The Computational Power of Simple Protocols for Self-Awareness on Graphs

Othon Michail
Joint work with:
Ioannis Chatzigiannakis
Stavros Nikolaou
Paul Spirakis

Computer Technology Institute & Press (CTI)

SSS 2011
October 2011
Grenoble, France

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Population Protocols

[AADFP, PODC '04]

- n mobile agents (finite-state)
- mobility
  - passive: uncontrollable by the agents
  - fair: protocols make progress
- rendezvous communication mechanism (ordered)
- no time-guarantee on causal influence: protocols can only stabilize
- Fairly weak model on complete graphs: semilinear predicates [AAE, PODC '06] even if memory becomes $o(\log \log n)$ [CMNPS, FOMC '11]

Mediated Population Protocols

[CMS, ICALP '09]

- An additional limited mediator:
  - modeled by a constant memory between each pair of agents
  - e.g. a fixed meeting point for each pair
  - or a limited global storage
- Very strong model on complete graphs: symmetric predicates in **NSPACE**(n^2) [CMNPS, MFCS '10].

Mediated Population Protocols

[CMS, ICALP '09]

- An additional limited mediator:
  - modeled by a constant memory between each pair of agents
  - e.g. a fixed meeting point for each pair
  - or a limited global storage
- Very strong model on complete graphs: symmetric predicates in **NSPACE**(n^2) [CMNPS, MFCS '10].

Mediated Population Protocols

[CMS, ICALP '09]

- An additional limited mediator:
  - modeled by a constant memory between each pair of agents
  - e.g. a fixed meeting point for each pair
  - or a limited global storage
- Very strong model on complete graphs: symmetric predicates in **NSPACE**(n^2) [CMNPS, MFCS '10].

Mediated Population Protocols

[CMS, ICALP '09]

- An additional limited mediator:
  - modeled by a constant memory between each pair of agents
  - e.g. a fixed meeting point for each pair
  - or a limited global storage
- Very strong model on complete graphs: symmetric predicates in **NSPACE**(n^2) [CMNPS, MFCS '10].

Mediated Population Protocols

[CMS, ICALP '09]

- An additional limited mediator:
  - modeled by a constant memory between each pair of agents
  - e.g. a fixed meeting point for each pair
  - or a limited global storage
- Very strong model on complete graphs: symmetric predicates in **NSPACE**(n^2) [CMNPS, MFCS '10].

Mediated Graph Protocols

MPPs and additionally:

- a complete communication graph $K_n = (V, E)$
- an input graph $G_\iota[K_n] = (V', E')$ specified by some network initialization function $\iota : V \cup E \to \{q_0, q_1, s_0, s_1\}$
- We are interested in protocols that (stably) decide properties of the input graph
- Different than GDMPPs [CMS, SSS '10]: There the population had to decide properties of the whole communication graph

Mediated Graph Protocols

MPPs and additionally:

- a complete communication graph $K_n = (V, E)$
- an input graph $G_\iota[K_n] = (V', E')$ specified by some network initialization function $\iota : V \cup E \to \{q_0, q_1, s_0, s_1\}$
- We are interested in protocols that (stably) decide properties of the input graph
- Different than GDMPPs [CMS, SSS '10]: There the population had to decide properties of the whole communication graph

Mediated Graph Protocols

MPPs and additionally:

- a complete communication graph $K_n = (V, E)$
- an input graph $G_\iota[K_n] = (V', E')$ specified by some network initialization function $\iota : V \cup E \to \{q_0, q_1, s_0, s_1\}$
- We are interested in protocols that (stably) decide properties of the input graph
- Different than GDMPPs [CMS, SSS '10]: There the population had to decide properties of the whole communication graph

Mediated Graph Protocols

MPPs and additionally:

- a complete communication graph $K_n = (V, E)$
- an input graph $G_\iota[K_n] = (V', E')$ specified by some network initialization function $\iota : V \cup E \to \{q_0, q_1, s_0, s_1\}$
- We are interested in protocols that (stably) decide properties of the input graph
- Different than GDMPPs [CMS, SSS '10]: There the population had to decide properties of the whole communication graph

Our Fundamental Question

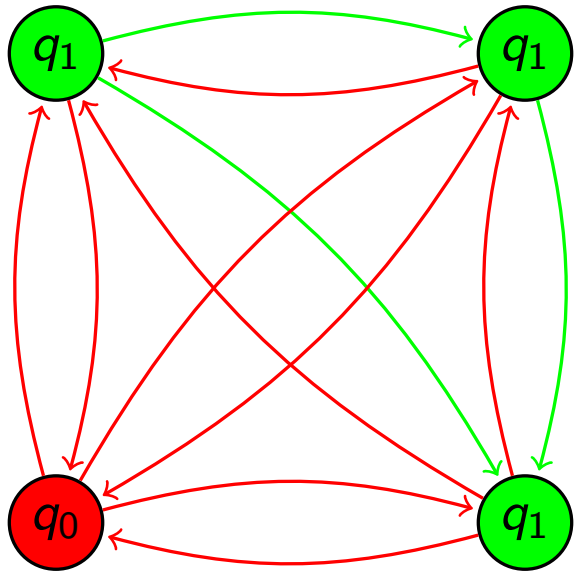
Can we characterize the class **GMGP** of decidable properties?

