What can be verified locally?

Alkida Balliu

IRIF Paris and GSSI L'Aquila

Goal

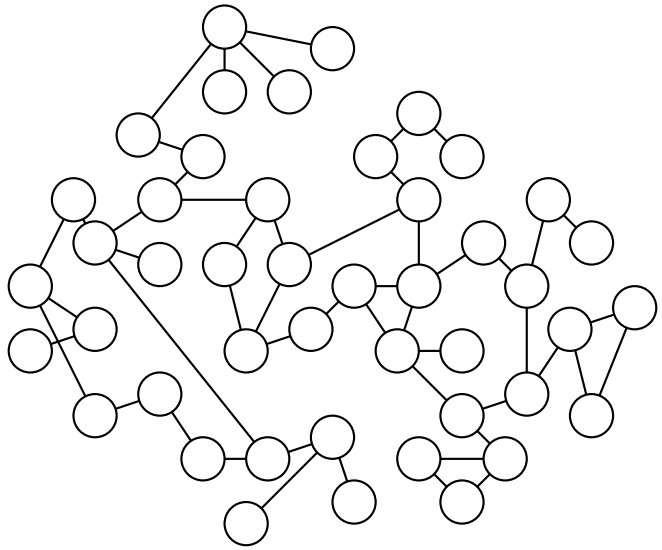
- Complexity theory in the distributed setting.
- Classify problems according to their difficulty.

Overview

- LOCAL model;
- Decision problems;
- Verification problems;
- Hierarchy of complexity classes in the LOCAL model.

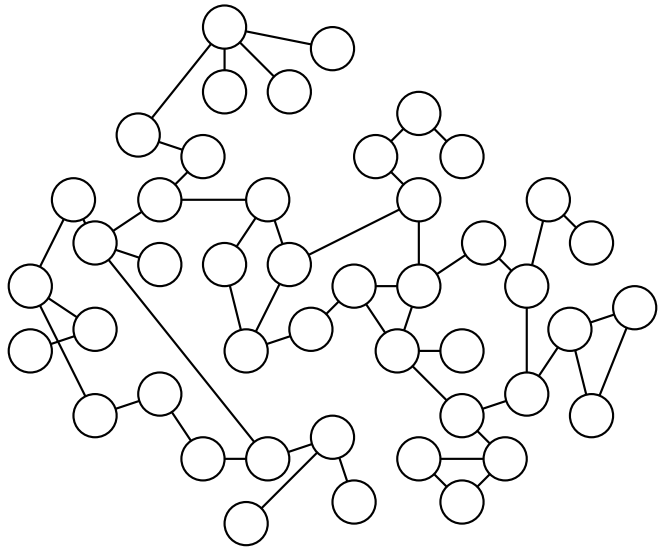
LOCAL Model

- The distributed network is represented by a graph.



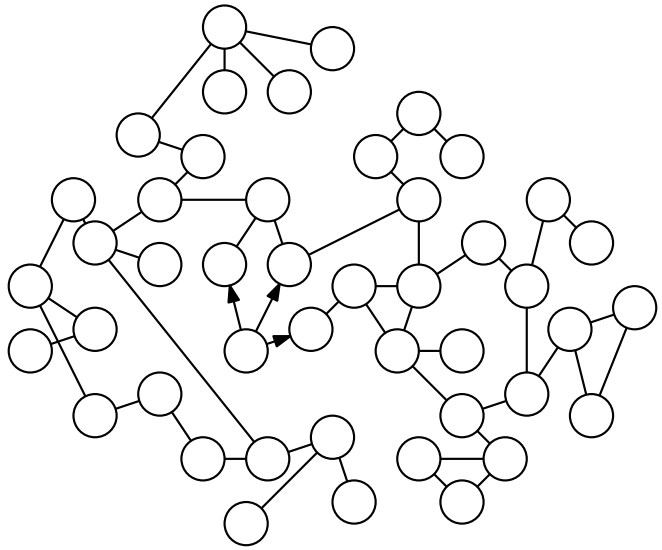
LOCAL Model

- The distributed network is represented by a graph.
- Synchronous model.



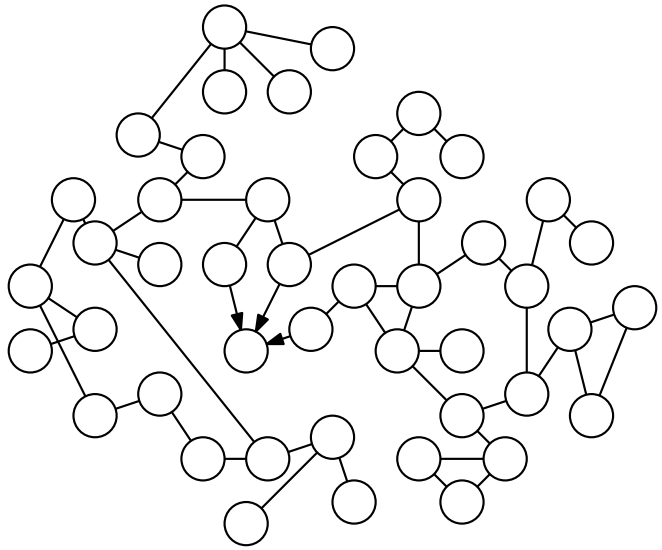
LOCAL Model

- The distributed network is represented by a graph.
- Synchronous model.



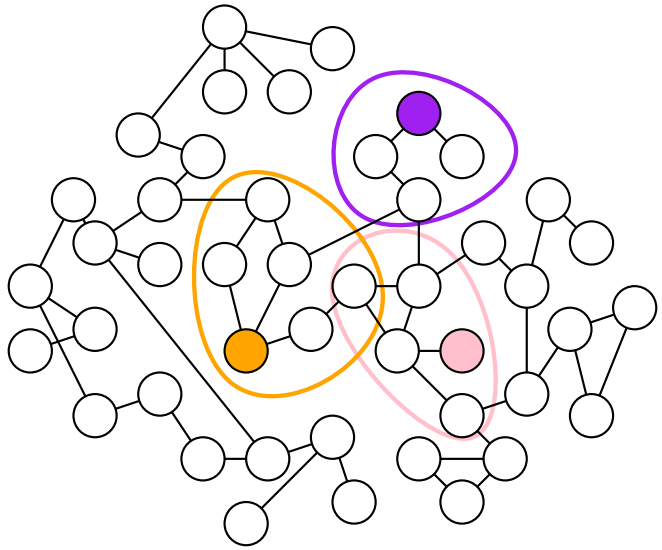
LOCAL Model

- The distributed network is represented by a graph.
- Synchronous model.



