Distributed local strategies in broadcast networks

Arnaud Sangnier

LIAFA - Université Paris Diderot-Paris 7

joint work with: Nathalie Bertrand and Paulin Fournier

ACTS - CMI - Chennai -10th February 2015

Motivation

Verify network of processes of unbounded size

Why to consider such networks?

- Classical distributed algorithms (*mutual exclusion, leader election,...*)
- Telecommunication protocols (*routing,...*)
- Algorithms for ad-hoc networks
- Model for biological systems
- and many more applications ...

Hypothesis

All the processes have the same behavior

In [Esparza, STACS'14], such networks are called **crowd**

More precisely:

- Each process will follow the same protocol
- Process can communicate
- Communication way:
  - Message passing
  - Shared variable
  - Rendez-vous communication
  - **Broadcast communication**
  - **Multi-diffusion (selective broadcast)**

Question:
Is there a network with N processes which allows to reach a goal ?

In this talk

Today:

Decidability and complexity of reachability problems on parameterized networks

Features:

- **Simple protocols with broadcast communication**
- **Simple reachability questions**
- **Take into account some locality assumptions**

Outline

1. Ad Hoc Networks
2. Clique and Reconfigurable Networks
3. Considering local strategies
4. Conclusion

Outline

Defining a model for Ad Hoc Networks

Main characteristics [Delzanno et al., CONCUR'10]

- No creation/deletion of nodes
- Each node executes the same finite state process
- Model based on the ω-calculus
- Broadcast of the messages to the neighbors
- Static topology represented by a connectivity graph

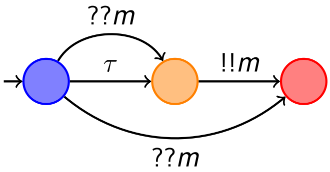
Ad Hoc Networks: syntax

A protocol $P = \langle Q, \Sigma, R, q_0 \rangle$

Finite state system whose transitions are labeled with:

1. broadcast of messages - $!!m$
2. reception of messages - $??m$
3. internal actions - τ

where m belongs to the finite alphabet Σ

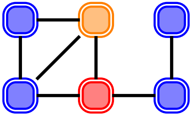


A protocol defines an Ad Hoc Network (AHN)

Ad Hoc Networks: configurations

A configuration is a graph $\gamma = \langle V, E, L \rangle$

- V : finite set of vertices
- $E : V \times V$: finite set of edges
- $L : V \to Q$: labeling function



- **Initial configurations**: **all** vertices are labeled with the initial state q_0
- *Notation : $L(\gamma)$ all the labels present in γ*

Remarks:

- The size of the considered graphs is not bounded
- Infinite number of configurations

$\Rightarrow$ **AHN are infinite state systems**

Ad Hoc Networks: semantics

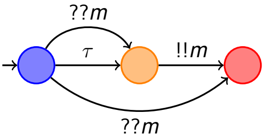
Transition system $AHN(P) = \langle \mathcal{C}, \rightarrow, \mathcal{C}_0 \rangle$ associated to P

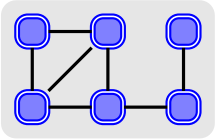
- $\mathcal{C}$: set of configurations
- $\rightarrow : \mathcal{C} \times \mathcal{C}$: transition relation
- $\mathcal{C}_0$: initial configurations

The relation $\rightarrow$ respects the following rules during an execution:

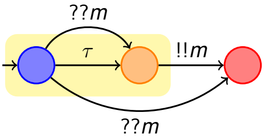
- The topology remains **static**
  - The number of vertices does not change
  - The edges do not change
  - Only the labels of the vertices can evolve
- Two kind of transitions according to the given protocol
  1. **local actions** - one process performs an internal action τ
  2. **broadcast** - one process emits a message with $!!m$, all its neighbors that can receive it with $??m$ have to receive it