An Example: count-3-in-neighbors

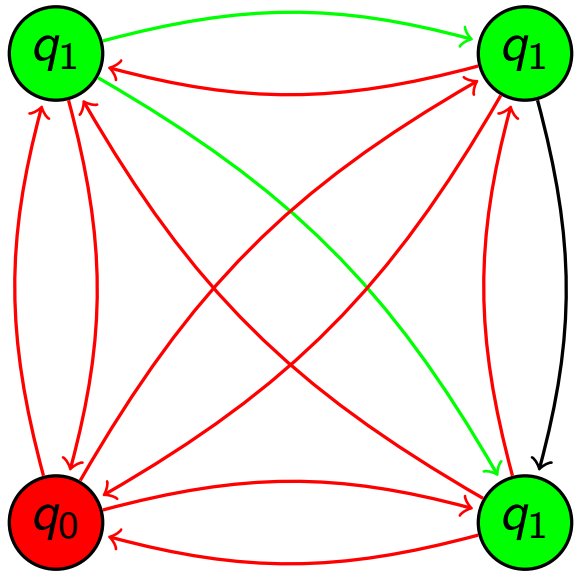
Is there some node with ≥ 3 in-neighbors?

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $S = \{s_0, s_1\}$
- only q_4 outputs 1
- if $i < 4$ and $j < 3$ then $\delta(q_i, q_j, s_1) = (q_i, q_{j+1}, s_0)$
- if $i = 4$ or $j \geq 3$ then $\delta(q_i, q_j, s_1) = (q_4, q_4, s_0)$
- if $i = 4$ or $j = 4$ then $\delta(q_i, q_j, s_0) = (q_4, q_4, s_0)$

