Applying Category Theory to the Study of Biregular LCL-Problems and Round Elimination

Selim Virtanen

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 14.11.2023

Supervisor

Prof. Camilla Hollanti

Advisors

Prof. Jukka Suomela

Prof. Jara Uitto

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Selim Virtanen

Title Applying Category Theory to the Study of Biregular LCL-Problems and Round Elimination

Degree programme Science and Technology

Major Mathematics **Code of major** SCI3054

Supervisor Prof. Camilla Hollanti

Advisors Prof. Jukka Suomela, Prof. Jara Uitto

Date 14.11.2023 **Number of pages** 82+2 **Language** English

Abstract

Recently, a lot of progress has been made in the study of distributed computational complexity in the class of so-called locally checkable labelling problems (LCLs). In particular, there is a new proof technique, round elimination, which resembles a function, and finding its periodic points in a further restricted biregular formalism has been fruitful for showing lower bounds. In this thesis I provide a rigorous definition for biregular LCL-problems based on category theory, which gives powerful tools for studying 0-round reductions and equivalence of problems.

I define the category $\mathrm{LCL}\,(d, \delta)$ with (d, δ)-biregular LCL-problems as objects and local correctness preserving (LCP) maps as morphisms. The use of categories gives access to terminology and definitions from category theory. LCP-maps imply 0-round reductions, and conversely I show that once impossible labels are removed, they capture all 0-round reductions that can be found with reasonable assumptions.

In this framework round elimination becomes a map between categories, and I prove that it preserves the existence of LCP-maps. Thus it can be defined over equivalence classes, generalising it as a proof technique and unifying existing definitions. My main result is that with reasonable assumptions the longest period of a periodic point of round elimination is 2. This implies that periodicity of a problem Π can be checked automatically by determining the existence of LCP-maps $\mathrm{Re}^2(\Pi) \to \Pi$, for which I provide an implementable algorithm.

Finally, the new tools make it possible to define a minimality condition for representatives of equivalence classes, which can be checked without referencing other problems. I prove that it is equivalent to an intuitive condition, and show how to minimise representatives.

This work provides a new perspective for the further study of biregular LCL-problems and improvements for the automated analysis tools. These in turn enhance our understanding of distributed computational complexity, which can help us develop more efficient algorithms for distributed systems.

Keywords applied category theory, biregular LCL-problems, distributed algorithms, distributed graph problems, round elimination

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Selim Virtanen

Työn nimi Katgoriateorian soveltaminen kaksisäännöllisten LCL-ongelmien ja kierroseliminaation tutkimiseen

Koulutusohjelma Matematiikka ja systeemianalyysi

Pääaine Matematiikka **Pääaineen koodi** SCI3054

Työn valvoja Prof. Camilla Hollanti

Työn ohjaaja Prof. Jukka Suomela

Päivämäärä 14.11.2023 **Sivumäärä** 82+2 **Kieli** Englanti

Tiivistelmä

Viime vuosina hajautettujen algoritmien laskennallisen vaativuuden tutkimuksessa on tehty harppauksia tarkastelemalla paikallisesti tarkistettavia merkitsemisongelmia (LCL-ongelmia). Erityisesti uuden funktiota muistuttavan todistustustekniikan, kierroseliminaation, jaksollisten pisteiden etsiminen on ollut antoisaa alarajojen todistamisessa. Tässä työssä annan kaksisäännöllisille LCL-ongelmille kategoriateoriaan pohjautuvan tarkan määritelmän, joka tarjoaa tehokkaita työkaluja nollan kierroksen reduktioiden ja ongelmien ekvivalenssin tutkimiseen.

Määrittelen kategorian LCL (d, δ), jonka objekteina ovat (d, δ)-kaksisäännölliset LCL-ongelmat ja morfismeina paikallisen paikkansapitävyyden säilyttävät kuvaukset (LCP-kuvaukset). Näytän, että LCP-kuvaukset vastaavat kohtuullisilla oletuksilla nollan kierroksen reduktioita. Kategorioiden käyttö mahdollistaa kategoriateorian määritelmien ja sanaston hyödyntämisen.

Kierroseliminaatio muuttuu tässä kontekstissa kuvaukseksi kategorioiden välillä, ja osoitan sen säilyttävän LCP-kuvausten olemassaolon. Niinpä kierroseliminaatio voidaan määritellä ekvivalenssiluokkien yli, mikä yleistää sitä todistustekniikkana yhdistäen sen eri määritelmiä. Työn päätulos on, että kohtuullisilla oletuksilla kierroseliminaation jaksollisten pisteiden pisin mahdollinen jakso on 2. Tästä seuraa, että ongelman Π jaksollisuus voidaan tarkistaa automaattisesti selvittämällä, onko LCP-kuvauksia $\mathrm{Re}^2(\Pi) \to \Pi$ olemassa, mihin annan algoritmin.

Lopuksi hyödynnän uusia työkaluja määrittelemällä ekvivalenssiluokkien edustajille minimaalisuusehdon, joka voidaan tarkistaa viittaamatta muihin ongelmiin. Todistan sen olevan yhtäpitävä luontevan ehdon kanssa, sekä lisäksi näytän, miten edustajia voi minimoida.

Työ tarjoaa uuden näkökulman LCL-ongelmien tutkimiseen ja parannuksia siihen liittyviin automaattityökaluihin. Nämä vuorostaan lisäävät ymmärrystä hajautetusta aikavaativuudesta, mikä voi auttaa tehokkaampien algoritmien kehityksessä hajautetuille järjestelmille.

Avainsanat sovellettu kategoriateoria, kaksisäännölliset LCL-ongelmat, hajautetut algoritmit, hajautetut verkko-ongelmat, kierroseliminaatio

Contents

1 Introduction

1.1 Motivation

Computer networks, where nodes operate independently and have a limited number of neighbours, have applications ranging from our communication infrastructure to robot swarms. The study of *distributed algorithms* seeks to answer what can be computed in these networks efficiently [1]. In recent years a lot of progress has been made by restricting to *locally checkable labelling problems* (LCLs), which are a distributed analogue to NP-problems [2] introduced by Naor and Stockmeyer in 1995 [3]. In particular, a big leap forward was the introduction of the *round elimination* proof technique by Brandt in 2019 [4], with which under certain assumptions a general LCL-problem solvable in t rounds can be transformed into another one solvable in $t-1$ rounds. If a repeated application of round elimination leads back to the original problem, we get a contradiction that it is solvable exactly in both t and some $t-k$ rounds, which we can turn to lower bound complexity results.

When we are interested in lower bounds, we can simplify the situation by considering the LCL-problems in the more restricted *biregular formalism*, where problems can be concisely described through *active and passive configurations* and we have software tools for computing round elimination steps [5]. This is, because for each LCL-problem solvable in t rounds we can derive $(d, 2)$-biregular LCL-problems solvable in $\mathcal{O}(t)$ rounds by using simulation arguments and restricting the underlying graph family[1]. For many problems we can also find more insightful biregular representations [1]. Furthermore, we can use methods such as randomisation and gap theorems or order-invariant algorithms to lift results obtained with round elimination in the underlying PN-model to the LOCAL model [4].

1.2 My Contribution

My initial goal was to study round elimination over biregular formalism as a function between problem spaces, and in particular its periodic points. However, a problem arises that in order to get periodic points we must consider reductions between problems, which is not a big issue when we use round elimination as a proof technique for a single problem, but in order to get a well behaving function we need to define it over equivalence classes. When we start to investigate 0-round reductions as solution-preserving functions, a mathematical structure called *category* starts appearing. This gives us a new perspective on the subject and allows us to harness the powerful language of category theory to describe it.

[1] This is only meaningful if the underlying graph family contains d-regular graphs, and ideally they need to capture the worst-case running times. I will show the method in Section 2

In this thesis we study how category theory can be applied to the study of LCL-problems and round elimination. We start by giving a short introduction to category theory and the background of distributed algorithms, biregular LCL-problems and round elimination in Section 2. Then in Section 3 we construct categories of biregular LCL-problems by representing them in a suitable form and defining local correctness preserving (LCP) maps between them. This immediately bears fruit as we can import the universal definitions of limits and colimits and find some of them in our category, and it is easy to define equivalence of problems. Our choice of LCP-maps is justified in Section 4, where we discuss how well they capture 0-round reductions, and in the general case issues are only caused by impossible labels that can safely be removed, which also simplifies things for round elimination.

In Section 5 we define round elimination as a map between categories and study its properties, including that it induces a functor which shows that round elimination respects equivalence, allowing us to unify its different definitions. Furthermore, only periodic points of periods 1 and 2 have previously been discovered [6], and my primary result is that 2 is indeed the longest possible period, given reasonable definitions. Its main implication is that checking periodicity reduces to performing round elimination twice and checking whether an LCP-map back to the original problem exists. To that end, we discuss what simplifying assumptions can be made about LCP-maps, and present an algorithm for checking if one exists, in Section 6. Finally, in Section 7 we study minimality in equivalence classes in search of a canonical representative.

To my understanding, no previous research of applying category theory to LCL-problems exists.

2 Background

2.1 Category Theory

Category theory is the study of mathematical structures called *categories*. A category consists of *objects*, and *morphisms* between those objects. A common example is the category of sets Set, which has sets as objects and functions as morphisms. A key feature of categories is that the morphisms can be composed, like one would functions, and and a fitting alternative name for category theory could indeed be the abstract theory of composition.

In this section we introduce some basic category theory concepts, amalgamated from textbooks [7, 8] and common knowledge, in a way that will be useful for us later.

Definition 2.1.1. A **category** $\mathcal{C}$ consists of a collection of **objects**, which we by abuse of notation denote $\mathcal{C}$, and for each pair of objects $A, B \in \mathcal{C}$ a collection of **morphisms** denoted by $\mathcal{C}(A, B)$, or $\mathrm{Mor}\,(A, B)$ if the underlying category is clear from context. Furthermore the **domain** A and **codomain** B of a morphism $f \in \mathcal{C}(A, B)$ can be denoted $f\colon A \to B$. Morphisms $f\colon A \to B, g\colon B \to C$ can be **composed** to $g \circ f = gf\colon A \to C$, so that the following axioms hold:

- For each object $A \in \mathcal{C}$ there is an **identity morphism** $1_A\colon A \to A$, so that for all objects $B \in \mathcal{C}$ and morphisms $f\colon A \to B$, $g\colon B \to A$ we have $f \circ 1_A = f$ and $1_A \circ g = g$.
- Composition is **associative**, i.e. for all morphisms $f\colon A \to B$, $g\colon B \to C$, $h\colon C \to D$ we have $h \circ (g \circ f) = (h \circ g) \circ f = hgf$.

Note that we use the ambiguous word 'collection' in order to skirt the problem that objects or morphisms do not necessarily fit in sets. For example, if we try to form a set of all sets, set-theoretical paradoxes arise, but the objects in the category Set are all sets. As this is not relevant to our work, we will just assume that good enough extensions to set theory exist, so that such collections make sense. However, we make the following distinction, that brings us slightly closer to safe territory.

Definition 2.1.2. A **locally small** category is a category $\mathcal{C}$, where for each pair of objects $A, B \in \mathcal{C}$ the morphisms $\mathcal{C}(A, B)$ form a set.

