Population protocols - Correction

Remark 1. For some exercices, other solutions are possible.

1 Basics

Consider population comprising two species of agents X, Z. Recall that we assume the following fairness condition.

Definition 1 (Global fairness). *An execution E is fair if and only if for every configuration C occurring infinitely often in E, for every configuration C' reachable from C, C' also occurs infinitely often in E.*

Intuitively, it means that if something is reachable infinitely often, then it is actually reached infinitely often.

Question 1. Design a protocol such that for any population size n composed of x agents of species X, z agents of species Z, eventually exactly x agents out of the z agents of type Z are "killed", i.e., set to some special null state $\perp$. If $x \geq z$, then all the agents of type Z must be killed.

Solution. State space $\{X_0, X_1, Z, \perp\}$. Agent with input symbol X (resp. Z) is initialized with state X_0 (resp. Z). The only effective rule is $X_0, Z \to X_1, \perp$. In every other cases, the agents states remains the same. □

Question 2. Same question, but eventually exactly $2 \cdot x$ agents of type Z are killed. Generalize so that eventually $k \cdot x$ agents of type Z are eventually killed, where k is any predefined constant. How the state space depends on k ?

Solution. State space $\{X_0, X_1, X_2, Z, \perp\}$. Agent with input symbol X (resp. Z) is initialized with state X_0 (resp. Z). The only effective rules are $X_0, Z \to X_1, \perp$ and $X_1, Z \to X_2, \perp$. In every other cases, the agents states remains the same.

Generalization: state space $\{X_0, \ldots, X_k, Z, \perp\}$. Agent with input symbol X (resp. Z) is initialized with state X_0 (resp. Z). The effective rules are, for each $0 \leq i < k$, $X_i, Z \to X_{i+1}, \perp$. We see that, with this solution, the memory size (in bits) is $O(\log k)$. □

Question 3. We consider another species Y. Design a protocol such that for any population size n composed of x agents of species X, y agents of species Y and z agents of species Z, eventually exactly $x + y$ agents out of the z agents of types Z are killed.

Solution. State space $\{X, X', Y, Y', Z, \perp\}$. Agent with input symbol X (resp. Y, Z) is initialized in state X (resp. Y, Z). The effective rules are

$$X, Z \to X', \perp$$
$$Y, Z \to Y', \perp$$

In other words, it suffices that each X (resp. Y) kills exactly one Z. □

Question 4. Can you design a protocol such that $x \cdot y$ agents out of the z agents of type Z are eventually killed ? (answer informally)

Solution. The answer is no, because there is not enough memory. Informally, this looks like killing $k \cdot x$ agents out of the Z agents, except that $y = k$ is not a constant any more. In some sense, the requirement that the state space size is independent of the population size is too harsh to compute this predicate. Note, however, that a fully detailed answer to this question is hard. □

2 Presburger arithmetics

Consider an arbitrary (finite) set of species Σ. An input assignment is then represented by a vector $v \in \mathbb{N}^\Sigma$ specifying the number of agents of each species. Using this representation, there is a zero assignment (representing a configuration with zero agents), and assignments can be additioned component-wise. A predicate ψ is then represented by a function on input assignments taking values in $\{0,1\}$: the function takes value 1 iff the input assignment satisfies the predicate. Recall that a predicate ψ is stably computable if there exists a population protocol such that fr any population size, for any input assignment v, eventually all the agents permanently output $\psi(v)$.

Question 5. Show that predicate of the form

$$a_1 \cdot \#X_1 + \cdots + a_k \cdot \#X_k = c \mod m$$

where the a_i's , c, and m are integer coefficients, is stably computable

Solution. The protocol is slightly simpler than the one given in the lecture. Recall that in the lecture, a protocol A was presented to solve the predicate $a_1 \cdot \#X_1 + \cdots + a_k \cdot \#X_k < c$. The idea was to initialize each agent of type X_i with weight a_i. In addition, a leader election mechanism is implemented, and the leader tries to collect the sum of all the weights. But, since the memory size must be independent of the population size, a threshold mechanism was required.

In the present case, things are simpler since it suffices to compute sums of weights modulo m. Formally, the state of an agent is given by

- a leader bit
- an output bit
- a counter taking values in $\{0, \ldots, m-1\}$; an agent with input symbol X_i is initialized with $a_i \mod m$

The rules are summed up by the following.

$$(l, *, u), (l', *, u') \to (1, b, (u + u') \mod m), (0, b, 0)$$

if l or l' is 1; b is 1 if $u + u' = c \mod m$ and 0 otherwise. If both l and l' are zero, then the agents states remains unchanged. The correctness of this protocol is proved in exactly the same way as in the lecture. At some point, there is a unique leader in the system, which eventually collects the sum of all the weights modulo m. At this point, the leader knows the answer to the predicate, and propagates the information. □

Question 6. Show that boolean combinations of stably computable predicates are also stably computable.