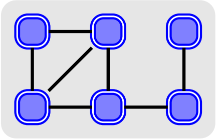
Ad Hoc Networks: an example



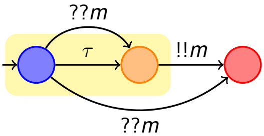


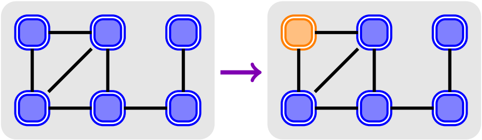
Ad Hoc Networks: an example



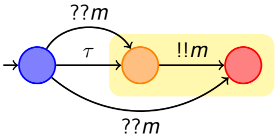


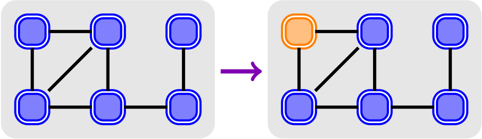
Ad Hoc Networks: an example



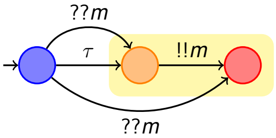


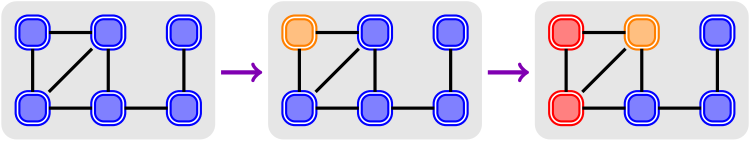
Ad Hoc Networks: an example



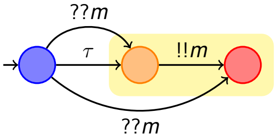


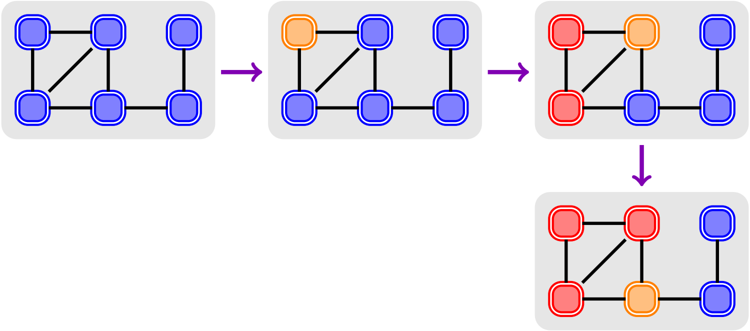
Ad Hoc Networks: an example



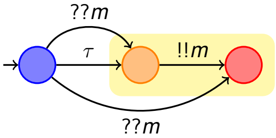


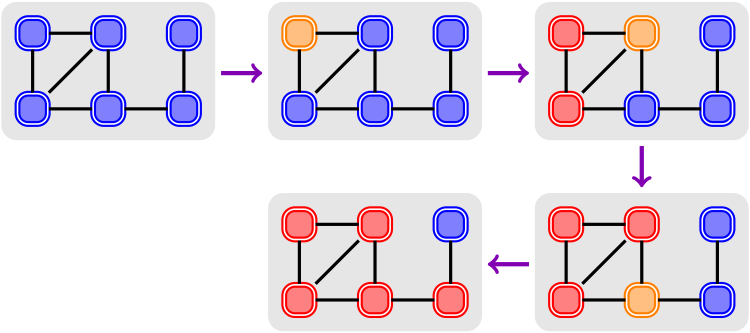
Ad Hoc Networks: an example





Ad Hoc Networks: an example





Reachability question

Parameters: Number of processes

Control State Reachability (REACH)

Input: A protocol and a control state $q \in Q$;

Output: Does there exist $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ s.t. $\gamma \rightarrow^* \gamma'$ and $q \in L(\gamma')$?

Target State Reachability (TARGET)

Input: A protocol and a set of control states $T \subseteq Q$;

Output: Does there exist $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ s.t. $\gamma \rightarrow^* \gamma'$ and $L(\gamma') \subseteq T$?

Remarks:

- These problems consider an infinite number of possible initial configurations
- Reachability of a configuration γ' is certainly feasible, **the number of processes is in fact fixed**

Encoding Minsky machine to prove undecidability

Minsky machine

- Manipulates two counters c_1 and c_2
- Finite set of labeled instructions of the form:
  1. $L : c_i := c_i + 1$; goto L'
  2. L : if $c_i = 0$ goto L' else $c_i := c_i - 1$; goto L''
- An initial label L_0
- A special label L_F with no output instruction

Halting problem: Is the label L_F eventually reached?

Theorem [Minsky, 67]

The halting problem for Minsky machines is undecidable.

Undecidability result

Theorem [Delzanno et al, CONCUR'10]

REACH and TARGET for Ad Hoc Networks are undecidable.

Idea of the proof:

- Ensure that a topology is in a certain form
- Simulate the behavior of a Minsky machine

Undecidability result

Theorem [Delzanno et al, CONCUR'10]

REACH and TARGET for Ad Hoc Networks are undecidable.