Example 2.1.3. Many familiar structures form locally small categories, where the objects are structured sets and the morphisms structure-preserving maps:

i) Set, where the objects are sets and the morphisms are functions,

ii) Group, where the objects are groups and the morphisms are group homomorphisms,

iii) Ring, where the objects are unital rings and the morphisms are ring homomorphisms,

iv) $\mathrm{Vect}_{\mathbb{K}}$, where the objects are vector spaces over a field $\mathbb{K}$ and the morphisms are $\mathbb{K}$-linear maps,

v) Top, where the objects are topological spaces and the morphisms are continuous functions,

vi) Graph, where the objects are graphs and the morphisms are graph morphisms.

However, not all categories are of this form, as more abstract structures are allowed by the definition:

vii) Any preorder (reflexive and transitive relation) $\sim$ over a set forms a category, where the objects are the elements of the set, and there is a unique morphism $a \to b$ if and only if $a \sim b$. Here identities are given by reflexivity and composition by transitivity. An example of such category is the power set 2^S of a set S with inclusion as the relation.

viii) Any monoid (group without inverses) can be expressed as a one-element category, where the morphisms correspond to the elements and composition to the binary operation of the monoid. In particular, any group G gives such a category BG, where every morphism is an isomorphism by the inverses.

ix) For any unital ring R, Mat_R is a category, where the objects are $\mathbb{Z}_+$ and morphisms $n \to m$ are the $m \times n$ matrices over R. Here composition is given by matrix multiplication.

In the above, vii) is an example of a thin category.

Definition 2.1.4. A **thin category** is a category, where for each pair of objects A and B there is at most one morphism $A \to B$.

Conversely, thin categories can be thought of as preorders over their collection of objects.

Some morphisms can have special properties.

Definition 2.1.5. A morphism $f\colon A \to B$ is called an **isomorphism**, if there is a morphism $g\colon B \to A$ s.t. $g \circ f = 1_A$ and $f \circ g = 1_B$. In this case g is the **inverse** of f, denoted by f^{-1}. If an isomorphism $A \to B$ exists, A and B are **isomorphic** and we write $A \cong B$.

Note that identity morphisms are isomorphisms, an isomorphism f is an inverse of its inverse f^{-1}, so f^{-1} is also an isomorphism, and the composite $f' \circ f$ of two composable isomorphisms f and f' is an isomorphism by the inverse $f^{-1} \circ f'^{-1}$. Hence $\cong$ can be seen as an equivalence relation over the objects of a category, and indeed objects are often considered to be the same, if they are isomorphic.

Example 2.1.6. Isomorphisms in some of the categories of Example 2.1.3 are:

i) Bijections in Set,

ii) Bijective homomorphisms in Group,

iii) Homeomorphisms, i.e. bijective continuous functions whose inverses are also continuous, in Top,

iv) In a thin category, A and B are isomorphic if and only if there are morphisms $A \to B$ and $B \to A$,

v) Invertible elements in a monoid,

vi) Invertible square matrices in Mat_R.

Definition 2.1.7. Morphisms $A \to A$ from an object to itself are called **endomorphisms**, or **automorphisms** if they are also isomorphisms.

Definition 2.1.8. An **epic** morphism, **epimorphism**, is a morphism $f\colon A \to B$, such that for any morphisms $g, h\colon B \to C$ it holds that if $g \circ f = h \circ f$, then $g = h$.

In the category of sets, epimorphisms correspond to surjections. Therefore epimorphisms are thought of as their category-theoretical generalisation. It is easy to see, that surjections are indeed epic in categories with underlying sets. The converse holds sometimes, but not always: for example the inclusion $\iota\colon \mathbb{Z} \to \mathbb{Q}$ in Ring is epic, as any ring homomorphism $\varphi\colon \mathbb{Q} \to R$ is uniquely determined by the restriction $\varphi|_{\mathbb{Z}} = \varphi \circ \iota$ through the homomorphism property $\varphi\left(\frac{a}{b}\right) = \varphi(a)\left(\varphi(b)\right)^{-1}$. Epimorphisms also generalise to abstract categories where surjections are not necessarily defined.

Functors are maps that behave like morphisms between categories, although defining a category of categories is difficult due to set-theoretical limitations.

Definition 2.1.9. A **functor** $F\colon \mathcal{C} \to \mathcal{D}$ between categories $\mathcal{C}$ and $\mathcal{D}$ maps every object $A \in \mathcal{C}$ to an object $F(A) \in \mathcal{D}$ and every morphism $f\colon A \to B$ to a morphism $F(f)\colon F(A) \to F(B)$, so that identities and composition are respected:

- For each object $A \in \mathcal{C}$, we have $F(1_A) = 1_{F(A)}$.

- For each composable pair of morphisms $f\colon A \to B$, $g\colon B \to C$, we have $F(g \circ f) = F(g) \circ F(f)$.

Example 2.1.10. i) Some functors are called forgetful, as they just forget part of the structure of a category. For example there is a forgetful functor Ring $\to$ Group that maps a ring to its additive group, thus forgetting multiplication, and a forgetful functor Group $\to$ Set that maps a group to its underlying set.

ii) There is a functor $P\colon$ Set $\to$ Set that maps a set A to its power set $P(A) = 2^A$ and a function $f\colon A \to B$ to the one that gives its images on subsets: $P(A) \to P(B)\colon A' \mapsto f(A')$ [7].

Definition 2.1.11. A functor $F\colon \mathcal{C} \to \mathcal{D}$ is **full** if $F(\mathcal{C}(A, B)) = \mathcal{D}(F(A), F(B))$ for every pair of objects $A, B \in \mathcal{C}$, i.e. F is surjective on morphisms.

Definition 2.1.12. Let $\mathcal{C}$ and $\mathcal{D}$ be categories. We say that $\mathcal{C}$ is a **subcategory** of $\mathcal{D}$, denoted $\mathcal{C} \subseteq \mathcal{D}$, if for all objects $A, B \in \mathcal{C}$ we have $A, B \in \mathcal{D}$ and $\mathcal{C}(A, B) \subseteq \mathcal{D}(A, B)$. In this case, there is an **inclusion functor** $\iota\colon \mathcal{C} \to \mathcal{D}$ that maps $A \mapsto A$ for objects and $(\varphi\colon A \to B) \mapsto (\varphi\colon A \to B)$ for morphisms. We say that the subcategory is **full** if ι is full, i.e. $\mathcal{C}$ contains all morphisms of $\mathcal{D}$ that are defined between its objects.

When we want to study the isomorphism classes of objects in a category, it is handy to refer to the skeleton of a category. Riehl [7, Definition 1.5.15] defines skeletons as follows:

"A category C is **skeletal** if it contains just one object in each isomorphism class. The **skeleton** SkC of a category C is the unique (up to isomorphism) skeletal category that is equivalent to C."

To avoid defining equivalence of categories, the following construction suffices for us.

Definition 2.1.13. Let $\mathcal{C}$ be a category. By (a generalisation of) axiom of choice, we can choose an object from each isomorphism class. We call the full subcategory $\mathrm{Sk}\,\mathcal{C} \subseteq \mathcal{C}$ determined by these objects the **skeleton** of $\mathcal{C}$.

Furthermore, we define a functor $\mathrm{Sk}\colon \mathcal{C} \to \mathrm{Sk}\,\mathcal{C}$ as follows: Sk maps each object $A \in \mathcal{C}$ to its corresponding isomorphic object $\mathrm{Sk}(A) \in \mathrm{Sk}\,\mathcal{C}$. For each object $A \in \mathcal{C}$, fix an isomorphism $\varphi_A\colon A \to \mathrm{Sk}(A)$. For morphisms, Sk maps $\psi\colon A \to B$ to $\varphi_B \circ \psi \circ \varphi_A^{-1}\colon \mathrm{Sk}(A) \to \mathrm{Sk}(B)$.

Due to the arbitrary choices, this construction is not very natural, and as mentioned, it is only unique up to isomorphism. However, any category isomorphic to it counts as the skeleton, so we should see the skeleton through an isomorphic

category, where the objects are isomorphism classes, instead of a subcategory. The morphisms between the equivalence classes are something abstract defined through representatives, which are picked by the above construction. Therefore our intuition should be that the functor Sk, for which the concrete choices are not specified anyway, gives a way to denote the isomorphism class $\mathrm{Sk}(A)$ of an object A, and for a morphism f its abstract counterpart $\mathrm{Sk}(f)$. In the following we show that Sk is indeed a functor, and as an extra for those who know from other sources, such as Riehl [7], about natural transformations and equivalence of categories, provide a proof that Sk and inclusion give an equivalence between the categories.

Proposition 2.1.13.1. Sk *is indeed a functor. Furthermore* $\mathrm{Sk}\colon \mathcal{C} \to \mathrm{Sk}\,\mathcal{C}$*, together with the inclusion functor* $\iota\colon \mathrm{Sk}\,\mathcal{C} \to \mathcal{C}$*, gives an equivalence of the categories* $\mathrm{Sk}\,\mathcal{C}$ *and* $\mathcal{C}$*.*

Proof. For each $A \in \mathcal{C}$, we have

$$\mathrm{Sk}(1_A) = \varphi_A \circ 1_A \circ \varphi_A^{-1} = 1_{\mathrm{Sk}(A)},$$

and for any composable pair of morphisms $f\colon A \to B$, $g\colon B \to C$, we have

$$\mathrm{Sk}(g \circ f) = \varphi_C \circ g \circ f \circ \varphi_A^{-1} = \varphi_C \circ g \circ \varphi_B^{-1} \circ \varphi_B \circ f \circ \varphi_A^{-1} = \mathrm{Sk}(g) \circ \mathrm{Sk}(f).$$

For equivalence, there is a natural isomorphism $\varphi\colon 1_{\mathrm{Sk}\,\mathcal{C}} \Rightarrow \mathrm{Sk} \circ \iota$, given as follows. For $\mathrm{Sk}(A) \in \mathrm{Sk}\,\mathcal{C}$, we have $1_{\mathrm{Sk}\,\mathcal{C}}(\mathrm{Sk}(A)) = \mathrm{Sk}(A) = \mathrm{Sk} \circ \iota(\mathrm{Sk}(A))$, and an isomorphism $\varphi_{\mathrm{Sk}(A)}\colon \mathrm{Sk}(A) \to \mathrm{Sk}(A)$, given by the definition of Sk. For any morphism $f\colon \mathrm{Sk}(A) \to \mathrm{Sk}(B)$, we have

$$(\mathrm{Sk} \circ \iota)(f) \circ \varphi_{\mathrm{Sk}(A)} = \mathrm{Sk}(f) \circ \varphi_{\mathrm{Sk}(A)} = \varphi_{\mathrm{Sk}(B)} \circ f \circ \varphi_{\mathrm{Sk}(A)}^{-1} \circ \varphi_{\mathrm{Sk}(A)} = \varphi_{\mathrm{Sk}(B)} \circ 1_{\mathrm{Sk}\,\mathcal{C}}(f),$$

and hence the following diagram commutes:

$$\begin{array}{ccc} 1_{\mathrm{Sk}\,\mathcal{C}}(\mathrm{Sk}(A)) & \xrightarrow{\varphi_{\mathrm{Sk}(A)}} & \mathrm{Sk} \circ \iota(\mathrm{Sk}(A)) \\ {\scriptstyle 1_{\mathrm{Sk}\,\mathcal{C}}(f)}\downarrow & & \downarrow{\scriptstyle \mathrm{Sk}\circ\iota(f)} \\ 1_{\mathrm{Sk}\,\mathcal{C}}(\mathrm{Sk}(B)) & \xrightarrow[\varphi_{\mathrm{Sk}(B)}]{} & \mathrm{Sk} \circ \iota(\mathrm{Sk}(B)) \end{array}$$

