Recent Advances in Population Protocols

Paul G. Spirakis

Joint work with: O. Michail, I. Chatzigiannakis

Research Academic Computer Technology Institute (RACTI)
Patras, Greece

Invited talk at MFCS 2009
August 2009

Outline I

Outline II

The Systems

- A **communication graph** $G = (V, E)$.
- V: A **population** of $|V| = n$ **agents** (sensor nodes).
- E: The permissible pairwise interactions between the agents.
- Each agent is a self-contained package consisting of
  - sensor, control unit, **constant** memory, low-power wireless communication mechanism, limited power supply.
  - Agents are **passively mobile**.
  - Communicate when they come sufficiently close to each other.
- Agents are represented as **communicating finite-state machines**.

Formal Definition of Population Protocols
[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

A PP $\mathcal{A}$ consists of

- finite **input and output alphabets** X and Y,
- finite set of **states** Q,
- **input function** $I : X \to Q$,
- **output function** $O : Q \to Y$,
- **transition function** $\delta : Q \times Q \to Q \times Q$.

$\delta(p, q) = (p', q')$ or simply $(p, q) \to (p', q')$ is called a **transition**.

Significant Properties

- **Uniformity:** Protocol descriptions are independent of the population size *n*.
- **Anonymity:** There is no room in the state of an agent to store a unique identifier.

Fairness Assumption

- A **communication graph** $G = (V, E)$ describes the permissible interactions.
- First graph considered: **all-pairs** (complete directed communication graph).
    - Corresponding model: **basic population protocol model**.
- Model passive movement by an **adversary scheduler** that picks members of E.
- Make the scheduler "*computation-friendly*" by a fairness assumption
    - Do not allow avoidance of a possible step forever.

Computation

- $C : V \rightarrow Q$, **population configuration** specifying the state of each agent
  - $C \xrightarrow{e} C'$, C can go to C' in one step via encounter $e \in E$.
  - $C \rightarrow C'$, C can go to C' in one step, i.e there exists $e \in E$ s.t. $C \xrightarrow{e} C'$.
  - $C \xrightarrow{*} C'$, C' is reachable from C (there exists sequence $C = C_0, C_1, \ldots, C_k = C'$ s.t. $C_i \rightarrow C_{i+1}$ for all $0 \leq i < k$).

Execution: Finite or infinite sequence $C_0, C_1, C_2, \ldots,$ s.t. $C_i \rightarrow C_{i+1}$ for all i.
Fairness Formally: For all C, C' s.t. $C \rightarrow C'$, if C occurs infinitely often in the execution the same holds for C'.
Computation: Infinite fair execution.

Why use in Sensor Nets

- Each agent is severely limited.
    - Constant storage capacity (independent of n).
- The designer cannot control the interactions between agents.
- Models societies of networked mobile tiny artefacts.
    - Limited sensing, signal processing and communication capabilities.
    - Limited energy.
- Pervasive, adaptive and scalable systems.

Stable Computation

- **Population protocols do not halt**.
- No fixed time to view the output of the population.
- Instead we talk about **stability**.
- Stability ensures that the computation reaches a point after which no agent can change its output.

Stable Computation

- $\mathcal{X}$: the set of all input assignments.
- $\mathcal{Y}$: the set of all output assignments.
- If $x \in \mathcal{X}$ then $I(x)$ is the **input configuration** corresponding to x.
- Each configuration C has a corresponding **output assignment** $O(C) = y_C \in \mathcal{Y}$.

Stable Computation

- C is **output-stable** if $O(C') = O(C)$ for all C' reachable from C (no agent changes its output).
- A computation that contains an output-stable configuration C **stabilizes to output** y_C.

Stable Computation

- A PP $\mathcal{A}$ **stably computes a function** $F_{\mathcal{A}} : \mathcal{X} \rightarrow \mathcal{Y}$ iff for every $x \in \mathcal{X}$ and every computation that starts from $I(x)$ (initial configuration) the computation reaches an output-stable configuration C, where $O(C) = F_{\mathcal{A}}(x)$.
- If $F_{\mathcal{A}} : \mathcal{X} \rightarrow \{0, 1\}$ and $Y = \{0, 1\}$, then we call $F_{\mathcal{A}}$ a **stably computable predicate**.
  - **predicate output convention**: we require that all agents eventually agree on the correct output value.