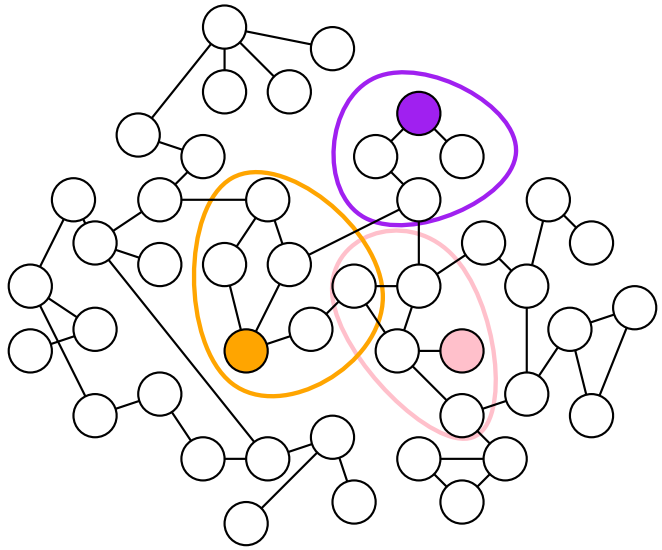
LOCAL Model

- The distributed network is represented by a graph.
- Synchronous model.
- Equivalent to a model where each node sees the network up to distance t.



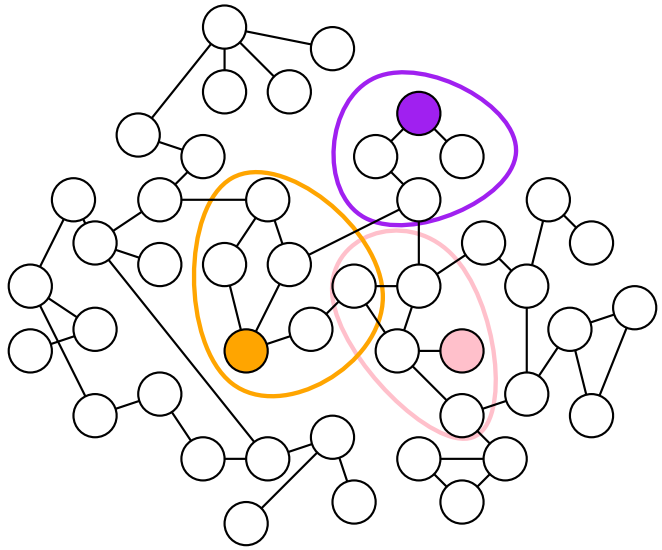
LOCAL Model

- The distributed network is represented by a graph.
- Synchronous model.
- Equivalent to a model where each node sees the network up to distance t.
- The time complexity of a local algorithm $\mathcal{A}$ is determined by the range t that it needs to explore.



LOCAL Model

- The distributed network is represented by a graph.
- Synchronous model.
- Equivalent to a model where each node sees the network up to distance t.
- The time complexity of a local algorithm $\mathcal{A}$ is determined by the range t that it needs to explore.
- We want t to be constant.

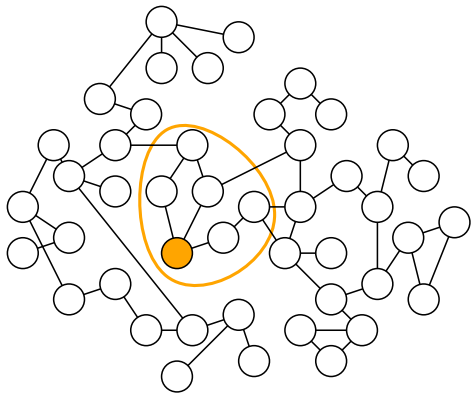


Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.

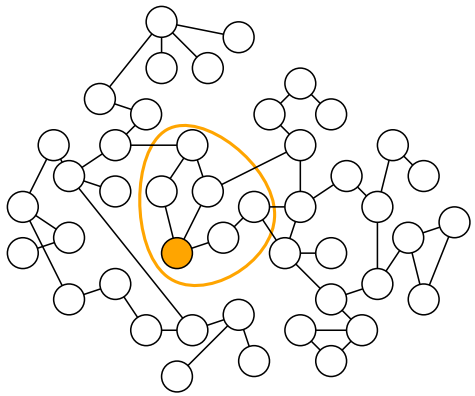
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;



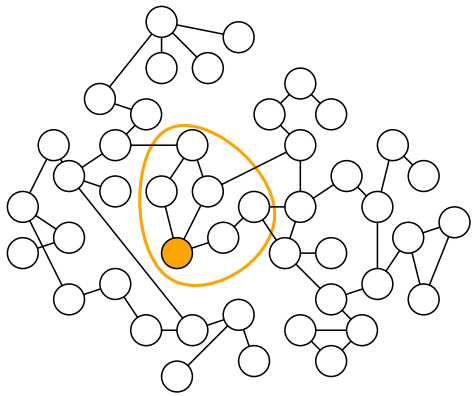
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;



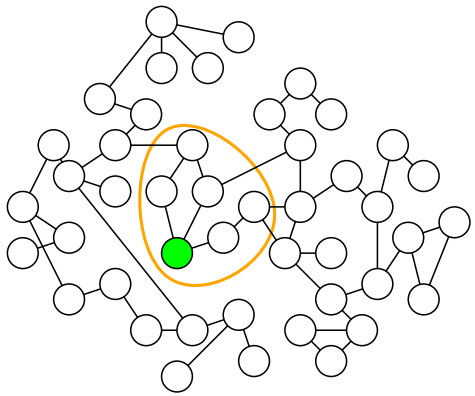
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision:



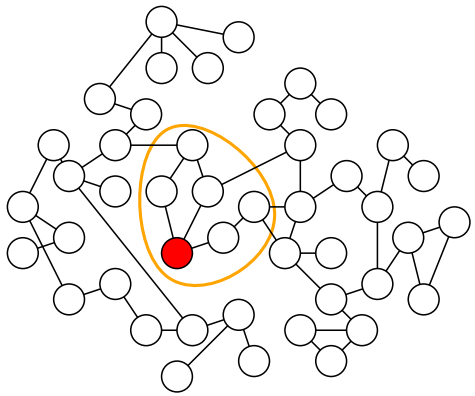
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept"



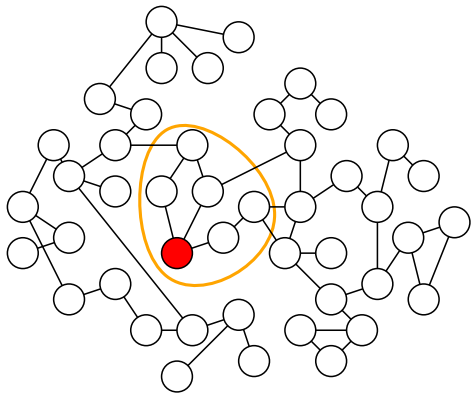
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



Decision Problems

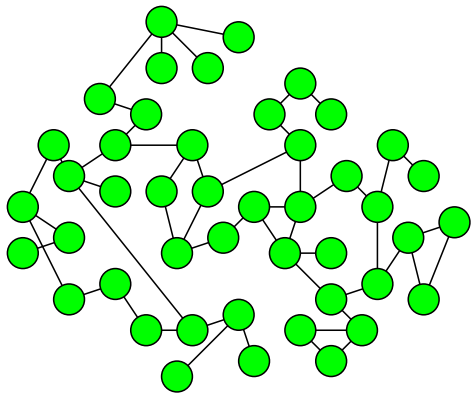
- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



- $global\_output = \bigwedge_{v \in V} local\_output(v)$.