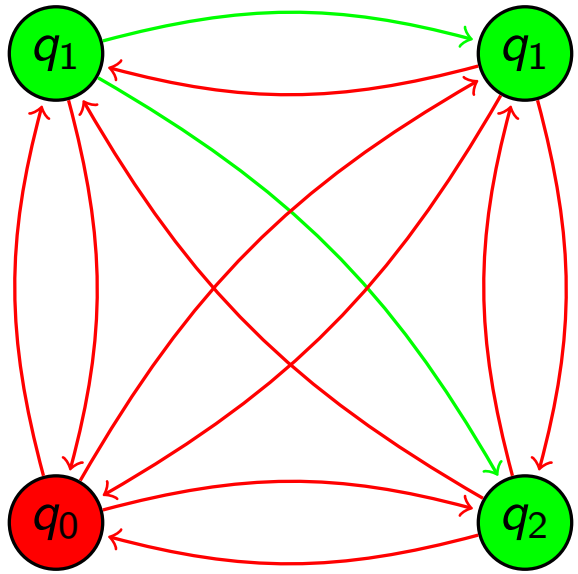
An Example: count-3-in-neighbors



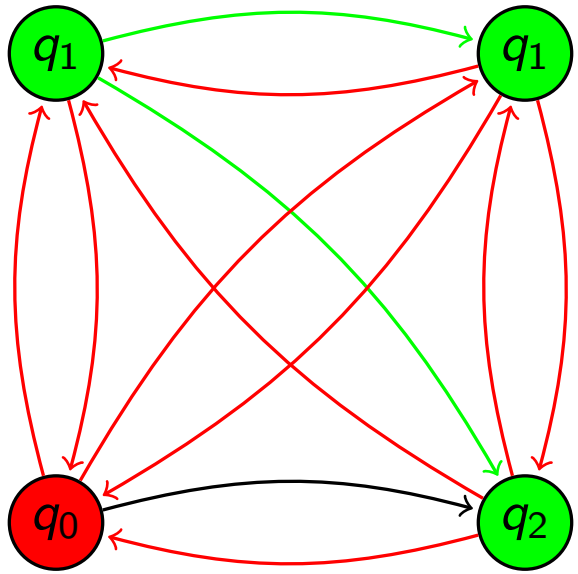
An Example: count-3-in-neighbors



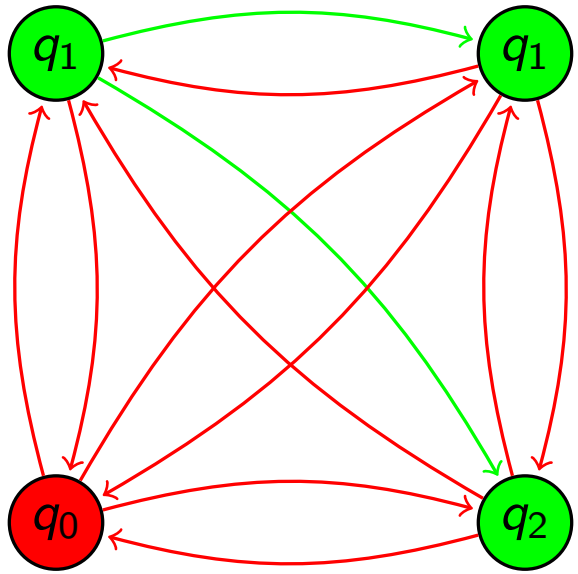
An Example: count-3-in-neighbors



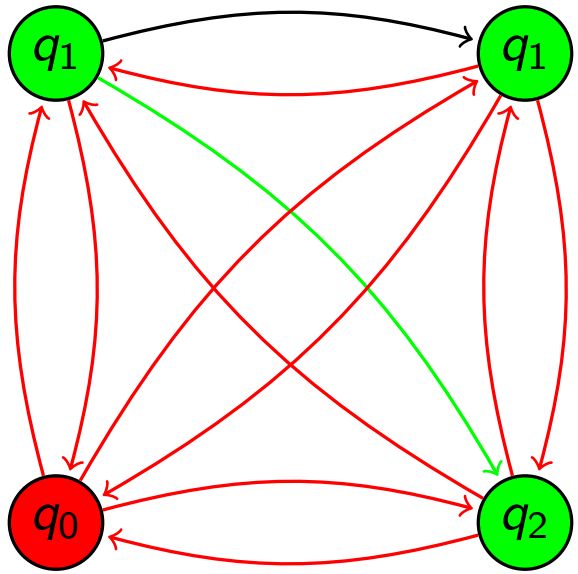
An Example: count-3-in-neighbors



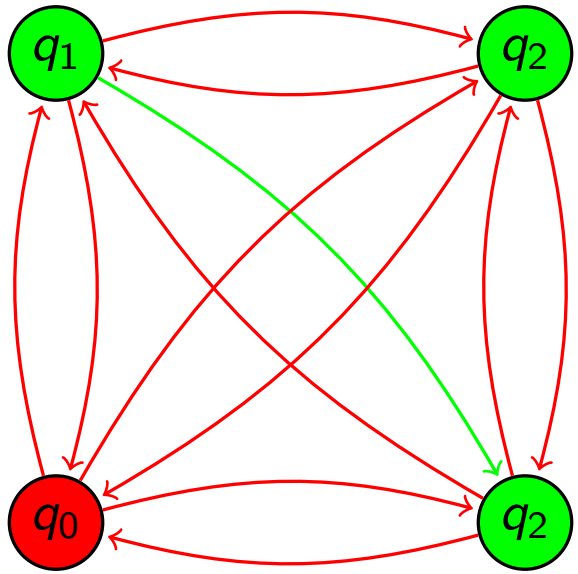
An Example: count-3-in-neighbors



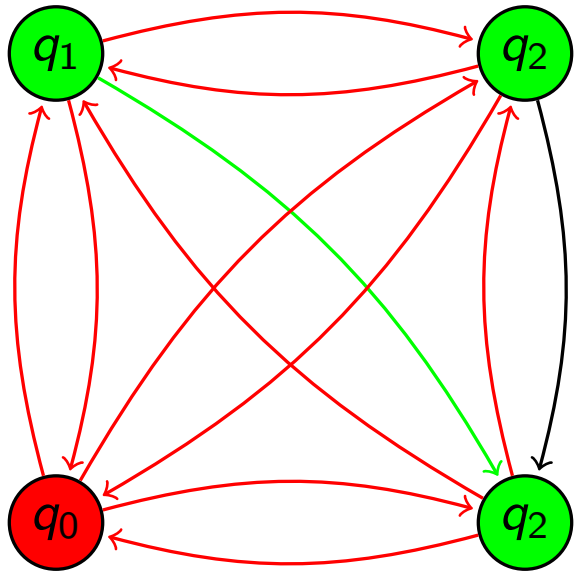
An Example: count-3-in-neighbors



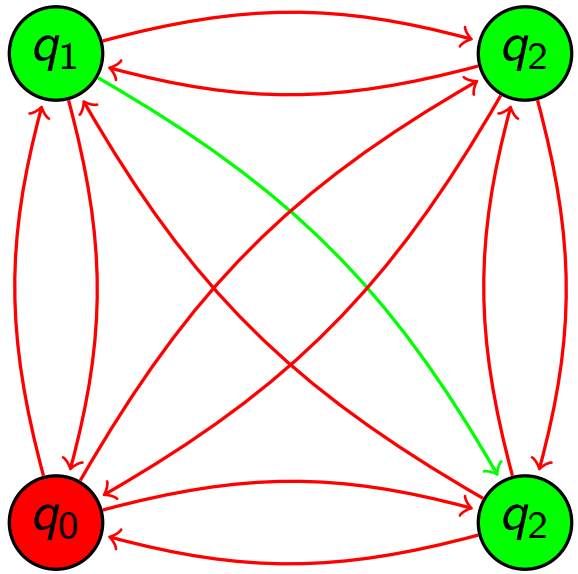
An Example: count-3-in-neighbors



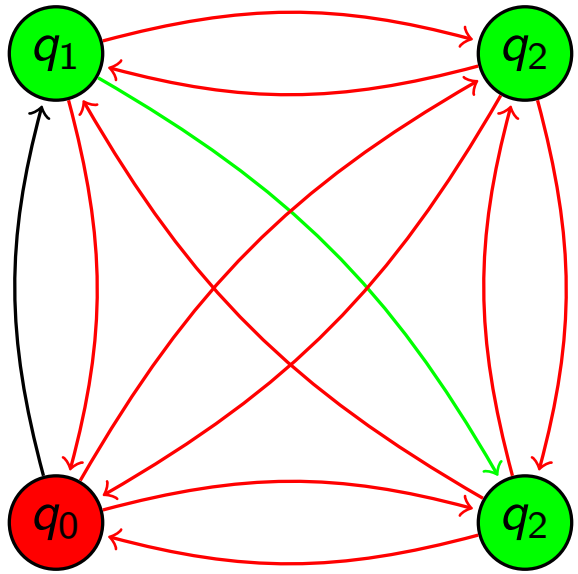
An Example: count-3-in-neighbors