A Canonical Example

- Assume a complete G.

"*Find if at least* 5 *sensors have detected elevated temperature*."

- Each agent senses the temperature of a distinct bird after a global start signal.
- If detected elevated temperature input 1, else 0 (i.e. $X = \{0, 1\}$).
- We want every agent to eventually output
  - 1, if at least 5 birds were found sick,
  - 0, otherwise.

A Protocol

- $X = Y = \{0, 1\}$, $Q = \{q_0, q_1, \ldots, q_5\}$,
- $I(0) = q_0$ and $I(1) = q_1$,
- $O(q_i) = 0$, for $0 \leq i \leq 4$, and $O(q_5) = 1$,
- δ:

$$\begin{aligned}(q_i, q_j) &\rightarrow (q_{i+j}, q_0), \text{ if } i + j < 5\\ &\rightarrow (q_5, q_5), \text{ otherwise.}\end{aligned}$$

Why it Works...

- Due to fairness, all agents with non-zero state index will eventually interact with each other.
- In each such interaction one of them keeps the sum.
- All indices are eventually aggregated in one agent's state index j (assuming that no faults can happen).
- If $j < 5$, then q_5 cannot occur, thus no agent ever outputs 1.
- Otherwise, state q_5 appears and floods the population (2nd rule), i.e. eventually every agent outputs 1.

Semilinear Predicates - Presburger Arithmetic

Semilinear Predicates

- A predicate on input assignments is **semilinear** if its support (input assignments mapped to 1) is a semilinear set.

Presburger Arithmetic [Presburger 1929]

- Arithmetic on natural numbers with addition **but not multiplication**.
- Formulas involving addition, $<$, mod-k congruence relation $\equiv_k$ for each constant k and usual logical connectives $\vee$, $\wedge$ and $\neg$.

Semilinear sets are those that can be defined in Presburger arithmetic [Ginsburg and Spanier, 1966].

Exact Characterization

Theorem

A predicate is computable in the basic population protocol model if and only if it is semilinear [Angluin et al. 2004, 2006].

Stably Computable (semilinear)

- "The number of a's is greater than 5" (i.e. $N_a > 5$).
- $(N_a = N_b) \vee (\neg(N_b > N_c))$.

Non-stably computable (non-semilinear)

- "The number of c's is the product of the number of a's and the number of b's" (i.e. $N_c = N_a \cdot N_b$).

Formal Definition of Mediated Population Protocols
[I. Chatzigiannakis, O. Michail, and P. G. Spirakis, ICALP '09]

Population V of $|V| = n$ agents forming a communication graph $G = (V, E)$. A MPP $\mathcal{A}$ consists of

- finite **input and output alphabets** X and Y,
- finite set of **agent states** Q, **agent input function** $I : X \rightarrow Q$, **agent output function** $O : Q \rightarrow Y$,
- finite set of **edge states** S, **edge input function** $\iota : X \rightarrow S$, **edge output function** $\omega : S \rightarrow Y$,
- **output instruction** r,
- (**totally ordered cost set** K, **cost function** $c : E \rightarrow K$) optional, and
- **transition function** $\delta : Q \times Q \times K \times S \rightarrow Q \times Q \times K \times S$.

Mediated Population Protocols

Transition function $\delta : Q \times Q \times S \rightarrow Q \times Q \times S$

- Assume that costs are not defined.
- When agents u_1, u_2 in states a, b, respectively, interact through (u_1, u_2) in state s then $(a, b, s) \rightarrow (a', b', s')$ is applied and
  - a goes to a',
  - b goes to b', and
  - s goes to s'.

New Assumptions about Edges...

- We assume that each edge is equipped with a buffer of $\mathcal{O}(1)$ storage capacity (independent of the population).
    - **Each pair of communicating agents shares a memory of constant size**.
- During interaction (u, v) the corresponding agents read the memory contents and update it according to δ.

Network Configurations

- A **network configuration** is a mapping $C : V \cup E \rightarrow Q \cup S$ specifying the agent state of each agent in the population and the edge state of each edge in the communication graph.

r-stability

- **Problem**: "*Given an undirected communication graph $G = (V, E)$ and a useful cost function $c : E \rightarrow K$ on the set of edges, design a protocol that will find the minimum cost edges of E*".
- **Example of instruction** r: "*Get each $e \in E$ for which $\omega(s_e) = 1$ (where s_e is the state of e)*".

