What can be verified locally?

Alkida Balliu, Gianlorenzo D'Angelo,
Pierre Fraigniaud, and Dennis Olivetti

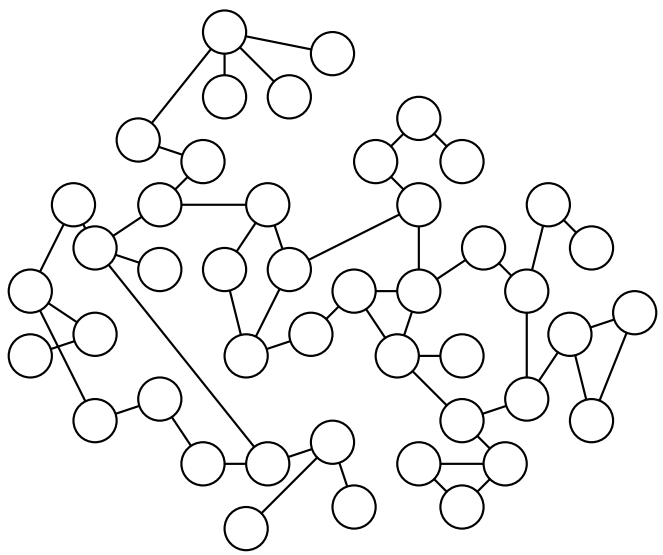
Gran Sasso Science Institute, Italy
CNRS and University Paris Diderot, France

Goal

- Classify problems in the distributed context, according to their difficulty.
- Build a hierarchy of complexity classes in the context of the LOCAL model.

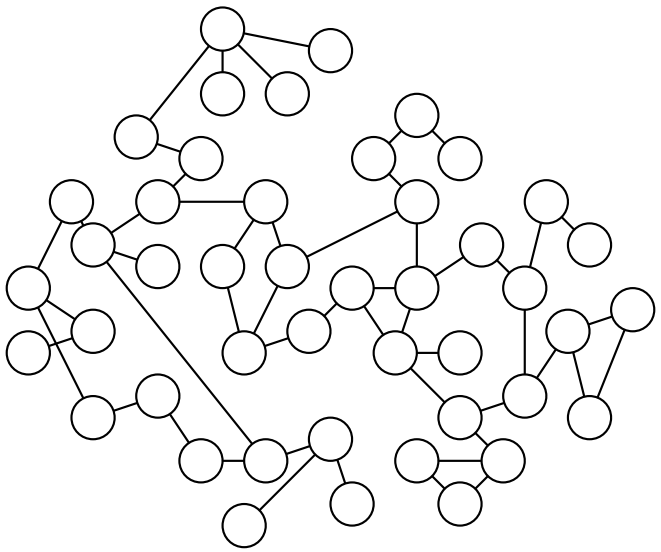
Local Model

- The distributed network is represented by a graph.



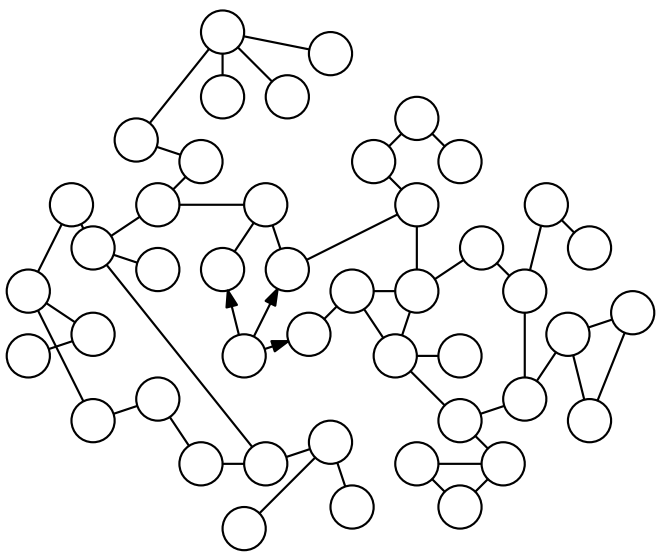
Local Model

- The distributed network is represented by a graph.
- Synchronous model: the computation proceeds in *rounds*.