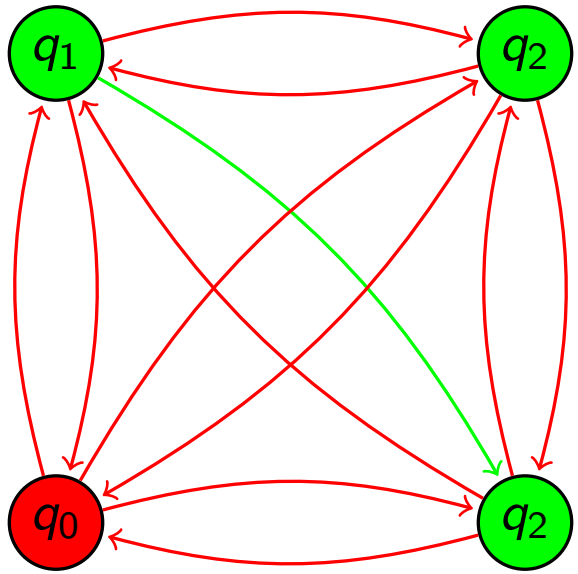
An Example: count-3-in-neighbors



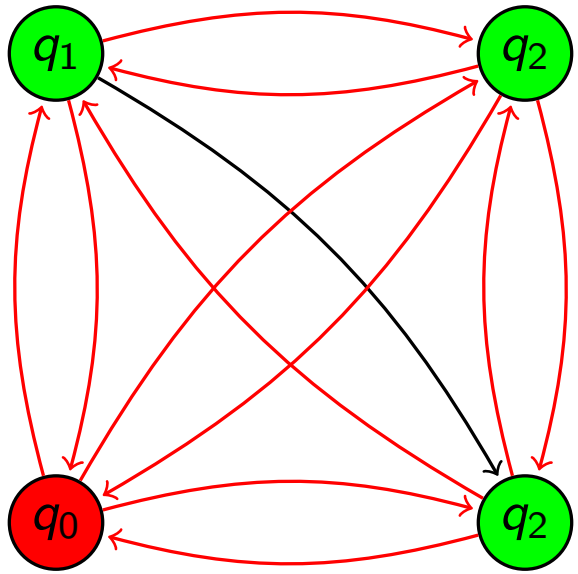
An Example: count-3-in-neighbors



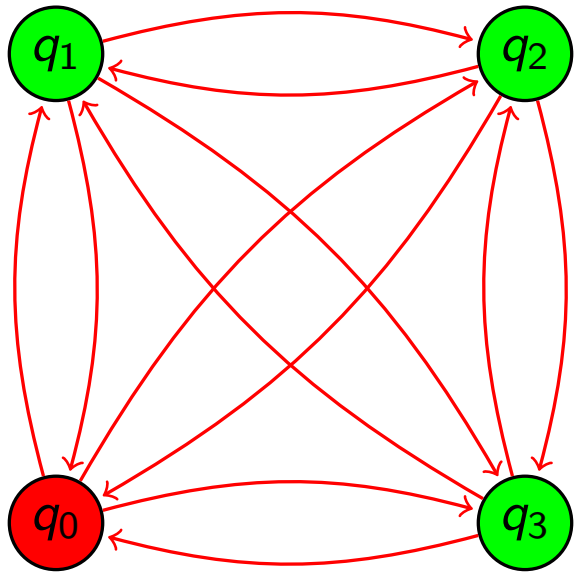
An Example: count-3-in-neighbors



An Example: count-3-in-neighbors



An Example: count-3-in-neighbors



All agents forever output 0

Properties of MGPs

Theorem

GMGP *is closed under union, intersection, complement, and inversion.*

- Let $G = (V, E)$ be a simple digraph and K the irreflexive subset of V^2. Then the inverse of G is defined as $H = (V, K \backslash E)$. Let $L^{-1} = \{H \mid \exists G \in L$ such that H is the inverse of $G\}$.