A configuration C is called **r-stable** if one of the following holds:

- If the problem concerns a subgraph to be found, then C should fix a subgraph that will not change in any configuration C' reachable from C.
- If the problem concerns a function to be computed by the agents, then an r-stable configuration drops down to an agent output-stable configuration.

- **Permits protocols that construct subgraphs of G.**

Stable Computation

Definition

A protocol $\mathcal{A}$ **stably solves** a problem Π, if for every instance I of Π and every computation of $\mathcal{A}$ on I, the network reaches an r-stable configuration C that gives the correct solution for I if interpreted according to the output instruction r. If Π is a function f to be computed we say instead that $\mathcal{A}$ **stably computes** f.

Definition

In the special case where Π is an optimization problem (like *minimum cost edges*), a protocol that stably solves Π will be called an **optimizing population protocol** for problem Π.

Formal Definition of Approximation Protocols

Definition

Let Π be a minimization problem, and let δ be a positive real number, $\delta \geq 1$. A protocol $\mathcal{A}$ is said to be a δ **factor approximation protocol** for Π if on each instance I of Π, and every fair execution of $\mathcal{A}$ on I, the network reaches an r-stable configuration C, that if interpreted according to the output instruction r of $\mathcal{A}$ gives a feasible solution s for I such that:

$$f_\Pi(I, s) \leq \delta \cdot \mathrm{OPT}(I).$$

Maximal Matching Problem

Problem

*(**Maximal matching**) Given an undirected communication graph $G = (V, E)$, find a **maximal matching**, i.e., a set $E' \subseteq E$ such that no two members of E' share a common end point in V and, moreover, there is no $e \in E - E'$ such that e shares no common end point with every member of E'.*

A protocol

MaximalMatching

- $X = \{0\}$,
- $Y = \{0, 1\}$,
- $Q = \{q_0, q_1\}$,
- $I(0) = q_0$,
- $O(q_0) = 0,\ O(q_1) = 1$,
- $S = \{0, 1\}$,
- $\iota(0) = 0$,
- $\omega(0) = 0,\ \omega(1) = 1$,
- r: "*Get each $e \in E$ for which $\omega(s_e) = 1$ (where s_e is the state of e)*",
- δ:

$$(q_0, q_0, 0) \rightarrow (q_1, q_1, 1)$$

Proof of correctness

Theorem

Protocol *MaximalMatching* stably solves the maximal matching problem.

Proof.

M: set of edges in state 1. Two interactions happening in parallel cannot concern adjacent edges (natural assumption about scheduler). $e = (u, v)$ gets in M iff both endpoints are in q_0. Then u, v go to q_1 to indicate adjacency with an edge of M. So any edge conflicting with M cannot get in M and M is always a matching. Any edge $e' = (u', v')$ not conflicting with M will eventually get in M because of fairness and because $q(u') = q(v') = q_0$ and $s(e') = 0$, so M eventually will become maximal. □

Minimum Vertex Cover Problem

Problem

*(**Minimum vertex cover**) Given an undirected communication graph $G = (V, E)$, find a minimum cardinality **vertex cover**, i.e., a set $V' \subseteq V$ such that every edge has at least one end point incident at V'.*

- The protocol *VertexCover* agrees on everything to *MaximalMatching* except for a new instruction r: "*Get each $\upsilon \in V$ for which $O(q_\upsilon) = 1$ (where q_υ is the state of υ)*".

Theorem: *By collecting the end points of edges in M, VertexCover protocol is clearly a 2 approximation protocol for the minimum vertex cover problem [see Vazirani 2001].*

Transitive Closure

Problem

*(**Transitive Closure**) Given a communication graph $G = (V, E)$ in $\mathcal{G}^d_{All}$ with a precomputed subgraph $G' = (V', E')$, find the transitive closure of G', that is, find a new edge set E^* that will contain a directed edge (u, υ) joining any nodes u, υ for which there is a non-null path from u to υ in G' (note that always $E' \subseteq E^*$).*

- We assume the existence of a **unique leader** in state l in the initial configuration.