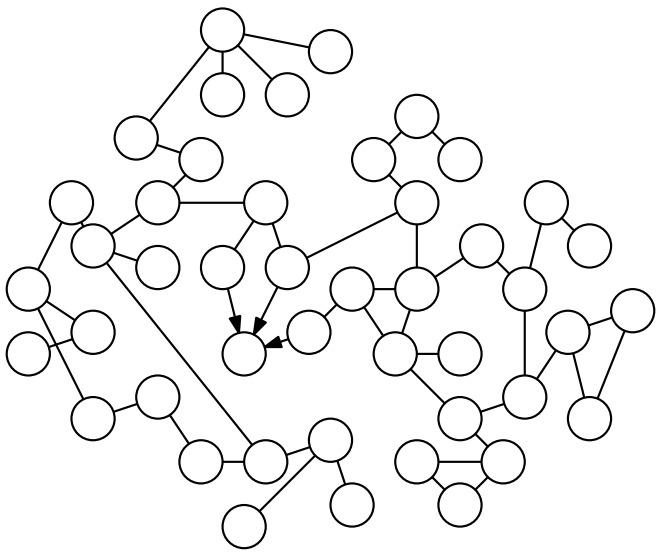
Local Model

- The distributed network is represented by a graph.
- Synchronous model: the computation proceeds in *rounds*.



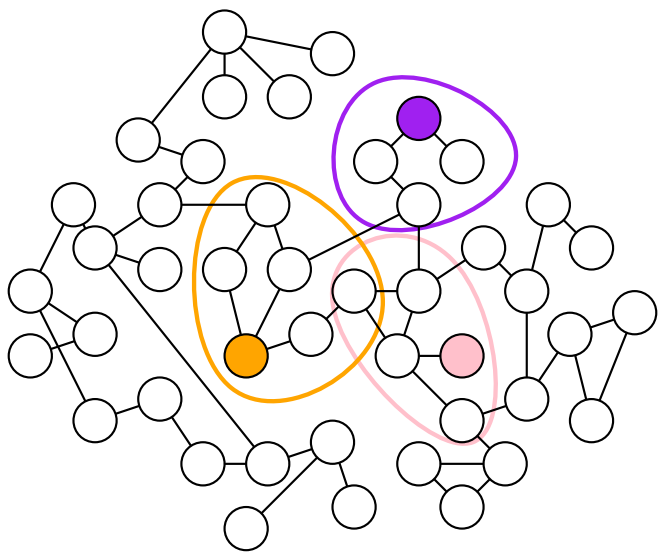
Local Model

- The distributed network is represented by a graph.
- Synchronous model: the computation proceeds in *rounds*.



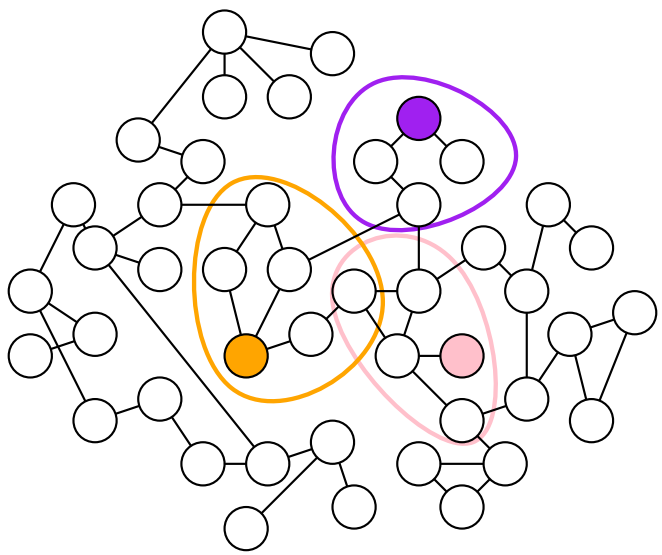
Local Model

- The distributed network is represented by a graph.
- Synchronous model: the computation proceeds in *rounds*.
- Equivalent to a model where each node sees the network up to distance t.



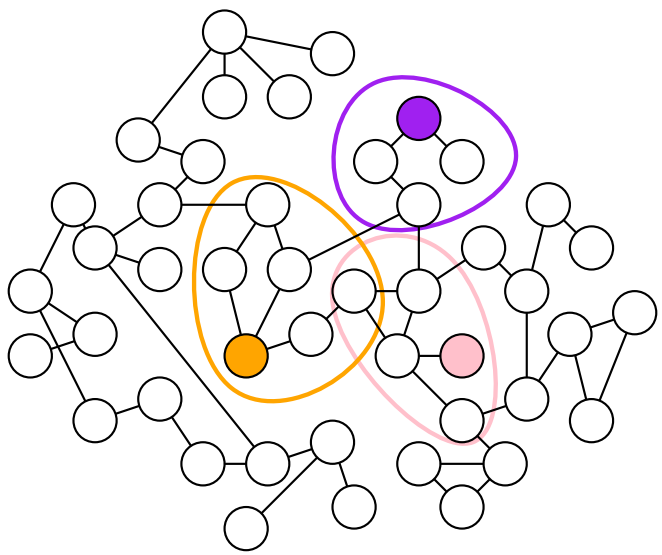
Local Model

- The distributed network is represented by a graph.
- Synchronous model: the computation proceeds in *rounds*.
- 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: the computation proceeds in *rounds*.
- 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.

