About the Termination Detection in the Asynchronous Message Passing Model

Jérémie Chalopin[1] Emmanuel Godard[2]

Yves Métivier[1] Gerard Tel[3]

[1]Université Bordeaux 1

[2]Université de Provence, Marseille

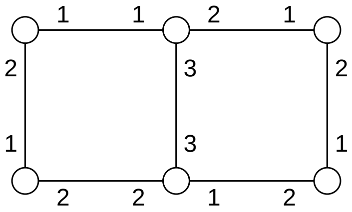
[3]University of Utrecht

SOFSEM 2007

Message passing systems

A network is represented as a graph G with a port-numbering δ where each process can

- modify its state,
- send a message via port p,
- receive a message via port q.



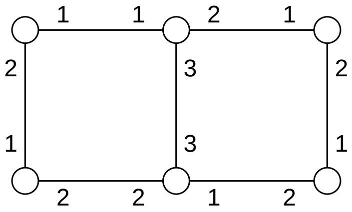
Message passing systems

A network is represented as a graph G with a port-numbering δ where each process can

- modify its state,
- send a message via port p,
- receive a message via port q.

We consider

- reliable networks,
- (potentially) anonymous systems.
- asynchronous systems.



Motivations

Different kinds of termination exist for distributed algorithms.

- Implicit termination:
  - in the final configuration, each process has computed a correct result,
  - the processes are not aware that the global computation has terminated.

Motivations

Different kinds of termination exist for distributed algorithms.

- Implicit termination:
  - in the final configuration, each process has computed a correct result,
  - the processes are not aware that the global computation has terminated.
- Explicit termination:
  - in the final configuration, each process has computed a correct result,
  - at least one process knows that the global result has been computed.

Motivations

Different kinds of termination exist for distributed algorithms.

- Implicit termination:
  - in the final configuration, each process has computed a correct result,
  - the processes are not aware that the global computation has terminated.
- Explicit termination:
  - in the final configuration, each process has computed a correct result,
  - at least one process knows that the global result has been computed.

Question

When can we transform a distributed algorithm with implicit termination into an algorithm with explicit termination ?

Motivations

What initial knowledge must we have about the network to enable this transformation ?

- the topology of the network,
- its size,
- a bound on its size,
- ...

Motivations

What initial knowledge must we have about the network to enable this transformation ?

- the topology of the network,
- its size,
- a bound on its size,
- ...

Equivalently, what are the families of graphs where we can transform any implicitly terminating distributed algorithm into an explicitly terminating algorithm ?

“Naive” strategy

- compute a snapshot,
- check that the computation is globally finished:
  - no process can modify its state (according to the algorithm),
  - no message is in transit.

“Naive” strategy

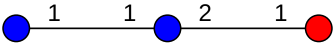
- compute a snapshot,
- check that the computation is globally finished:
  - no process can modify its state (according to the algorithm),
  - no message is in transit.

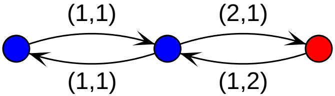
Problem

This ideal strategy cannot always be successfully applied.

From graphs to digraphs

We represent the network by a labelled digraph (G, δ, λ) where the state of each process is encoded by its label.

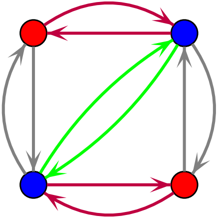




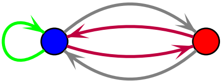
Coverings

Definition

D is a covering of D' via φ if φ is a locally bijective homomorphism.



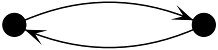
D

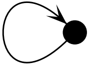


D'

Lifting Lemma

(G, δ_G)

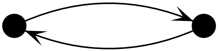


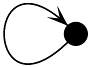


(D, δ_D)

Lifting Lemma

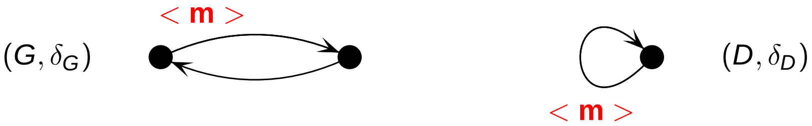
(G, δ_G)





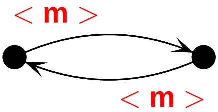
(D, δ_D)

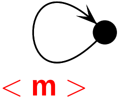
Lifting Lemma



Lifting Lemma

(G, δ_G)

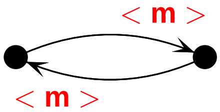


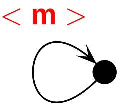


(D, δ_D)

Lifting Lemma

(G, δ_G)

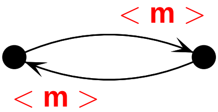


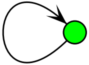


(D, δ_D)