A Protocol

TranClos

- $X = Y = \{0, 1\}$,
- $Q = \{l, q_0, q_1, q_1', q_2, q_2', q_3\}$, $S = \{0, 1\}$,
- *controlled input assignment*: "$W(e') = 1$, *for all* $e' \in E'$, *and* $W(e) = 0$, *for all* $e \in E - E'$",
- $\iota(x) = x$, for all $x \in X$, $\omega(s) = s$, for all $s \in S$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ:

$$\begin{array}{rcl} (l, q_0, 0) & \rightarrow & (q_0, l, 0) \\ (l, q_0, 1) & \rightarrow & (q_1, q_2, 1) \\ (q_1, q_2, 1) & \rightarrow & (q_0, l, 1) \end{array} \qquad \begin{array}{rcl} (q_2, q_0, 1) & \rightarrow & (q_2', q_3, 1) \\ (q_1, q_3, x) & \rightarrow & (q_1', q_0, 1), \text{ for } x \in \{0, 1\} \\ (q_1', q_2', 1) & \rightarrow & (q_0, l, 1) \end{array}$$

Why it Works...

- Initially (due to the controlled input assignment) all arcs in E' are in state 1 and all the other arcs in state 0.
- The protocol tries to find two consecutive arcs (e.g. (u_1, u_2) and (u_2, u_3)) both in state 1.
- If it does, it waits until interaction (u_1, u_3) happens. Then (u_1, u_3) goes to state 1 (transivity) and the population remains again with a unique leader.
- Eventually, due to fairness, all consequtive arcs in state 1 will be visited by the leader and the transitive closure is correctly constructed.
- The leader does not get stuck since the rule $(q_1, q_2, 1) \rightarrow (q_0, l, 1)$ backtracks the protocol.

MPP is stronger than PP

- Obviously, PP model is a special case of MPP model.
    - Ignore edge functions, states, costs, and instruction r to get PP.
- The edge buffers enable each pair of agents to remember a **pairwise history of up to $\mathcal{O}(1)$ interactions**.

For Example...

Assume a complete directed graph $G = (V, E)$.

- $N_c = N_a \cdot N_b$.
- "**The number of c's is the product of the number of a's and the number of b's**"
- Rephrase it: "**Is N_c equal to the number of links leading from agents with input a to agents with input b?**"
- This predicate is **not semilinear**, since Presburger arithmetic does not allow multiplication of variables.

$N_c = N_a \cdot N_b$ protocol

VarProduct

- $X = \{a, b, c, 0\}$,
- $Y = \{0, 1\}$,
- $Q = \{a, \dot{a}, b, c, \bar{c}, 0\}$,
- $I(x) = x$, for all $x \in X$,
- $O(a) = O(b) = O(\bar{c}) = O(0) = 1$, and $O(c) = O(\dot{a}) = 0$,
- $S = \{0, 1\}$,
- $\iota(x) = 0$, for all $x \in X$,
- r: *"If there is at least one agent with output 0, reject, else accept."*,
- δ:

$$(a, b, 0) \rightarrow (\dot{a}, b, 1)$$
$$(c, \dot{a}, 0) \rightarrow (\bar{c}, a, 0)$$
$$(\dot{a}, c, 0) \rightarrow (a, \bar{c}, 0)$$

Proof Sketch

- For each a the protocol tries to erase b c's.
- Each a is able to remember the b's that has already counted by marking the corresponding links.
- If the c's are less than the product then at least one $\dot{a}$ remains and if the c's are more at least one c remains. In both cases at least one agent that outputs 0 remains.
- If $N_c = N_a \cdot N_b$ then every agent eventually outputs 1.

Discussion

- A MPP **strongly stably computes** a predicate if in every computation all agents eventually agree on the correct output value.
- This is not the case for *VarProduct* (sometimes only one agent eventually gives output 0 and the answer is “reject”).
- But it is easy to see that that *VarProduct* has **stabilizing states**:
    - In every computation all agents eventually stop changing their state (stronger than stabilizing outputs).
- Moreover, instruction r defines a **semilinear predicate on multisets of *VarProduct*’s states** (we can write it formally as $(N_c > 0) \vee (N_{\dot{a}} > 0)$).
- We exploit these properties to prove that with a slight modification *VarProduct* strongly stably computes $N_c = N_a \cdot N_b$.