Idea of the proof:

- Ensure that a topology is in a certain form
- Simulate the behavior of a Minsky machine

One way to regain decidability:
restrict the considered graphs or change the semantics

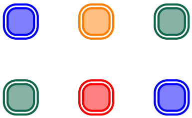
Outline

Clique Networks

Clique Networks are Ad Hoc Networks restricted to clique graphs

A configuration is a multiset $\gamma : Q \mapsto \mathbb{N}$

- $\gamma(q)$ gives the number of process in state q
- We forget about the graphs since it always the same

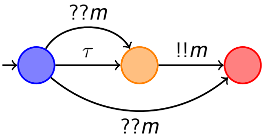


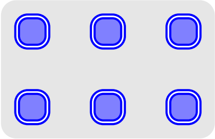
- **Initial configurations**: $\gamma(q) > 0$ iff $q \in Q_0$

Remarks:

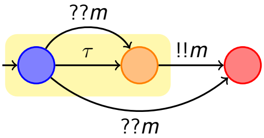
- Clique Networks are Broadcast Networks with no rendez-vous communication [Esparza et al., LICS'99]
- **In clique networks, a broadcast message is received by all the processes**

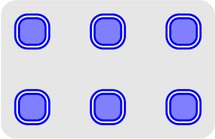
Clique Networks: an example



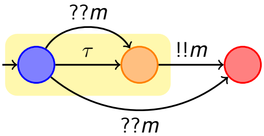


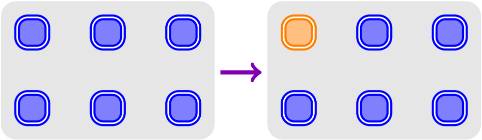
Clique Networks: an example



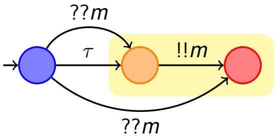


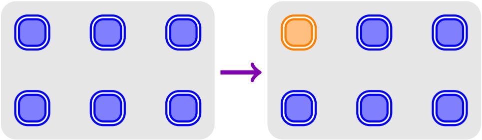
Clique Networks: an example



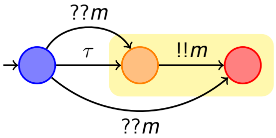


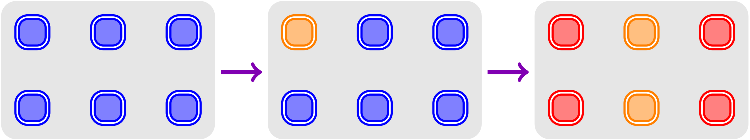
Clique Networks: an example



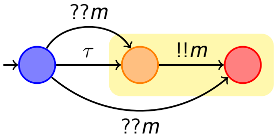


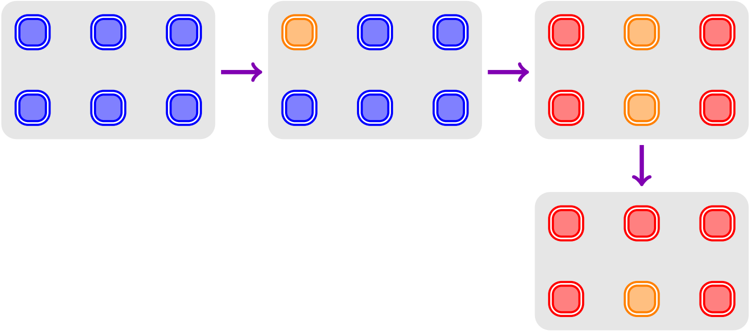
Clique Networks: an example



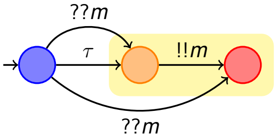


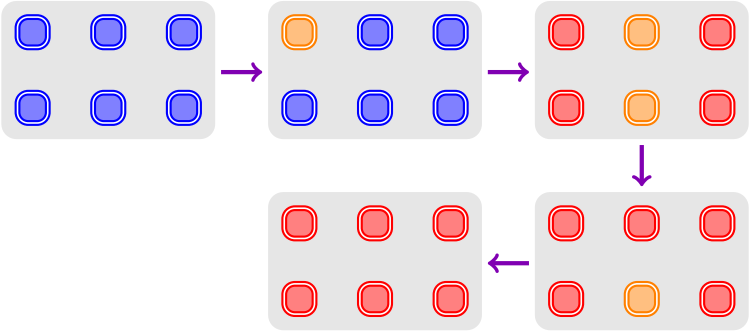
Clique Networks: an example





Clique Networks: an example





Deciding REACH in Broadcast Networks

Theorem [Esperza et al., LICS'99] [Schmitz & Schnoebelen, CONCUR'13]

REACH is decidable in Clique Networks and Ackermann-complete.