Lifting Lemma

(G, δ_G)

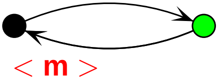


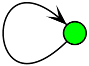


(D, δ_D)

Lifting Lemma

(G, δ_G)

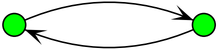


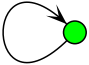


(D, δ_D)

Lifting Lemma

(G, δ_G)

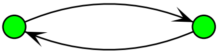


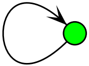


(D, δ_D)

Lifting Lemma

(G, δ_G)





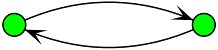
(D, δ_D)

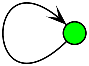
Lemma (from Angluin'80)

$$
\begin{array}{ccc}
(G, \lambda, \delta_G) & & \\
\text{covering} \downarrow & & \\
(D, \eta, \delta_D) & \xrightarrow[\mathcal{A}]{} & (D, \eta', \delta_D)
\end{array}
$$

Lifting Lemma

(G, δ_G)





(D, δ_D)

Lemma (from Angluin'80)

$$\begin{array}{ccc}
(G, \lambda, \delta_G) & \xrightarrow{\mathcal{A}} & (G, \lambda', \delta_G) \\
\text{covering} \downarrow & & \downarrow \text{covering} \\
(D, \eta, \delta_D) & \xrightarrow[\mathcal{A}]{} & (D, \eta', \delta_D)
\end{array}$$

A “better” strategy

- compute a “snapshot” up to a covering,
- check that the computation is globally finished:
  - no process can modify its state (according to the algorithm),
  - no message is in transit.

An algorithm to construct coverings

Proposition (from Mazurkiewicz'97)

There exists an algorithm $\mathcal{M}$ that terminates implicitly on any graph (G, λ, δ_G) that computes a graph (D, η, δ_D) such that:

- *(G, λ, δ_G) is a covering of (D, η, δ_D) via a homomorphism γ,*

An algorithm to construct coverings

Proposition (from Mazurkiewicz'97)

There exists an algorithm $\mathcal{M}$ that terminates implicitly on any graph (G, λ, δ_G) that computes a graph (D, η, δ_D) such that:

- *(G, λ, δ_G) is a covering of (D, η, δ_D) via a homomorphism γ,*
- *each vertex v knows (D, η, δ_D) and $\gamma(v)$.*

An algorithm to construct coverings

Proposition (from Mazurkiewicz'97)

There exists an algorithm $\mathcal{M}$ that terminates implicitly on any graph (G, λ, δ_G) that computes a graph (D, η, δ_D) such that:

- *(G, λ, δ_G) is a covering of (D, η, δ_D) via a homomorphism γ,*
- *each vertex v knows (D, η, δ_D) and $\gamma(v)$.*

One can obtain an algorithm $\mathcal{M}(\mathcal{A})$ that terminates implicitly on any graph $(G, \lambda, \delta) \in \mathcal{F}$ and that yields to a final labelling $(G, (\lambda, res_{\mathcal{A}}, res_{\mathcal{M}}), \delta)$ such that :

- $(G, res_{\mathcal{A}}, \delta)$ is the final configuration of an execution of $\mathcal{A}$ on (G, λ, δ),

An algorithm to construct coverings

Proposition (from Mazurkiewicz'97)

There exists an algorithm $\mathcal{M}$ that terminates implicitly on any graph (G, λ, δ_G) that computes a graph (D, η, δ_D) such that:

- *(G, λ, δ_G) is a covering of (D, η, δ_D) via a homomorphism γ,*
- *each vertex v knows (D, η, δ_D) and $\gamma(v)$.*

One can obtain an algorithm $\mathcal{M}(\mathcal{A})$ that terminates implicitly on any graph $(G, \lambda, \delta) \in \mathcal{F}$ and that yields to a final labelling $(G, (\lambda, res_{\mathcal{A}}, res_{\mathcal{M}}), \delta)$ such that :

- $(G, res_{\mathcal{A}}, \delta)$ is the final configuration of an execution of $\mathcal{A}$ on (G, λ, δ),
- $(G, res_{\mathcal{M}}, \delta)$ is the final configuration of the execution of $\mathcal{M}$ on $(G, (\lambda, res_{\mathcal{A}}), \delta)$.

A “better” strategy

- compute a “snapshot” up to a covering,
- check that the computation is globally finished:
  - no process can modify its state (according to the algorithm),
  - no message is in transit.

Problem

- The “snapshot” up to a covering can be computed only in the final configuration.
- The processes cannot detect the termination.

A “better” strategy

- compute a “snapshot” up to a covering,
- check that the computation is globally finished:
  - no process can modify its state (according to the algorithm),
  - no message is in transit.

Problem

- The “snapshot” up to a covering can be computed only in the final configuration.
- The processes cannot detect the termination.

