Compact local certification

Laurent Feuilloley
CNRS and University of Lyon 1

Distributed Algorithms: past, present & future trends
June 13, 2023

Based on joint (and disjoint) work with Nicolas Bousquet, Pierre Fraigniaud, Pedro Montealegre, Théo Pierron, Eric Rémila, Ivan Rapaport, Ioan Todinca...

A typical theorem shape

In some distributed computing setting,

we can efficiently solve

the tasks of some type

in systems with nice enough properties.

Setting: Origins

The origin of our setting: the state model of self-stabilization.

Very roughly:

- A network represented as a graph
- Every node has a local memory (and a unique identifier)
- The computation is performed by local steps: read the memory of your neighbors, and update your memory.

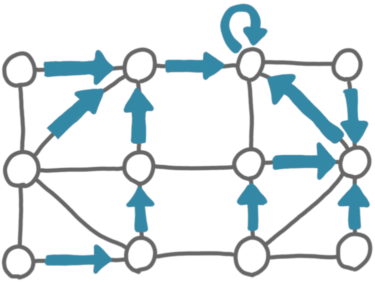
The main challenge is fault-tolerance: the algorithm should to converge to a correct configuration in spite of arbitrary changes to the memories (when given enough time to recover).

Setting: spanning tree example

Spanning tree:

Set of pointers such that:

- 1 out-pointer per node.
- No cycle.
- Connected.

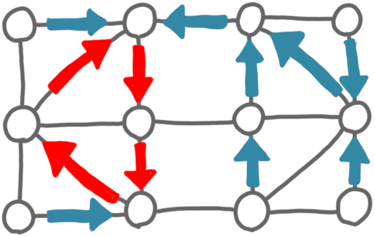


Setting: spanning tree example

Spanning tree:

Set of pointers such that:

- 1 out-pointer per node.
- No cycle.
- Connected.

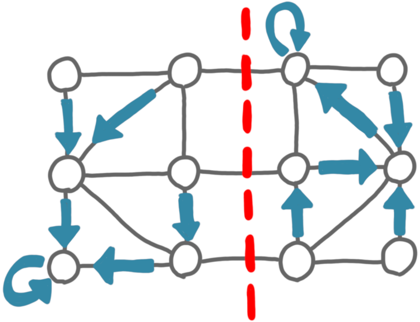


Setting: spanning tree example

Spanning tree:

Set of pointers such that:

- 1 out-pointer per node.
- No cycle.
- Connected.

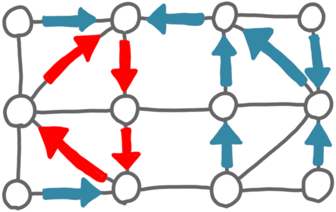


Setting: checking correctness

Our setting is “stop-or-continue self-stabilization”:

- If the configuration is correct, no update should be performed anymore.
- If the configuration is not correct, at least one node wants to take a step.

Not trivial: the output does not suffice.

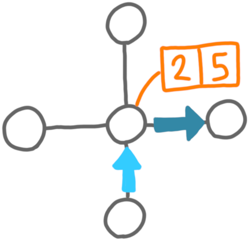


Setting: local certification

Key idea: keep additional info in memory (represented by a label).

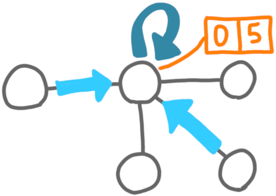
Label:

- Distance to the root.
- The ID of the root.



Sanity check:

- The distances locally make sense.
- Same "ID of the root".
- The root is the root.



Setting: local certification

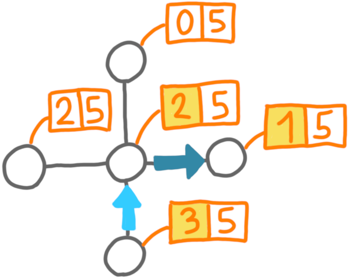
Key idea: keep additional info in memory (represented by a label).

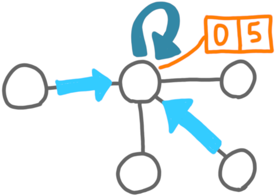
Label:

- Distance to the root.
- The ID of the root.

Sanity check:

- The distances locally make sense.
- Same "ID of the root".
- The root is the root.