Conversely, there is a natural isomorphism $\varphi^{-1}\colon \iota \circ \mathrm{Sk} \Rightarrow 1_{\mathcal{C}}$, given as follows. For $A \in \mathcal{C}$, we have $\iota \circ \mathrm{Sk}(A) = \mathrm{Sk}(A)$ and $1_{\mathcal{C}}(A) = A$, and an isomorphism $\varphi_A^{-1}\colon \mathrm{Sk}(A) \to A$, given by the definition of Sk. For any morphism $f\colon A \to B$, we have

$$1_{\mathcal{C}}(f) \circ \varphi_A^{-1} = \varphi_B^{-1} \circ \varphi_B \circ f \circ \varphi_A^{-1} = \varphi_B^{-1} \circ \mathrm{Sk}(f) = \varphi_B^{-1} \circ (\iota \circ \mathrm{Sk})(f),$$

and hence the following diagram commutes:

$$\begin{array}{ccc} \iota \circ \mathrm{Sk}(A) & \xrightarrow{\varphi_A^{-1}} & 1_{\mathcal{C}}(A) \\ {\scriptstyle \iota \circ \mathrm{Sk}(f)}\downarrow & & \downarrow{\scriptstyle 1_{\mathcal{C}}(f)} \\ \iota \circ \mathrm{Sk}(B) & \xrightarrow[\varphi_B^{-1}]{} & 1_{\mathcal{C}}(B) \end{array}$$

□

Some familiar constructions that appear in different fields of mathematics arise from category-theoretical **limits** and **colimits**. While the definition of a limit is too technical for our needs, the general idea behind them is that a diagram of a specific shape should determine an object and some associated morphisms up to a unique isomorphism. The defining conditions for a given limit or colimit can therefore be stated in a general category, and then it can be checked, if a construction fulfilling them exists in any given category. In other words, a limit or colimit does not necessarily exist in a given category, but if it does, its definition is given by a **universal property**. We introduce some limits and colimits that will appear in our LCL-categories.

Definition 2.1.14. For objects A and B, a **product** is an object L with morphisms $\pi_A \colon L \to A$ and $\pi_B \colon L \to B$ called **projections**, such that for any object C and morphisms $f_A \colon C \to A$ and $f_B \colon C \to B$, there is a unique morphism $f \colon C \to L$, for which the following diagram commutes:

$$\begin{array}{ccccc} & & C & & \\ & {\scriptstyle f_A}\swarrow & \vdots\ {\scriptstyle \exists! f} & \searrow{\scriptstyle f_B} & \\ A & \xleftarrow[\pi_A]{} & L & \xrightarrow[\pi_B]{} & B \end{array}$$

The product is often denoted $A \times B$, and the morphism f by $f_A \times f_B$.

We usually only refer to the object when talking about a limit or colimit, but the accompanying projections or coprojections are included implicitly. Furthermore, often call any product "the" product, and similarly for any limit or colimit, since a product is indeed unique up to unique isomorphism by the following.

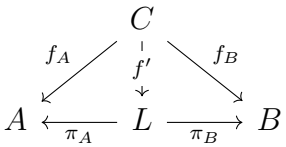
Consider products L, L' of A and B. By the definition, there are unique maps f and f' such that the following diagram commutes:

$$\begin{array}{ccccc} & & L & & \\ & {\scriptstyle \pi_A}\swarrow & \vdots\ {\scriptstyle \exists! f'} & \searrow{\scriptstyle \pi_B} & \\ A & \xleftarrow[\pi'_A]{} & L' & \xrightarrow[\pi'_B]{} & B \\ & {\scriptstyle \pi_A}\nwarrow & \vdots\ {\scriptstyle \exists! f} & \nearrow{\scriptstyle \pi_A} & \\ & & L & & \end{array}$$

In particular, the definition gives that $f \circ f'$ is unique, forcing $f \circ f' = 1_L$, and similarly $f' \circ f = 1_{L'}$. Hence f is an isomorphism and f' its inverse.

Example 2.1.15. In Set, the cartesian product $A \times B = \{(a, b) \mid a \in A, b \in B\}$, together with projections $\pi_A \colon (a, b) \mapsto a$ and $\pi_B \colon (a, b) \mapsto b$, is the product of sets A and B.

Proof. Let C be a set with functions $f_A \colon C \to A$ and $f_B \colon C \to B$. Now there is a function $f \colon C \to A \times B \colon c \mapsto (f_A(c), f_B(c))$, so that $\pi_A \circ f = f_A$ and $\pi_B \circ f = f_B$. For uniqueness, let f' be a function, for which the diagram



commutes. For any $c \in C$, $\pi_A \circ f'(c) = f_A(c)$ and $\pi_B \circ f'(c) = f_B(c)$, i.e. $f'(c) = (f_A(c), f_B(c))$. In conclusion, $f' = f$. □

The choice of product is not unique, as we could have just as well chosen $B \times A$ with $\pi_A \colon (b, a) \mapsto a$ and $\pi_B \colon (b, a) \mapsto b$, or even $A \times \{1\} \times B$ with $\pi_A \colon (a, 1, b) \mapsto a$ and $\pi_B \colon (a, 1, b) \mapsto b$. However, they are isomorphic to $A \times B$ by the isomorphisms $(b, a) \mapsto (a, b)$ and $(a, 1, b) \mapsto (a, b)$ respectively, which are unique by the universal property.

Products can be extended to more than two objects by adding more objects and associated morphisms similar to A and B to the diagram of Definition 2.1.14. Conversely, we can take an empty product by removing them as follows.

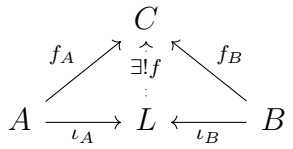
Definition 2.1.16. An object L is **terminal** if for any object C there is a unique morphism $C \to L$.

Again, we can see by uniqueness, that the morphism between two terminal objects is an isomorphism, and therefore we speak of "the" terminal object, if one exists.

Example 2.1.17. In Set, a singleton $\{\bullet\}$ is terminal, since from any set C, there is exactly one function to it, namely $c \mapsto \bullet$. As any sets of the same cardinality are isomorphic, all singletons are indistinguishable from the perspective of category theory, and therefore "the" singleton is referred to by its cardinality 1.

In category theory there is a **duality** that arises from reversing all arrows. For any category $\mathcal{C}$ we can construct the **opposite** or **dual** category $\mathcal{C}^{\mathrm{op}}$, where the objects are the same, and $\mathcal{C}^{\mathrm{op}}(A, B) = \mathcal{C}(B, A)$ with reversed composition. Definitions, statements and proofs in the opposite category can give dual ones in the original, and indeed, for any limit there is a colimit obtained by reversing the arrows in the defining universal property diagram.

Definition 2.1.18. For objects A and B, a **coproduct** or a **sum** is an object L with morphisms $\iota_A \colon A \to L$ and $\iota_B \colon B \to L$ called **coprojections**, such that for any object C and morphisms $f_A \colon A \to C$ and $f_B \colon B \to C$, there is a unique morphism $f \colon L \to C$, for which the following diagram commutes:



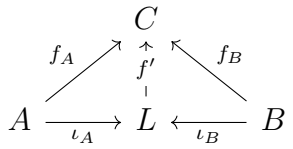
The coproduct is often denoted $A \sqcup B$ or $A \oplus B$, and the morphism f by $f_A \sqcup f_B$ or $f_A \oplus f_B$.

Example 2.1.19. In Set, the disjoint union $A \sqcup B = \{(a,1) \mid a \in A\} \cup \{(b,2) \mid b \in B\}$ (which is isomorphic to $A \cup B$ if A and B are disjoint), with the inclusions $\iota_A \colon A \to A \sqcup B \colon a \mapsto (a,1)$ and $\iota_B \colon B \to A \sqcup B \colon b \mapsto (b,2)$, is the coproduct of sets A and B.

Proof. Let C be a set with functions $f_A \colon A \to C$ and $f_B \colon B \to C$. Now there is a function

$$f \colon A \sqcup B \to C \colon (x,n) \mapsto \begin{cases} f_A(x), & n = 1\,, \\ f_B(x), & n = 2\,, \end{cases}$$

so that $f \circ \iota_A = f_A$ and $f \circ \iota_B = f_B$. For uniqueness, let f' be a function, for which the diagram



commutes. For any $(a,1) \in A \sqcup B$, $f'(a,1) = f' \circ \iota_A(a) = f_A(a)$ and for any $(b,2) \in A \sqcup B$, $f'(b,2) = f' \circ \iota_B(b) = f_B(b)$, i.e.

$$f'(x,n) = \begin{cases} f_A(x), & n = 1\,, \\ f_B(x), & n = 2\,. \end{cases}$$

In conclusion, $f' = f$. □

The dual of terminal is initial, and dually to the empty product, it is the empty sum.

Definition 2.1.20. An object L is **initial** if for any object C there is a unique morphism $L \to C$.

Example 2.1.21. In Set, the empty set $\emptyset$ is initial, since for any set C there is a unique empty function $\iota \colon \emptyset \to C$.

2.2 Distributed Graph Problems

Our work applies category theory to the study of distributed graph problems. Imagine a computer network, where each computer node can communicate with its neighbours, but no node holds full information about the network. We try to solve a graph problem, such as colouring, by making each node run the same distributed algorithm that proceeds in communication rounds. On each communication round each node sends a message to all of its neighbours, receives a message from each neighbour and can then modify its local state. These rounds keep looping until the node reaches a stopping state, after which it cannot modify its state any more, and it gives a local output such as its own colour in the graph. After all nodes have stopped, the local outputs should together form a solution to the problem.

As such a system would tend to be bottlenecked by communication latency rather than the speed of local computations, we often allow the local computations to be arbitrarily complex and measure the complexity of the algorithm in the number of communication rounds. Since it takes k communication rounds for any information to propagate distance k in the network, the complexity of a problem is directly tied to its locality.

We will introduce in more detail a couple of common models of distributed computing from the textbook [1]. While our work abstracts away the underlying model, understanding them can be useful for intuition, and they are domains for the complexity results obtainable through our theory. In particular, we will take a look at the PN-model, which Brandt [4] used as the underlying model for round elimination and to which the results apply directly, and the LOCAL model, to which these results can be lifted to using techniques such as randomisation.

We assume a basic knowlede of graph theory. In the following definition is some notation related to distances in graphs.

Definition 2.2.1. We denote the distance between nodes u and v of a graph G by $\mathrm{dist}_G(u, v)$, the diameter of G, i.e. the maximum distance of any pair of nodes, by $\mathrm{diam}(G)$, and the subgraph of G induced by the nodes within radius r of v by $\mathrm{ball}_G(v, r)$.

The models are built on top of *port-numbered networks*.

Definition 2.2.2 ([1])**.** A **port-numbered network** is a triple (V, P, p), where V is a set of nodes, $P \subseteq V \times \mathbb{Z}_+$ a set of ports and $p\colon P \to P$ an involution that tells us which ports are connected to each other. We say that $(v, i) \in P$ is the port i of the node v, and that the ports (v, i) and (u, j) are connected if $(u, j) = p(v, i)$, in which case also $p(u, j) = p \circ p(v, i) = (v, i)$ by the involution.

We say that the port-numbered network is **simple**, if there are no loops, i.e. $p(v, i)_1 \neq v$ for all $(v, i) \in P$, and no multiple connections between two nodes, i.e.