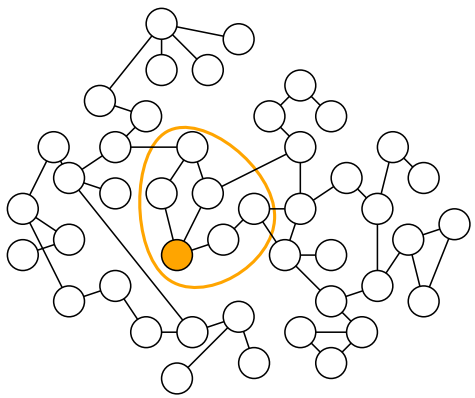


Decision Problems

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

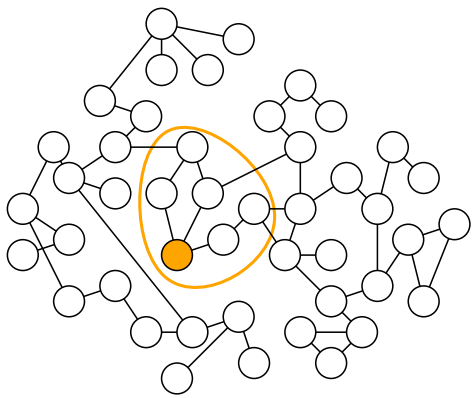
Decision Problems

- 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;



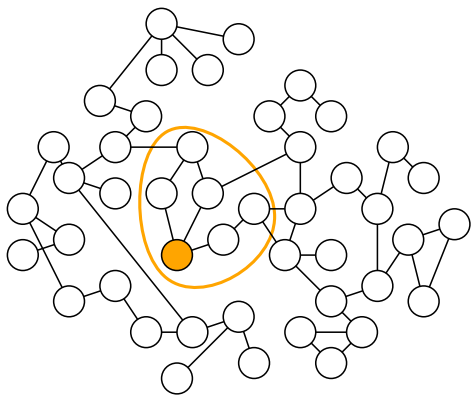
Decision Problems

- 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;



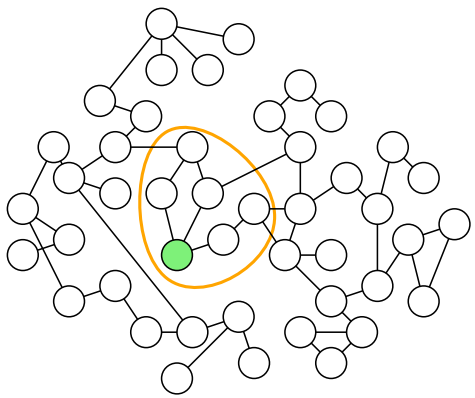
Decision Problems

- 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:



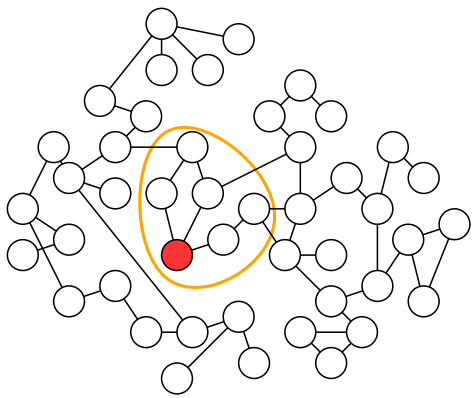
Decision Problems

- 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"



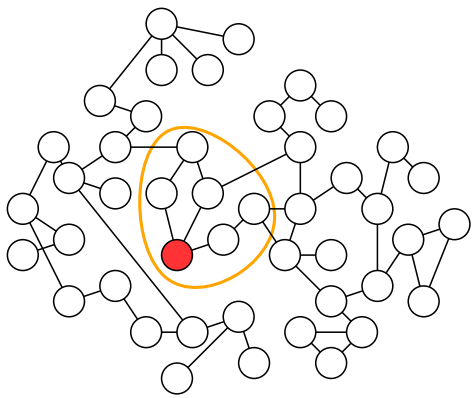
Decision Problems

- 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

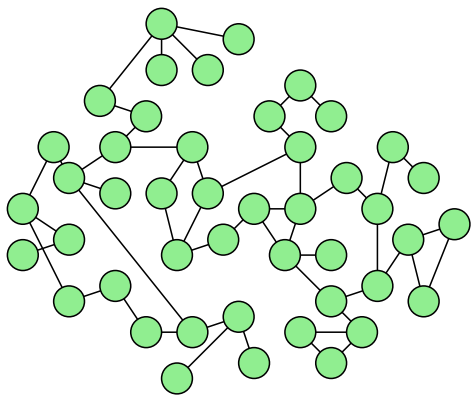
- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



