1 Definitions

1.1 Notation

We will use $\delta(v)$ to refer to the set of edges adjacent to v. We identify half-edges in a graph $G = (V, E)$ with a pair (e, v) where $e \in \delta(v)$, referring to the half of e adjacent to v.

We use $\{\{\star\}\}$ to refer to multisets, and the following notation for multisubsets of a set A:

- $A^{\{\{k\}\}}$ refers to the set of all multisubsets of A of size $k \in \mathbb{N}$,
- $A^{\{\{\mathbb{N}\}\}}$ refers to the set of all finite multisubsets of A,
- $A^{\{\{\star\}\}}$ refers to the set of all multisubsets of A.

1.2 Common and uncommon assumptions

T-hop neighbourhoods: we define a *T-hop centered neighbourhood* as a *centered graph of radius T* using the definitions from page 4 of [NS93]. Note that, in the case of subgraphs, the T-hop neighbourhood of v may be different from the subgraph induced by all nodes that have distance at most T from v.

Local model: we work in the *deterministic* LOCAL model with input labels. No assumptions are made about the number of input or output labels. We assume the nodes are aware of the exact number n of nodes in the input graph, but not of any information about the maximum degree Δ. Because of this information, we equivalently describe a LOCAL algorithm as a $T(n)$-round communication algorithm with unbounded message size, or as a *possibly uncomputable* function from $T(n)$-hop centered neighbourhoods to output labels. Additionally, we assume there is a finite integer c such that every node is assigned a unique ID in the set $\{1, \ldots, n^c\}$, but the nodes are not aware of the value of c.

Solvability: we say that a problem is *weakly unsolvable* if there are finitely many unsolvable instances for it. We will treat weakly unsolvable problems as solvable problems by adding one output label U and requiring the problem to output U on all nodes if the instance is unsolvable; this requires only constant time, so it does not affect asymptotic complexity. We call the maximum diameter of an unsolvable instance the *solvability horizon* of the weakly unsolvable problem. We say that a problem is *strongly unsolvable* if there are infinitely many unsolvable instances for it.

LCL problems: we define LCL problems as tuples $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, r, \mathcal{C})$ where:

- Σ_{in} is a set of input labels,
- Σ_{out} is a set of output labels,
- r is a finite integer, called the *radius* of Π, and
- $\mathcal{C}$ is a *finite* set of r-hop centered neighbourhoods, where each node is labelled by a pair in $\Sigma_{\text{in}} \times \Sigma_{\text{out}}$.

Though we haven't explicitly stated this in the definition, WLOG we can and will also assume that Σ_{in} is finite (else the problem would be strongly unsolvable) and that Σ_{out} is finite (any labels that don't appear in $\mathcal{C}$ will not appear in any valid solution and can be ignored).

1.3 New-ish definitions

Hanging trees: *connected* tree graphs containing a finite number of incomplete half-edges, called *hooks*. Specifically, we will call hanging trees with one hook *ornaments* and hanging trees with two hooks *tinsels*.

Class: given an ornament T and a node-edge checkable problem Π, we define the *class* of T to be the set $\mathfrak{Cl}(T) \subseteq \Sigma_{\text{out}}$ containing exactly the labellings of the hook that can be completed to a valid labelling for T according to Π. Up to the changed role of nodes and edges, this is the *class* definition from Section 3 of [CP17].

1.4 Actually new definitions

Finitely Represented Configuration: given an alphabet of symbols Σ, we call a *finitely represented configuration* of Σ a pair of a *finite* multisubset of Σ (the *requirement*) and a *finite* subset of Σ (the *filler*). We denote the set of finitely represented configurations in Σ as

$$\Sigma^{FSL} := \Sigma^{\{\{\mathbb{N}\}\}} \times [\Sigma]^{<\omega}$$

Extended Finite State Labelling problems: we define Extended FSL problems as tuples $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, f, \mathcal{N}, \mathcal{E})$ where:

- Σ_{in} is a *finite* set of input labels,
- Σ_{out} is a *finite* set of output labels,
- $f : \mathbb{N} \to 2^{\Sigma_{\text{in}}}$ is a *computable* function describing which input labels are allowed to appear for a node of degree $d \in \mathbb{N}$ which is surjective on some cover of Σ_{in} (in other words, each label in Σ_{in} appears in some set in $f(\mathbb{N})$),
- $\mathcal{N} : \Sigma_{\text{in}} \to 2^{(\Sigma_{\text{out}})^{FSL}}$ is a function assigning to each input label a set of *finitely represented configurations* of Σ_{out}, and
- $\mathcal{E}$ is a set of pairs of elements of Σ_{out}.

An *instance* for Π is a graph $G = (V, E)$ together with a labelling function $l : V \to \Sigma_{\text{in}}$ such that $l(v) \in f(\deg(v))$, that is, the labelling is coherent with the "allowed" labels for each degree. A *solution* for this instance is a labelling s of the half-edges of G such that:

- $\forall v \in V$, let $S(v) := \{\{s(e, v) \,|\, e \in \delta(v)\}\}$; then there exists $(R, F) \in \mathcal{N}(l(v))$ such that $R \subseteq S(v)$ and $(S(v) \smallsetminus R) \subseteq F^{\{\{\star\}\}}$, that is to say, every symbol in the requirement appears exactly once in $S(v)$, and every other symbol of $S(v)$ is a filler symbol, and
- $\forall e = \{u, v\} \in E$ we have $\{s(e, u), s(e, v)\} \in \mathcal{E}$.

