All Symmetric Predicates in $NSPACE(n^2)$ are Stably Computable by the Mediated Population Protocol Model

Stavros Nikolaou

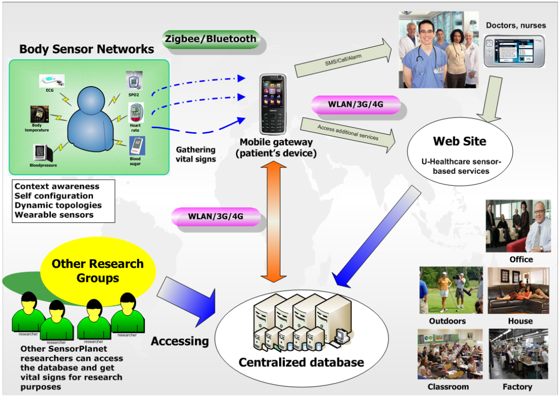
Joint work with: I. Chatzigiannakis, O. Michail
A. Pavlogiannis, P. G. Spirakis

Research Academic Computer Technology Institute (RACTI)
Patras, Greece

MFCS 2010
Brno, Czech Republic
August 2010

Outline I

The Motivation



- Wireless Sensor Networks have received great attention recently due to their wide range of applications.

Population Protocols - An Introduction

- Theoritical models for WSNs have become significantly important in order to understand their capabilities and limitations.
- Population Protocols [Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04] is a model for WSNs where:
  - Each node: limited computational device $\longrightarrow$ a *finite-state* machine + *sensing* + *communicating* device: **agent**.
  - **Passively mobile** agents: incapable to control or predict.
    - How: unstable environment, like water flow or wind, or the natural mobility of their carriers.
- Significant properties:
  - **Uniformity:** Protocol descriptions are independent of the population size. (*constant* size)
  - **Anonymity:** There is no room in the state of an agent to store a unique identifier.

The Population Protocols Model and Characteristics

- Agents interact in pairs according to a **communication graph** $G = (V, E)$ where:
  - V: A **population** of $|V| = n$ agents.
  - E: The permissible interactions between the agents.
- Interaction pattern: **adversary**
- Adversarial choices: **fairness condition**
- **fairness condition**: population partition (the adversary cannot avoid a possible step forever)

Computation

In every execution of a PP(X, Y, Q, I, O, δ):

- Initially: Each agent senses its environment $\longrightarrow$ an input symbol from a finite **input alphabet** X.
  - **input assignment**: tuple specifying an input for each agent.
- the input symbol is mapped by the **input function** $I : X \to Q$ to a state from a finite set of **agent states** Q
  - **population configuration**(C): tuple specifying the state of each agent.
- each state is mapped by the **output function** $O : Q \to Y$ to an output symbol from a finite **output alphabet** Y (agent's output).
- Interaction: **transition function** $\delta : Q \times Q \to Q \times Q \Longrightarrow$ agents update their states according to δ.
  - population configuration(C) changes(C'): goes from C to C' in one step ($C \to C'$).

Stable Computation

- **Computation**: Infinite fair sequence $C_0, C_1, C_2, \ldots$, s.t. $C_i \rightarrow C_{i+1}$ for all i.
- **Population protocols do not halt. They stabilize.**
- **stability**: there is a point/configuration in the computation after which no agent can change its output.
- **stable computation**: regular computation + stabilization

Computational Power

- Due to the minimalistic nature of the model the class of computable predicates is fairly small.
- In [Angluin et al. 2004, 2006] it was proven that it is exactly the class of **semilinear predicates**.
- Predicates such as $N_a \geq 10$ or $N_a < N_b$ capturing scenarios such as the infection of a percentage of a fish population or fire detection by a majority of sensors scattered in a forest.
- This class does not include multiplication, exponentiation and other important operations on input variables.

Modifying the PP model

- The next step is to enhance the model with extra realistic and implementable assumptions in order to gain more computational power.
- **Mediated Population Protocols**: We assume that each edge is equipped with a buffer of $\mathcal{O}(1)$ storage capacity (independent of the size of the population).
- This minor addition greatly enhances the computability of the model.

