Terminating Population Protocols via some Minimal Global Knowledge Assumptions

Paul G. Spirakis

joint work with
Othon Michail
Ioannis Chatzigiannakis

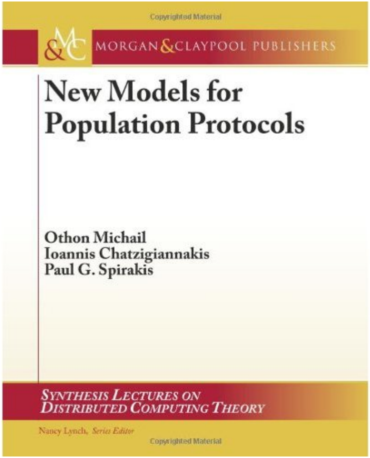
Computer Technology Institute & Press "Diophantus" (CTI)
Dept. of Computer Engineering & Informatics (CEID), Univ. of Patras

14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)
October 1-4, 2012
Toronto, Canada

Population Protocols (PPs)

- n finite-state anonymous agents
- Passively mobile
  - Modeled via a fair adversary scheduler
  - Abstract way to capture probabilistic mobility
- Only stabilizing computations
  - Inability to terminate
  - Agents cannot tell when they have heard from everybody else

Population Protocols



Previous Work

- PPs compute precisely the semilinear predicates [AADFP04]
  - Holds for local space up to $o(\log \log n)$ [CMNPS11]
- Mediated PPs are much more powerful [CMS09]
  - Equivalent to NTMs of space $O(n^2)$
- So are the Community Protocols that extend PPs with unique IDs
  - Equivalent to NTMs of space $O(n \log n)$
- The Stabilizing Inputs variant provides a means of sequentially composing protocols even in the absence of termination [AACFJP05]

Cover-time Service

- We augment PPs with a cover-time service

Definition

The cover-time service (CTS) informs a swapping state every time it covers the whole population.

- The CTS is a natural means of giving to finite-state nodes access to a known bound on the cover time of a random walk
- e.g. in a complete graph the cover time of a random walk is $n \log n$

Cover-time Service

- Imagine now a unique leader-state in the population that jumps from agent to agent
- Intuitively, the leader-state knows an upper bound on the cover time of its own random walk via the CTS
- We call this model the CTS model

Our Goal

- Study the computability of the CTS model
  - Functions on input assignments to the agents
- We do this via a reduction to an oracle model
  - The Absence Detection (AD) model
- The oracle is capable of detecting the presence or absence of every state from the population
- The AD model serves as a convenient abstraction for PP models that have the ability to detect termination

Population Protocol with Absence Detector

A Population Protocol with Absence Detector (AD) is a 7-tuple:

1. X is the input alphabet
2. Y is the output alphabet
3. Q is a set of states
4. $I : X \to Q$ is the input function
5. $\omega : Q \to Y$ is the output function
6. δ is the transition function $\delta : Q \times Q \to Q \times Q$
7. γ is the detection transition function $\gamma : Q \times \{0,1\}^{|Q|} \to Q$

We call a transition every

- $(q_1, q_2) \to (q_1', q_2')$ where $\delta(q_1, q_2) = (q_1', q_2')$ and $q_1, q_2, q_1', q_2' \in Q$ and every
- $(q, a) \to q'$ where $\gamma(q, a) = q'$ and $q, q' \in Q$, $a \in \{0,1\}^{|Q|}$

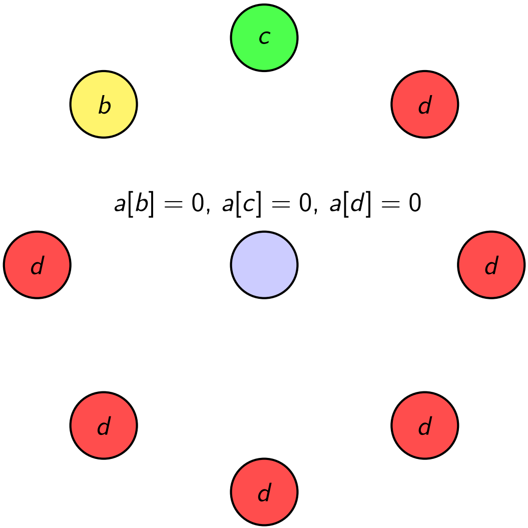
Setting

- Complete interaction graph $G = (V, E)$ (simple, directed)
- Population of n agents
- 1 absence detector
- The state of the absence detector is an absence vector $a \in \{0, 1\}^{|Q|}$ representing the absence or not of each state from the population
- $q \in Q$ is absent from the population in the current configuration iff $a[q] = 1$
- Each agent initially senses its environment receiving an input symbol from X
  - This results in an input assignment $x \in X^n$
- The absence vector is initially
  - $a[q] = 0$ for all $q \in Q$ so that $\exists \sigma_k \in x : I(\sigma_k) = q$ and
  - $a[q] = 1$ for all other $q \in Q$