Properties of MGPs

Theorem

GMGP *is closed under union, intersection, complement, and inversion.*

- Let $G = (V, E)$ be a simple digraph and K the irreflexive subset of V^2. Then the inverse of G is defined as $H = (V, K \backslash E)$. Let $L^{-1} = \{H \mid \exists G \in L$ such that H is the inverse of $G\}$.

Composition

Theorem

MGPs can be sequentially composed.

- *The first protocol must provide a stable input graph to the second protocol.*

Proof.

- Any MGP running on fixed input graphs has an equivalent MGP whose input graph may change but eventually stabilizes
  - Organize the agents into a linear graph and
  - reinitialize computation whenever the input graph changes
- The first and the second protocol composed: The first one provides a stabilizing input graph and the second one computes correctly once the input graph stabilizes

□

Composition

Theorem

MGPs can be sequentially composed.

- *The first protocol must provide a stable input graph to the second protocol.*

Proof.

- Any MGP running on fixed input graphs has an equivalent MGP whose input graph may change but eventually stabilizes
  - Organize the agents into a linear graph and
  - reinitialize computation whenever the input graph changes
- The first and the second protocol composed: The first one provides a stabilizing input graph and the second one computes correctly once the input graph stabilizes

□

Composition

Theorem

MGPs can be sequentially composed.

- *The first protocol must provide a stable input graph to the second protocol.*

Proof.

- Any MGP running on fixed input graphs has an equivalent MGP whose input graph may change but eventually stabilizes
  - Organize the agents into a linear graph and
  - reinitialize computation whenever the input graph changes
- The first and the second protocol composed: The first one provides a stabilizing input graph and the second one computes correctly once the input graph stabilizes

□

Composition

Theorem

MGPs can be sequentially composed.

- *The first protocol must provide a stable input graph to the second protocol.*

Proof.

- Any MGP running on fixed input graphs has an equivalent MGP whose input graph may change but eventually stabilizes
  - Organize the agents into a linear graph and
  - reinitialize computation whenever the input graph changes
- The first and the second protocol composed: The first one provides a stabilizing input graph and the second one computes correctly once the input graph stabilizes

□

Composition

Theorem

MGPs can be sequentially composed.

- *The first protocol must provide a stable input graph to the second protocol.*

Proof.

- Any MGP running on fixed input graphs has an equivalent MGP whose input graph may change but eventually stabilizes
  - Organize the agents into a linear graph and
  - reinitialize computation whenever the input graph changes
- The first and the second protocol composed: The first one provides a stabilizing input graph and the second one computes correctly once the input graph stabilizes

□

Deciding Properties of Disconnected Graphs

- GDMPPs General Impossibility [CMS, SSS '10]:
  - No non-trivial graph language is decidable if the graph universe contains disconnected graphs
- In contrast, MGPs can exploit the complete communication infrastructure to decide properties of disconnected input graphs.

Deciding Properties of Disconnected Graphs

- GDMPPs General Impossibility [CMS, SSS '10]:
  - No non-trivial graph language is decidable if the graph universe contains disconnected graphs
- In contrast, MGPs can exploit the complete communication infrastructure to decide properties of disconnected input graphs.

Deciding Connectivity

Theorem

Connectivity is MGP-decidable.

Deciding Connectivity

Protocol 1 *Connectivity Protocol (CP)*

1: $Q = \{q_0, q_1, t, t', l, l'\}$, $S = \{s_0, s_1\}$,
2: $O(q_0) = 0, O(q_1) = 0, O(t) = 1, O(t') = 0, O(l) = 1, O(l') = 0$,
3: δ:

a single leader is generated
$(q_1, q_1, s_1) \to (l, t, s_1)$

the single leader turns all nodes of the input graph it can reach to followers
$(l, t, s_1) \to (t, l, s_1), (t, l, s_1) \to (l, t, s_1), (l, q_1, s_1) \to (t, l, s_1), (q_1, l, s_1) \to (l, t, s_1)$

the single leader turns all nodes that do not belong to the input graph to followers of a single leader
$(l, q_0, s_0) \to (l, t, s_0)$

two single leaders meet in the same connected component of the input graph; one is turned to follower
$(l, l, s_1) \to (l, t, s_1)$

two non-adjacent single leaders meet in the same connected component of the input graph or in different connected components (in the case of disconnected input graph); they become non-unique leaders
$(l, l, s_0) \to (l', l', s_0)$

the non-unique leaders turn all the non-leaders they meet into their followers
$(l', t, s_1) \to (t', l', s_1), (t, l', s_1) \to (l', t', s_1), (l', q_1, s_1) \to (t', l', s_1), (q_1, l', s_1) \to (l', t', s_1), (l', q_0, s_0) \to (l', t', s_0), (l', t, s_0) \to (l', t', s_0)$

two leaders of any type meet in the same component; only one single leader remains
$(l', l, s_1) \to (l, t, s_1), (l, l', s_1) \to (l, t, s_1), (l', l', s_1) \to (l, t, s_1)$