Idea of the proof (for decidability)

- Use the fact that there is a well-quasi-oder on the set of configurations
- And that this order is a simulation
  - What can be done from a configuration, can be done from a bigger one
- Class of Well Structured Transitions Systems

Concerning TARGET

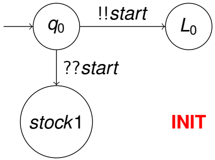
Theorem

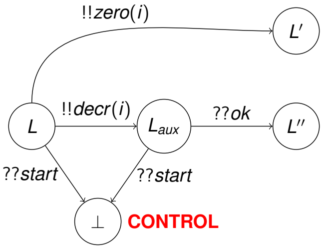
TARGET is undecidable in Clique Networks.

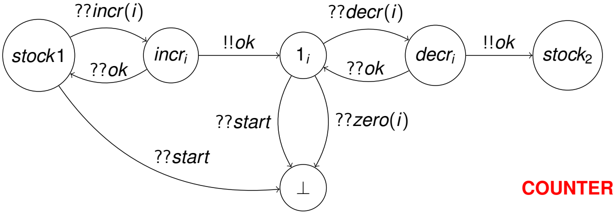
Idea of the proof:

- Simulate a two counter Minsky machines
- Isolate one process (controller) thanks to the clique property
- The other processes will simulate the counter values
  - Number of processes in state 1_i: value of counter i
- For zero-test, the controller can 'cheat'
- Use the target set to know when this happens

Protocol for TARGET in Clique Networks







Reconfigurable Networks

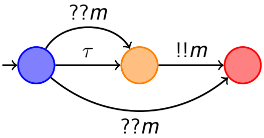
Transition system $RN(P) = \langle \mathcal{C}, \rightarrow, \mathcal{C}_0 \rangle$ associated to P

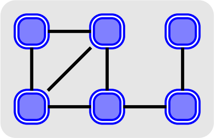
- $\mathcal{C}$: set of configurations
- $\rightarrow: \mathcal{C} \times \mathcal{C}$: transition relation
- $\mathcal{C}_0$: initial configurations

The relation $\Rightarrow$ respects the following rules during an execution:

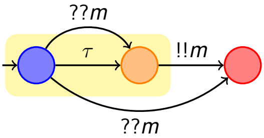
- The topology is **not static** anymore
  - The number of vertices does not change
  - The edges can change non deterministically
  - The labels of the vertices can evolve
- Three kind of transitions according to the given protocol
  1. **local actions**
  2. **broadcast**
  3. **reconfiguration** - the edges can change with no restriction