Setting: local certification

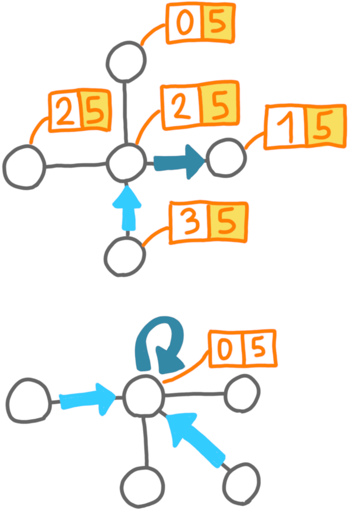
Key idea: keep additional info in memory (represented by a label).

Label:

- Distance to the root.
- The ID of the root.

Sanity check:

- The distances locally make sense.
- Same "ID of the root".
- The root is the root.



Setting: local certification

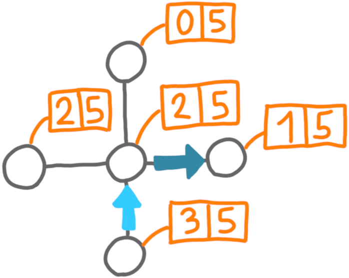
<u>Key idea:</u> keep additional info in memory (represented by a label).

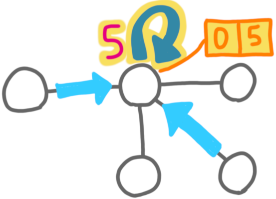
Label:

- Distance to the root.
- The ID of the root.

Sanity check:

- The distances locally make sense.
- Same "ID of the root".
- The root is the root.



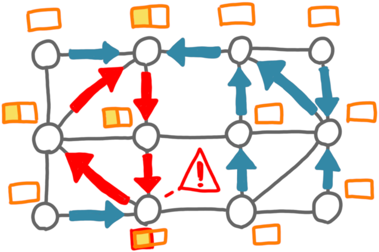


Setting: local certification

We have the two following properties → it is a local certification.

Yes For every **correct** configuration
there exists a label assignment such that
the sanity check **accepts at every node**.

No For every **incorrect** configuration,
for all label assignments,
the sanity check **rejects in at least one node.**



A typical theorem shape

In **local certification**,

we can efficiently solve

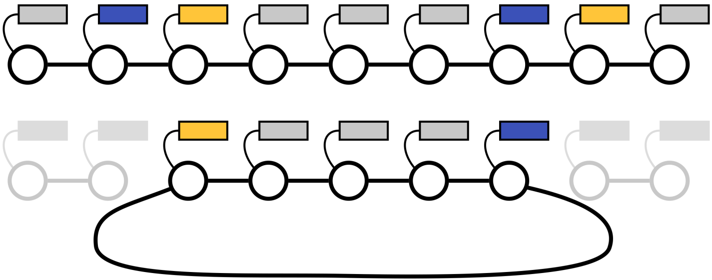
the tasks of some type

in systems with nice enough properties.

Efficiency: certificate size

The time/locality is already restricted: one communication round with the direct neighbors. $\rightarrow$ Focus on space.

For spanning tree certification: labels of size $O(\log n)$ per node. And this is optimal.

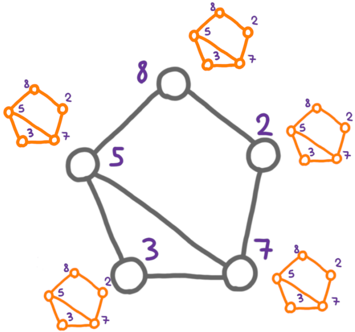


Efficiency: compact certification

Compact certification: certificates of size $O((\text{poly}) \log n)$.

Can be achieved for: planarity, tree problems, approximation of combinatorial problems.

But not for everything. Sometimes we need to use up to $\Theta(n^2)$



A typical theorem shape

In **local certification**,

we can **certify compactly**

the tasks of some type

in systems with nice enough properties.

Restricting tasks: logic on graphs

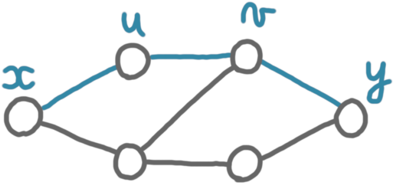
Ways to classify tasks: CSPs/LPs, using substructures...

Here → logic on graphs.

First-order (FO) logic on graphs: basically classic formula shape but with a predicate for edges (denoted $\sim$ in this talk).