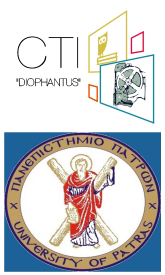
two leaders of any type meet in different components; they both become non-unique leaders
$(l', l, s_0) \to (l', l', s_0), (l, l', s_0) \to (l', l', s_0)$

a single leader restores all followers of multiple leaders to followers of single leaders
$(l, t', s_0) \to (l, t, s_0), (l, t', s_1) \to (t, l, s_1), (t', l, s_1) \to (l, t, s_1)$

Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

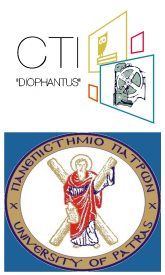
1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around



Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around



Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around

Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around

Deciding Connectivity

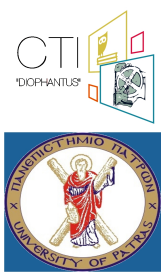
- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around

Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around



Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around

Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around

Deciding Connectivity

- C-leaders: believe the graph is Connected
- D-leaders: believe the graph is Disconnected
- Followers: always adopt the output of leaders

1. All agents of the input graph are initially C-leaders and all other agents are followers.
2. 2 leaders interact via s_1-edge
   - one becomes C-leader and the other follower
3. 2 C-leaders interact via s_0-edge
   - both become D-leaders
4. Leaders always keep moving around

Deciding Connectivity: Proof of Correctness

- If Connected: Eventually, the last 2 leaders interact via s_1-edge, a single C-leader remains and all followers output "connected".
- If Disconnected: Each component eventually has a single leader. Eventually, these leaders interact with each other via s_0-edges and all become D-leaders. No way to become L again. All followers output "disconnected". □

Deciding Connectivity: Proof of Correctness

- If Connected: Eventually, the last 2 leaders interact via s_1-edge, a single C-leader remains and all followers output "connected".
- If Disconnected: Each component eventually has a single leader. Eventually, these leaders interact with each other via s_0-edges and all become D-leaders. No way to become L again. All followers output "disconnected". □

Deciding Properties of Disconnected Graphs

- $N_{G,L}$: the # of components of G that belong to a language L
- $N_{G,\bar{L}}$: the # of components of G that do not

Theorem

Let L be a GDMPP-decidable language. Let p be a semilinear predicate on $\mathbb{N}^2$. Then $L' = \{G \mid p(N_{G,L}, N_{G,\bar{L}}) = 1\}$ is MGP-decidable.

- e.g. 25% of the components contain a 2-cycle

Deciding Properties of Disconnected Graphs

- $N_{G,L}$: the # of components of G that belong to a language L
- $N_{G,\bar{L}}$: the # of components of G that do not

Theorem

Let L be a GDMPP-decidable language. Let p be a semilinear predicate on $\mathbb{N}^2$. Then $L' = \{G \mid p(N_{G,L}, N_{G,\bar{L}}) = 1\}$ is MGP-decidable.

- e.g. 25% of the components contain a 2-cycle

Deciding Properties of Disconnected Graphs

- $N_{G,L}$: the # of components of G that belong to a language L
- $N_{G,\bar{L}}$: the # of components of G that do not

Theorem

Let L be a GDMPP-decidable language. Let p be a semilinear predicate on $\mathbb{N}^2$. Then $L' = \{G \mid p(N_{G,L}, N_{G,\bar{L}}) = 1\}$ is MGP-decidable.

- e.g. 25% of the components contain a 2-cycle

Deciding Properties of Disconnected Graphs

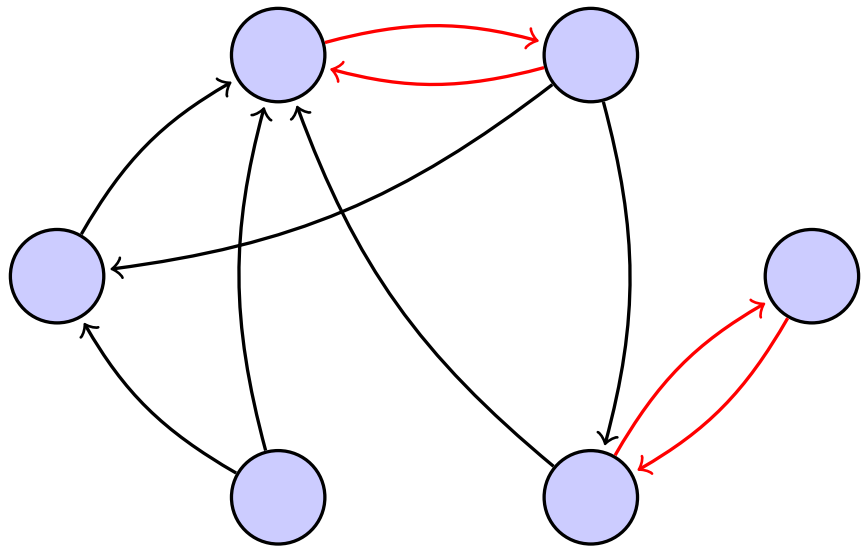
- $N_{G,L}$: the # of components of G that belong to a language L
- $N_{G,\bar{L}}$: the # of components of G that do not