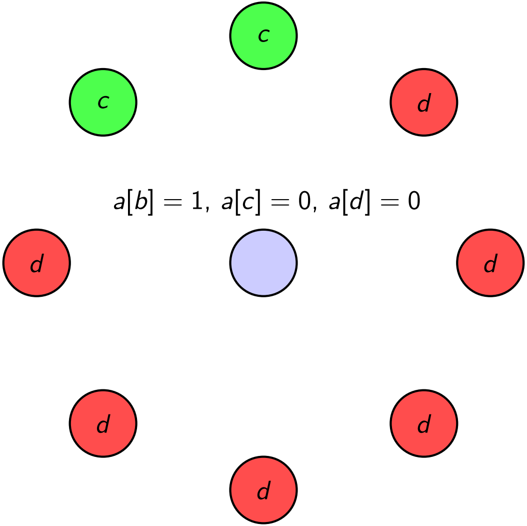
An Example

- $Q = \{b, c, d\}$
- $(c, b) \rightarrow (c, c)$
- $(c, d) \rightarrow (c, c)$
- $a \in \{0, 1\}^3$
  - e.g. $(0, 0, 1)$ means that b and c are present and d is absent

An Example

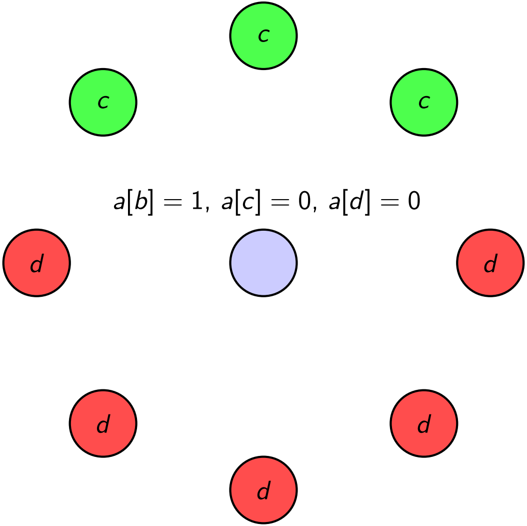


An Example



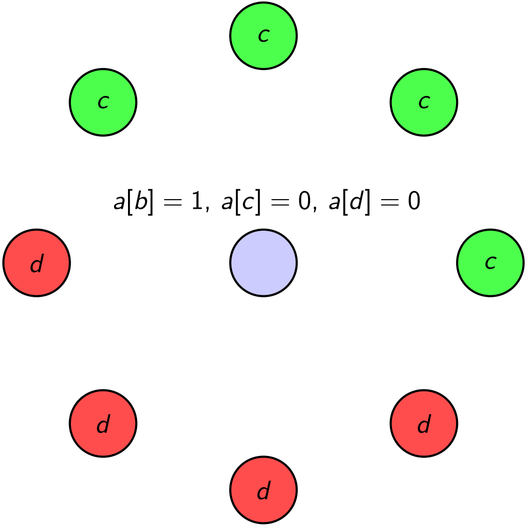
$$a[b] = 1,\ a[c] = 0,\ a[d] = 0$$

An Example



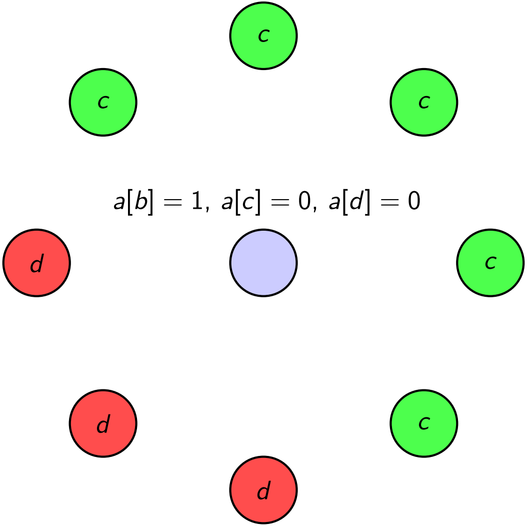
$a[b] = 1,\ a[c] = 0,\ a[d] = 0$

An Example

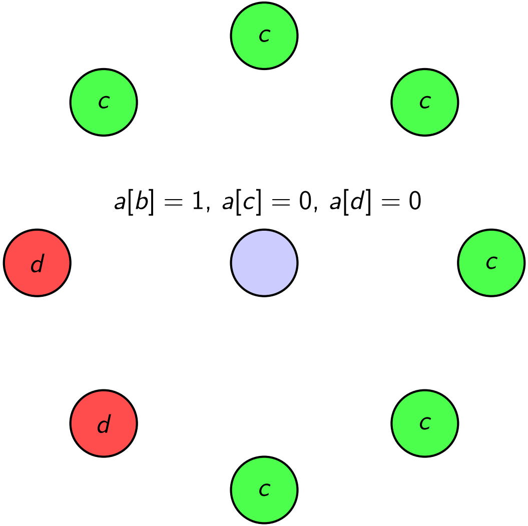


$a[b] = 1, \ a[c] = 0, \ a[d] = 0$

An Example

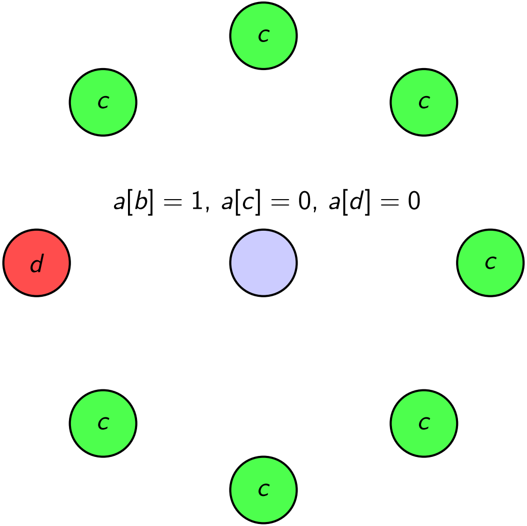


An Example



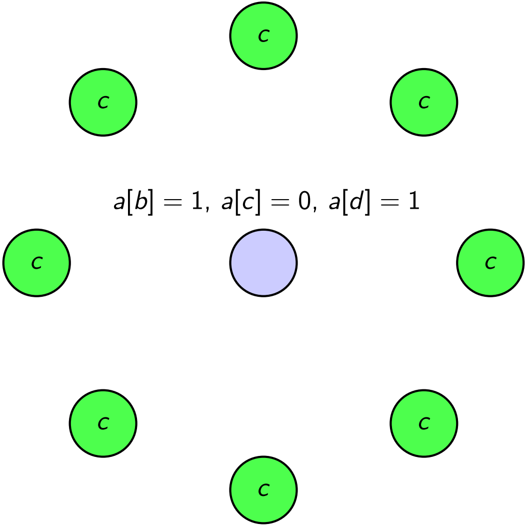
$a[b] = 1,\ a[c] = 0,\ a[d] = 0$

An Example



$a[b] = 1,\ a[c] = 0,\ a[d] = 0$

An Example



$a[b] = 1,\ a[c] = 0,\ a[d] = 1$

A Leader-Election AD

- $X = \{1\}$,
- $Q = \{l, f, q_{halt}\}$,
- $I(1) = f$,
- δ is defined as $(l, f) \rightarrow (l, q_{halt})$, and
- γ as
  - $(f, a) \rightarrow l$, if $a[l] = 1$ and
  - $(l, a) \rightarrow q_{halt}$, if $a[f] = 1$
- The idea is that the 1st agent that meets the absence detector becomes a leader while agents meeting the detector in subsequent rounds remain followers

Interesting Properties

Proposition

Any AD with stabilizing states has an equivalent halting AD.

Proposition

Halting ADs can be sequentially composed.

Proposition

Any AD has an equivalent AD that assumes a unique leader which does not obtain any input.