Decision Problems

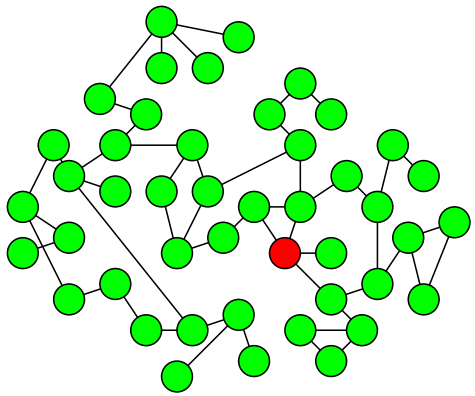
- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



- $global\_output = \bigwedge_{v \in V} local\_output(v)$.

Decision Problems

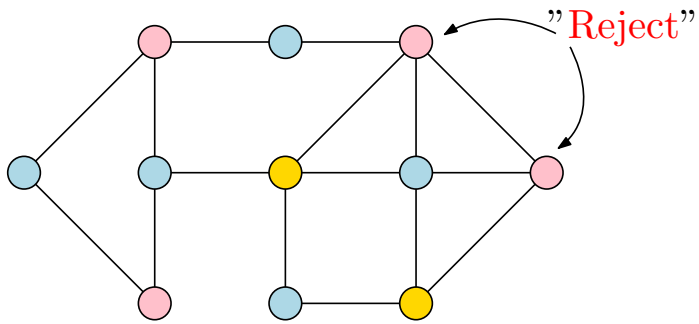
- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



- $global\_output = \bigwedge_{v \in V} local\_output(v)$.

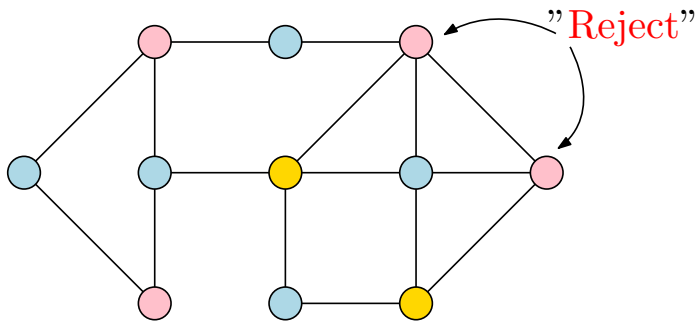
Example: Proper Coloring

- Node input: a color.
- Each node checks the colors of its neighbors.



Example: Proper Coloring

- Node input: a color.
- Each node checks the colors of its neighbors.



- *Local Decision (LD)* is the class of distributed languages that can be locally decided [NS '95].

LD Class

LD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ satisfying the following: for every input instance (G, x),

$$\begin{array}{rcl} (G,x) \in \mathcal{L} & \Rightarrow & \forall \mathrm{id} \in \mathrm{ID}(G), \forall u \in V(G), \mathcal{A}(G,x,\mathrm{id},u) = \text{accept} \\ (G,x) \notin \mathcal{L} & \Rightarrow & \forall \mathrm{id} \in \mathrm{ID}(G), \exists u \in V(G), \mathcal{A}(G,x,\mathrm{id},u) = \text{reject} \end{array}$$

LD Class

LD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ satisfying the following: for every input instance (G, x),

$$\begin{aligned} (G,x) \in \mathcal{L} &\Rightarrow \forall \mathrm{id} \in \mathrm{ID}(G), \forall u \in V(G), \mathcal{A}(G,x,\mathrm{id},u) = \text{accept} \\ (G,x) \notin \mathcal{L} &\Rightarrow \forall \mathrm{id} \in \mathrm{ID}(G), \exists u \in V(G), \mathcal{A}(G,x,\mathrm{id},u) = \text{reject} \end{aligned}$$

$\mathcal{L} \in \mathrm{P}$ if there is a polynomial time algorithm A such that,

$$x \in \mathcal{L} \iff A \text{ accepts } x.$$

Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.

Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, arbitrary size and independent from the id assignment;

Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, arbitrary size and independent from the id assignment;
  - gathers its local information from the network;

Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, arbitrary size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;

Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, arbitrary size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision, that is ether "accept" or "reject".

Verification Problems

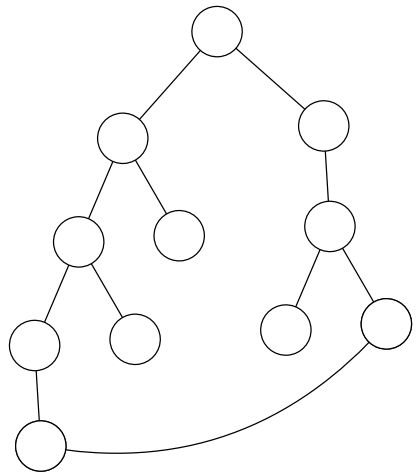
- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, arbitrary size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision, that is ether "accept" or "reject".
- $global\_output = \bigwedge_{v \in V} local\_output(v)$.

Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, arbitrary size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision, that is ether "accept" or "reject".
- $global\_output = \bigwedge_{v \in V} local\_output(v)$.
- Similar to PLS, but with id-independent certificates.

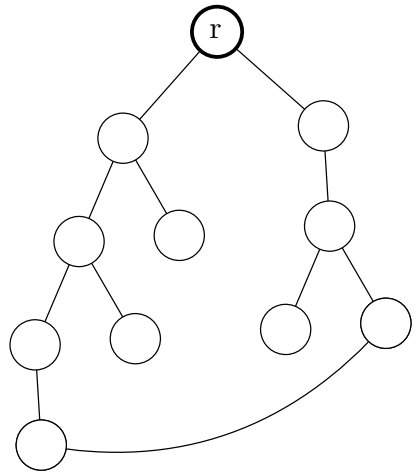
Example: Is the Given Graph a Tree?

- Not locally decidable, but locally verifiable.



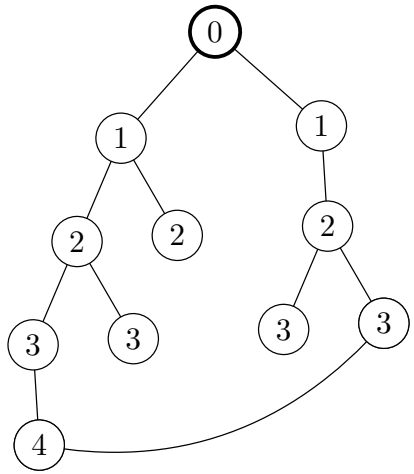
Example: Is the Given Graph a Tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.



