Problem Solving in Computer Science Lecture Notes
Date: 18/05/05
Scribe: Wojciech Galuba

Gregory's team summarizes their results:

- Given 2 bits of shared memory, there is an algorithm that has $O(n^2 b)$ time complexity on a bounded scheduler
- Any algorithm needs at least $3b - n + 2$ steps to run assuming a bounded scheduler
- There exists a trivial version of the originally proposed algorithm that requires $3bn \log n$ steps.
- Conjecture: the space-time complexity of any algorithm is $\Omega(bn \log n)$

Khaled's team:

- It is not possible to transfer any information between two processes when there is only one-bit of shared memory and an unbounded fair scheduler. Proof sketch: Assume process p_1 has a bit q that it wants to send to process p_2. Any process when scheduled, can either toggle the shared bit or leave it unchanged. If it leaves it unchanged then the other process will not be able to distinguish between two situations: when process was scheduled and left the bit unchanged and when the process was not scheduled. Therefore at any time a process is scheduled it must toggle the shared bit. If that is so, then the process has only one action available to it, toggling the bit, but it needs at least two possible actions on the shared memory so that the other process can distinguish between the situations in which $q = 0$ and $q = 1$.
- Question: An algorithm for 2-bits is possible, it is impossible for 1-bit but is it possible for 1.5-bits (3 allowed states of shared memory)?
- The solution to the "cheating problem" can be based on reasoning about knowledge: which cardinal knows which votes. Tom's remark: it may be impractical to define and track the knowledge for a given protocol.

Alex's team:

- Considering data overwrites in the shared memory as a way of cheating. However, it is still not a generic enough definition.
- idea: memory exclusion. Each part of shared memory is writable by some process not others. Violations easy to detect.
- two rules that must be satisfied by a cheat free protocol:
  1. When you express favorites you must not know the other's favorites.
  2. When you vote, you vote on the most frequent favorite.

Summary - optimal cost for bounded schedulers, two different classes of $O(bn \log n)$ space-time protocols

- space: $O(1)$, time: $O(bn \log n)$
- space: $O(n * \log n)$, time: $O(b)$

A semantically secure public-key encryption, page 33 of the survey: http://www.wisdom.weizmann.ac.il/ oded/PS/foc-sur04c.ps

---

Tom's outline of how the cheating should be defined:

- private state of each process:

  $F_i \in \{1, dots, n\}$, the initial favorites

  $P_i \in \{0, dots, 2^p - 1\}$, the private memory

- shared state:

  $T \in \{0, 1\}$, termination flag (the smoke)

  $V_i \in \{1, \ldots, n\}$, write only, the final votes

  $S \in \{0, \ldots, 2^s - 1\}$, shared memory

- proper protocol, for each process $1 \leq i \leq n$ function $f_i : F_i \times P_i \times T \times S \rightarrow P_i \times V_i \times T \times S$
- cheating protocol, for each process $1 \leq i \leq n$ function $f_i : F_i \times P_i \times T \times S \rightarrow P_i \times V_i \times T \times S \times F_i$
- Define:
  1. *Trace* of a protocol, given initial state and scheduler
  2. given process i and trace t, observation of i for t (essentially a projection in time and space)
  3. given protocol $F = \{f_i | 1 \leq i \leq n\}$ process i, observation θ of i is *consistent* with F
  4. given two protocols F and F', process i can *distinguish* F and F' if every observation θ of i consistent with F, θ is not consistent with F' and for every θ of i consistent with F', θ not consistent with F
  5. given a protocol F and initial state q_0, *outcome* of F for q_0 is the eventual winner (eventual value of V_i fro all i)
  6. F is *cheat-free* if for all processes i, initial states q_0, cheating protocols F', if F' and F have different outcomes for q_0, then i can distinguish F and F'
- Discussion: do we need "every" in point 4, or "some" is sufficient
- Problem: How can we modify the definitions so that changing your F_i in the first scheduling is still cheat-free/