$p(v,i)_1 \neq p(v,j)_1$ for all $(v,i),(v,j) \in P$ with $i \neq j$. For a simple port-numbered network, we can construct the **underlying graph** G by turning the connections to edges:

$$G = (V, \{(x_1, p(x)_1) \mid x \in P\}) .$$

Intuitively, a simple port-numbered network is just a simple graph, where each node has an indexing for its incident edges. This allows a computer node to distinguish its neighbours by which port they are connected and send different messages to different neighbours. Furthermore, port numbers allow us to represent a variety of graph-theoretic data through node labelling, since for example edge labels can be represented as vector-formatted node labels indexed by port numbers corresponding to the edges. Thus we can define problems as follows.

Definition 2.2.3 ([1])**.** A **distributed graph problem** Π associates for each simple port-numbered network $N = (V, P, p)$ a collection $\Pi(N)$ of **solutions**, each of which is a node labelling $f\colon V \to Y$ for some set of local outputs Y.

Distributed algorithms can be defined through the following collection of sets and functions.

Definition 2.2.4 ([1])**.** A **distributed algorithm** A consists of

- a set Input_A of possible **local inputs**
- a set States_A of possible **local states**
- a set $\mathrm{Output}_A \subseteq \mathrm{States}_A$ of possible **local outputs**, which are the **stopping states**
- a set Msg_A of possible **messages**

and for each $d \in \mathbb{Z}_{\geq 0}$ the functions

- $\mathrm{init}_{A,d}\colon \mathrm{Input}_A \to \mathrm{States}_A$
- $\mathrm{send}_{A,d}\colon \mathrm{States}_A \to \mathrm{Msg}_A^d$
- $\mathrm{receive}_{A,d}\colon \mathrm{States}_A \times \mathrm{Msg}_A^d \to \mathrm{States}_A$, for which

$$\mathrm{receive}_{A,d}|_{\mathrm{Output}_A \times \mathrm{Msg}_A^d} = \pi_{\mathrm{Output}_A}.$$

The algorithm is run on a port-numbered network $N = (V, P, p)$ with the input node labelling $f\colon V \to \mathrm{Input}_A$ as follows. Each node is given an identical copy of the state machine determined by A. The states on each round are represented by

labellings $x_0, \ldots : V \to \text{States}_A$. The initial states are determined from the local input using the $\text{init}_{A,d}$ function:

$$x_0(v) = \text{init}_{A,\deg(v)}(f(v)),$$

where $\deg(v) = |\{(v,i) \in P\}|$ is the number of ports the node v has in N.

Now on each round the nodes compute messages based on their current state and send them to their neighbours, and update their state based on the incoming messages and their current state. On round t, the message being sent through port (v,i) is $\text{send}_{A,\deg(v)}(x_{t-1}(v))_i$, where the tuple is indexed by the port numbers of v, which can be assumed to be $1, \ldots, \deg(v)$. The message is received by the port $p(v,i)$, and conversely the port (v,i) receives a message from $p(v,i)$. Thus we denote the incoming messages by the port labelling $m_t \colon P \to \text{Msg}_A$ determined by

$$m_t(v,i) = \text{send}_{A,\deg(p(v,i)_1)}(x_{t-1}(p(v,i)_1))_{p(v,i)_2}.$$

Now the new state is computed using the $\text{receive}_{A,d}$ function:

$$x_t(v) = \text{receive}_{A,\deg(v)}\left(x_{t-1}, m_t(v,1), \ldots, m_t(v,\deg(v))\right).$$

The above loop keeps repeating. The restriction of the $\text{receive}_{A,d}$ function ensures that when the node hits a stopping state, its state is no longer modified. Thus at that time its local output can be read. Once $x_T(v) \in \text{Output}_A$ for all $v \in V$, the algorithm stops, with $x_T \colon V \to \text{Output}_A$ being its global output and the first such $T \in \mathbb{Z}_{\geq 0}$ its running time.

When we say that an algorithm solves a problem, we assume that the port numbering can be chosen adversarially to be the worst possible for the algorithm.

Definition 2.2.5 ([1])**.** Let $\mathscr{G}$ be a family of simple graphs, Π and Π' distributed graph problems and A a distributed algorithm. In the **PN-model**, we say that A **solves** Π **on** $\mathscr{G}$ **given** Π', if for all simple port-numbered networks $N = (V,P,p)$ that have their underlying graphs in $\mathscr{G}$ and all inputs $f \in \Pi'(N)$, A stops when run on (N,f) and produces an output $g \in \Pi(N)$. In such case, if A always stops in time $T(|V|)$ for some function T, we say that A solves Π in time T.

Example 2.2.6. Let $\mathscr{G}$ be the family of k-colourable simple graphs for some $k \in \mathbb{Z}_+$, and Π' the problem of k-colouring a graph with colours $\{1, \ldots, k\}$. Let Π be the problem of finding a maximal independent set, defined by the labelling $f \colon V \to \{0,1\}$ belonging to $\Pi(N)$ when the following conditions hold:

- If $f(v) = 1$, then $f(u) = 0$ for all neighbours u of v (independence).
- If $f(v) = 0$, then $f(u) = 1$ for some neighbour u of v (maximality).

Consider the following algorithm A:

$$\begin{aligned}
&\mathrm{Input}_A = \{1, \ldots, k\}\,, \\
&\mathrm{States}_A = \{1, \ldots, k\}^2 \cup \{0, 1\}\,, \\
&\mathrm{Output}_A = \{0, 1\}\,, \\
&\mathrm{Msg}_A = \{0, 1\}\,, \\
&\mathrm{init}_{A,d} \colon c \mapsto (c, 1)\,, \\
&\mathrm{send}_{A,d} \colon x \mapsto \begin{cases} (1, \ldots, 1), & \text{if } x = 1\,, \\ (0, \ldots, 0), & \text{otherwise}\,, \end{cases} \\
&\mathrm{receive}_{A,d} \colon (x, m) \mapsto \begin{cases} x, & \text{if } x \in \{0, 1\}\,, \\ (x_1, x_2 + 1), & \text{otherwise, if } x_2 < x_1\,, \\ 1, & \text{otherwise, if } m = (0, \ldots, 0)\,, \\ 0, & \text{otherwise}\,. \end{cases}
\end{aligned}$$

The intuition behind it is as follows. The algorithm takes as an input the colour c of the node in a k-colouring and saves it to state together with a counter. On each round the nodes send 1 to all their neighbours if they have been assigned to the maximal independent set, or 0 if they have not. Nodes whose assignment has been decided have a binary label as their state that they can no longer modify. The nodes whose assignment has not yet been decided count rounds until the round number matches their colour, after which they assign themselves in the maximal independent set if none of their neighbours are yet in, or outside it otherwise.

We claim that A solves Π on $\mathscr{G}$ given Π' in k rounds. Hence we have to show that it terminates in k rounds and that the output is correct.

Firstly, as $\mathrm{receive}_{A,d}$ restricts correctly to Output_A, the algorithm cannot modify a stopping state once it has been reached. Therefore, as all nodes start with counter 1, and on each round either increment the counter or enter a stopping state, the counter matches the round number for nodes that have not stopped. Furthermore, each node enters a stopping state on the first round that the condition $x_2 < x_1$ breaks, i.e. on the round matching their colour. Since the largest possible colour is k, all nodes have to stop in k rounds.

When making a decision, a node will always join the output set if none of its neighbours have joined yet, which gives us that the output satisfies maximality. If a node with colour c joins the output set, none of its neighbours with a colour smaller than c output 1 as they have chosen the stopping state 0 on an earlier round, the node has no neighbours of colour c by the correctness of the input colouring, and all its neighbours with colour larger than c make their decision on later rounds and are forced to pick 0. Therefore independence is also satisfied, and the output is correct.

Without useful inputs the PN-model is weak. The following examples illustrate how the model cannot break symmetries or gather information about the structure of the network by itself.

Example 2.2.7. Consider the network

$$v \overset{1}{\underset{1}{\text{———}}} u \ ,$$

i.e. $(\{v, u\}, \{(v, 1), (u, 1)\}, p)$ with p connecting the two ports.

If an algorithm A is given the same input for both nodes, the nodes will be initialized to the same state. Now on each round both nodes start with the same state, generate the same message with $\text{send}_{A,d}$, send it to each other, and update to the same state with $\text{receive}_{A,d}$. Hence if A terminates, both nodes will have the same output. In conclusion, in the PN-model it is impossible to solve any problem, where v and u would have different labels without symmetry-breaking inputs.

Example 2.2.8. Consider the networks

$$N = \begin{array}{ccccc} v & \overset{1}{\text{———}} & & \overset{2}{\text{———}} & u \\ & {\scriptstyle 2}\diagdown & & \diagup{\scriptstyle 1} & \\ & & {\scriptstyle 1}\ w\ {\scriptstyle 2} & & \end{array} \ , N' = \begin{array}{ccccccc} v & \overset{1}{\text{———}} & & \overset{2}{\text{———}} & u & \overset{1}{\text{———}} & \overset{2}{w} \\ {\scriptstyle 2}\big| & & & & & & \big|{\scriptstyle 1} \\ {\scriptstyle 1}\big| & & & & & & \big|{\scriptstyle 2} \\ w' & \underset{2}{\text{———}} & & \underset{1}{u'} & & \underset{2}{\text{———}} & \underset{1}{v'} \end{array} .$$

If the nodes v', u' and w' are given the same inputs as their twins v, u and w respectively, an algorithm A cannot distinguish the networks in the PN-model. Namely, the nodes in N' are initialised to the same state as their counterparts in N, and similarly to the previous example, on every round the information a node in N' sees is the same as its counterpart in N sees, so they update their state identically. In particular, a node cannot distinguish between information propagated from itself or its twin. Thus the running time and local outputs of the counterparts are same in both networks, and in conclusion, in the PN-model it is impossible to solve any problem, where a node would give a different output in N or N' without giving the twins different inputs.

These issues are solved in the LOCAL model, where every node is given a unique identifier, and the size of the identifiers is bounded to be polynomial with respect to the number of nodes.

Definition 2.2.9 ([1])**.** Let $\mathscr{G}$ be a family of simple graphs, Π and Π' distributed graph problems and A a distributed algorithm. In the **LOCAL model**, we say that A **solves** Π **on** $\mathscr{G}$ **given** Π', if given a fixed constant c, for all simple port-numbered networks $N = (V, P, p)$ that have their underlying graphs in $\mathscr{G}$ and all inputs $f \in \Pi'(N)$ and injections $\text{ident}\colon V \to \{1, \ldots, |V|^c\}$, A stops when run on

$(N, f \times \text{ident})$ and produces an output $g \in \Pi(N)$. In such case, if A always stops in time $T(|V|)$ for some function T, we say that A solves Π in time T. The injection ident is called the **assignment of unique identifiers**.

Example 2.2.10 ([1])**.** Any graph problem that can be solved with a centralised deterministic algorithm is solvable in LOCAL model in linear time for connected graphs. Namely, by storing a set of nodes with their inputs, where nodes are named by their unique identifiers, and a set of edges, nodes can on round t gather their radius t neighbourhood, the messages being each node's radius $t-1$ neighbourhood. Hence in $\text{diam}(G) \leq |V|$ rounds all nodes know the structure and inputs of the whole graph. Now all nodes can solve the problem with the centralised algorithm, which can be as slow as needed since we allow arbitrarily complex computations, and get the same global result, after which each node outputs their own part of the solution.

By Example 2.2.10 we see that unlike in the PN-model, solvability is trivial in the LOCAL model. However, as we can sometimes do much better than linear number of rounds, we are instead interested in how efficiently problems can be solved. Since we can gather in t rounds all information in the t-neighbourhoods but conversely nothing outside them, complexity in the LOCAL model also directly corresponds to how locally a problem can be solved.