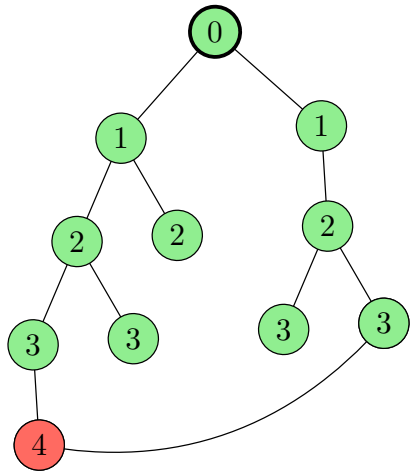
Example: Is the Given Graph a Tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
- Certificate of a node v: its hop-distance from the chosen root.



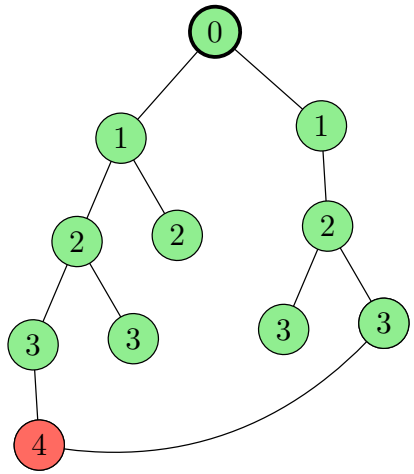
Example: Is the Given Graph a Tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
- Certificate of a node v: its hop-distance from the chosen root.



Example: Is the Given Graph a Tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
- Certificate of a node v: its hop-distance from the chosen root.



- *Nondeterministic LD (NLD)* is the class of distributed languages that can be locally verified [FKP '11].

NLD Class

NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ satisfying the following: for every input instance (G, x),

- $(G, x) \in \mathcal{L} \Rightarrow \exists c \in \mathcal{C}(G), \forall \mathrm{id} \in \mathrm{ID}(G), \forall u \in V(G),$
  $\mathcal{A}(G, x, c, \mathrm{id}, u) = \mathrm{accept}$
- $(G, x) \notin \mathcal{L} \Rightarrow \forall c \in \mathcal{C}(G), \forall \mathrm{id} \in \mathrm{ID}(G), \exists u \in V(G),$
  $\mathcal{A}(G, x, c, \mathrm{id}, u) = \mathrm{reject}$

NLD Class

NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ satisfying the following: for every input instance (G, x),

- $(G, x) \in \mathcal{L} \Rightarrow \exists c \in \mathcal{C}(G), \forall \mathrm{id} \in \mathrm{ID}(G), \forall u \in V(G),$
  $\mathcal{A}(G, x, c, \mathrm{id}, u) = \text{accept}$
- $(G, x) \notin \mathcal{L} \Rightarrow \forall c \in \mathcal{C}(G), \forall \mathrm{id} \in \mathrm{ID}(G), \exists u \in V(G),$
  $\mathcal{A}(G, x, c, \mathrm{id}, u) = \text{reject}$

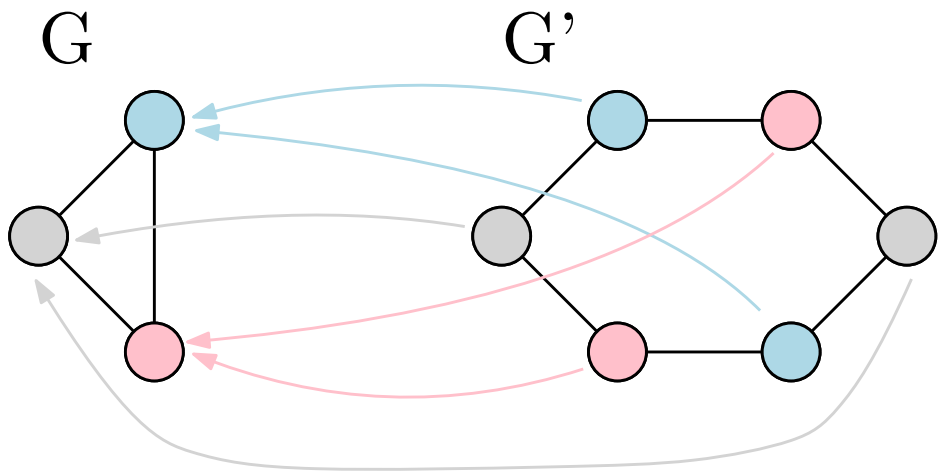
$\mathcal{L} \in$ NP if there is a polynomial time algorithm A such that,

$$x \in L \iff \exists c \text{ s.t. } A \text{ accepts } x \text{ with } c.$$

More About NLD

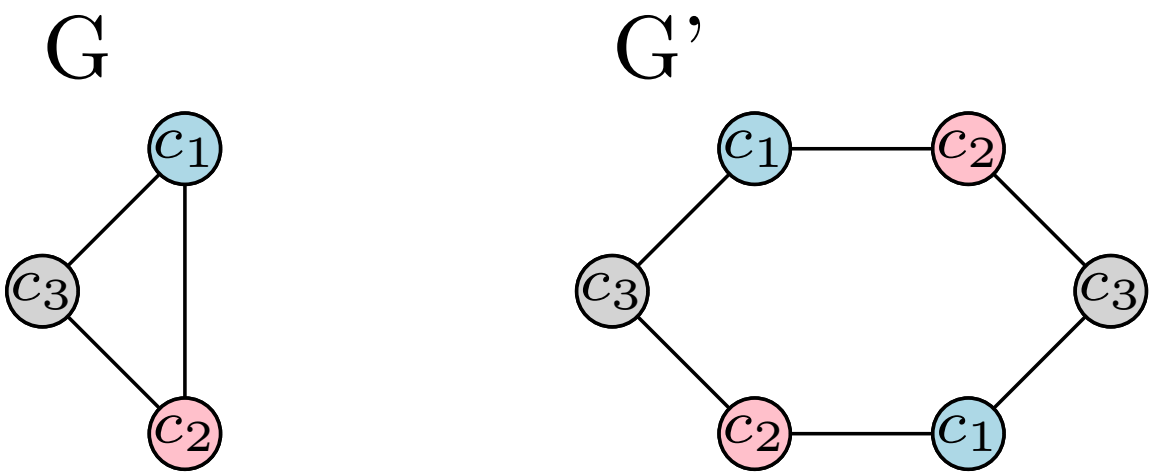
NLD is the class of all problems closed under lift [FKP '11].

- Let (G, x) and (G', x') be two input instances.
- (G', x') is a lift of (G, x) if there exists a function f such that: $f : V(G') \to V(G)$ preserving the local view of each node.



NLD is Closed Under Lift

- Let $\mathcal{L}$ be a language in NLD.
- If $(G, x) \in \mathcal{L} \ \wedge \ (G', x')$ is a lift of (G, x), then $(G', x') \in \mathcal{L}$.



Goal

- Build a hierarchy of complexity classes in the distributed setting.

Complexity Classes

- $\mathsf{LD} = \Sigma_0^{loc} = \Pi_0^{loc}$ (similar to P in the sequential setting).