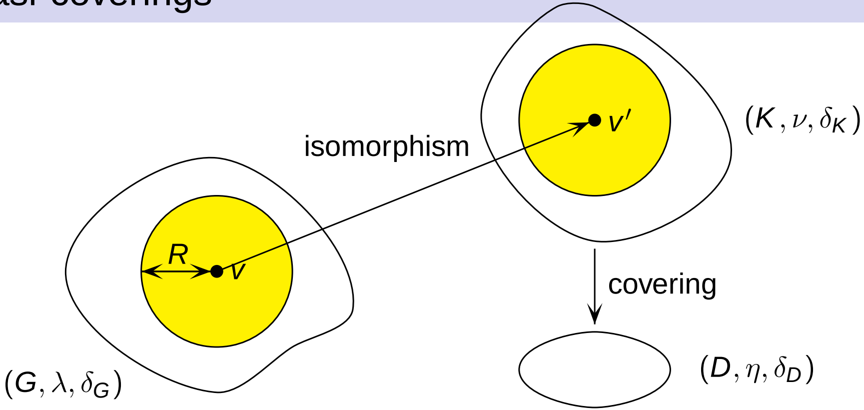
What is the meaning of the states of the processes during the execution of our “snapshot” algorithm ?

Problems

In the litterature, some impossibility results exist

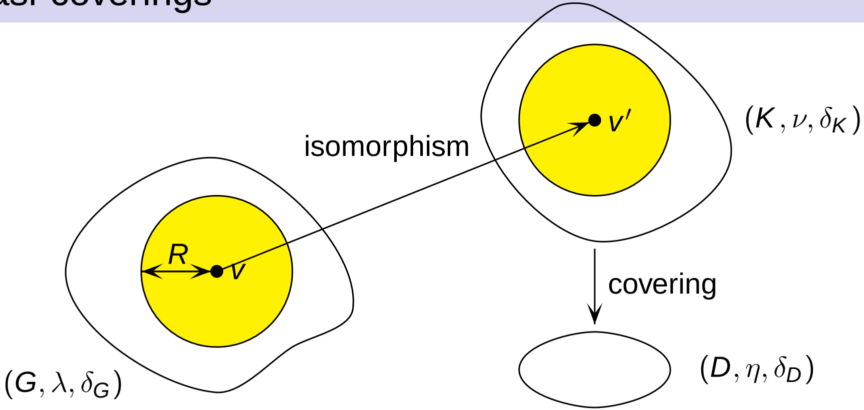
- Angluin '80 : about the detection of the termination.
- Métivier, Muscholl, Wacrenier '97 : introduction of quasi-coverings.

Quasi-coverings



(G, λ, δ_G) is a quasi-covering of (D, η, δ_D) of radius R of center v.

Quasi-coverings

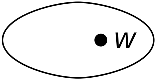


(G, λ, δ_G) is a quasi-covering of (D, η, δ_D) of radius R of center v.

If $V(G) \setminus B_G(v, k) \neq \emptyset$, the quasi-covering is strict.

Impossibility result (from MMW'97)

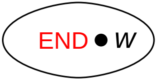
$(D, \eta, \delta_D) \in \hat{\mathcal{F}}$



- Let $\hat{\mathcal{F}} = \{(D, \eta, \delta_D) \mid \exists (G, \lambda, \delta_G)$ that covers $(D, \eta, \delta_D)\}$.
- Consider $\mathcal{T}$ that detects termination on $\mathcal{F}$ and a synchronous execution ρ of $\mathcal{T}$ on (D, η, δ_D).

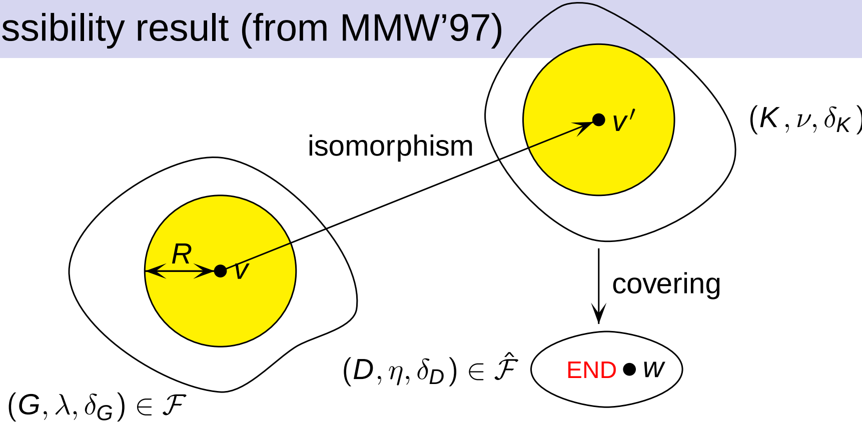
Impossibility result (from MMW'97)

$(D, \eta, \delta_D) \in \hat{\mathcal{F}}$



- Let $\hat{\mathcal{F}} = \{(D, \eta, \delta_D) \mid \exists (G, \lambda, \delta_G)$ that covers $(D, \eta, \delta_D)\}$.
- Consider $\mathcal{T}$ that detects termination on $\mathcal{F}$ and a synchronous execution ρ of $\mathcal{T}$ on (D, η, δ_D).
- Let r be the number of rounds of ρ.