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

Decision Problems

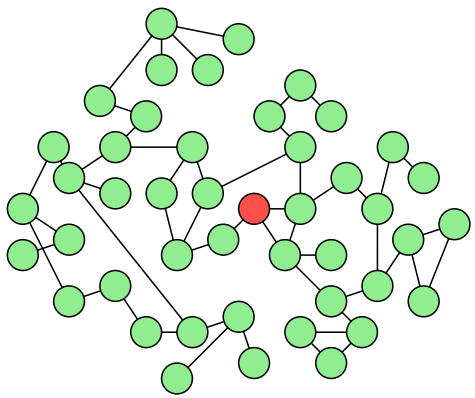
- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



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

Decision Problems

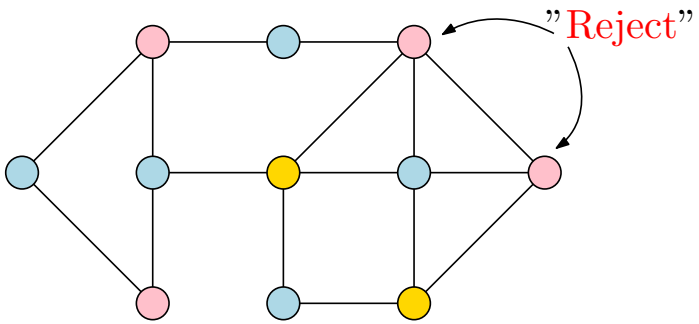
- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".



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

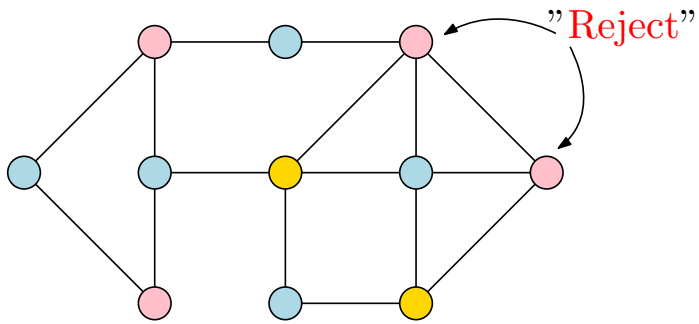
Example: Proper Coloring

- 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].

Distributed vs Centralized

Definition of LD

LD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that

$$(G, x) \in \mathcal{L} \iff \mathcal{A} \text{ accepts } (G, x).$$

Distributed vs Centralized

Definition of LD

LD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that

$$(G, x) \in \mathcal{L} \iff \mathcal{A} \text{ accepts } (G, x).$$

Definition of P

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

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

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**, unbounded 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**, unbounded 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**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;

Verification Problems

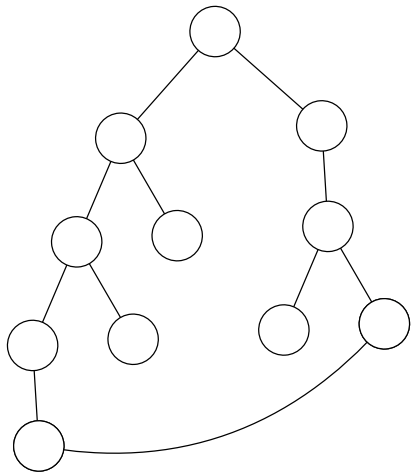
- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  -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

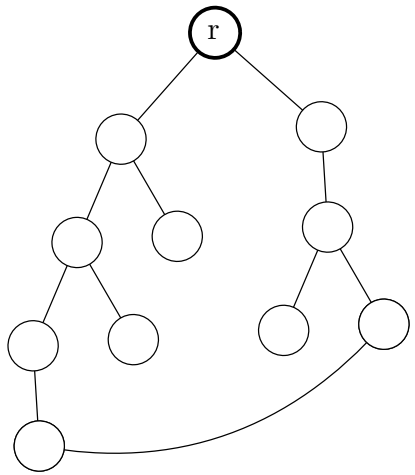
Example: is the given graph a tree?