Composition Theorem

Theorem

Any MPP $\mathcal{A}$, that stably computes a predicate p with stabilizing states in some family of directed and connected communication graphs $\mathcal{G}$, containing an instruction r that defines a semilinear predicate t on multisets of $\mathcal{A}$'s agent states, can be composed with a provably existing MPP $\mathcal{B}$, that strongly stably computes t with stabilizing inputs in $\mathcal{G}$, to give a new MPP $\mathcal{C}$ satisfying the following properties:

- *$\mathcal{C}$ is formed by the composition of $\mathcal{A}$ and $\mathcal{B}$,*
- *its input is $\mathcal{A}$'s input,*
- *its output is $\mathcal{B}$'s output, and*
- *$\mathcal{C}$ strongly stably computes p in $\mathcal{G}$.*

Non-uniform Upper Bounds

Let *DMP* be the class of predicates stably computable by the MPP model in any family of directed communication graphs.

Theorem

All predicates in DMP are also in $NSPACE(m)$.

Proof

Any network configuration can be represented explicitly by storing a state per node and a state per edge. This takes $\mathcal{O}(m)$ *space since* G *is always connected. Any stably computable predicate corresponds to a unique stably computable language (its support). Let* L *be a stably computable language. We present a NTM* M_A *that decides* L *in space* $\mathcal{O}(m)$. *To accept input* x, M_A *must verify two conditions:*

1. *That there exists a configuration* C *reachable from* $I(x)$ *satisfying* r.
2. *There is no* C' *reachable from* C *in which* r *is violated.*

Non-uniform Upper Bounds

The first condition is verified by guessing C_{i+1} to replace C_i. Obviously, a C satisfying r can be found by using at most $\mathcal{O}(m)$ space in any branch of the M_A's computation. The second condition is the complement of the similar reachability problem: "There is some C' reachable from C in which r is violated" and NSPACE is closed under complement for all space functions $\geq \log n$ [Immerman 1988]. $\square$

Graph Decision MPPs (GDM model)
[I. Chatzigiannakis, O. Michail, and P. G. Spirakis, To appear in DISC '09]

A **GDM** is simply a special case of MPP.

- binary **output alphabet** $Y = \{0, 1\}$,
- finite set of **agent states** Q,
- **agent output function** $O : Q \rightarrow Y$,
- finite set of **edge states** S,
- **output instruction** r,
- **initial agent state** $q_0 \in Q$,
- **initial edge state** $s_0 \in S$, and
- **transition function** $\delta : Q \times Q \times S \rightarrow Q \times Q \times S$.

Initially all agents are in q_0 and all edges in s_0 (**no input**).

Graph Languages

- A **graph universe** $\mathcal{U}$ is any set of communication graphs.
- A **graph language** L is a subset of $\mathcal{U}$ containing communication graphs that **share some common property**.
  - e.g. $L = \{G \in \mathcal{U} \mid G$ contains a directed hamiltonian path$\}$.

Deciding Graph Languages

Definition

A GDM $\mathcal{A}$ **stably computes** a predicate $p : \mathcal{U} \rightarrow \{0, 1\}$ with the **predicate output convention**, if for any input graph $G \in \mathcal{U}$ s.t. $p(G) = 1$ and any computation of $\mathcal{A}$ on G the protocol reaches an output-stable configuration C, where $y_C(u) = 1$ for all $u \in V$ and for any $G' \in \mathcal{U}$ s.t. $p(G') = 0$ and any computation of $\mathcal{A}$ on G' the protocol reaches an output-stable configuration C', where $y_{C'}(u) = 0$ for all $u \in V$.

Definition

We say that a GDM $\mathcal{A}$ **decides** a graph language L if it stably computes the predicate $p : \mathcal{U} \rightarrow \{0, 1\}$, defined as $p(G) = 1$ iff $G \in L$, with the predicate output convention (i.e. in every computation, all agents eventually agree in the correct output value).

Decidable Graph Languages

Some examples:

- Node Parity.
- Edge Parity.
- Any node has less than $k = \mathcal{O}(1)$ outgoing neighbors (bounded out-degree).
- Some node has more incoming than outgoing neighbors.
- G has some directed path of length at least $k = \mathcal{O}(1)$.