Computing the Non-Semilinear Predicate ($N_1 = 2^d$)

Protocol 1 *Power of 2*

1: $X = \{1\}, Q = (\{l\} \times \{q_0, q_1, q_2, q_3, q_4\}) \cup (\{n\} \times \{1, \bar{1}, 1'\}) \cup \{q_{accept}, q_{reject}\}$,
2: $I(1) = (n, 1)$ only for the non-leaders,
3: the leader is initialized to (l, q_0),
4: δ:

$$\begin{aligned}
(l, q_0), (n, 1) &\rightarrow (l, q_1), (n, \bar{1}) \\
(l, q_1), (n, 1) &\rightarrow (l, q_2), (n, \bar{1}) \\
(l, q_2), (n, 1) &\rightarrow (l, q_3), (n, 1') \\
(l, q_3), (n, 1) &\rightarrow (l, q_2), (n, \bar{1}) \\
(l, q_4), (n, 1') &\rightarrow (l, q_4), (n, 1)
\end{aligned}$$

5: γ:

$$\begin{aligned}
(l, q_2), a &\rightarrow q_{accept}, \text{ if } a[n, 1] = a[n, 1'] = 1 \\
&\rightarrow (l, q_4), \text{ if if } a[n, 1] = 1 \text{ and } a[n, 1'] = 0 \\
(l, q_3), a &\rightarrow q_{reject}, \text{ if } a[n, 1] = 1 \text{ and } a[n, 1'] = 0 \\
(l, q_4), a &\rightarrow (l, q_1), \text{ if } a[n, 1'] = 1
\end{aligned}$$

Computing the Non-Semilinear Predicate ($N_1 = 2^d$)

- Implements the classical TM algorithm
- The unique leader plays the role of the TM head
- In each iteration it halves the number of remaining 1s
  - by marking red half of them and green the rest
  - then restores the green to begin the next iteration
- The absence detector informs the protocol if the current iteration is complete
  - If no uncolored 1 has remained then the head has visited all 1s

CTS-AD Equivalence

Theorem

The CTS model is computationally equivalent to the leader-AD model.

Proof.

- The CTS-leader may form an absence vector by walking around and keeping track of present states until it covers the whole population
- The AD-leader detects the completion of a covering by marking all nodes that it meets and asking the absence detector whether all nodes have been marked

□

- Thus we may explore the computational power of the CTS model via the AD model

Some Notation

- **SEM**: the class of semilinear predicates
- **HAD**: the class of computable predicates by halting ADs with leader
- k-truncate of a configuration $c \in \mathbb{N}^Q$: $\tau_k(c)[q] := \min(k, c[q])$ for all $q \in Q$

PPs vs ADs

Theorem

SEM *is a proper subset of* **HAD**.

Proof.

- For any stabilizing PP $\exists$ finite k such that a configuration is output stable iff its k-truncate is output stable
- For all finite k and any initial configuration $c \in \mathbb{N}^Q$, there is an AD that aggregates in one agent $\tau_k(c)$
- Let the AD know the k corresponding to the simulated PP
- The AD-leader every l (constant) simulation steps, collects $\tau_k(c)$ and checks whether it is output-stable
  - As k is fixed it can do this in its fixed memory
- Thus, any PP can be simulated by some AD and since there is a non-semilinear AD (power of 2) the theorem follows

□

A Better Lower Bound

Theorem

Any predicate of the form

$$\sum_{d_1,d_2,\ldots,d_k=0}^{l} a_{d_1,d_2,\ldots,d_k} N_1^{d_1} N_2^{d_2} \cdots N_k^{d_k} < c,$$

where $a_{d_1,d_2,\ldots,d_k}$ and c are integer constants and l and k are nonnegative constants, is in **HAD**.

- The construction is based on a protocol for the much simpler $(bN_1^d < c)$

Computing the Predicate ($bN_1^d < c$)

Protocol 2 *VarPower*

1: $X = \{s_1\}, Q = (\{l_1, l_2, \ldots, l_d, l_1^e, l_2^e, \ldots, l_d^e\} \times [c]) \cup \{0,1\}^d \cup \{q_{accept}, q_{reject}\}$,
2: $I(s_1) = 0^d$,
3: the initial state of the leader is $(l_1, -c)$,
4: δ:

$$\begin{aligned}
(l_i, w), (0, u_{-i}) &\to (l_{i+1}, w), (1, u_{-i}), \text{ if } i < d \\
&\to q_{accept}, \text{ if } i = d \text{ and } c \geq 0, w + b \leq -c \text{ or } c < 0, w + b < 0 \\
&\to q_{reject}, \text{ if } i = d \text{ and } c \geq 0, w + b \geq 0 \text{ or } c < 0, w + b \geq -c \\
&\to (l_i, w + b), (1, u_{-i}), \text{ if } i = d \text{ and } c \geq 0, -c \leq w + b < 0 \text{ or} \\
&\qquad\qquad c < 0, 0 \leq w + b < |c| \\
(l_i^e, w), (1, u_{-i}) &\to (l_i^e, w), (0, u_{-i})
\end{aligned}$$