- Not locally decidable, but locally verifiable.



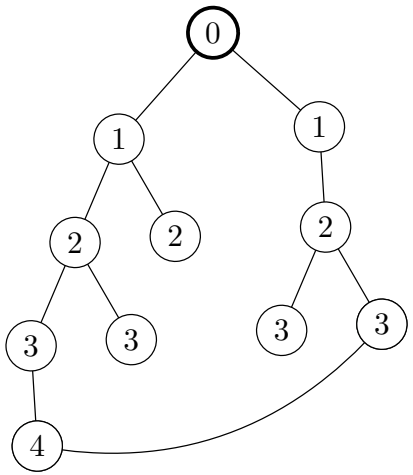
Example: is the given graph a tree?

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



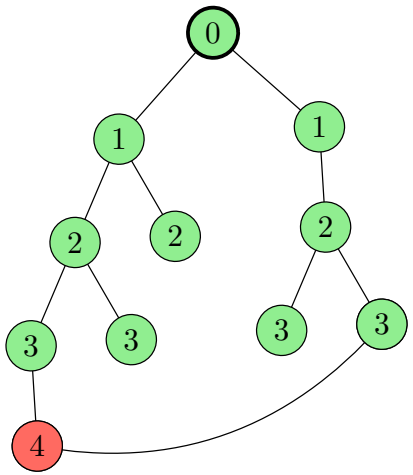
Example: is the given graph a tree?

- 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.



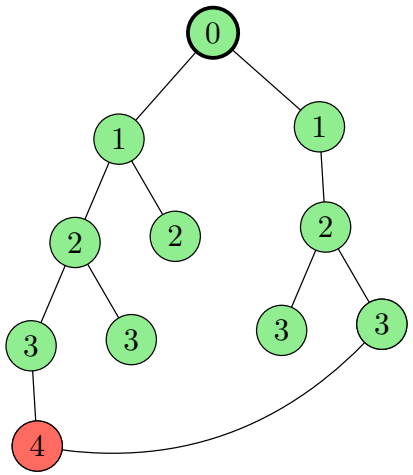
Example: is the given graph a tree?

- 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].

Distributed vs Centralized

Definition of NLD

NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that,

$$(G, x) \in \mathcal{L} \iff \exists c \text{ s.t. } \mathcal{A} \text{ accepts } (G, x) \text{ with c.}$$

Distributed vs Centralized

Definition of NLD

NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that,

$$(G, x) \in \mathcal{L} \iff \exists c \text{ s.t. } \mathcal{A} \text{ accepts } (G, x) \text{ with c.}$$

Definition of NP

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

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

Goal

- Build a hierarchy of complexity classes in the distributed setting.
- Distributed hierarchies in other setting:
  - [Reiter '14] in the context of automata;
  - [FFH '16] in a model inspired by the CONGEST one.

Complexity Classes

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

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).