Path of Constant Length

Theorem

*(**Directed Path of Constant Length**) The graph language $P_k = \{G \in \mathcal{G} \mid G$ has at least one directed path of at least k edges$\}$ is decidable for any $k = \mathcal{O}(1)$ (the same holds for $\overline{P}_k$).*

Proof

If $k = 1$ the protocol that decides P_1 is trivial, since it accepts iff at least one interaction happens (in fact it can always accept since all graphs have at least two nodes and they are weakly connected, and thus $P_1 = \mathcal{G}$). We give a general protocol, DirPath, that decides P_k for any constant $k > 1$.

Path of Constant Length

DirPath

- $Q = \{q_0, q_1, 1, \ldots, k\}$, $S = \{0, 1\}$,
- $O(k) = 1$, $O(q) = 0$, *for all* $q \in Q - \{k\}$,
- *r: "Get any* $u \in V$ *and read its output",*
- δ:

$$\begin{aligned}
(q_0, q_0, 0) &\rightarrow (q_1, 1, 1) \\
(q_1, x, 1) &\rightarrow (x-1, q_0, 0), \ \textit{if } x \geq 2 \\
&\rightarrow (q_0, q_0, 0), \ \textit{if } x = 1 \\
(x, q_0, 0) &\rightarrow (q_1, x+1, 1), \ \textit{if } x+1 < k \\
&\rightarrow (k, k, 0), \ \textit{if } x+1 = k \\
(k, \cdot, \cdot) &\rightarrow (k, k, \cdot) \\
(\cdot, k, \cdot) &\rightarrow (k, k, \cdot)
\end{aligned}$$

Path of Constant Length

- *The protocol* **expands non-communicating active paths**.
- *The* **head** *of each path p* **counts its length** l_p.
- *If* l_p *becomes equal to k then state k giving output* 1 *is propagated.*
- *All the other states give output* 0.
- *To avoid getting stuck, the protocol keeps* **backtracking** *and even totally* **releasing the active paths** *(the graph is not necessarily complete).*
- *Due to fairness, if a path of length at least k exists, then DirPath eventually finds it.*

□

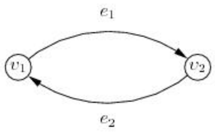
Closure Results

The **class of decidable graph languages** (those for which exists some GDM decider) is **closed under**

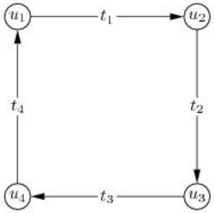
- **complement**,
- (constant) **union**, and
- (constant) **intersection** operations.

An Impossibility Result

- Here the universe is $\mathcal{G}$ containing all **weakly connected graphs**.
- $2C = \{G \in \mathcal{G} \mid G$ has at least two nodes u, υ s.t. both $(u, \upsilon), (\upsilon, u) \in E(G)$ (in other words, G has at least one 2-cycle)$\}$.



(a) Graph G



(b) Graph G'

Figure: $G \in 2C$, $G' \notin 2C$.

An Impossibility Result

Lemma

For any GDM $\mathcal{A}$ and any computation (infinite fair execution) $C_0, C_1, C_2, \dots$ of $\mathcal{A}$ on G (Figure 1(a)) there exists a computation $C'_0, C'_1, C'_2, \dots, C'_i, \dots$ of $\mathcal{A}$ on G' (Figure 1(b)) s.t.

$$\begin{aligned} C_i(\upsilon_1) &= C'_{2i}(u_1) = C'_{2i}(u_3) \\ C_i(\upsilon_2) &= C'_{2i}(u_2) = C'_{2i}(u_4) \\ C_i(e_1) &= C'_{2i}(t_1) = C'_{2i}(t_3) \\ C_i(e_2) &= C'_{2i}(t_2) = C'_{2i}(t_4) \end{aligned}$$

for any finite $i \geq 0$.

Proof: The proof is by induction on i (completely technical).

An Impossibility Result

Theorem

There exists no GDM with **stabilizing states** *to decide the graph language* $2C$.

An Impossibility Result

Proof.