5: γ:

$$\begin{aligned}
(l_i, w), a &\to (l_i^e, w), \text{ if } a[0, u_{-i}] = 1 \text{ and } i > 1 \\
&\to q_{accept}, \text{ if } a[0, u_{-i}] = 1, i = 1 \text{ and } w < 0 \\
&\to q_{reject}, \text{ if } a[0, u_{-i}] = 1, i = 1 \text{ and } w \geq 0 \\
(l_i^e, w), a &\to (l_{i-1}, w), \text{ if } a[1, u_{-i}] = 1
\end{aligned}$$

Simulating a Counter Machine

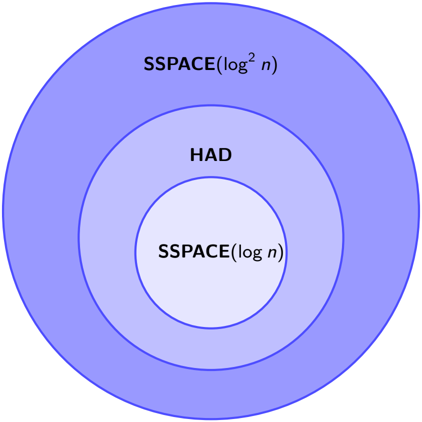
- ADs and one-way (online) counter machines (CMs) can simulate each other

Theorem

SSPACE$(\log n) =$ **SCMSPACE**$(n) \subseteq$ **HAD** $\subseteq$ **SNSPACE**$(\log n) \subseteq$ **SSPACE**$(\log^2 n)$.

- **SSPACE**: deterministic TM space with input commutativity
- **SNSPACE**: nondeterministic TM space with input commutativity
- **SCMSPACE**: deterministic CM space with input commutativity

Our Bounds on **HAD**



Simulating a Counter Machine

- CM: a control unit, an input terminal, and a constant number of counters
- The AD:
  - Simulates the control unit by its unique leader
  - The input slots of the agents simulate the input terminal
  - The k counters are stored by creating a k-vector of bits in the memory of each agent
  - Each counter is distributed across the agents
  - The value of the ith counter at any time is determined by the number of 1s appearing in the ith components of the agents
  - A crucial operation of the CM is to determine the set of strictly positive counters
  - The AD can do the same by detecting the absence of an all-0 component (all agents have 0 in the corresponding place)

Conclusions

- We proposed the CTS model a new extension of PPs that additionally assumes the existence of a cover-time service
- By reduction to the AD oracle model we were able to investigate and almost completely characterize the computational power of the new model
- The introduced minimal global knowledge enables CTS to perform halting computations, a feature that was missing from PPs
- We showed that **HAD** is somewhere between **SSPACE**$(\log n)$ and **SSPACE**$(\log^2 n)$
- In the full paper we also show that ADs can simulate some interesting linear bounded automata

Open Problems

- Give an exact characterization of **HAD**
- Make the AD model fault-tolerant, e.g. self-stabilizing
- What happens in the case where the detector does not always correctly detect the existing states in the population?
- How is the computability of graph properties of the interaction graph affected by the presence of an absence detector?
- Are there other realistic variants of PPs that have the ability to terminate?

Thank You!

References

O. Michail, I. Chatzigiannakis, and P. G. Spirakis.
New Models for Population Protocols.
N. A. Lynch (Ed), Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2011.

D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta.
Computation in networks of passively mobile finite-state sensors.
In *23rd annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 290–299, New York, NY, USA, 2004. ACM.

I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis.
Passively mobile communicating machines that use restricted space.
Theor. Comput. Sci., 412[46]:6469–6483, October 2011.

References

I. Chatzigiannakis, O. Michail, and P. G. Spirakis.
Mediated population protocols.
In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5556 of *LNCS*, pages 363–374. Springer-Verlag, July 2009.

D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, and R. Peralta.
Stably computable properties of network graphs.
In *Distributed Computing in Sensor Systems: First IEEE International Conference DCOSS*, volume 3560 of *LNCS*, pages 63–74. Springer-Verlag, June 2005.