Finite State Labelling problems: we define FSL problems as tuples $\Pi = (\Sigma_{\text{out}}, \mathcal{N}, \mathcal{E})$ where:

- Σ_{out} is a *finite* set of output labels,
- $\mathcal{N} \subseteq 2^{(\Sigma_{\text{out}})^{FSL}}$ is a *finite* set of finitely represented configurations of Σ_{out},
- $\mathcal{E}$ is a set of pairs of elements of Σ_{out}.

An *instance* for Π is any graph; a solution for this instance is a labelling s of the half-edges of G such that:

- $\forall v \in V$, let $S(v) := \{\{s(e, v) \,|\, e \in \delta(v)\}\}$; then there exists $(R, F) \in \mathcal{N}$ such that $R \subseteq S(v)$ and $(S(v) \smallsetminus R) \subseteq F^{\{\{\star\}\}}$, that is to say, every symbol in the requirement appears exactly once in $S(v)$, and every other symbol of $S(v)$ is a filler symbol, and

- $\forall e = \{u, v\} \in E$ we have $\{\{s(e, u), s(e, v)\}\} \in \mathcal{E}$.

Minimum degree of a label: we define the *minimum degree* of an input label $\chi \in \Sigma_{\text{in}}$ as $\deg_{\min}(\chi) := \min \{d \in \mathbb{N} \mid \chi \in f(d)\}$ (the minimum degree for which that input label is allowed). This set is never empty. Generally, we can assume that $\forall (R, F) \in \mathcal{N}(\chi)$ we have $|R| = \deg_{\min}(\chi)$, since a configuration with a longer requirement would be unattainable by a finite number of degrees (which can be encoded by putting them in separate input classes) and any configuration with a shorter requirement implicitly requires using a number of filler symbols, which can be encoded by adding all combinations of filler symbols to the requirements multiset.

Width: we define the *width* of a EFSL problem Π as the maximum length of one of its requirements: accounting for hidden requirements encoded in the degree, we get

$$\max \left\{ \max_{\chi \in \Sigma_{\text{in}}} \deg_{\min}(\chi), \max \{|R| \mid \exists \chi \in \Sigma_{\text{in}}, \exists F \subset \Sigma_{\text{out}} : (R, F) \in \mathcal{N}(\chi)\} \right\}$$

For FSL problems we define it as $\max \{|R| \mid \exists F (\subset \Sigma_{\text{out}} : R, F) \in \mathcal{N}\}$.

Height: we define the *height* of an EFSL problem as $h(\Pi) := \max \{|\mathcal{N}(\chi)| \mid \chi \in \Sigma_{\text{in}}\}$ (the maximum number of configurations any node can be in). For FSL problems we define it as $|\mathcal{N}|$.

Restricted (E)FSL: we define the *restriction* of an (E)FSL the *node-edge checkable* problem with input labels obtained by replacing every finitely represented configuration (R, F) with the configurations $\{R \cup \{\{x\}\} \mid x \in F\}$.

2 List of results

1. Extended FSL = FSL (up to changing the number of labels) input/degree info in the structure and output labels
2. Node-edge checkable problems $\subseteq$ FSL
3. Weak and strong solvability are decidable on FSL

3 List of possible results

1. If a FSL problem is known to be $\Omega(\log n)$ on trees, its complexity on trees is decidable and an efficient algorithm can be found. (This holds because of Gustav's proof which should be stronger, it's in here because I still haven't written out all the details of the FSL based one)
2. (Belief I haven't been able to fully prove) For each FSL problem Π there is a value d that only depends on the problem description (specifically, number of labels, height and width) such that there is a worst-case family of graphs for Π of maximum degree d. My theory is either width+1 or width·height (also possible: width+height, using one of each filler). Latter is required for solvability.
3. (Investigating) A form of the Round Elimination procedure applies to FSL problems.

4 Proofs

4.1 Equivalence of EFSL and FSL

In this section, we show that the EFSL and FSL can encode the same problems.

Theorem 4.1. *For each FSL problem* Π *that has a LOCAL algorithm* A *with complexity* $T(n)$*, there is an EFSL problem* Π' *which has a LOCAL algorithm* A' *with complexity* $O(T(n))$*.*

Proof. Let $\Pi = (\Sigma_{\text{out}}, \mathcal{N}, \mathcal{E})$. We define $\Pi' = (\Sigma'_{\text{in}}, \Sigma'_{\text{out}}, f', \mathcal{N}', \mathcal{E}')$ as follows:

- $\Sigma'_{\text{in}} = \{\star\}$,
- $\Sigma'_{\text{out}} = \Sigma_{\text{out}}$,
- f' is the constant function $f(n) = \{\star\}$,
- $\mathcal{N}'(\star) = \mathcal{N}$,
- $\mathcal{E}' = \mathcal{E}$.