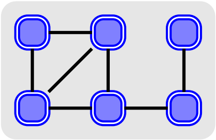
Reconfigurable Networks: an example



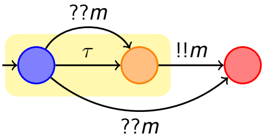


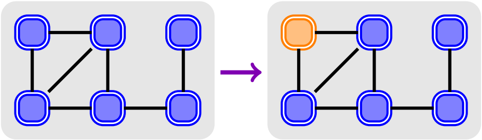
Reconfigurable Networks: an example



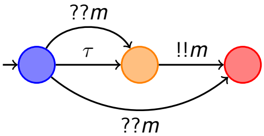


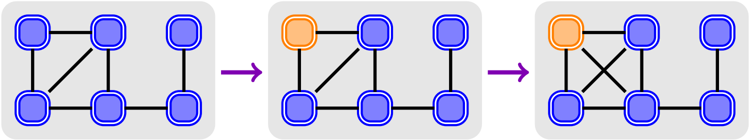
Reconfigurable Networks: an example



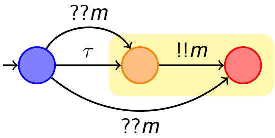


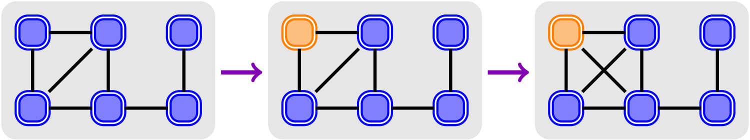
Reconfigurable Networks: an example



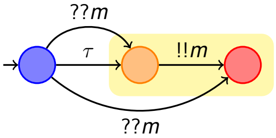


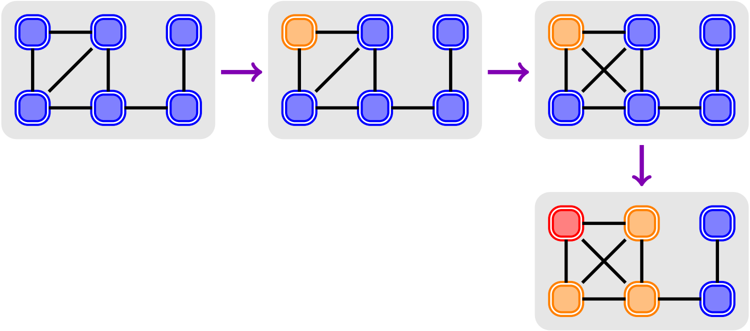
Reconfigurable Networks: an example



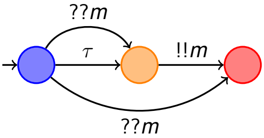


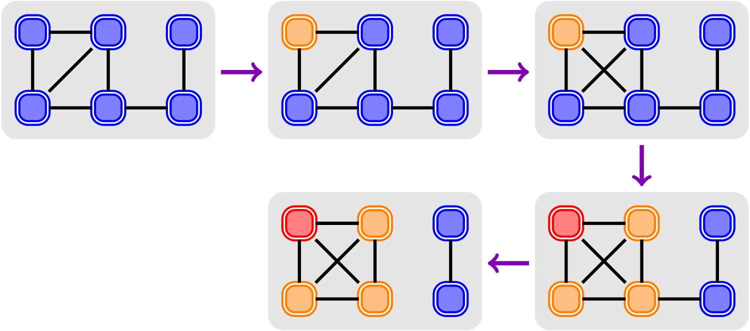
Reconfigurable Networks: an example



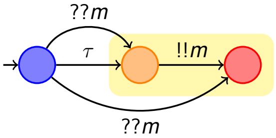


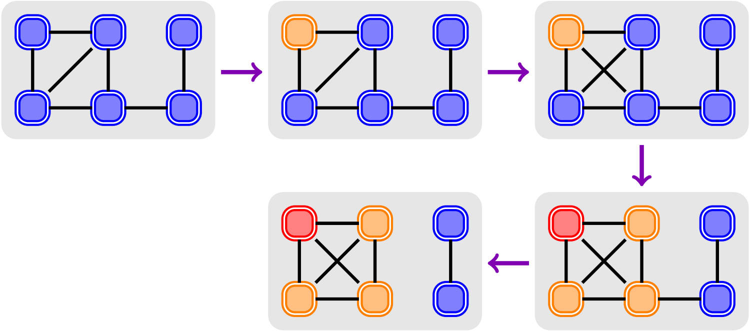
Reconfigurable Networks: an example



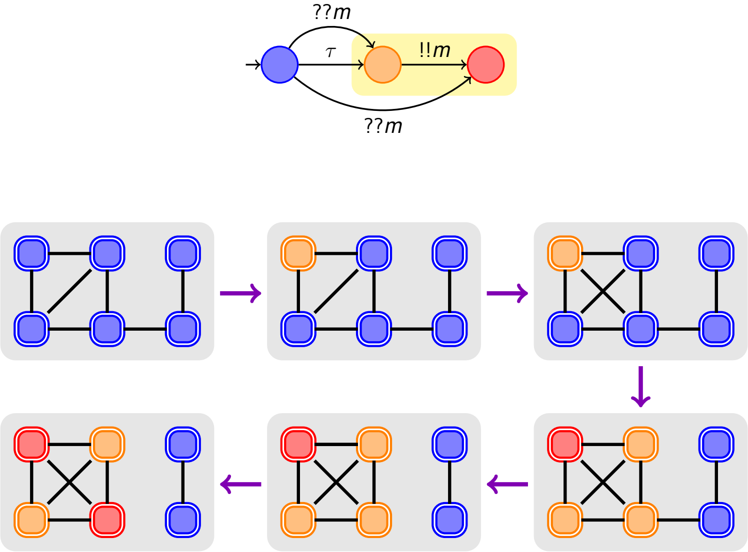


Reconfigurable Networks: an example





Reconfigurable Networks: an example



Results in Reconfigurable Networks

Theorem [Delzanno et al.,FSTTCS'12]

REACH in reconfigurable networks is PTIME-complete

Idea of the proof:

- **Lower bound:** LOGSPACE reduction from the Circuit Value Problem
- **Upper bound:** algorithm which builds the set of reachable states

Solving REACH in Reconfigurable Networks

PTIME algorithm to compute the set of reachable states

Input : $P = \langle Q, \Sigma, R, q_0 \rangle$ a protocol
Output : $S \subseteq Q$ the set of reachable control states in $RAN(P)$

