#(J_{i,j}) \in \text{Failed}(s)\}$.
6. For all $x \in \text{Failed}(s)$, set Neg-Required(s, x) to be ***undefined***.

   For all $x \in \text{NDef}_{s-1}$ such that $x \notin \text{Failed}(s)$, set Neg-Required$(s, x)$ to be Neg-Required$(s - 1, x)$.

   Leave Neg-Required(s, x) undefined for all $x \in \mathbb{N}$ such that $x \notin \text{NDef}_{s-1}$, so that

$$\text{NDef}_s = \text{NDef}_{s-1} \setminus \text{Failed}(s).$$

Things to Notice

Things to Notice:

- If A_s, Achieved(s), B_s, Neg-Required(m, s) for $m \in \mathbb{N}$ and NDef_s are produced from A_{s-1}, Achieved($s-1$), B_{s-1}, Neg-Required($m, s-1$) for $m \in \mathbb{N}$ and NDef_{s-1} as described in this process, then these are finite sets that have the properties that were described for them when they were introduced.
- Either $A_s = A_{s-1}$ or $A_s = A_{s-1} \cup \{k\}$ for some number $k \in \mathbb{N}$, so that $A_{s-1} \subseteq A_s$ and $|A_s| \leq s+1$ for all $s \in \mathbb{N}$.
- No priority $\#(I_i)$, such that $i \in \mathbb{N}$, is ever removed when forming Achieved(s) from Achieved($s-1$) — so that if $\#(I_i) \in$ Achieved(s) then $\#(I_i) \in$ Achieved(t) for every number $t \in \mathbb{N}$ such that $t \geq s$.

Things to Notice

Things to Notice:

- One the other hand, it *is* possible that $\#(J_{i,j})$ is "achieved" at some point but this achievement is removed again, after that.
  Suppose that $\#(J_{i,j})$ is "achieved" at step s, for numbers $i, j, s \in \mathbb{N}$, so that

$$\#(J_{i,j}) \in \text{Achieved}(s) \setminus \text{Achieved}(s-1).$$

  We will say that $\#(J_{i,j})$ is ***temporarily achieved*** at step s, in this case, if the achievement is removed later, so that there exists a number $t \in \mathbb{N}$ such that $t > s$ and $\#(J_{i,j}) \notin \text{Achieved}(t)$.
  We will say that $\#(J_{i,j})$ is ***permanently achieved*** at step s, in this case, if it is never removed again — so that $\#(J_{i,j}) \in \text{Achieved}(t)$ for all $t \in \mathbb{N}$ such that $t > s$, as well.

Proof That A Exists

Theorem #1: Let $A_0, A_1, A_2, A_3, \ldots$ be a sequence of subsets of $\mathbb{N}$ such that $A_i \subseteq A_{i+1}$ for all $i \in \mathbb{N}$. Then this sequence has a limit. Indeed, the limit of this sequence is $\bigcup_{n \geq 0} A_n$.

The proof is a straightforward consequence of the definition of a "limit" of a sequence of subsets of $\mathbb{N}$, and is included in the supplemental document for this lecture.

Since the the sequence of sets $A_0, A_1, A_2, \ldots$ produced using the previous method satisfies the requirements in the theorem it follows that this sequence has limit $A = \bigcup_{n \geq 0} A_n$.

Proof That A is Not Computable

Lemma #2: Let $i, j \in \mathbb{N}$ and let $n = \#(J_{i,j})$. Then, during the construction of sets $A_0, A_1, A_2, A_3, \ldots$, the objective $J_{i,j}$ is ***temporarily achieved*** at most $\lfloor (n-1)/2 \rfloor + 1$ times.

Sketch of Proof: An achievement of $J_{i,j}$ can only be "temporary" if it is removed after that, and an examination of the process confirms that the number of times that this can be happen is bounded by the number of objectives I_ℓ such that $\ell \in \mathbb{N}$ and $\#(I_\ell) < \#(J_{i,j})$. A (slightly more detailed) proof is included in the supplemental document.

Proof That *A* is Not Computable

Lemma #3:

(a) Each of the following sequences of subsets of $\mathbb{N}$ has a limit:

i. Achieved(0), Achieved(1), Achieved(2), Achieved(3), ...
ii. $B_0, B_1, B_2, B_3, \ldots$
iii. $\text{NDef}_0, \text{NDef}_1, \text{NDef}_2, \ldots$

(b) Let NDef $\subseteq \mathbb{N}$ be the limit of the above sequence $\text{NDef}_0, \text{NDef}_1, \text{NDef}_2, \text{NDef}_3, \ldots$. Then, for every number $n \in \text{NDef}$, there exists a number $m \in \mathbb{N}$ such that $n \in \text{NDef}_i$ for all $i \geq m$ and, furthermore, such that

$$\text{Neg-Required}(s, n) = \text{Neg-Required}(t, n)$$

for all $s, t \in \mathbb{N}$ such that $s, t \geq m$.

Proof That *A* is Not Computable

The claim follows by the fact that every achievement of every objective I_i is "permanent", Lemma #2, and an examination of the process that has been provided. A proof is included in the supplementary document.