In recent years, a lot of progress has been made researching locally checkable labelling (LCL) problems, an important subclass of locally verifiable problems, which in turn are analogous to NP-problems in traditional computing [2]. For example, we now know well, which complexity classes are possible for LCLs, and have methods for proving lower bounds or speeding up algorithms that fall in the gaps between possible complexities.

A *locally verifiable problem* is one, where the correctness of an output can be checked by verifying correctness in constant-sized neighbourhoods. For example, colouring is locally verifiable, as it is enough to check that adjacent nodes have different colours, whereas leader election, where one node in the network outputs 1 and others 0, is not, as accepting leaderless neighbourhoods, which we have to allow for large networks, would imply that a global leaderless output is also accepted. LCLs restrict local verifiability further by requiring that the collection of accepted neighbourhoods is finite, meaning for example that colouring in general is not LCL as the 1-neighbourhoods of nodes with arbitrarily large degree would have to be described, but a colouring in a bounded-degree graph with a finite set of colours is.

Naor and Stockmeyer [3] define LCLs separately for vertex labelling problems in graphs, giving us the following definition:

Definition 2.2.11 ([3])**.** A **locally checkable vertex labelling problem** is a distributed graph problem Π described by a finite set of **labels** Σ, a radius

$r \in \mathbb{Z}_{\geq 0}$ and a finite set of **locally consistent labellings** L, where each element $(H, c, g) \in L$ consists of a graph $H = (V', E')$ centered at $c \in V'$ with radius at most r, i.e. $\mathrm{dist}_H(u, c) \leq r$ for all $u \in V'$, and a function $g\colon V' \to \Sigma$. For a port numbered network N with an underlying graph $G = (V, E)$, $\Pi(N)$ consists of the labellings $f\colon V \to \Sigma$, for which for every $u \in V$ there is a $(H, c, g) \in L$ and a graph isomorphism $\varphi\colon \mathrm{ball}_G(u, r) \to H$ for which $\varphi(u) = c$ and $f(v) = g(\varphi(v))$ for all $v \in \mathrm{ball}_G(u, r)$.

However, we also want to consider problems, where part of an output label is interpreted on the edges through indexing by port numbers. Therefore we make the following generalisations in order to restrict the edge orientations in the locally consistent neighbourhoods.

Firstly, for a simple graph $G = (V, E)$ we have the edge set $E \subseteq V^2$ contain two directed elements $(u, v), (v, u) \in E$ for each undirected edge uv. They can be thought of as unnumbered ports, and we can interpret indexed labellings on them through the port-numbered network as follows.

Definition 2.2.12. Let $N = (V, P, p)$ be a simple port-numbered network with an underlying graph $G = (V, E)$ and $f\colon \{1, \ldots, d\} \to \Sigma$ a function for some $d \in \mathbb{Z}_{\geq 0}$. For any $v \in V$ with $\deg(v) = d$, f induces a function $f_v\colon \{(v, u) \in E\} \to \Sigma$ by the following: For each $(v, u) \in E$ there is a unique port $(v, i) \in P$ for which $p(v, i)_1 = u$. If i is the j:th of the d port numbers of v in ascending order, we define $f_v(v, u) = f(j)$.

Definition 2.2.13. A **locally checkable labelling problem** (**LCL**) is a distributed graph problem Π described by a finite sets of **vertex and edge labels** Σ_V and Σ_E, a radius $r \in \mathbb{Z}_{\geq 0}$ and a finite set of **locally consistent labellings** L, where each element $(H, c, g_V, g_E) \in L$ consists of a graph $H = (V', E')$ centered at $c \in V'$ with radius at most r, i.e. $\mathrm{dist}_H(u, c) \leq r$ for all $u \in V'$, and functions $g_V\colon V' \to \Sigma_V$ and $g_E\colon E' \to \Sigma_E$. For a port numbered network N with an underlying graph $G = (V, E)$, $\Pi(N)$ consists of the labellings

$$f\colon V \to \Sigma_V \times \bigcup_{d=0}^{\infty} \mathrm{Set}\left(\{1, \ldots, d\}, \Sigma_E\right),$$

for which for every $u \in V$, $f(u)_2$ is $\{1, \ldots, \deg(u)\} \to \Sigma_E$, and there is a $(H, c, g_V, g_E) \in L$ and a graph isomorphism $\varphi\colon \mathrm{ball}_G(u, r) \to H$ for which $\varphi(u) = c$, $f(v)_1 = g_V(\varphi(v))$ for all $v \in \mathrm{ball}_G(u, r)$ and $(f(v)_2)_v(v, w) = g_E(\varphi(v), \varphi(w))$ for all $(v, w) \in \mathrm{ball}_G(u, r)$

Note that if only vertex labels are needed, we can choose the set of edge labels to be a singleton and vice versa. Furthermore, as the set of locally consistent labellings is finite, solutions exist only in graphs where the maximum degree is

bounded by some constant Δ, and therefore the output labels can also be defined to be the finite set

$$\Sigma_V \times \bigcup_{d=0}^{\Delta} \operatorname{Set}\left(\{1,\ldots,d\}, \Sigma_E\right).$$

When we are studying lower bounds, we can make further simplifying assumptions by restricting the graph family, as that can allow faster algorithms but not make existing ones slower. If we limit a radius r LCL into graphs that have no cycles of length less than $2r+2$, all r-neighbourhoods are trees, and therefore a node can gather their r-neighbourhood even in the PN-model. Now we can transform such LCL-problem into a radius 1 LCL by having each node output the structure, oriented by port numbers, and original outputs of their entire $(r-1)$-neighbourhood, and accepting the labelling if all outputs in the radius 1 neighbourhood combine into a valid r-neighbourhood output of the original problem. As the new problem can be solved in constant $r-1$ rounds given the original problem by gathering, and the old in 0 rounds given the new one by only outputting the label of the node itself, the two problems fall into the same complexity class. In conclusion, with such restriction it is enough to study LCL-problems of radius 1.

Further restricting radius 1 problems to regular graphs gives rise to the following biregular formalism.

Definition 2.2.14. A (d,δ)**-biregular LCL-problem** is a triple (Σ, A, P), where Σ is a finite set of labels, and A and P respectively collections of size d and size δ multisets over Σ, called the **active and passive configurations**. The triple defines a radius 1 LCL-problem as follows:

- $\Sigma_V = \{0,1\}$ and $\Sigma_E = \Sigma \sqcup \{\bullet\}$
- For every active configuration $a \in A$ there is a locally consistent labelling $(S_d, c, g_V, g_E) \in L$, where S_d is the star graph with a central node c and outer nodes $\{1,\ldots,d\}$, with

  $$g_V : v \mapsto \begin{cases} 1 & \text{if } v = c \\ 0 & \text{otherwise} \end{cases}$$

  and $g_E(c,v) \in \Sigma$, $g_E(v,c) = \bullet$ for each outer node v so that $a = [g_E(c,1),\ldots,g_E(c,d)]$.
- For every passive configuration $p \in P$ there is a locally consistent labelling $(S_\delta, c, g_V, g_E) \in L$, where S_δ is the star graph with a central node c and outer nodes $\{1,\ldots,\delta\}$, with

  $$g_V : v \mapsto \begin{cases} 0 & \text{if } v = c \\ 1 & \text{otherwise} \end{cases}$$

and $g_E(c, v) = \bullet$, $g_E(v, c) \in \Sigma$ for each outer node v so that $p = [g_E(1, c), \ldots, g_E(\delta, c)]$.

Another way to look at biregular LCL-problems is that they are only solvable over biregular graphs, which have a bipartition to active and passive nodes denoted by the vertex labels 1 and 0, and the active nodes, which have degree d, output labels to their incident edges according to an active configuration, and the passive nodes, which have degree δ, accept the output if the labels they see on their incident edges form a passive configuration.

Any radius 1 LCL over d-regular graphs without 3-cycles with $d \geq 1$, determined by Σ_V, Σ_E and L, can be converted to a $(d, 2)$-biregular LCL-problem (Σ, A, P) as follows:

- Let $\Sigma = \Sigma_V^2 \times \Sigma_E^2$.
- For each $l = (H, c, g_V, g_E) \in L$, we can assume $H = S_d$ with a central node c and outer nodes $\{1, \ldots, d\}$ by d-regularity without 3-cycles. Denote $t(l, i) = (g_V(c), g_V(i), g_E(c, i), g_E(i, c))$.
- Let $A = \{[t(l, 1), \ldots, t(l, d)] \mid l \in L\}$.
- Let $P = \{[(x, x', y, y'), (x', x, y', y)] \mid x, x' \in \Sigma_V, y, y' \in \Sigma_E\}$.

Here allowed 1-neighbourhoods are encoded into the active configurations while passive nodes simulate edges and the passive configurations ensure that the encoding stays consistent over an edge. If an algorithm can solve the original problem in T rounds in a network, then the same algorithm can be simulated in the network obtained by adding a node into each edge, by running the algorithm on odd rounds and propagating the information over the edge nodes on even rounds, to solve the new problem in $2T + 1$ rounds. Conversely, an algorithm that solves the new problem in T rounds can be converted to solve the original in T rounds by simulating the edge nodes in the vertices.

In conclusion, to study lower bounds of general LCLs, we can instead study problems in the biregular formalism. One proof technique that is simpler to formulate and automatise in the biregular formalism is **round elimination**. The core idea of round elimination is that a problem solvable in T rounds can under certain assumptions be transformed into another one that is solvable in $T - 1$ rounds. The technique was defined by Brandt [4] more generally for certain types of locally verifiable problems, but we will use a versions adapted for the biregular formalism.

Definition 2.2.15 ([1])**.** Let $\Pi = (\Sigma, A, P)$ be a (d, δ)-biregular LCL-problem. We define the **output problem** $\mathrm{Re}(\Pi) = (S, B, Q)$ to be the (δ, d)-biregular LCL-problem given as follows:

- S is the set of non-empty subsets of Σ.
- B is the set of multisets $[b_1, \ldots, b_\delta]$, where $b_1, \ldots, b_\delta \in S$ and every choice $p_1 \in b_1, \ldots, p_\delta \in b_\delta$ gives a passive configuration $[p_1, \ldots, p_\delta] \in P$ for the original problem.
- Q is the set of multisets $[q_1, \ldots, q_d]$, where $q_1, \ldots, q_d \in S$ and there is a choice $a_1 \in q_1, \ldots, a_d \in q_d$ that gives an active configuration $[a_1, \ldots, a_d] \in A$ for the original problem.

Notice that round elimination flips the roles of active and passive nodes for the output problem. Given a solution to $\mathrm{Re}(\Pi)$, Π can be solved in one round as follows: The active nodes of $\mathrm{Re}(\Pi)$ send their edge outputs to their respective neighbours, which are the active nodes of Π. For each neighbour, the received subsets of Σ form a passive configuration from Q by the validity of the solution, and thus the node can output a label from each subset so that the choice forms an active configuration in A. As the original outputs formed active configurations in B, an active node of $\mathrm{Re}(\Pi)$, which is passive in Π, sees a passive configuration in P regardless of the choice, and therefore the output is a valid solution of Π.