It is trivial to observe that this is the same problem, and can be solved with the same complexity as Π. □

Theorem 4.2. *For each EFSL problem* Π *that has a LOCAL algorithm* A *with complexity* $T(n)$*, there is an FSL problem* Π' *which has a LOCAL algorithm* A' *with complexity* $O(T(n))$*.*

This is the more complex of the two proofs, so we provide an overview before getting into the technical details: we encode the k-th input label as a line of degree 3 nodes of length k, and add leaves as needed to make sure every node of the original graph is of degree ≥ 4. This at most multiplies the number of nodes by $3|\Sigma_{\text{in}}|$, and every node can in finite time compute its own input label by looking at its $(|\Sigma_{\text{in}}| + 1)$-hop neighbourhood. To make sure

Proof. Let $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, f, \mathcal{N}, \mathcal{E})$. Up to a bijection we can identify the set Σ_{in} with the set $\{1, \ldots, k\}$. We define $\Pi' = (\Sigma'_{\text{out}}, \mathcal{N}', \mathcal{E}')$ as:

- $\Sigma'_{\text{out}} = \Sigma_{\text{out}} \cup \Sigma_I$, where $\Sigma_I := \{(I, n) \mid n \in \Sigma_{\text{in}}\} \cup \{(I, L), (I, Err), Err\}$,
- $\mathcal{E}' = \mathcal{E} \cup \{\{\{X, X\}\} \mid X \in \Sigma_I\}$ (every edge containing the new input labels is symmetric).

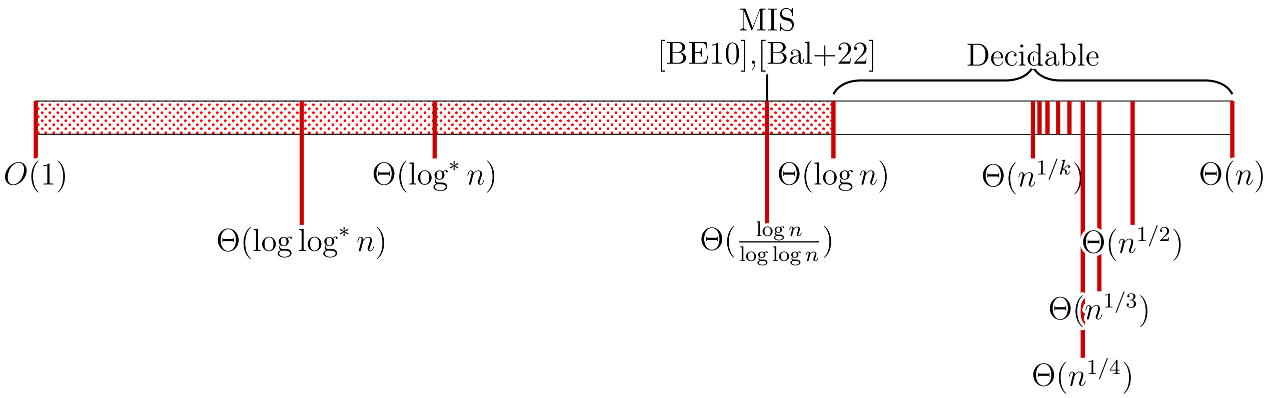
$\mathcal{N}'$ has a more complex formula, so we provide an algorithm to derive it from f and $\mathcal{N}$ instead:

1.

□

5 What do we know about the complexity landscape?

5.1 Trees



5.2 General graphs

Bibliography

[NS93] Moni Naor and Larry Stockmeyer. "What can be computed locally?" In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*. STOC '93. New York, NY, USA: Association for Computing Machinery, June 1, 1993, pp. 184–193. ISBN: 978-0-89791-591-5. DOI: `10.1145/167088.167149`. URL: `https://www.wisdom.weizmann.ac.il/~naor/PAPERS/lcl.pdf` (visited on 08/11/2025).

[BE10] Leonid Barenboim and Michael Elkin. "Sublogarithmic Distributed MIS Algorithm for Sparse Graphs Using Nash-Williams Decomposition". In: *Distributed Computing* 22.5 (Aug. 1, 2010), pp. 363–379. ISSN: 1432-0452. DOI: `10.1007/s00446-009-0088-2`. URL: `https://doi.org/10.1007/s00446-009-0088-2` (visited on 09/16/2025).

[CP17] Yi-Jun Chang and Seth Pettie. "A Time Hierarchy Theorem for the LOCAL Model". In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). Berkeley, CA, Oct. 2017, pp. 156–167. ISBN: 978-1-5386-3464-6. DOI: `10.1109/FOCS.2017.23`. URL: `http://ieeexplore.ieee.org/document/8104055/` (visited on 08/11/2025).

[Bal+22] Alkida Balliu et al. "Distributed Δ-Coloring Plays Hide-and-Seek". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2022. New York, NY, USA: Association for Computing Machinery, June 10, 2022, pp. 464–477. ISBN: 978-1-4503-9264-8. DOI: `10.1145/3519935.3520027`. URL: `https://doi.org/10.1145/3519935.3520027` (visited on 09/16/2025).