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

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

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

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

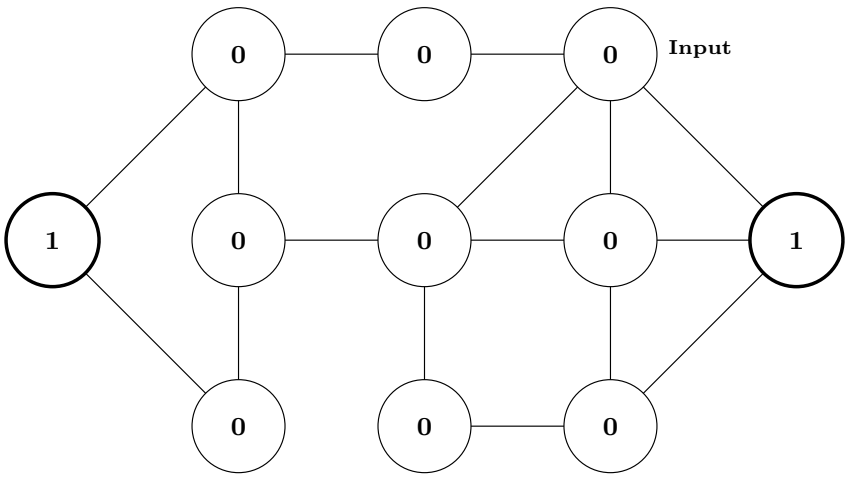
Π_2^{loc} Class

- Π_2 class: An input instance satisfies a certain property in Π_2 iff

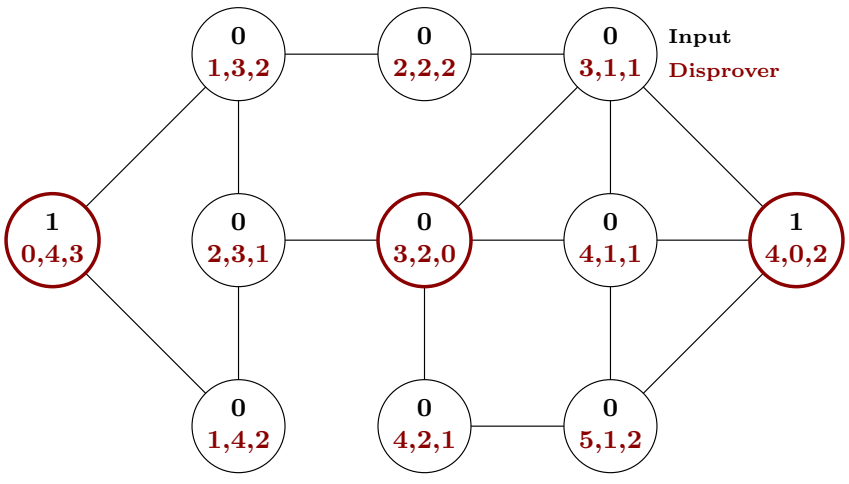
$$\forall c_1, \exists c_2, \text{ all nodes accept.}$$

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

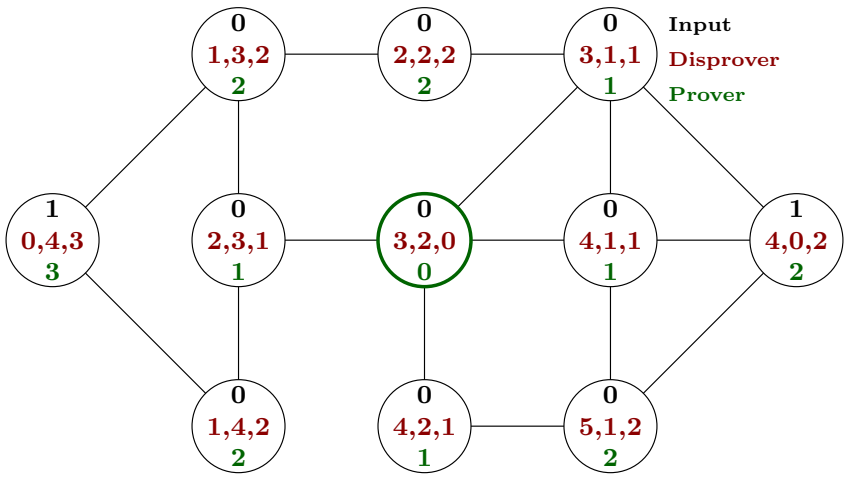
Exactly Two Selected



Exactly Two Selected



Exactly Two Selected



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}:

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

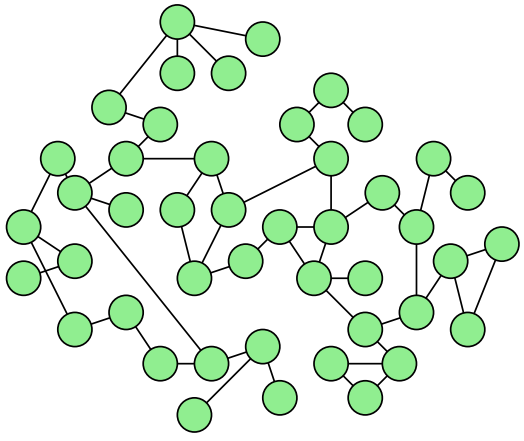
- LD:

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

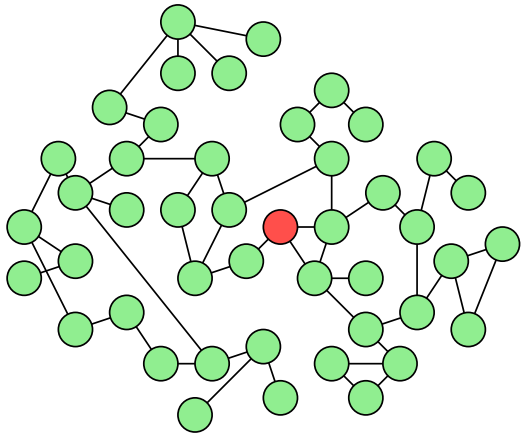
- Problems that can be solved only if a specific node knows (an upper bound of) the size of the network!