The Mediated PP model
[I. Chatzigiannakis, O. Michail, and P. G. Spirakis, ICALP '09]

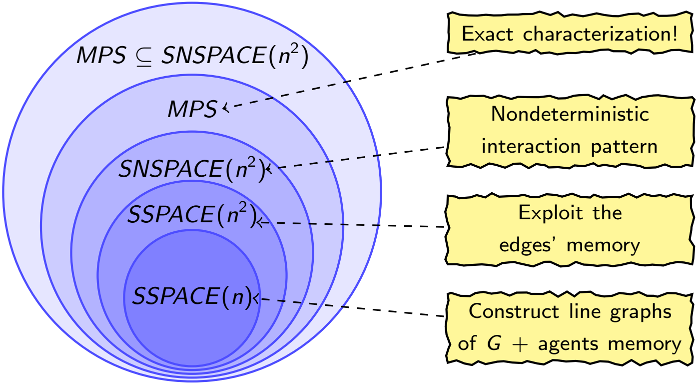
- Model's description:
  - each edge $\longrightarrow$ a state from a finite set of **edge states** S
  - **transition function** $\delta : Q \times Q \times S \to Q \times Q \times S$
    - When agents u_1, u_2 in states a, b, respectively, interact through (u_1, u_2) in state s then $(a, b, s) \to (a', b', s')$ is applied and
    - a goes to a', b goes to b' and s goes to s'.
- Configuration: includes the states of the edges $\longrightarrow$ **network configuration**.
- The *computation* of the model is in every other respect the same as in the PP model.

Symmetric MPP: An MPP variation

- **Symmetric MPP** (SMPP): *complete communication graph* G + *all edges are initially in a common state* s_0.
- An SMPP protocol $\mathcal{A}$: **symmetric predicates** (permuting the input symbols does not affect the outcome of the predicate).
- **Mediated Predicates in the fully Symmetric case** (MPS): class of stably computable predicates.
- Let $SSPACE(f(n))$ and $SNSPACE(f(n))$ be $SPACE(f(n))$'s and $NSPACE(f(n))$'s restrictions to symmetric predicates, respectively.

Road Map

- From [I. Chatzigiannakis *et al.* ICALP, 2009.] we have that all predicates in MPS are also in $SNSPACE(n^2)$.



Correctly Labeled Line Graphs

- *label*: agents and edges (**label component** of the state)
- The labels are used to define line graphs into the communication graph.

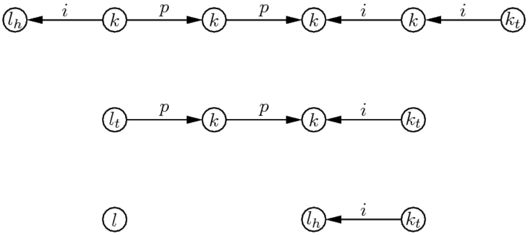


Figure: Some correctly labeled line subgraphs. We assume that all edges not appearing are in state 0 (inactive).

$SSPACE(n) \subseteq MPS$

Theorem (Angluin *et al.*, 2006)

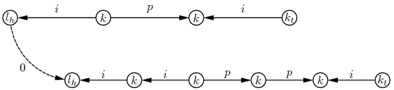
For any predicate $p \in SSPACE(n)$ there is an (M)PP $\mathcal{A}$ that stably computes p in a spanning line graph of n agents.

- SMPP $\mathcal{I}$: constructs a correctly labeled spanning line subgraph of any complete G.
- Initially all agents are simple leaders and all edges are inactive.
- *Idea*: leaders interact via inactive edges $\xrightarrow{\text{labels}}$ merge linegraphs.

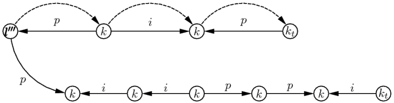
(a) (b) (c) (d)

Figure: (a)-(b): Individual agents create non trivial line subgraphs. (c)-(d): Simple leaders extend already formed line subgraphs.

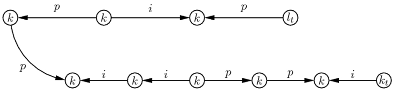
$SSPACE(n) \subseteq MPS$ Merging Process



(a) Before: Leader endpoints interact via an inactive edge.



(b) Update the labels appropriately.



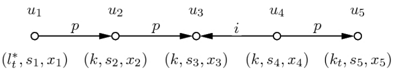
(c) After: The resulting line graph include all agents from both line graphs and an additional edge.