Conversely, under certain assumptions $\mathrm{Re}(\Pi)$ can be solved one round faster than Π. Assume that Π is solvable in $T \geq 1$ rounds by an algorithm A on a family of graphs with no cycles shorter than $2T + 4$. As all $(T + 1)$-neighbourhoods are trees, each node u that is active in $\mathrm{Re}(\Pi)$ can gather their $(T - 1)$-neighbourhood even in the PN-model. For each neighbour v, the node u can simulate all potential T-neighbourhoods of v that coincide with the $(T - 1)$-neighbourhood of u, and run the algorithm A on them, collecting the outputs for the edge (v, u) in a set. When these sets are output for $\mathrm{Re}(\Pi)$ by u, the node v sees a valid passive configuration in Q as the simulation outputs corresponding to its actual T-neighbourhood can be chosen. Furthermore, we assume that the input in the T-neighbourhood of v outside the $(T - 1)$-neighbourhood of u is independent of such inputs for the other neighbours. For example if the input is unique identifiers, the assumption does not hold, as an identifier appearing in one of the extensions blocks it from the others, whereas in a loose enough graph family a colouring would fulfill the assumption, as there would be coloured graphs with any of the combinations as subgraphs. The simulated neighbourhoods only overlap on the fixed $(T - 1)$-neighbourhood of u, and thus any combination of the simulated outputs is a valid configuration in P, i.e. u outputs a configuration in B.

It is possible, that a repeated application of round elimination outputs a problem that the original problem is 0-round reducible to. These problems are called **periodic points of round elimination**, and they are interesting since they give a contradiction to the above assumptions. For example, if Π is 0-round reducible to $\mathrm{Re}(\mathrm{Re}(\Pi))$, it can be solved in 0 rounds given $\mathrm{Re}(\mathrm{Re}(\Pi))$, but at the same time

the assumptions would give that $\mathrm{Re}(\mathrm{Re}(\Pi))$ can be solved exactly 2 rounds faster that Π. Therefore if Π is not 0-round solvable, which is easy to check, and the independence assumption is satisfied for example by giving each node the same input, we get a lower bound that depends on the length of the shortest cycle in the graph[2]. Now if the underlying graph family has graphs, where the shortest cycles are relatively long, the bound can be good.

We can observe, that as round elimination produces a problem over the non-empty subsets of the original alphabet, the definitions of the output problems get complicated quickly. Therefore it is important to simplify them as much as possible. One common simplification is through maximality of active configurations, which is used for example by Round Eliminator up to relabelling [5].

Definition 2.2.16. Let $\Pi = (\Sigma, A, P)$ be a (d, δ)-biregular LCL-problem and $\mathrm{Re}(\Pi) = (S, B, Q)$. We define the **maximality-simplified output problem** $\mathrm{Re}'(\Pi) = (S', B', Q')$ to be the (δ, d)-biregular LCL-problem given as follows:

- B' consists of the **maximal active configurations** in B, i.e.

$$B' = \{b' \in B \mid \nexists b \in B \setminus \{b'\} : b'_1 \subseteq b_1, \dots, b'_\delta \subseteq b_\delta\}.$$

- S' consists of those labels in S that actually appear in B', i.e.

$$S' = \{s \in S \mid \exists b' \in B' \colon s \in b'\}.$$

- Q' consists of those configurations in Q that are over the alphabet S', i.e.

$$Q' = \{q' \in Q \mid q_1, \dots, q_d \in S'\}.$$

Since $S' \subseteq S$, $B' \subseteq B$ and $Q' \subseteq Q$, any solution to $\mathrm{Re}'(\Pi)$ is also a solution to $\mathrm{Re}(\Pi)$. Conversely, for every $b \in B$ we can find $b' \in B'$ with $b_1 \subseteq b'_1, \dots, b_\delta \subseteq b'_\delta$ by maximising the partial order, since B is finite. Therefore in any solution of $\mathrm{Re}(\Pi)$ an active configuration b can be mapped to b' through $b_1 \mapsto b'_1, \dots, b_\delta \mapsto b'_\delta$, and the passive conditions do not break because for a passive configuration $q \in Q$, any q' with $q_1 \subseteq q'_1, \dots, q_d \subseteq q'_d$ is in Q, since a choice $a_1 \in q_1, \dots, a_d \in q_d$ for which $[a_1, \dots, a_d] \in A$ also applies for q'. In conclusion, $\mathrm{Re}(\Pi)$ and $\mathrm{Re}'(\Pi)$ are 0-round equivalent.

[2]We have only defined the biregular formalism over genuine biregular graphs, where all nontrivial finite graphs have cycles. It is also possible to make similar definitions for example over biregular trees, and the limiting assumption becomes the largest radius of a neighbourhood without leaves, giving us logarithmic lower bounds, or linear in the case of paths [1]. Nevertheless, symbolically the round elimination process stays the same and the periodic points are important in both cases.

3 LCL-problems in Categories

3.1 Objects

The traditional definition of LCL-problems in the biregular formalism uses multisets. They are sufficient for describing a problem and checking solutions, since permutations of a configuration are not important. However, when studying 0-round reductions, we need to keep track of permutations. Thus we will consider the multisets as equivalence classes of tuples.

Let Σ be a set and $k \in \mathbb{Z}_{\geq 0}$.

Definition 3.1.1. Let $\mathbb{S}_k$ be the symmetric group over the index set $\{1, \ldots, k\}$. It gives us an action on Σ^k by

$$\mathbb{S}_k \times \Sigma^k \to \Sigma^k \colon (\sigma, x) \mapsto \left(x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(k)}\right),$$

i.e. for a permutation $\sigma \in \mathbb{S}_k$ and a tuple $x \in \Sigma^k$ we will denote

$$\sigma(x) = \left(x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(k)}\right),$$

which corresponds to moving the element at each index i to the index $\sigma(i)$.

Proposition 3.1.1.1. *The function defined above is indeed an action.*

Proof. We have the identity

$$\mathrm{id}(x) = \left(x_{\mathrm{id}(1)}, \ldots, x_{\mathrm{id}(k)}\right) = x$$

and compatibility

$$\begin{aligned}\sigma(\tau(x)) &= \sigma\left(x_{\tau^{-1}(1)}, \ldots, x_{\tau^{-1}(k)}\right) = \left(x_{\tau^{-1}(\sigma^{-1}(1))}, \ldots, x_{\tau^{-1}(\sigma^{-1}(k))}\right) \\ &= \left(x_{(\sigma\tau)^{-1}(1)}, \ldots, x_{(\sigma\tau)^{-1}(k)}\right) = (\sigma\tau)(x).\end{aligned}$$

□

Definition 3.1.2. We denote by $\sim_\sigma$ the permutation equivalence in Σ^k, i.e. for all $x, y \in \Sigma^k$:

$$x \sim_\sigma y \iff \exists \tau \in \mathbb{S}_k : x = \tau(y).$$

Proposition 3.1.2.1. *The relation $\sim_\sigma$ is an equivalence relation.*

Proof. Reflexivity is given by $x = \mathrm{id}(x)$, symmetry by

$$x = \tau(y) \iff \tau^{-1}(x) = \tau^{-1}\tau(y) = y,$$

and transitivity by

$$x = \tau(y) \wedge y = \tau'(z) \Rightarrow x = \tau(\tau'(z)) = (\tau\tau')(z).$$

□

Definition 3.1.3. We denote by $\Sigma^k/\sim_\sigma$ the quotient set that consists of the equivalence classes of $\sim_\sigma$. There is a canonical projection

$$\pi\colon \Sigma^k \to \sigma^k/\sim_\sigma\colon x \mapsto \overline{x} = \left\{y \in \Sigma^k \mid y \sim_\sigma x\right\},$$

which maps a tuple x to its equivalence class $\overline{x}$. If the tuple is represented by its elements $x = (x_1, \ldots, x_k)$, we use $\overline{x} = [x_1, \ldots, x_k]$ for the equivalence class.

The following definition and lemma help us extend functions to the quotient.

Definition 3.1.4. Given a function $\varphi\colon \Sigma \to \Sigma'$, we denote by $\varphi\colon \Sigma^k \to \Sigma'^k$ the induced function $(x_1, \ldots, x_k) \mapsto (\varphi(x_1), \ldots, \varphi(x_k))$, and by $\varphi\colon \Sigma^k/\sim_\sigma \to \Sigma'^k/\sim_\sigma$ the induced function $[x_1, \ldots, x_k] = \overline{x} \mapsto \overline{\varphi(x)} = [\varphi(x_1), \ldots, \varphi(x_k)]$.

Proposition 3.1.4.1. *The function $\varphi\colon \Sigma^k/\sim_\sigma \to \Sigma'^k/\sim_\sigma$ is well-defined.*

Proof. Due to elementwise application of $\varphi\colon \Sigma \to \Sigma'$, we have $\varphi(\sigma(x)) = \sigma(\varphi(x))$ for all $x \in \Sigma^k$ and $\sigma \in \mathbb{S}_k$. □

Lemma 3.1.5. *Let $\varphi\colon \Sigma \to \Sigma'$ and $\varphi'\colon \Sigma' \to \Sigma''$. Now $\varphi' \circ \varphi = (\varphi' \circ \varphi) : \Sigma^k \to \Sigma''^k$ and $\varphi' \circ \varphi = (\varphi' \circ \varphi) : \Sigma^k/\sim_\sigma \to \Sigma''^k/\sim_\sigma$, i.e. composition of induced functions gives the function induced by composed functions.*

Proof. We have

$$\varphi' \circ \varphi(x) = \varphi'(\varphi(x_1), \ldots, \varphi(x_k)) = (\varphi' \circ \varphi(x_1), \ldots, \varphi' \circ \varphi(x_k)) = (\varphi' \circ \varphi)(x)$$

and

$$\varphi' \circ \varphi(\overline{x}) = \varphi'[\varphi(x_1), \ldots, \varphi(x_k)] = [\varphi' \circ \varphi(x_1), \ldots, \varphi' \circ \varphi(x_k)] = (\varphi' \circ \varphi)(\overline{x}). \quad \square$$

As we use the overline $\bar{\cdot}$ to move from tuples to equivalence classes, we will use the following underline notation to lift back to tuples.

Definition 3.1.6. Let $X \subseteq \Sigma^k/\sim_\sigma$. We denote by $\underline{X}$ the preimage

$$\underline{X} = \pi^{-1}(X) = \left\{x \in \Sigma^k \mid \overline{x} \in X\right\}.$$

By abuse of notation we will use $\sim_\sigma$, π and $\bar{\cdot}$ for all Σ, k and restrictions of π, if the specifics are clear from context, or implicit differentiating notation such as π'. The square bracket notation is sometimes used for multisets, which is also intentional abuse of notation as $\sigma^k/\sim_\sigma$ is in clear bijective correspondence with multisets of size k over Σ. This also makes it easy to see that the following definition of LCL-problems is equivalent to Definition 2.2.14.

Definition 3.1.7. Let $d, \delta \in \mathbb{Z}_+$. A (d, δ)-biregular LCL-problem is a pair (A, P), where $A \subseteq \Sigma^d / \sim_\sigma$ and $P \subseteq \Sigma^\delta / \sim_\sigma$ for some finite set Σ. We denote the class of such pairs $\mathrm{LCL}(d, \delta)$.

Here Σ is the underlying alphabet, A the active configurations and P the passive configurations. The alphabet is usually unimportant and can be inferred from (A, P) to be the one that contains exactly the labels that appear in A or P, but sometimes it is convenient to refer to it explicitly, in which case we can denote the problem by (Σ, A, P). In order to construct categories with $\mathrm{LCL}(d, \delta)$ as objects, lifting to $\underline{A}$ and $\underline{P}$ allows us to define morphisms between them.