Complementary Classes

In a class:

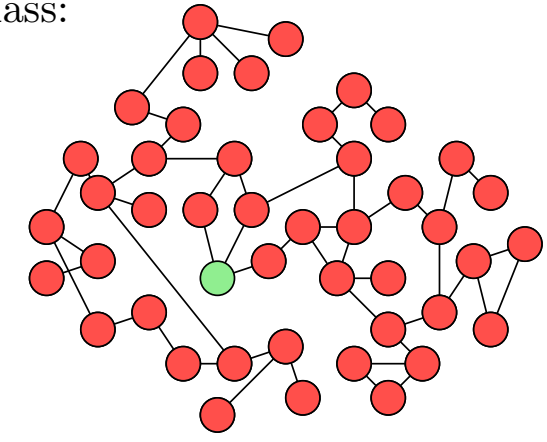


A globaly accepted input instance.

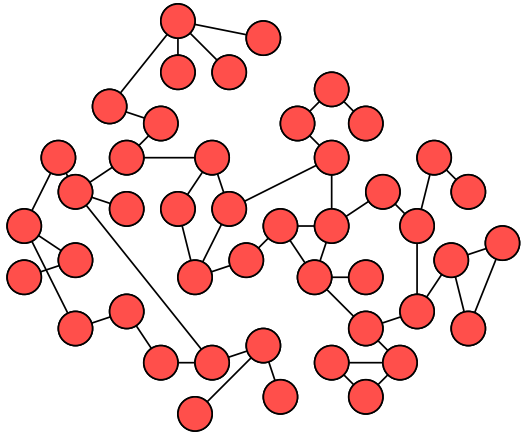


A globaly rejected input instance.

In a complementary class:

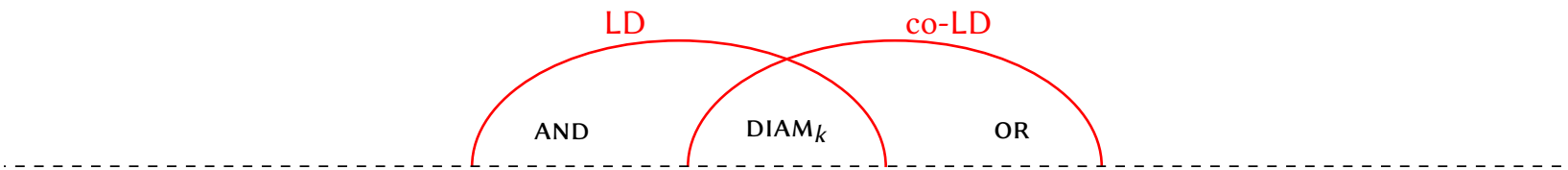


A globaly accepted input instance.

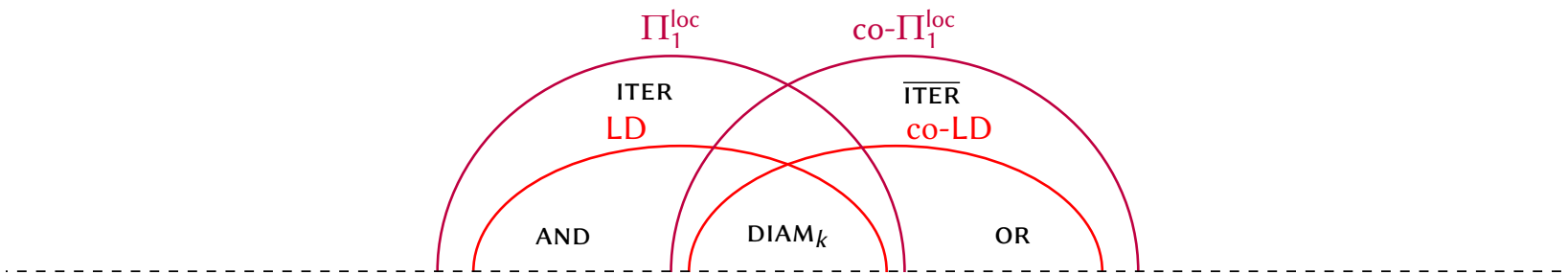


A globaly rejected input instance.