Solution. Assume that the protocol A_1 (resp. A_2), with state space Q_1 (resp. Q_2), stably computes the predicate P_1 (resp. P_2). Let $out : Q_1 \cup Q_2 \to \{0,1\}$ denote the map giving the output value associated with each state. We show how to compute the predicate $P_1 \wedge P_2$ (logical and). The basic idea is to run both protocols in parallel, and computing the output based on the output of the two protocols. We define the protocol A as follows. Its state space is $Q = Q_1 \times Q_2 \times \{0,1\}$. The last bit stores the output. The rules are given by

$$(p_1, p_2, *), (q_1, q_2, *) \to (p_1', p_2', x)(q_1', q_2', y) \Leftrightarrow \begin{cases} p_1, q_1 \to p_1', q_1' \text{ is a rule of } A_1 \\ p_2, q_2 \to p_2', q_2' \text{ is a rule of } A_2 \end{cases}$$

where x denotes $out(p_1) \wedge out(p_2)$, and y denotes $out(q_1) \wedge out(q_2)$.

The logical or is implemented accordingly. The negation is even simpler: it suffices to flip the output. □

Now, we assume the following lemma proven in [1].

Lemma 1 (Pumping lemma for population protocols). *Let ψ be a predicate over $\mathbb{N}^\Sigma$. If ψ is stably computable, then there exists a finite number of couples $(v_1, M_1), \ldots, (v_s, M_s)$ such that*

$$\psi(v) = 1 \Leftrightarrow v \in \bigcup_{i=1}^{s} (v_i + M_i)$$

where each $v_i \in \mathbb{N}^\Sigma$ is an input assignment, and each M_i is a set of input assignments containing the zero assignment, and closed under addition.

Question 7. Use the pumping lemma to show that the predicate

$$(\#X)^2 = \#Y$$

is not stably computable.

Solution. Let $V = \{(x, y) \in \mathbb{N}^2,\ \ x^2 = y\}$. By the pumping lemma (here, Σ is the set $\{X, Y\}$), if the above predicate is stably computable, then there exists $(v_1, M_1), \ldots, (v_s, M_s)$ such that

$$V = \bigcup_{i=1}^{s} (v_i + M_i)$$

where each $v_i \in \mathbb{N}^2$, and each $M_i \subseteq \mathbb{N}^2$ contains $(0, 0)$ and is closed under addition.

If all the M_i were reduced to $(0, 0)$, then V would be reduced to the finite set $\{v_1, \ldots, v_s\}$; this is a contradiction since V is infinite. Hence, there is some i such that M_i contains an element $(a, b) \neq (0, 0)$. Let's write $v_i = (\alpha, \beta)$. Since M_i is closed under addition, for all $k \in \mathbb{N}$, the quantity $(k \cdot a, k \cdot b) = (a, b) + \ldots (a, b)$ k times belongs to M_i. Thus, $(\alpha + k \cdot a, \beta + k \cdot b) \in V$ for all k. By the definition of V, this means that for all k

$$(\alpha + k \cdot a)^2 = \beta + k \cdot b$$

But this implies $(a, b) = (0, 0)$; whence a contradiction. □

This last example confirms that the population protocol model is not strong enough to compute multiplication of variables. The following example insists on the fact that the coefficients must be integers. Indeed consider the predicate

$$P :\ \#X \leq \sqrt{2} \cdot \#Y$$

and assume that it is stably computable. Our goal is to derive a contradiction. By the pumping lemma, we thus have

$$M \underset{def}{=} \{(a, b) \in \mathbb{N}^2,\ b\sqrt{2} - a \geq 0\} = \bigcup_{i=1}^{s} (x_i + M_i)$$

For any $u = (a, b), v = (c, d)$ in $\mathbb{N}^2$, we write $u \cdot v = ac + bd$. Let $z = (-1, \sqrt{2})$.

Question 8. – (a) Show that $x \in M$ if and only if $z \cdot x > 0$.
– (b) Let $\epsilon = \min\{z \cdot x_1, \ldots, z \cdot x_s\}$. Show that there exists $y \in M$ such that $0 < z \cdot y < \epsilon$.
– (c) Show that for some i, $(y - x_i) \cdot z < 0$.
– (d) Considering the quantity $(x_i + n(y - x_i)) \cdot z$ for arbitrary large $n \in \mathbb{N}$, derive a contradiction.

Remark 2. My mistake. This exercice is way harder (point b) than I thought.

References

1. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.