3.2 LCP-maps

We want morphisms between LCL-problems to have a good balance of capturing 0-round reductions and being easy to work with. Therefore we restrict ourselves to reductions given by maps between active tuples, which fulfill sufficient correctness-preserving conditions that depend on the passive configurations, and furthermore use the heuristic that these conditions should be local. This leads us to the following definition of local correctness preserving (LCP) maps. We will further discuss the choice in Section 4.

Definition 3.2.1. Let $\Pi = (A, P), \Pi' = (A', P') \in \mathrm{LCL}(d, \delta)$. We define a local correctness preserving map $\varphi \colon \Pi \to \Pi'$ to be a function $\varphi \colon \underline{A} \to \underline{A'}$ that satisfies the implication

$$\left(a^{(1)}_{i_1}, \dots, a^{(\delta)}_{i_\delta}\right) \in \underline{P} \Rightarrow \left(\varphi\left(a^{(1)}\right)_{i_1}, \dots, \varphi\left(a^{(\delta)}\right)_{i_\delta}\right) \in \underline{P'} \qquad \text{(LCP-rule)}$$

for all $a^{(1)}, \dots, a^{(\delta)} \in \underline{A}$ and $i_1, \dots, i_\delta \in \{1, \dots, d\}$. We denote the set of all LCP-maps $\Pi \to \Pi'$ by $\mathrm{LCP}(\Pi, \Pi')$.

The intuition here is that the active nodes can map their output for Π into an output of Π', and the LCP-rule ensures that if a passive node was satisfied with the labels it saw for Π, it will be satisfied for Π' after the mapping, no matter how the tuples in $\underline{A}$ gave the original labels.

The LCP-rule can be handy for writing proofs, as we will see for example in Section 5. However, checking it naively can be costly, as it contains unnecessary detail about the active tuples, while it is enough to know the possible images of each label. Therefore we will introduce an equivalent condition, where all this has been abstracted into a relation.

Definition 3.2.2. Let $\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}(d, \delta)$ and $\sim$ be a relation on $\Sigma \times \Sigma'$. The relation $\sim$ induces a relation $\sim^\delta$ on $\Sigma^\delta \times \Sigma'^\delta$ by

$$q \sim^\delta q' \iff q_1 \sim q'_1, \dots, q_\delta \sim q'_\delta.$$

We say that the pair (P, P') is compatible with the relation $\sim$, if it holds that

$$\forall p \in \underline{P} : p \sim^{\delta} q' \Rightarrow q' \in \underline{P'}.$$

Definition 3.2.3. A function $\varphi\colon \underline{A} \to \underline{A'}$ induces a label relation $\sim_\varphi$ on $\Sigma \times \Sigma'$ by

$$l \sim_\varphi l' \iff \exists a \in \underline{A}, i \in \{1, \ldots, d\} : a_i = l, \varphi(a)_i = l'.$$

Essentially, we can view $\sim_\varphi$ as a multi-valued map, that associates with each label in Σ the set of its possible images in Σ' under φ. $\sim_\varphi^\delta$ gives the same for δ-tuples, and thus compatibility of (P, P') with $\sim_\varphi$ would give that the image of $\underline{P}$ is in $\underline{P'}$, therefore corresponding to a correctness-preserving condition. By the following lemma LCP-maps could equivalently be defined through compatibility.

Lemma 3.2.4. *The LCP-rule holds for φ if and only if (P, P') is compatible with the induced relation $\sim_\varphi$.*

Proof. "⇒": Assume that φ is LCP and $p \sim_\varphi^\delta q'$ for some $p \in \underline{P}$ and $q' \in \Sigma'^\delta$. Now

$$p_1 \sim_\varphi q'_1, \ldots, p_\delta \sim_\varphi q'_\delta,$$

and therefore there are $a^{(1)}, \ldots, a^{(\delta)} \in \underline{A}$ and $i_1, \ldots, i_\delta \in \{1, \ldots, d\}$ s.t.

$$a^{(1)}_{i_1} = p_1, \ldots, a^{(\delta)}_{i_\delta} = p_\delta$$

and

$$\varphi\left(a^{(1)}\right)_{i_1} = q'_1, \ldots, \varphi\left(a^{(\delta)}\right)_{i_\delta} = q'_\delta.$$

Hence by the LCP-rule $q' \in \underline{P'}$, and thus (P, P') is compatible with $\sim_\varphi$.

"⇐": Assume now that (P, P') is compatible with $\sim_\varphi$ and $\left(a^{(1)}_{i_1}, \ldots, a^{(\delta)}_{i_\delta}\right) \in \underline{P}$ for some $a^{(1)}, \ldots, a^{(\delta)} \in \underline{A}$ and $i_1, \ldots, i_\delta \in \{1, \ldots, d\}$. We have

$$a^{(1)}_{i_1} \sim_\varphi \varphi\left(a^{(1)}\right)_{i_1}, \ldots, a^{(\delta)}_{i_\delta} \sim_\varphi \varphi\left(a^{(\delta)}\right)_{i_\delta},$$

and therefore compatibility gives us

$$\left(\varphi\left(a^{(1)}\right)_{i_1}, \ldots, \varphi\left(a^{(\delta)}\right)_{i_\delta}\right) \in \underline{P'}.$$

Hence the LCP-rule holds. □

One advantage with the compatibility approach is that it can be also checked for relations that are not necessarily induced by a function. Therefore, when finding LCP-maps, we can build the relation one label pair at a time, and check compatibility only for parts involving the new additions, as we will see in Section 6. Furthermore, as all subrelations of a compatible relation are compatible, it is easy to show that some modifications to functions preserve the LCP-properties.

Lemma 3.2.5. *If a pair (P, P') is compatible with a relation $\sim$ and $\sim' \subseteq \sim$, then the pair is compatible with $\sim'$. Here the relations are interpreted as subsets of $\Sigma \times \Sigma'$.*

Proof. $\sim'$ being a subrelation gives us the implication $p \sim'^{\delta} q' \Rightarrow p \sim^{\delta} q'$, and therefore compatibility with $\sim'$ follows from compatibility with $\sim$. □

3.3 Constructing the LCL-category

We get $\mathrm{LCP}((A,P),(A',P'))$ as a subset of functions $\underline{A} \to \underline{A'}$ by restricting to those that satisfy the LCP-rule, and therefore LCP-maps inherit composition and local smallness from Set. However, we still have to show that the restriction respects identities and composition.

Lemma 3.3.1. *Let* $\Pi, \Pi', \Pi'' \in \mathrm{LCL}(d,\delta)$. *We have:*

a) $1_\Pi = 1_{\underline{A}} \in \mathrm{LCP}(\Pi,\Pi)$.

b) $\varphi \in \mathrm{LCP}(\Pi,\Pi') \wedge \varphi' \in \mathrm{LCP}(\Pi',\Pi'') \Rightarrow \varphi' \circ \varphi \in \mathrm{LCP}(\Pi,\Pi'')$.

Proof. a) We have

$$\left(1_{\underline{A}}\left(a^{(1)}\right)_{i_1}, \ldots, 1_{\underline{A}}\left(a^{(\delta)}\right)_{i_\delta}\right) = \left(a^{(1)}_{i_1}, \ldots, a^{(\delta)}_{i_\delta}\right),$$

so the LCP-rule holds trivially.

b) Assume

$$\left(a^{(1)}_{i_1}, \ldots, a^{(\delta)}_{i_\delta}\right) \in \underline{P}.$$

The LCP-rule for φ gives

$$\left(\varphi\left(a^{(1)}\right)_{i_1}, \ldots, \varphi\left(a^{(\delta)}\right)_{i_\delta}\right) \in \underline{P'},$$

and now the LCP-rule for φ' gives

$$\left(\varphi' \circ \varphi\left(a^{(1)}\right)_{i_1}, \ldots, \varphi' \circ \varphi\left(a^{(\delta)}\right)_{i_\delta}\right) \in \underline{P''}.$$

Hence the LCP-rule holds for $\varphi' \circ \varphi$. □

We are now ready to expand $\mathrm{LCL}(d,\delta)$ into a category.

Definition 3.3.2. We denote by $\mathrm{LCL}(d,\delta)$ the locally small category, whose objects are $\mathrm{LCL}(d,\delta)$ and morphisms LCP-maps.

$\mathrm{LCL}(d,\delta)$ is the main category we will be working with, and from which we derive other related categories, which are illustrated in Figure 1. In Section 3.5 we will construct the thin version $\mathrm{TLCL}(d,\delta)$ by collapsing parallel morphisms into one, and its skeletal version $\mathrm{SkTLCL}(d,\delta)$ by identifying equivalent objects. On the other hand, in Section 4 we expand the category by allowing less strict global correctness preserving conditions, and explore how the resulting categories can sometimes be similar to $\mathrm{LCL}(d,\delta)$, when reduced to $\mathrm{LCL}^*(d,\delta)$ by removing impossible labels. We hop between categories of opposite degrees in Section 5, where round elimination is considered, and finally in Section 6 we investigate the more restricted configuration-respecting LCP-maps with $\mathrm{CRLCL}(d,\delta)$ and the more general nondeterministic LCP-maps with $\mathrm{NDLCL}(d,\delta)$.

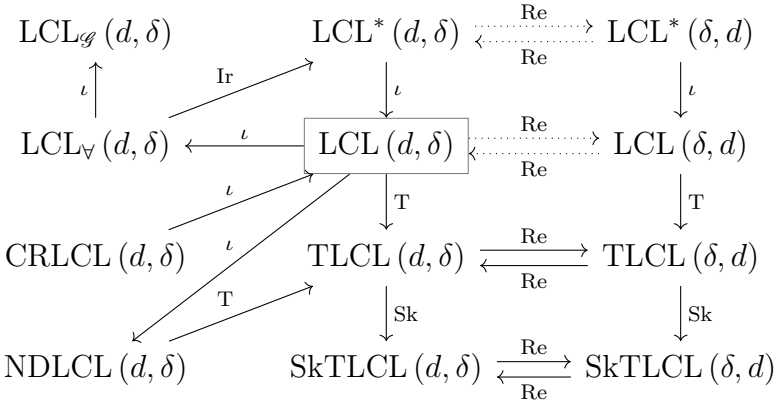


Figure 1: LCL-based categories and their relationships

3.4 Limits and Colimits

Before we start constructing related categories, let us check that the limits introduced in Section 2.1 exist in $\mathrm{LCL}(d, \delta)$.

Definition 3.4.1. We call a problem of the form $(\emptyset, P) \in \mathrm{LCL}(d, \delta)$ an **empty problem**, and denote $\Pi_\emptyset = (\emptyset, \emptyset) \in \mathrm{LCL}(d, \delta)$.

Theorem 3.4.2. *A problem is an initial object in* $\mathrm{LCL}(d, \delta)$ *if and only if it is empty.*

Proof. Observe that $\underline{\emptyset} = \emptyset$.

"$\Leftarrow$": Given problems $(\emptyset, P), (A', P') \in \mathrm{LCL}(d, \delta)$, there is precisely one function $\emptyset \to \underline{A'}$, namely the empty function, and it is trivially LCP, as the $a^{(1)}, \ldots, a^{(\delta)}$ in the LCP-rule are picked from an empty set. Hence empty problems are initial.