Theorem

Let L be a GDMPP-decidable language. Let p be a semilinear predicate on $\mathbb{N}^2$. Then $L' = \{G \mid p(N_{G,L}, N_{G,\bar{L}}) = 1\}$ is MGP-decidable.

- e.g. 25% of the components contain a 2-cycle

Resolving an Open Question: A GDMPP Detecting 2-cycles



- 2CYCLE $= \{G \in \mathcal{G}_{wcon} \mid G$ has at least one 2-cycle)

Theorem

2CYCLE is GDMPP-decidable.

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

Theorem

2CYCLE is GDMPP-decidable.

- Initially all agents are leaders and all edges are inactive
- The agents maintain a counter with the # of the active edges incident to them (at most 2)
- When 2 leaders interact only one survives (at least one leader survives forever)
- Leaders form lines and cycles by activating edges (active subgraphs)
- Any 2-cycle that becomes active cannot be deactivated
- Any other active subgraph is eventually deactivated

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

A GDMPP: Detecting 2-cycles

- $G \in$ 2CYCLE
  - There is always at least one leader, which will eventually activate a 2-cycle
  - Both agents of an activated 2-cycles output 1
  - The predicate $(N_1 \geq 1)$ (there is at least 1 agent giving 1 as output) is semilinear so we can compose our protocol with a stabilizing inputs protocol computing the predicate
- $G \notin$ 2CYCLE
  - Eventually, a unique leader remains
  - Since it cannot detect a 2-cycle it forever outputs 0, while keeping moving around, and all nonleaders eventually copy this output □

GDMPPs are Weak: An Insight

- Any GDMPP-decidable language known so far is also PP-decidable
- Focus for a moment on MPPs running on weakly-connected graphs (not necessarily complete)
- **MPU**: the corresponding class of stably computable predicates ('U' standing for "Unrestricted graphs")

GDMPPs are Weak: An Insight

- Any GDMPP-decidable language known so far is also PP-decidable
- Focus for a moment on MPPs running on weakly-connected graphs (not necessarily complete)
- **MPU**: the corresponding class of stably computable predicates ('U' standing for "Unrestricted graphs")

GDMPPs are Weak: An Insight

- Any GDMPP-decidable language known so far is also PP-decidable
- Focus for a moment on MPPs running on weakly-connected graphs (not necessarily complete)
- **MPU**: the corresponding class of stably computable predicates ('U' standing for "Unrestricted graphs")

GDMPPs are Weak: An Insight

Theorem

MPU = SEM.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders)

□

Remark: Leader does not seem to help PP-like models w.r.t. computational power!

GDMPPs are Weak: An Insight

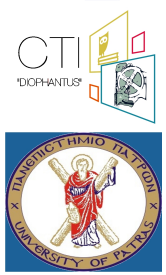
Theorem

MPU $=$ **SEM**.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders)

□

Remark: Leader does not seem to help PP-like models w.r.t. computational power!



GDMPPs are Weak: An Insight

Theorem

MPU $=$ **SEM**.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders) □

Remark: Leader does not seem to help PP-like models w.r.t. computational power!

GDMPPs are Weak: An Insight

Theorem

MPU $=$ **SEM**.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders) □

Remark: Leader does not seem to help PP-like models w.r.t. computational power!

GDMPPs are Weak: An Insight

Theorem

MPU $=$ **SEM**.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders)

Remark: Leader does not seem to help PP-like models w.r.t. computational power!

GDMPPs are Weak: An Insight

Theorem

MPU $=$ **SEM**.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders)

□

Remark: Leader does not seem to help PP-like models w.r.t. computational power!

GDMPPs are Weak: An Insight

Theorem

MPU $=$ **SEM**.

- Any $p \in$ **MPU** is also stably computable on star graphs (1 internal node and n-1 leaves).
- The latter can be simulated by a PP with a unique leader on complete graphs
  - The leader plays the role of the internal node and stores the edge states on the leaves (which are not allowed effective communication with each other)
- Leader-PPs can be simulated by classical PPs
  - We compose a leader election protocol with a stabilizing-inputs implementation of the above PP (extended with ineffective transitions between 2 leaders)

□

Remark: Leader does not seem to help PP-like models w.r.t. computational power!