$S := \{q_0\}$
$oldS := \emptyset$
while $S \neq oldS$ **do**
  $oldS := S$
  for all $\langle q_1, !!a, q_2 \rangle \in R$ such that $q_1 \in oldS$ **do**
    $S := S \cup \{q_2\} \cup \{q' \in Q \mid \langle q, ??a, q' \rangle \in R \wedge q \in oldS\}$
  end for
end while

- Each time, do all the possible transactions in the network
- Terminates in at most $|P|$ iterations of the main loop

What about TARGET

Theorem [Fournier,Phd's thesis'15]

TARGET in reconfigurable networks is in PTIME

Idea of the proof:

- Same idea as for REACH
- First compute the reachable states from q_0
- Then compute the reachable states S from the target set (by inversing the transition relation)
- If these two sets match, the algorithm returns S
- Otherwise it repeats the preceding actions by restricting the protocols to states in S

Outline

Local strategies

Do all the processes really behave the same in the previous networks ?

- No, they all follow the same protocol P
- If the protocol is non-deterministic, each process can make a different choice!
- How to enforce, that each process behaves exactly the same ?

Local strategy $\sigma = (\sigma_a, \sigma_r)$

- σ_a : $\text{Path}(P) \mapsto (Q \times (\{!!m\} \cup \{\varepsilon\}) \times Q) \cup \bot$ (for actions)
- σ_r : $\text{Path}(P) \times \Sigma \mapsto (Q \times \{??m\} \times Q) \cup \bot$ (for receptions)
- These two functions continue paths in the protocols

Local strategies tell a process what to do according to its (local) past

Two processes with the same past will behave similarly

Reachability question with local strategies

An execution respects a local strategy iff each process during the execution does a choice matching with the strategy

Control State Reachability (REACH[L])

Input: A protocol and a control state $q \in Q$;

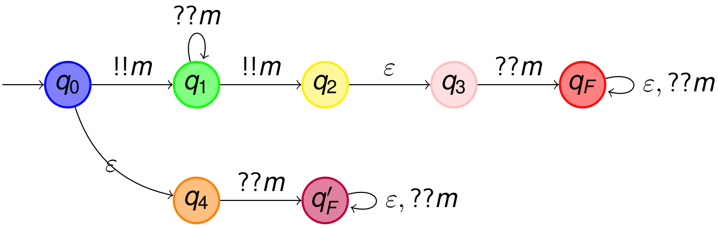
Output: Does there exist $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ and a local strategy σ s.t. $\gamma \rightarrow^* \gamma'$ respects σ and $q \in L(\gamma')$?

Target State Reachability (TARGET[L])

Input: A protocol and a set of control state $T \subseteq Q$;

Output: Does there exist $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ and a local strategy σ s.t. $\gamma \rightarrow^* \gamma'$ respects σ and $L(\gamma') \subseteq T$?

Example of reachability questions under local strategies



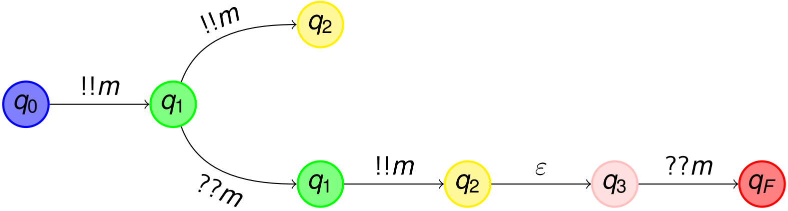
- There exists a local strategy to reach q_F in Clique and Reconfigurable Networks
- There does not exists a local strategy to reach q'_F in Clique and Reconfigurable Networks
  - Either all the process will move in their first step to q_1 or they will all move to q_4

Strategy patterns for reconfigurable networks

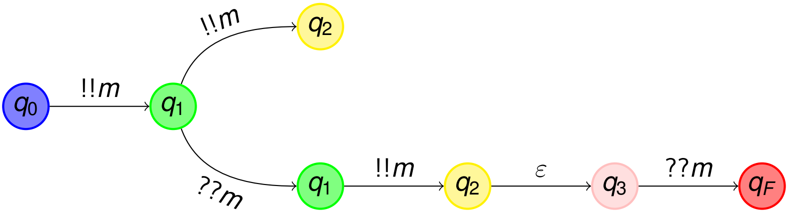
To represent local strategies in reconfigurable networks, we will use trees

- Each path in the tree will be an unfolded path of the protocol
- From each node in the tree:
  - **At most one edge labelled by an action (broadcast or internal action)**
  - **At most one edge per message m labelled with $??m$**
- Those trees can be seen as underspecified local strategies
- They represent sets of local strategies