Complexity Classes

- LD $= \Sigma_0^{loc} = \Pi_0^{loc}$ (similar to P in the sequential setting).
- NLD $= \Sigma_1^{loc}$ (similar to NP in the sequential setting).

Complexity Classes

- LD $= \Sigma_0^{loc} = \Pi_0^{loc}$ (similar to P in the sequential setting).
- NLD $= \Sigma_1^{loc}$ (similar to NP in the sequential setting).
- Σ_k^{loc}: An input instance satisfies a certain property in Σ_k^{loc} iff

$$\exists c_1, \forall c_2, \ldots, Q c_k, \text{ all nodes accept.}$$

Complexity Classes

- $\mathrm{LD} = \Sigma_0^{loc} = \Pi_0^{loc}$ (similar to P in the sequential setting).
- $\mathrm{NLD} = \Sigma_1^{loc}$ (similar to NP in the sequential setting).
- Σ_k^{loc}: An input instance satisfies a certain property in Σ_k^{loc} iff

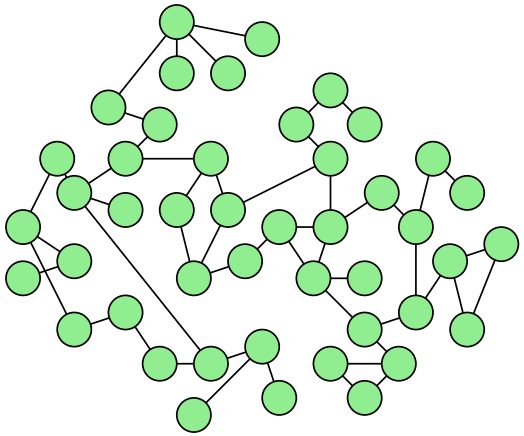
$$\exists c_1, \forall c_2, \ldots, Qc_k, \text{ all nodes accept.}$$

- Π_k^{loc}: An input instance satisfies a certain property in Π_k^{loc} iff

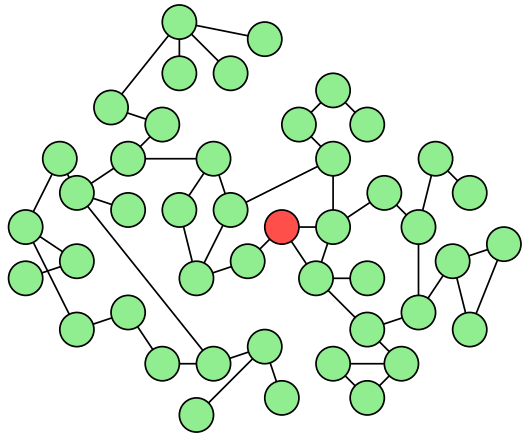
$$\forall c_1, \exists c_2, \ldots, Qc_k, \text{ all nodes accept.}$$

Complementary Classes

In a class:

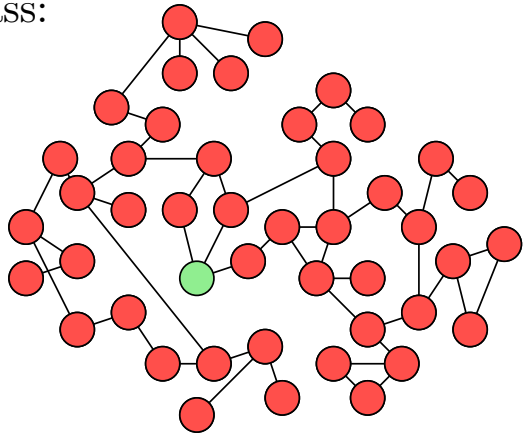


A globaly accepted input instance.

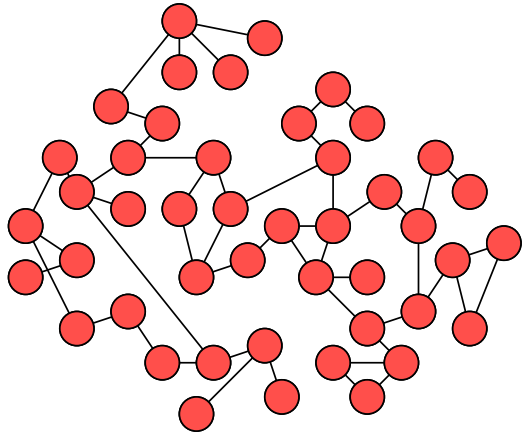


A globaly rejected imput instance.

In a complementary class:



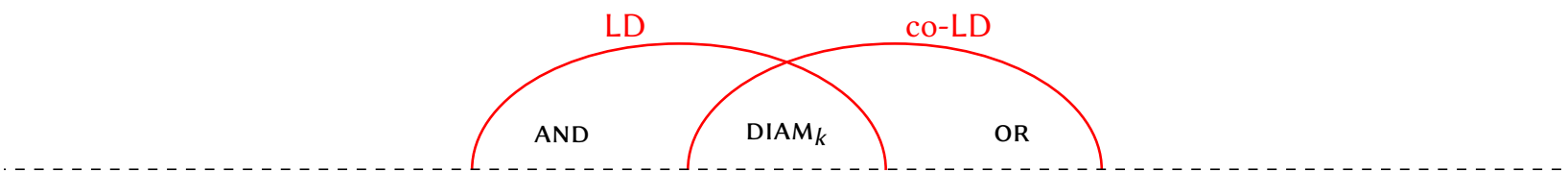
A globaly accepted input instance.



A globaly rejected imput instance.

Level 0 of the Hierarchy

- AND : $\left|\{u \in V(G) : x(u) = 1\}\right| = 0$
- OR : $\left|\{u \in V(G) : x(u) = 1\}\right| \geq 1$



Π_1^{loc}: The Role of the Last Universal Quantifier

- Π_1^{loc}:

$$(G, x) \in \mathcal{L} \Leftrightarrow \forall c \text{ all nodes accept.}$$

- LD:

$$(G, x) \in \mathcal{L} \Leftrightarrow \text{ all nodes accept}$$

Π_1^{loc}: The Role of the Last Universal Quantifier

- Π_1^{loc}:

$$(G, x) \in \mathcal{L} \Leftrightarrow \forall c \text{ all nodes accept.}$$

- LD:

$$(G, x) \in \mathcal{L} \Leftrightarrow \text{ all nodes accept}$$

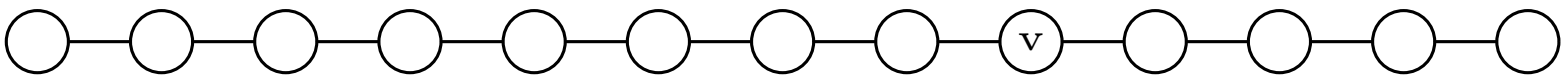
- Problems that can be solved only if a specific node knows (an upper bound of) the size of the network!

ITER

- Let f be a function and a and b two non-negative integers.