With that noted, let Achieved, B and NDef be the limits of the sequences

$$\text{Achieved}(0), \text{Achieved}(1), \text{Achieved}(2), \text{Achieved}(3), \ldots$$

$$B_0, B_1, B_2, B_3, \ldots$$

and

$$\text{NDef}_0, \text{NDef}_1, \text{NDef}_2, \text{NDef}_3, \ldots$$

respectively.

Proof That A is Not Computable

Theorem #4: Let $A \subseteq \mathbb{N}$ be the limit of the sequence

$$A_0, A_1, A_2, A_3, \ldots .$$

Then A is not computable.

It suffices to argue that the characteristic function of A is not computed by the $\mathcal{S}$-program $\mathcal{P}$ such that $\#(\mathcal{P}) = n$, for any $n \in \mathbb{N}$. The proof of this is straightforward for numbers $n \in$ Achieved. A somewhat more complicated argument, involving the set NDef and the functions Neg-Required(s, m) for $s, m \in \mathbb{N}$, can be given to establish this when $n \notin$ Achieved as well. Once again, a proof is included in the supplemental document for this lecture.

Proof That A is Computably Enumerable

Recall, from the previous lecture, that if $G : \mathbb{N} \to \mathbb{N}$ is a total function then a sequence

$$S_0, S_1, S_2, S_3, \ldots$$

of subsets of $\mathbb{N}$ is ***G-computable*** if the function $H : \mathbb{N}^2 \to \{0, 1\}$ such that, for all $n, x \in \mathbb{N}$,

$$H(n, x) = p_{S_n}(x) = \begin{cases} 1 & \text{if } x \in S_n \\ 0 & \text{if } x \notin S_n \end{cases}$$

is G-computable. Let $I : \mathbb{N} \to \mathbb{N}$ be the identify function, so that $I(n) = n$ for all $n \in \mathbb{N}$.

Proof That A is Computably Enumerable

Lemma #5: Each of the sequences of sets

$$A_0, A_1, A_2, A_3, \ldots$$

and

$$B_0, B_1, B_2, B_3, \ldots$$

is I-computable.

It suffices to examine the process to compute $A_0, A_1, A_2, \ldots$ that has now been described and to introduce representations of finite sets as natural numbers in order to establish this claim. Once again, details are available in the supplemental document.

Proof That A is Computably Enumerable

Theorem #6: Let $A \subseteq \mathbb{N}$ be the limit of the sequence $A_0, A_1, A_2, A_3, \ldots$. Then A is computably enumerable.

This is a reasonably straightforward consequence of Lemma #5, above, and the fact that $A_s \subseteq A_{s+1}$ for all $s \in \mathbb{N}$ — so that $A = \bigcup_{i \in \mathbb{N}} A_i$. Details are included in the supplemental document, once again.

Proof That A is not Turing-Complete

Recall, from the previous lecture, that if $G : \mathbb{N} \to \mathbb{N}$ is a total function then the ***leap*** G'' of G is the set

$$G'' = \{x \in \mathbb{N} \mid \ell(x) \in W^G_{r(x)}\} \subseteq \mathbb{N}$$

and that if $S \subseteq \mathbb{N}$ then the ***leap*** S'' of S is the leap p''_S of the characteristic function of S.

As noted in the previous lecture $G' \equiv_m G''$ for every total function $G : \mathbb{N} \to \mathbb{N}$, so that $S' \equiv_m S''$ for every subset $S \subseteq \mathbb{N}$, as well.

Proof That A is not Turing-Complete

The following results are reasonably straightforward consequences of results from Lecture #16 and #19; proofs can be found in the supplemental document for this lecture.

Lemma #7: Let $S, T \subseteq \mathbb{N}$. If $S \preceq_T T$ then $S'' \preceq_1 T''$.

Lemma #8: Let $S \subseteq \mathbb{N}$. If S'' is $\emptyset^{(1)}$-computable then S is not Turing-complete for the set Σ_1 of computably enumerable sets.

Proof That A is not Turing-Complete

Once again, let $A \subseteq \mathbb{N}$ be the limit of the sequence of sets

$$A_0, A_1, A_2, A_3, \ldots$$

and let $B \subseteq \mathbb{N}$ be the limit of the sequence of sets

$$B_0, B_1, B_2, B_3, \ldots$$

Lemma #9: $A'' = \{\langle y, x \rangle \mid \langle x, y \rangle \in B\}$.

Proof That A is not Turing-Complete

Theorem #10: A is not Turing complete for the set Σ_1 of computably enumerable sets.

Proof:

- Lemma #5 implies that the sequence $B_0, B_1, B_2, B_3, \ldots$ is I-computable and, since the identify function $I : \mathbb{N} \to \mathbb{N}$ is computable, this sequence is $\emptyset$-computable as well.
- Lemma #3 implies that this sequence has a limit, B.
- It follows by the ***Limit Lemma***, stated and proved during Lecture #19, that B is $\emptyset^{(1)}$-computable.
- The previous lemma now implies that A'' is $\emptyset^{(1)}$-computable.
- It now follows by Lemma #8 that A is not Turing-complete for the set Σ_1 of computably enumerable sets, as claimed. □

What Else Can Be Shown?

- Post's problem was solved independently, twice, at almost the same time in the mid-1950's. The solution given in these notes is based on a description of a solution by Richard Friedberg, as described in the reference *Degrees of Unsolvability*.
- Another solution for this problem will be considered in the presentation for this lecture.