Example of strategy patterns



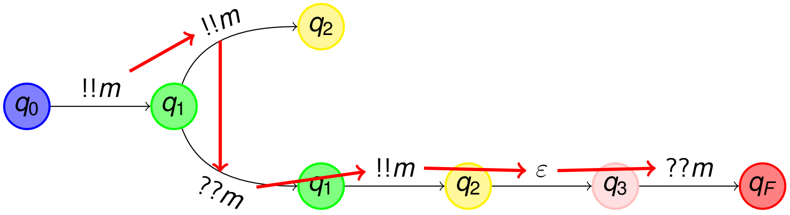
Admissible strategy patterns



An admissible strategy pattern:

- A strategy pattern

Admissible strategy patterns

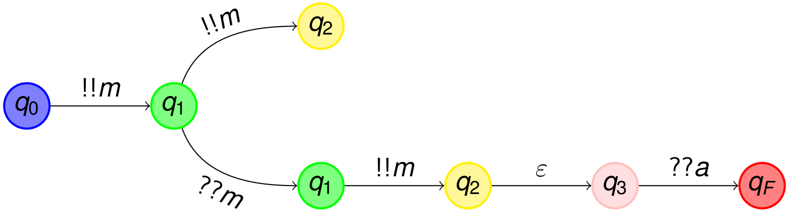


ADMISSIBLE

An admissible strategy pattern:

- A strategy pattern **+ a total order on the edge s.t.:**
  - The order in the tree is satisfied
  - Each $??m$ is preceded by $!!m$

Admissible strategy patterns

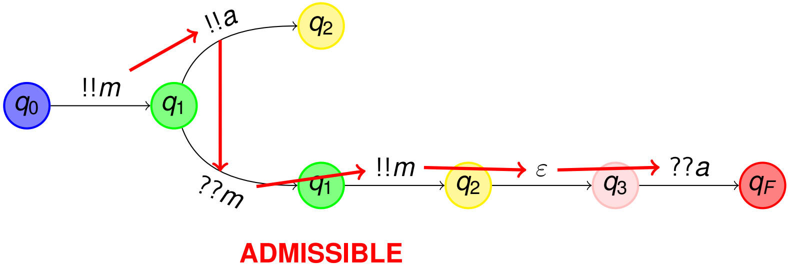


NOT ADMISSIBLE

An admissible strategy pattern:

- A strategy pattern **+ a total order on the edge s.t.:**
  - The order in the tree is satisfied
  - Each ??m is preceded by !!m

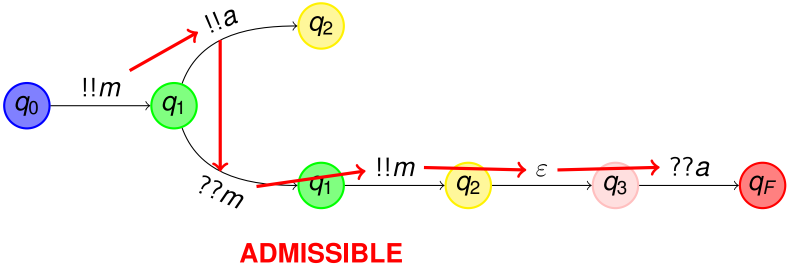
Admissible strategy patterns



An admissible strategy pattern:

- A strategy pattern **+ a total order on the edge s.t.:**
  - The order in the tree is satisfied
  - Each $??m$ is preceded by $!!m$

Admissible strategy patterns



An admissible strategy pattern:

- A strategy pattern **+ a total order on the edge s.t.:**
  - The order in the tree is satisfied
  - Each $??m$ is preceded by $!!m$

Checking whether a strategy pattern is admissible can be done in polynomial time

Results

Why reason on strategy patterns ?

Soundness and correctness

A state is reachable in Reconfigurable Networks iff there is an admissible strategy pattern containing it.

Results

Why reason on strategy patterns ?

Soundness and correctness

A state is reachable in Reconfigurable Networks iff there is an admissible strategy pattern containing it.

Minimization

If there exists an admissible strategy pattern containing q there exists one of polynomial size (in the size of P).

Results

Why reason on strategy patterns ?

Soundness and correctness

A state is reachable in Reconfigurable Networks iff there is an admissible strategy pattern containing it.

Minimization

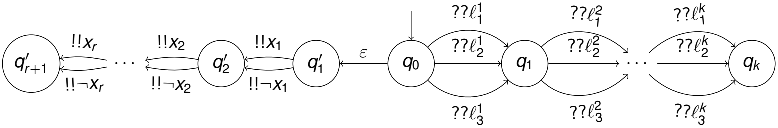
If there exists an admissible strategy pattern containing q there exists one of polynomial size (in the size of P).