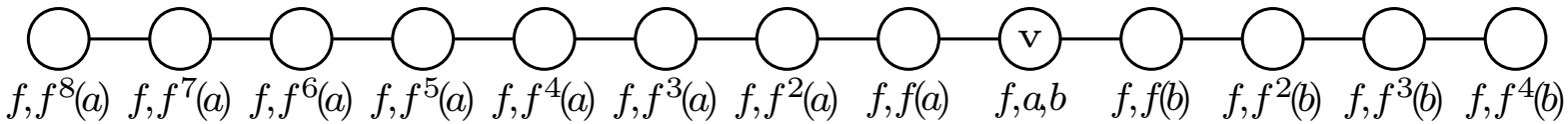
ITER

- Let f be a function and a and b two non-negative integers.
- A configuration in ITER consists in a path $P = LvR$ with a special node v (*pivot*).



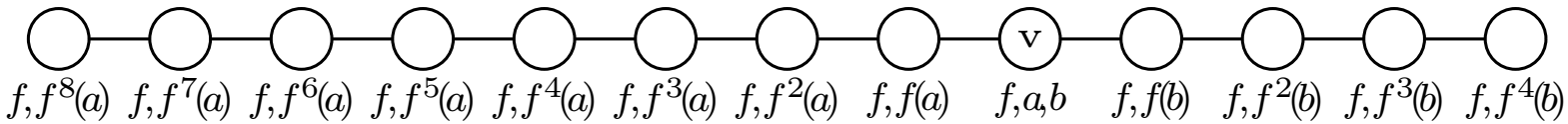
ITER

- Let f be a function and a and b two non-negative integers.
- A configuration in ITER consists in a path $P = LvR$ with a special node v (*pivot*).
- Nodes in L (resp., in R) are given as input $f, f^i(a)$ (resp., $f, f^i(b)$); to v is given in input f, a, b.



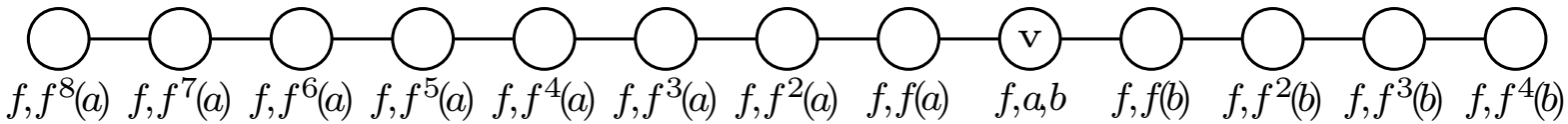
ITER

- Let f be a function and a and b two non-negative integers.
- A configuration in ITER consists in a path $P = LvR$ with a special node v (*pivot*).
- Nodes in L (resp., in R) are given as input $f, f^i(a)$ (resp., $f, f^i(b)$); to v is given in input f, a, b.
- f is s.t. $f(0) = 0$ and $f(1) = 1$

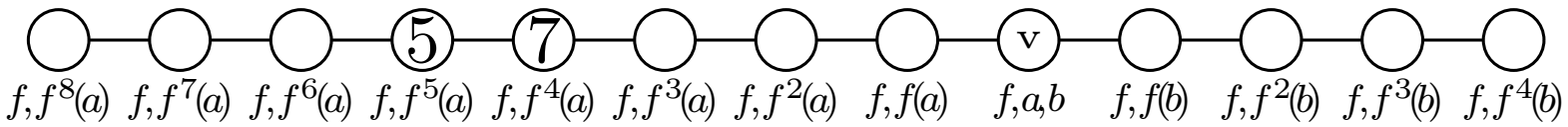


ITER

- Let f be a function and a and b two non-negative integers.
- A configuration in ITER consists in a path $P = LvR$ with a special node v (*pivot*).
- Nodes in L (resp., in R) are given as input $f, f^i(a)$ (resp., $f, f^i(b)$); to v is given in input f, a, b.
- f is s.t. $f(0) = 0$ and $f(1) = 1$
- An input instance is in ITER if and only if:
  - $f^{|L|}(a) \in \{0, 1\}$ and $f^{|R|}(b) \in \{0, 1\}$
  - $f^{|L|}(a) = 0$ or $f^{|R|}(b) = 0$



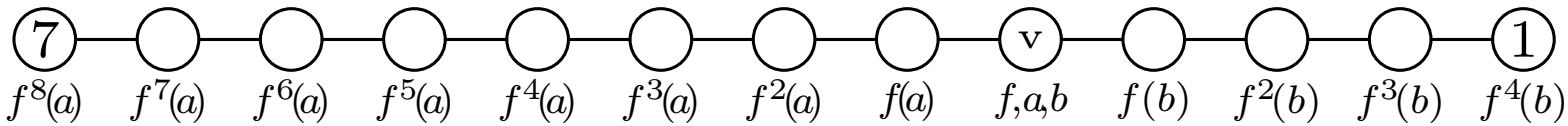
ITER



$$f(7) = 6 \neq 5$$

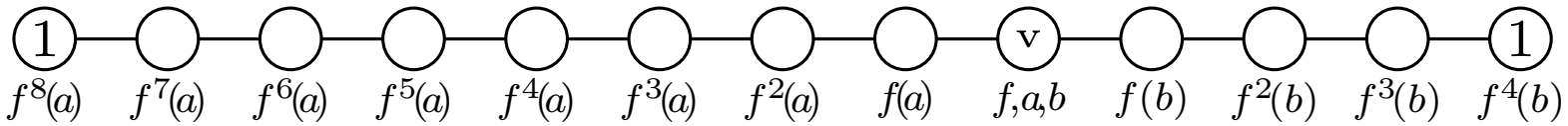
- Nodes reject if they notice local inconsistencies.

ITER



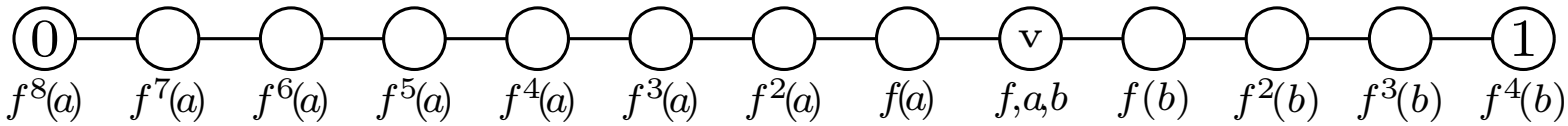
- An endpoint node rejects only if it has in input something different from 1 or 0; otherwise accepts.
- In this case, the left endpoint node rejects.