- Assume that a GDM $\mathcal{A}$ decides $2C$ with stabilizing states.
- When $\mathcal{A}$ runs in a fair manner on G, after finitely many steps all agents output the value 1 (i.e. $\mathcal{A}$ **"accepts"** G).
- But according to the previous Lemma there exists some unfair execution of $\mathcal{A}$ on G' simulating that of $\mathcal{A}$ on G.
- Since the states of $\mathcal{A}$ on G have stabilized there exists no transition to fix the wrong decision that $\mathcal{A}$ has made on G'.
- If we allow the scheduler that runs $\mathcal{A}$ on G' to decome fair now (leading to a computation) we have a fair execution that also **"accepts"** G'.
- But this is a **CONTRADICTION**: By assumption $\mathcal{A}$ decides $2C$ so it **cannot "accept"** $G' \notin 2C$. □

An Immediate Consequence

Corollary

There exists no GDM that in any computation on any $G \in 2C$ stabilizes to a single state, e.g. by propagating an alert state when it finds a 2-cycle, to decide the graph language $2C$.

A General Impossibility Result

Here the universe is $\mathcal{H}$ containing all **directed graphs** (**also those that are disconnected**).

Lemma

For any nontrivial graph language L (i.e. $L \neq \emptyset$ and $L \neq \mathcal{H}$), there exists some disconnected graph G in L where at least one component of G does not belong to L or there exists some disconnected graph G' in $\overline{L}$ where at least one component of G' does not belong to $\overline{L}$ (or both).

A General Impossibility Result

Proof.

If the statement **does not hold**: Any disconnected graph in L has all its components in L and any disconnected graph in $\overline{L}$ has all its components in $\overline{L}$.

1. All connected graphs belong to L. Then $\overline{L}$ contains at least one disconnected graph (since it is nontrivial) that has all its components in L. **CONTRADICTION**
2. All connected graphs belong to $\overline{L}$. **CONTRADICTION by symmetry**
3. L and $\overline{L}$ contain connected graphs G and G', respectively. Their disjoint union $U = (V \cup V', E \cup E')$ is disconnected, belongs to L or $\overline{L}$ but one of its components belongs to L and the other to $\overline{L}$. **CONTRADICTION - by assumption both components should belong to the same language**

$\square$

A General Impossibility Result

Theorem

Any nontrivial graph language $L \subset \mathcal{H}$ is undecidable by GDM.

Proof.

Assume that GDM $\mathcal{A}$ decides a nontrivial graph language L. Closure under complement implies that $\exists$ GDM $\mathcal{B}$ deciding $\overline{L}$. By previous Lemma $\exists$ disconnected G in L with some component in $\overline{L}$ or in $\overline{L}$ with some component in L.

1. L contains such a G. Since $\mathcal{A}$ decides L all agents of G should eventually answer "accept". But $\mathcal{A}$ runs on a component that belongs to $\overline{L}$ whose agents cannot communicate with the other components. Thus $\mathcal{A}$ answers "reject" in this component. **CONTRADICTION**
2. $\overline{L}$ contains such a G. **CONTRADICTION by symmetry**

□

An Immediate Consequence

Corollary

GDM cannot decide Connectivity.

Summary

- **Population Protocol model** was the **first step** in this widely unexplored field of societies of tiny networked artefacts.

Our research:

- **Mediated Population Protocol model**: A natural extension of the basic model gives birth to a **promising new area of research**.
- Many new directions: **finding subgraphs**, **deciding graph properties**, **optimization**, **approximation**...
- Moreover the MPP model is computationally stronger than the Population Protocol model.
- Some useful graph languages are decidable by MPPs but many others seem difficult without a leader.
- Nothing is decidable if the universe contains disconnected graphs.
- **Verification** is the key for applying such protocols in real, critical systems.

Open Problems

- **Major Open Problem**: An exact characterization of the class of decidable graph languages.
  - An alternative: **A general method for impossibility results that suits the GDM model**.
  - **Ad-hoc proofs require a lot of effort**.
- The adopted notion of fairness (Angluin et al.) seems to cause **performance and correctness problems** in the case of probabilistic schedulers.
  - Our research on those problems will appear soon...
- Is there some other notion of fairness that would be more suitable for real-life applications?
  - Can it resolve the existing problems?
- The GDM model that aditionally assumes a **unique leader** seems to be stronger (no proof exists).

FRONTS

- This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).
- FRONTS is a joint effort of eleven academic and research institutes in foundational algorithmic research in Europe.
- The effort is towards establishing the **foundations of adaptive networked societies of tiny artefacts**.

Thank You!