Figure: Two line subgraphs are merged together.

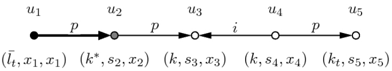
$SSPACE(n) \subseteq MPS$ continued

- By repeatedly merging line graphs, as shown above, we construct a spanning line subgraph of G.
- This process is called **spanning process** and terminates in a finite number of steps.
- We can construct an SMPP $\mathcal{B}$ that is a composition of $\mathcal{A}$ and a protocol $\mathcal{I}$ which is responsible for the spanning process.
- Agents don't know when spanning process ends.
- These protocols run in "parallel" in different state components.
- While the spanning process is ongoing the execution of $\mathcal{A}$ does not take place in a spanning line graph of G. Therefore it has to be **reinitialized** after each merging.

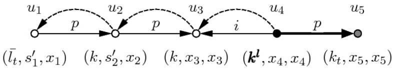
$SSPACE(n) \subseteq MPS$ Reinitialization



(a) Just after merging. The leader endpoint has the special star mark.



(b) The agent who has that mark get reinitialized and passes it to the right neighbor.



(c) The process ends when the tail non-leader gets reinitialized.

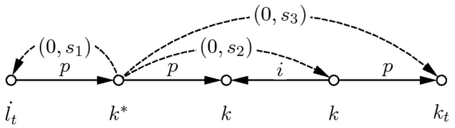
Figure: An example of the reinitialization process. When an agent gets reinitialized, it restores $\mathcal{A}$'s initial state, which it stores in an input backup component of the state.

- The **reinitialization process** terminates in a finite number of steps.
- When the spanning process terminates, $\mathcal{A}$ will be executed in a correctly labeled line subgraph stably computing p.

$SSPACE(n^2) \subseteq MPS$

Theorem

Given a correctly labeled spanning line subgraph of a complete communication graph G, there is MPP A that running on such a graph can simulate a deterministic TM of $\mathcal{O}(n^2)$ space computing symmetric predicates.



(a) The agent in k^* controls now the simulation. The dot label marks k^*'s outgoing inactive edge that will be used by the simulation as a tape cell.

Figure: The simulation uses the inactive edges as tape cells.

$SSPACE(n^2) \subseteq MPS$ continued

- By moving those special labels across the line graph according the movement of the head of the simulated machine we can **access the inactive edges of the communication graph in a systematic fashion**.
- The inactive outgoing edges provide the $\mathcal{O}(n^2)$ **space** needed for the simulation.
- Therefore for any predicate $p \in SSPACE(n^2)$ there is a MPP $\mathcal{A}$ that stably computes it in a correctly labeled line graph of n agents.
- Using the same ideas as before: SMPP $\mathcal{B} = \mathcal{A} + \mathcal{I}$.
- $\mathcal{I}$ also reinitializes inactive edges.

$SNSPACE(n^2) \subseteq MPS$ Construction

- To construct a nondeterministic TM $\mathcal{N}$: **inherent nondeterminism of the interaction pattern**.
- Whenever a nondeterministic choice has to be made the selected choice is determined by the **next arbitrary interaction** of the agent in control of the simulation with another agent.
- If $\mathcal{N}$ rejects then its computation is reinitialized in order to follow another path in the tree representing the nondeterministic computation.
- Fairness guarantees that all the computation paths will eventually be followed.

$SNSPACE(n^2) = MPS$

Theorem

A predicate is in MPS iff it is symmetric and is in $NSPACE(n^2)$

Summary

Our research:

- We have given an **exact characterization** of for the fully symmetric case of the MPP model.
- We showed we can organize a population of tiny artifacts into a TM machine that computes symmetric predicates and **makes use of all the available space from both agents and edges**.

Further research and open problems:

- **Stable decidability of properties of the communication graph** (see e.g. [Chatzigiannakis, Michail, Spirakis, SSS '10 - 2] for a first attempt for MPPs),
- **Expected time complexity** of predicates under some **probabilistic scheduling assumption**
- **Fault-tolerance**

Thank You!

SEVENTH FRAMEWORK PROGRAMME

COMPUTER TECHNOLOGY INSTITUTE CTI
UNIVERSITY OF PATRAS