Impossibility result (from MMW'97)



- Let $\hat{\mathcal{F}} = \{(D, \eta, \delta_D) \mid \exists (G, \lambda, \delta_G)$ that covers $(D, \eta, \delta_D)\}$.
- Consider $\mathcal{T}$ that detects termination on $\mathcal{F}$ and a synchronous execution ρ of $\mathcal{T}$ on (D, η, δ_D).
- Let r be the number of rounds of ρ.
- Suppose there exists $(G, \lambda, \delta_G) \in \mathcal{F}$ that is a strict quasi-covering of (D, η, δ_D) of radius $R > r$.

Main result

Theorem

Given a recursive family $\mathcal{F}$ of networks, one can detect the termination of $\mathcal{A}$ on $\mathcal{F}$

$$\Longleftrightarrow$$

there exists a computable function $r : \hat{\mathcal{F}} \to \mathbb{N}$ such that for any $(D, \lambda, \delta) \in \hat{\mathcal{F}}$, there is no strict quasi-covering of (D, λ, δ) of radius $r(D, \lambda, \delta)$ in $\mathcal{F}$.

A “good” strategy

- compute a “snapshot” up to a quasi-covering,
- check that the computation is globally finished:
  - no process can modify its state (according to the algorithm),
  - no message is in transit.

Detecting the termination of $\mathcal{M}(\mathcal{A})$

- During the computation of $\mathcal{M}(\mathcal{A})$ on (G, λ, δ_G), each vertex knows a graph $(D, \eta, \delta_D) \in \hat{\mathcal{F}}$ such that (G, λ, δ_G) is a quasi-covering of (D, η, δ_D).

Detecting the termination of $\mathcal{M}(\mathcal{A})$

- During the computation of $\mathcal{M}(\mathcal{A})$ on (G, λ, δ_G), each vertex knows a graph $(D, \eta, \delta_D) \in \hat{\mathcal{F}}$ such that (G, λ, δ_G) is a quasi-covering of (D, η, δ_D).
- Using an adaptation of an algorithm of Szymanski, Shy and Prywes ('85), we can compute in each node the radius R of this quasi-covering.

Detecting the termination of $\mathcal{M}(\mathcal{A})$

- During the computation of $\mathcal{M}(\mathcal{A})$ on (G, λ, δ_G), each vertex knows a graph $(D, \eta, \delta_D) \in \hat{\mathcal{F}}$ such that (G, λ, δ_G) is a quasi-covering of (D, η, δ_D).
- Using an adaptation of an algorithm of Szymanski, Shy and Prywes ('85), we can compute in each node the radius R of this quasi-covering.
- If $R > r(D, \eta, \delta_D)$, then the quasi-covering cannot be strict (by definition of the function r).

Detecting the termination of $\mathcal{M}(\mathcal{A})$

- During the computation of $\mathcal{M}(\mathcal{A})$ on (G, λ, δ_G), each vertex knows a graph $(D, \eta, \delta_D) \in \hat{\mathcal{F}}$ such that (G, λ, δ_G) is a quasi-covering of (D, η, δ_D).
- Using an adaptation of an algorithm of Szymanski, Shy and Prywes ('85), we can compute in each node the radius R of this quasi-covering.
- If $R > r(D, \eta, \delta_D)$, then the quasi-covering cannot be strict (by definition of the function r).
- (G, λ, δ_G) is a covering of (D, η, δ_D) and all vertices of G have computed their final values.

Applications

- Known corollaries :
  One can detect termination of an algorithm $\mathcal{A}$ on a network (G, λ, δ) if one of the following conditions is satisfied.
  - existence of a leader,
  - unique ids,
  - initial knowledge of a bound on the size of the network.

Applications

- Known corollaries :
  One can detect termination of an algorithm $\mathcal{A}$ on a network (G, λ, δ) if one of the following conditions is satisfied.
  - existence of a leader,
  - unique ids,
  - initial knowledge of a bound on the size of the network.

- New corollaries :
  One can detect termination of an algorithm $\mathcal{A}$ on a network (G, λ, δ) if one of the following conditions is satisfied.
  - existence of at most k distinguished vertices,
  - initial knowledge of a bound on the multiplicity of each initial label.