ITER



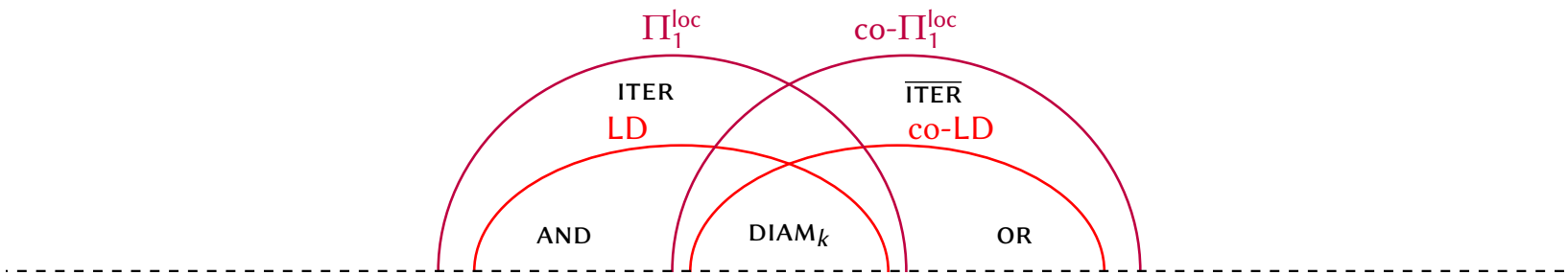
- $(G, x) \notin \mathcal{L} \Rightarrow \exists c$ s.t. at least one node rejects.
- v rejects only if $f^{|L|}(a) = f^{|R|}(b) = 1$; otherwise accepts.
- Certificate: an upper bound of the size of the network.

ITER

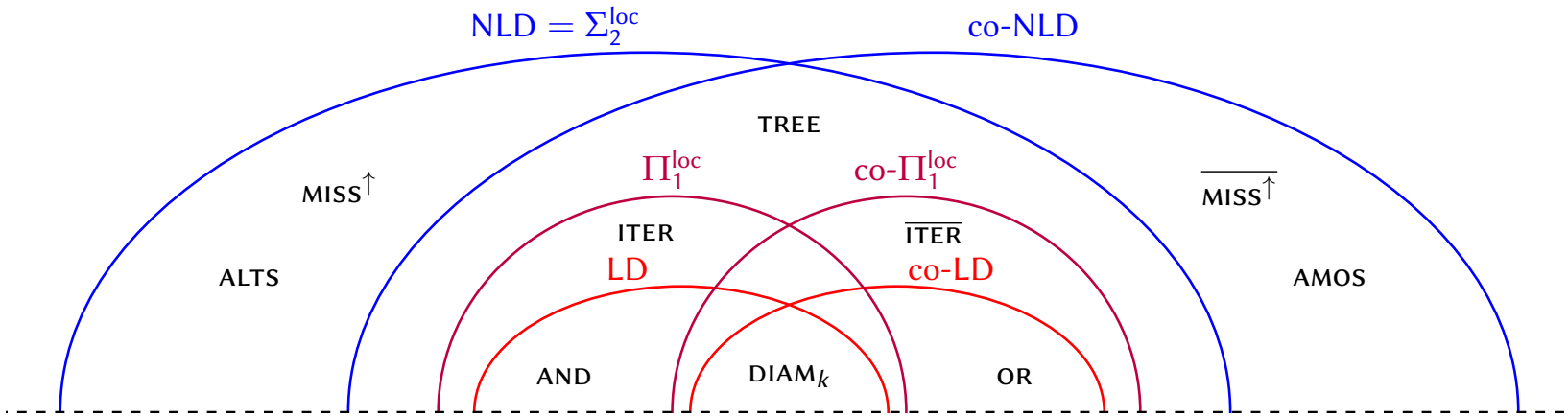


- $(G, x) \in \mathcal{L} \Rightarrow \forall c$ s.t. all nodes accept.
- Whatever certificate v has, it will never compute $f^{|L|}(a) = f^{|R|}(b) = 1$.

Local Hierarchy



Local Hierarchy



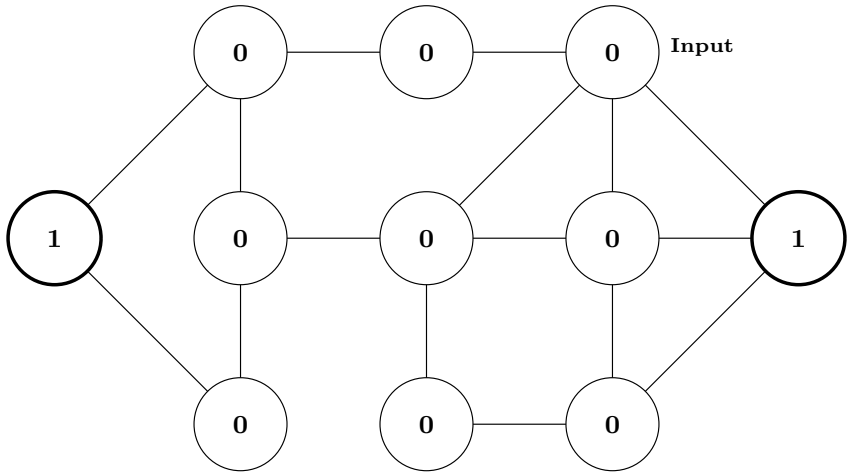
Π_2^{loc} Class

- Π_2^{loc} class: An input instance satisfies a certain property in Π_2 iff

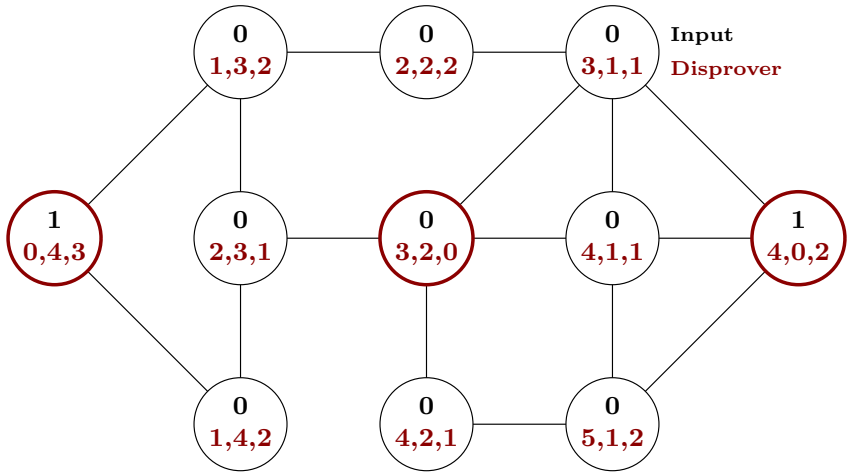
$$\forall c_1, \exists c_2, \text{ all nodes accept.}$$

- Two party game between a *disprover* and a *prover*.