GDMPPs are Weak: An Insight

- Gaining some insight into GDMPPs
  - They may also run on worst-case instances like star graphs
  - Star graphs are hard because the internal node has an uncountable ($O(n)$) number of neighbors (the leaves)

GDMPPs are Weak: An Insight

- Gaining some insight into GDMPPs
  - They may also run on worst-case instances like star graphs
  - Star graphs are hard because the internal node has an uncountable ($O(n)$) number of neighbors (the leaves)

GDMPPs are Weak: An Insight

- Gaining some insight into GDMPPs
  - They may also run on worst-case instances like star graphs
  - Star graphs are hard because the internal node has an uncountable ($O(n)$) number of neighbors (the leaves)

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

MGPs are Strong: An Exact Characterization

Theorem

GMGP *is equal to the class of all decidable graph languages by a NTM of linear space which receives the input graph by its adjacency matrix representation.*

Proof.

- Construct a spanning pseudo-path subgraph
- Visit one after the other all distributed cells (from left to right)
- Keep count of the distance of the corresponding nodes from the left endpoint
- Store each s_1 edge (u, v) in the leftmost cells of the memory by putting a 1 in the adjacency matrix position (u, v)
- Then simply simulate the NTM on the constructed input
- In the worst case, the adjacency matrix is of the same order as the interaction graph and needs $O(n^2)$ space, which is available

□

Conclusions

- Investigated MPPs ability to decide graph properties of input graphs over a complete communication infrastructure
- Some Properties: closure, sequential composition
- Some Protocols: connectivity, cardinalities of components
- Exact Characterization: Equivalent to linear-space NTM
- Resolved open questions: GDMPPs can detect 2-cycles, MPPs are semilinear on unrestricted graphs, leaders do not enhance PPs
- Almost resolved: GDMPPs are weak (probably semilinear)

Conclusions

- Investigated MPPs ability to decide graph properties of input graphs over a complete communication infrastructure
- Some Properties: closure, sequential composition
- Some Protocols: connectivity, cardinalities of components
- Exact Characterization: Equivalent to linear-space NTM
- Resolved open questions: GDMPPs can detect 2-cycles, MPPs are semilinear on unrestricted graphs, leaders do not enhance PPs
- Almost resolved: GDMPPs are weak (probably semilinear)

Conclusions

- Investigated MPPs ability to decide graph properties of input graphs over a complete communication infrastructure
- Some Properties: closure, sequential composition
- Some Protocols: connectivity, cardinalities of components
- Exact Characterization: Equivalent to linear-space NTM
- Resolved open questions: GDMPPs can detect 2-cycles, MPPs are semilinear on unrestricted graphs, leaders do not enhance PPs
- Almost resolved: GDMPPs are weak (probably semilinear)

Conclusions

- Investigated MPPs ability to decide graph properties of input graphs over a complete communication infrastructure
- Some Properties: closure, sequential composition
- Some Protocols: connectivity, cardinalities of components
- Exact Characterization: Equivalent to linear-space NTM
- Resolved open questions: GDMPPs can detect 2-cycles, MPPs are semilinear on unrestricted graphs, leaders do not enhance PPs
- Almost resolved: GDMPPs are weak (probably semilinear)

Conclusions

- Investigated MPPs ability to decide graph properties of input graphs over a complete communication infrastructure
- Some Properties: closure, sequential composition
- Some Protocols: connectivity, cardinalities of components
- Exact Characterization: Equivalent to linear-space NTM
- Resolved open questions: GDMPPs can detect 2-cycles, MPPs are semilinear on unrestricted graphs, leaders do not enhance PPs
- Almost resolved: GDMPPs are weak (probably semilinear)

Conclusions

- Investigated MPPs ability to decide graph properties of input graphs over a complete communication infrastructure
- Some Properties: closure, sequential composition
- Some Protocols: connectivity, cardinalities of components
- Exact Characterization: Equivalent to linear-space NTM
- Resolved open questions: GDMPPs can detect 2-cycles, MPPs are semilinear on unrestricted graphs, leaders do not enhance PPs
- Almost resolved: GDMPPs are weak (probably semilinear)

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!

References

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.

D. Angluin, J. Aspnes, and D. Eisenstat.
Stably computable predicates are semilinear.
In *25th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 292–299, New York, NY, USA, 2006. ACM Press.

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.

References

I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis.
Passively mobile communicating machines that use restricted space.
In *Proceedings of the 7th ACM ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing (FOMC)*, San Jose, California, 2011.

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.

I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis.
All symmetric predicates in $NSPACE(n^2)$ are stably computable by the mediated population protocol model.
In *35th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 6281 of *LNCS*, pages 270–281. Springer-Verlag, August 23–27 2010.

References

I. Chatzigiannakis, O. Michail, and P. G. Spirakis.
Stably Decidable Graph Languages by Mediated Population Protocols.
In *12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 6366 of *LNCS*, pages 252–266. Springer-Verlag, September 2010.