Theorem

REACH[L] in Reconfigurable Networks is NP-complete.

NP-hardness

- Reduction from 3SAT
- 3SAT formula of the form $\bigwedge_{i\in[1..k]} \ell_1^i \vee \ell_2^i \vee \ell_3^i$ over the variables $\{x_1, \ldots, x_r\}$



- The local strategy ensures that even if many processes broadcast the x_i or $\neg x_i$, they will all make the same choices
- The choices of the local strategy corresponds to a valuation satisfying the formula

Concerning target

Theorem

TARGET[L] in Reconfigurable Networks is NP-complete.

Idea of the proof:

- Used again the strategy pattern
- Refine the notion of admissible
- The order needs to ensure we can 'empty' some nodes not in the target set
- The admissible tree might be bigger but is still of polynomial size

Local strategies in clique networks

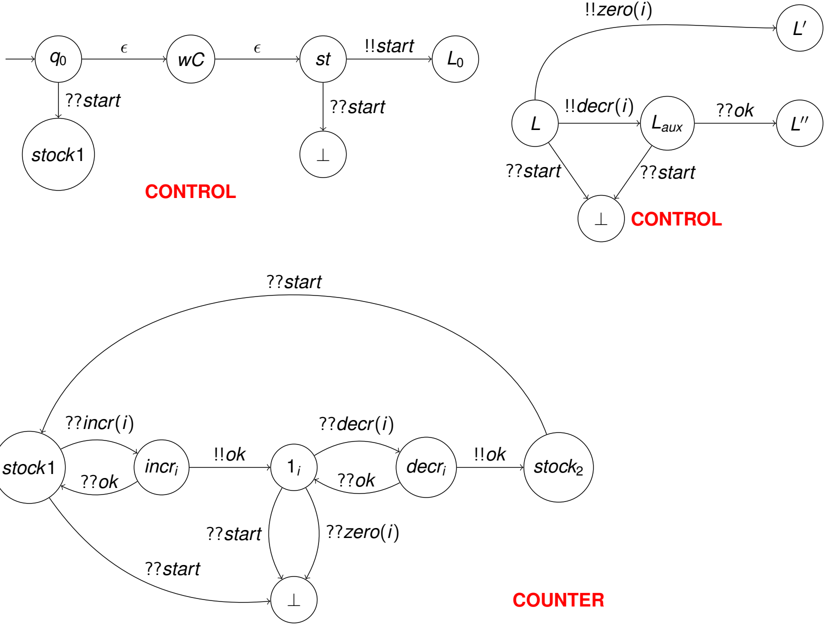
Theorem

REACH[L] and TARGET[L] are undecidable in Clique Networks.

Idea of the proof:

- Encode the behavior of a Minsky machine
- For TARGET[L], as for TARGET in Clique Networks
- For REACH[L]:
  - Simulate the same run twice
  - Locality ensures that we can do the same simulation
  - On the second run we ensure that we will use at most as manu processes for the counters as in the first run
  - As for TARGET in Clique Networks, cliques are used to guarantee that at most one process at a time changes state

Protocol for REACH[L] in Clique Networks



How to regain decidability ?

A complete protocol

- From each state, at least one edge labelled with an action (internal or broadcast)
- From each state, for each message m, an edge labelled with $??m$

For a complete protocol in a clique network, at each broadcast, all processes change their past

Theorem

REACH[L] in Clique Networks is decidable when restricted to complete protocols.

Idea of the proof:

- Use an abstract system
- Encode the number of process with the same history in a single process
- Such a system is then well-structured (the order on the configuration is a simulation)

Outline

Conclusion

Results

	Reconfigurable Networks	**Clique Networks**
REACH	Ptime	Ackermann-complete
TARGET	Ptime	Undecidable
REACH[L]	**NP-complete**	**Undecidable** **Decidable for complete protocols**
TARGET[L]	**NP-complete**	**Undecidable**

- When we get decidability, we obtain also a cutoff.

Last remarks

Many many papers on this subject

- See the survey [Esparza, STACS'14]
- Aminof et al. studied model-checking with branching time logic
- Esparza & Ganty studied communication through shared variables with no locking mechanism
- Bollig et al. studied expressivity of parameterized networks
- Bertrand et al. studied Broadcast Networks and Ad Hoc Networks with probability

And now ?

- How can this knowledge be used to verify or synthesize real distributed algorithms ?
- Often you need identity (from an infinite alphabet)
- You might have message passing systems with queues
- Or parameterized shared memory (an array whose size depends on the number of processes)