<u>Example:</u> diameter at most 3:

$$\forall x, \forall y, \exists u, \exists v, (x \sim u \land u \sim v \land v \sim y)$$



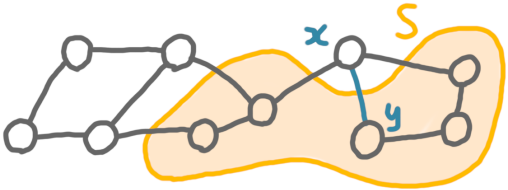
Restricting tasks: MSO

FO is too weak for many interesting properties. :(

MSO is a popular extension: can quantify on sets of vertices/edges.

Example: connectivity

$$\forall S \subseteq V, (S \neq \emptyset, S \neq V) \Rightarrow \exists x \notin S, \exists y \in S, x \sim y.$$



A typical theorem shape

In **local certification**,

we can **certify compactly**

all MSO properties

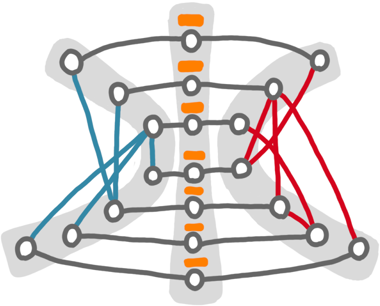
in systems <u>with nice enough properties.</u>

Restricting the system: necessary?

We do have to restrict the graph class.

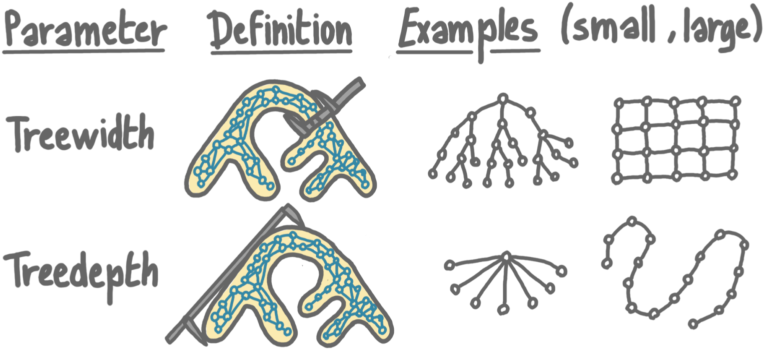
$\rightarrow$ "Diameter $\leq k$" has an FO formula and requires $\tilde{\Omega}(n)$ bits in general graphs.

The lower bound needs arbitrarily complicated structure.



Restricting the system: bounding graph parameters

Inspiration → Courcelle's theorem: Any MSO property can be checked in linear time in graphs of bounded treewidth.



Meta-theorem(s)

In **local certification**,

we can **certify compactly**

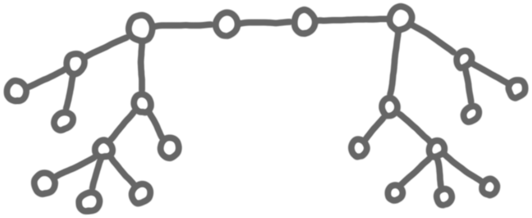
all MSO properties

graphs of bounded **treedepth/treewidth**.

Restricting the system only?

At least some task restriction is necessary.

$\rightarrow$ "Symmetry" requires $\Omega(n)$ bits even on trees.



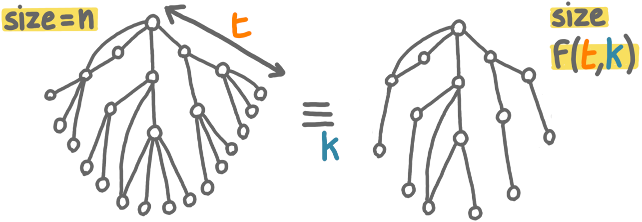
No MSO formula can express symmetry (we "need" to quantify on a mapping function, and not simply on a set).

The proof in a nutshell

Step 1: Certify the underlying structure that captures the parameter (embedding in extended tree / tree decomposition)

The proof in a nutshell

Step 2 (for treedepth only): Provide and certify a kernel.



Step 3 (for treedepth only): Let the nodes check the MSO property on this model.

Conclusion

In **local certification**,

we can **certify compactly**

all MSO properties

graphs of bounded **treedepth/treewidth.**

Open questions:

- More general graph parameters?
- Minor-free graphs?
- Other meta-theorems approaches?
- Consequences in self-stabilization?