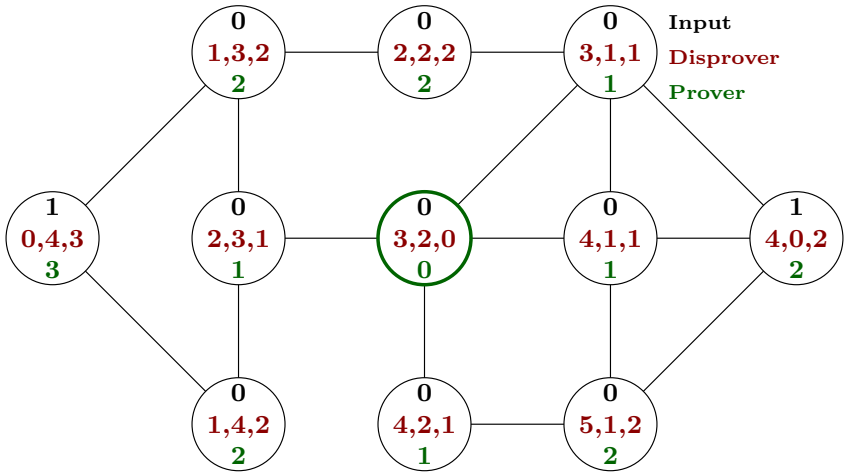
Exactly Two Selected



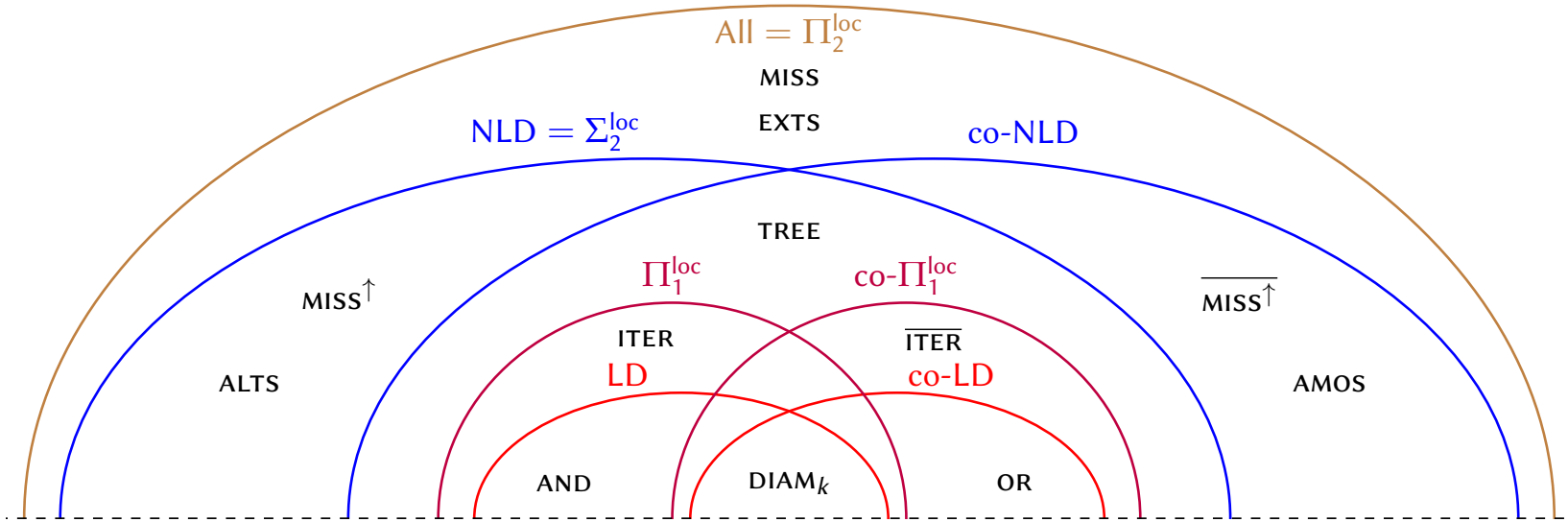
Exactly Two Selected



Exactly Two Selected



Local Hierarchy [B., D'Angelo, Fraigniaud, Olivetti '17]



$$\text{LD} \subset \Pi_1^{loc} \subset \text{NLD} = \Sigma_2^{loc} \subset \Pi_2^{loc} = \text{All (all inclusions are strict).}$$

Another local hierarchy [Feuilloley, Fraigniaud, Hirvonen '16]

The Context

- LOCAL model.
- Bounded-size certificates: $O(\log n)$ bits.
- Id-dependent certificates.

Another local hierarchy [Feuilloley, Fraigniaud, Hirvonen '16]

The Context

- LOCAL model.
- Bounded-size certificates: $O(\log n)$ bits.
- Id-dependent certificates.

- **The hierarchy**: $\forall i \geq 0$, λ_i is the class of distributed languages $\mathcal{L}$ s.t. $\exists A$, s.t. $\forall$ Id-assignment, the input instance is in $\mathcal{L}$ iff:

$$Qc_1 \ldots \forall c_{i-1} \exists c_i \text{ s.t. } \forall v \in V,\ A(G, x, c_1, \ldots, c_i, Id, v) = \text{accept}$$

where $Qc_1 = \forall$ if i is even and $Qc_1 = \exists$ otherwise.

The λ_i hierarchy

- $\lambda_0 = \text{LD}$.
- λ_1 similar to NLD.
- λ_2 similar to Π_2^{loc}.

  $\vdots$

- $\lambda_i = Qc_1 \ldots \forall c_{i-1} \exists c_i$ s.t. $\forall v \in V,\ A(G, x, c_1, \ldots, c_i, Id, v) = \text{accept}$

The λ_i hierarchy

- $\lambda_0 = \mathsf{LD}$.
- λ_1 similar to NLD.
- λ_2 similar to Π_2^{loc}.

  $\vdots$
- $\lambda_i = Qc_1 \ldots \forall c_{i-1} \exists c_i$ s.t. $\forall v \in V,\ A(G, x, c_1, \ldots, c_i, Id, v) =$ accept
- Let λ_i^o be the corresponding class of λ_i where certificates are of $O(\log n)$ bits and id-oblivious.

Mapping between the 2 hierarchies

- $\lambda_1^o \subsetneq \lambda_1$
  Certifying a spanning tree: not closed under lift.

Mapping between the 2 hierarchies

- $\lambda_1^o \subsetneq \lambda_1$
  Certifying a spanning tree: not closed under lift.
- $\lambda_i^o = \lambda_i, \forall i \geq 2$
  Idea: give an Id assignment with the first certificate and let the other certificates to depend on this Id assignment (also check the correctness of this Id assignment).

Open Problems

- Unbounded size id-independent certificates:
  - find a complete problem for Π_1^{loc} and co-Π_1^{loc};
  - find a problem in the intersection between the classes Π_1^{loc} and co-Π_1^{loc}.
- Bounded size ($O(\log n)$) id-dependent certificates
  - we don't know if the hierarchy collapses;
  - there are no separating problems for λ_2 and λ_3 (neither for classes higher in the hierarchy).

Thank you!

References

- Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. What can be verified locally?, *STACS 2017*.
- Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. Brief Announcement: Local distributed verification, *DISC 2016*.
- Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing, *J. ACM 2013 (FOCS 2011)*.
- Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A hierarchy of local decision, *ICALP 2016*.
- Fabian Reiter. Distributed graph automata, *LICS 2015*.
- Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM J. Comput. 1995*.