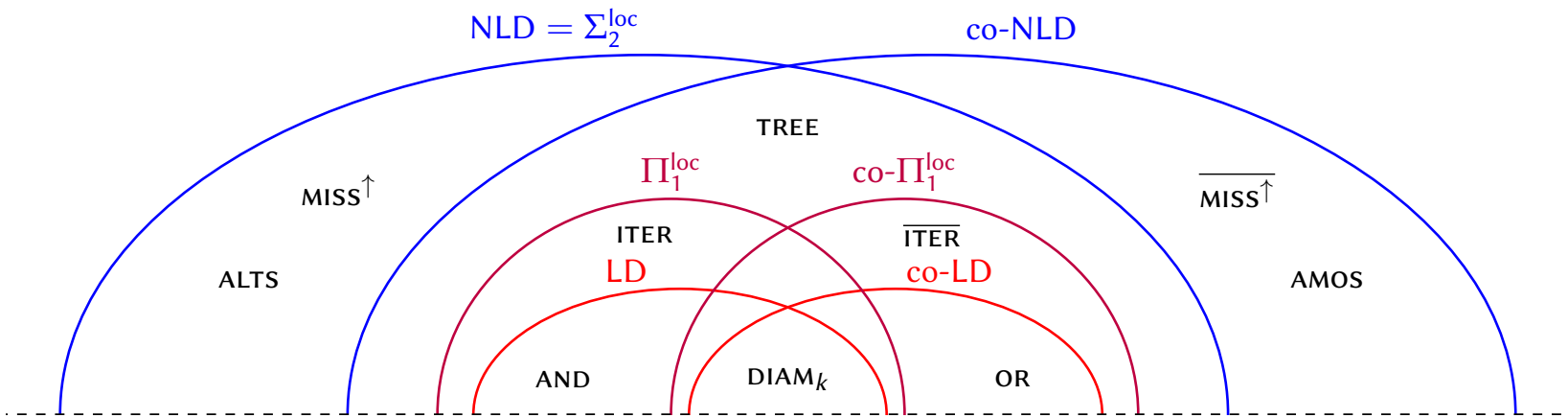
Local Hierarchy



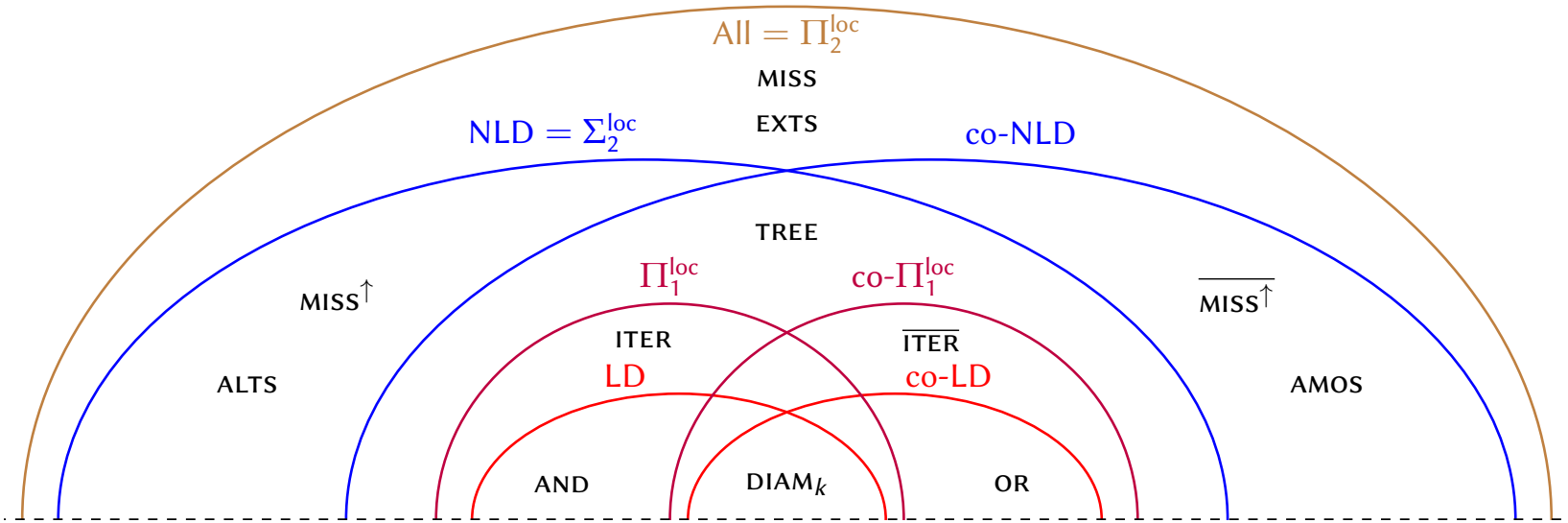
Local Hierarchy



Local Hierarchy



Local Hierarchy



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

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 [FFH '16]
  - we don't know if the hierarchy collapses;
  - there are no separating problems for Σ_2^{loc} and Σ_3^{loc} (neither for classes higher in the hierarchy).

Thank you!