"$\Rightarrow$": There are no maps from a non-empty problem (A, P) to $\Pi_\emptyset$, since $\emptyset$ has no element to map $a \in \underline{A}$ to. Thus $(A, P) \ncong \Pi_\emptyset$, and hence (A, P) is not initial. □

Definition 3.4.3. We call a problem of the form

$$(\{[\bullet, \ldots, \bullet]\}, \{[\bullet, \ldots, \bullet]\} \cup P) \in \mathrm{LCL}(d, \delta)$$

a **trivial problem**, and denote

$$\Pi_\bullet = (\{[\bullet, \ldots, \bullet]\}, \{[\bullet, \ldots, \bullet]\}) \in \mathrm{LCL}(d, \delta).$$

Here $\bullet$ denotes the element of a general singleton set $\{\bullet\}$, and $[\bullet, \ldots, \bullet]$ has d copies of it on tha active side and δ on the passive.

Theorem 3.4.4. *A problem is a terminal object if and only if it is trivial.*

Proof. "$\Leftarrow$": Given problems $(\{[\bullet, \ldots, \bullet]\}, \{[\bullet, \ldots, \bullet]\} \cup P), (A', P') \in \mathrm{LCL}(d, \delta)$, there is precisely one function $\underline{A'} \to \underline{\{[\bullet, \ldots, \bullet]\}} = \{(\bullet, \ldots, \bullet)\}$, namely the one that maps everything to the single element $(\bullet, \ldots, \bullet)$, and it is trivially LCP, as the consequent of the LCP-rule becomes the tautology $(\bullet, \ldots, \bullet) \in \{(\bullet, \ldots, \bullet)\} \cup \underline{P}$. Hence trivial problems are terminal.

"$\Rightarrow$": Let (A', P') be a terminal object in $\mathrm{LCL}(d, \delta)$. We know by the universal property $(A', P') \cong \Pi_\bullet \in \mathrm{LCL}(d, \delta)$, and as LCP-maps are inherited from set functions, the isomorphism has to be a bijection $\underline{A'} \cong \{(\bullet, \ldots, \bullet)\}$. The single element of $\underline{A'}$ has to be a fixed point of all of its permutations, and therefore of the form $(a', \ldots, a')$. Furthermore, there is only one set map $\{(\bullet, \ldots, \bullet)\} \to \{(a', \ldots, a')\} = \underline{A'}$, which has to be the LCP-map $\Pi_\bullet \to (A', P')$, and therefore by the LCP-rule $(a', \ldots, a') \in \underline{P'}$, i.e. $[a', \ldots, a'] \in P'$. In conclusion, $(A', P') = (\{[a', \ldots, a']\}, \{[a', \ldots, a']\} \cup P')$ is trivial. □

The following lemmas help us deal with coproducts.

Lemma 3.4.5. *A label map $\varphi\colon \Sigma \to \Sigma'$ induces an LCP-map $\varphi\colon (\Sigma, A, P) \to (\Sigma', \varphi(A), \varphi(P))$ by the induced maps introduced in Definition 3.1.4. If φ is bijective, the induced map is an isomorphism.*

Proof. Because of the elementwise application of $\varphi\colon \Sigma \to \Sigma'$, the induced label relation $\sim_{\varphi|_{\underline{A}}}$ is a subrelation of $\sim_\varphi$ defined by

$$l \sim_\varphi l' \iff l' = \varphi(l).$$

As $\varphi(\underline{P}) \subseteq \underline{\varphi(P)}$, the pair $(P, \varphi(P))$ is compatible with $\sim_\varphi$, and therefore by Lemma 3.2.5 also with its subrelation $\sim_{\varphi|_{\underline{A}}}$, giving us by Lemma 3.2.4 that that the LCP-rule holds for $\varphi|_{\underline{A}}$, which we therefore denote $\varphi\colon (A, P) \to (\varphi(A), \varphi(P))$.

If $\varphi\colon \Sigma \to \Sigma'$ is bijective, its inverse is a label map, which induces the inverse LCP-map $\varphi^{-1}\colon (\Sigma', \varphi(A), \varphi(P)) \to (\Sigma, A, P)$, since by Lemma 3.1.5

$$\varphi^{-1} \circ \varphi(A) = 1_A(A) = A,$$

$$\varphi^{-1} \circ \varphi(P) = 1_P(P) = P,$$

$$\varphi^{-1} \circ \varphi = 1_{\underline{A}} = 1_{(A,P)}$$

and

$$\varphi \circ \varphi^{-1} = 1_{\underline{\varphi(A)}} = 1_{(\varphi(A),\varphi(P))}.$$ □

Lemma 3.4.6. *Let $A \subseteq A'$ and $P \subseteq P'$. Now $\subseteq\colon \underline{A} \to \underline{A'}\colon a \mapsto a$ is LCP, which we denote $(A, P) \subseteq (A', P')$.*

Proof. The label relation $\sim_\subseteq$ induced by $\subseteq\colon \underline{A} \to \underline{A'}\colon a \mapsto a$ through Definition 3.2.3 is a subrelation of $=$. $P \subseteq P'$ gives $\underline{P} \subseteq \underline{P'}$, and hence the pair (P, P') is compatible with $=$. Therefore by Lemmas 3.2.5 and 3.2.4, $\subseteq$ is LCP. □

Definition 3.4.7. If $(A, P) \subseteq (A', P')$, we call (A, P) a **subproblem** of (A', P') and (A', P') a **superproblem** of (A, P)

Definition 3.4.8. Given two problems

$$\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}\,(d, \delta)\,,$$

we denote

$$\Pi \oplus \Pi' = (\Sigma \sqcup \Sigma', \iota_\Sigma(A) \cup \iota_{\Sigma'}(A'), \iota_\Sigma(P) \cup \iota_{\Sigma'}(P')) \in \mathrm{LCL}\,(d, \delta)\,,$$

where $\iota_\Sigma \colon \Sigma \to \Sigma \sqcup \Sigma'$ and $\iota_{\Sigma'} \colon \Sigma' \to \Sigma \sqcup \Sigma'$ are the coprojections of the disjoint union $\Sigma \sqcup \Sigma'$, induced to the quotients through Definition 3.1.4. Furthermore, we denote

$$\iota_\Pi = \iota_\Sigma|_{\underline{A}} : \underline{A} \to \underline{\iota_\Sigma(A) \cup \iota_{\Sigma'}(A')}$$

and

$$\iota_{\Pi'} = \iota_{\Sigma'}|_{\underline{A'}} : \underline{A'} \to \underline{\iota_\Sigma(A) \cup \iota_{\Sigma'}(A')},$$

and given a problem $\Pi'' = (A'', P'') \in \mathrm{LCL}\,(d, \delta)$ and LCP-maps $f_\Pi \colon \Pi \to \Pi''$ and $f_{\Pi'} \colon \Pi' \to \Pi''$, by $f = f_\Pi \oplus f_{\Pi'} \colon \Pi \oplus \Pi' \to \Pi''$ the map

$$\begin{cases} (\iota_\Pi(a)) \mapsto f_\Pi(a)\,, \\ (\iota_{\Pi'}(a')) \mapsto f_{\Pi'}(a')\,. \end{cases}$$

Remark. If Σ and Σ' are disjoint, we have $\Pi \oplus \Pi' \cong (A \cup A', P \cup P')$, where A and A', and likewise P and P', are disjoint. The problem corresponds to solving either Π or Π'.

Theorem 3.4.9. *$\Pi \oplus \Pi' \in \mathrm{LCL}\,(d, \delta)$, together with the LCP-maps $\iota_\Pi \colon \Pi \to \Pi \oplus \Pi'$ and $\iota_{\Pi'} \colon \Pi' \to \Pi \oplus \Pi'$, is the sum (coproduct) of the problems $\Pi, \Pi' \in \mathrm{LCL}\,(d, \delta)$, and the map $f_\Pi \oplus f_{\Pi'}$ in Definition 3.4.8 is a well-defined LCP-map.*

Proof. Recall that ι_Σ and $\iota_{\Sigma'}$ are injections, and thus bijections to their disjoint images $\iota_\Sigma(\Sigma)$ and $\iota_{\Sigma'}(\Sigma')$, for which $\iota_\Sigma(\Sigma) \cup \iota_{\Sigma'}(\Sigma') = \Sigma \sqcup \Sigma'$. Thus by Lemma 3.4.5, ι_Σ and $\iota_{\Sigma'}$ induce isomorphisms $\Pi \cong (\iota_\Sigma(\Sigma), \iota_\Sigma(A), \iota_\Sigma(P))$ and $\Pi' \cong (\iota_{\Sigma'}(\Sigma'), \iota_{\Sigma'}(A'), \iota_{\Sigma'}(P'))$ to the subproblems, which by composition with inclusions give the LCP-maps ι_Π and $\iota_{\Pi'}$.

As the two subproblems are over disjoint alphabets, $\underline{A}$ and $\underline{A'}$, and similarly $\underline{P}$ and $\underline{P'}$, are disjoint. Thus for $f_\Pi \oplus f_{\Pi'}$, we can identify which subproblem the input in $\underline{\iota_\Sigma(A) \cup \iota_{\Sigma'}(A')} = \underline{\iota_\Sigma(A)} \cup \underline{\iota_{\Sigma'}(A')}$ belongs to, and map it to Π'' through Π by $f_\Pi \circ \iota_\Pi^{-1}$, or through Π' by $f_{\Pi'} \circ \iota_{\Pi'}^{-1}$. Thus $f_\Pi \oplus f_{\Pi'}$ is well-defined as a function $\underline{\iota_\Sigma(A) \cup \iota_{\Sigma'}(A')} \to \underline{A''}$. To show that $f_\Pi \oplus f_{\Pi'}$ is LCP, we observe that by the disjoint alphabets, in cases where the antecedent of the LCP-rule is true, all entries of the tuple must be from the same subproblem, and then the LCP-rule for $f_\Pi \circ \iota_\Pi^{-1}$ or $f_{\Pi'} \circ \iota_{\Pi'}^{-1}$, which are composites of LCP-maps, gives the consequent.

To conclude that the construction is a categorical coproduct, we have to show that f is unique. Let $f' \colon \Pi \oplus \Pi' \to \Pi''$ s.t. $f_\Pi = f' \circ \iota_\Pi$ and $f_{\Pi'} = f' \circ \iota_{\Pi'}$. We have $f'(\iota_\Pi(a)) = f_\Pi(a) = f(\iota_\Pi(a))$ for all $a \in \underline{A}$ and $f'(\iota_{\Pi'}(a')) = f_{\Pi'}(a') = f(\iota_{\Pi'}(a'))$ for all $a' \in \underline{A'}$, and therefore $f' = f$ by f being well-defined. □

The following notation will help us construct a product of problems.

Definition 3.4.10. We define the operator $\boxtimes\colon \Sigma^k \times \Sigma'^k \to (\Sigma \times \Sigma')^k$ by

$$x \boxtimes x' = ((x_1, x'_1), \ldots, (x_k, x'_k)),$$

and for $X \subseteq \Sigma^k$ and $X' \subseteq \Sigma'^k$ denote

$$X \boxtimes X' = \{x \boxtimes x' \mid x \in X, x' \in X'\}.$$

We extend the notation to quotients by the rule

$$Y \boxtimes Y' = \overline{\underline{Y} \boxtimes \underline{Y'}},$$

where $Y \subseteq \Sigma^k / \sim_\sigma$ and $Y' \subseteq \Sigma'^k / \sim_\sigma$.

Remark. Notice, that here we fix the set product $\Sigma \times \Sigma'$ to be the usual $\{(x, x') \mid x \in \Sigma, x' \in \Sigma'\}$, with projections $\pi_\Sigma\colon \Sigma \times \Sigma' \to \Sigma\colon (x, x') \mapsto x$ and $\pi_{\Sigma'}\colon \Sigma \times \Sigma' \to \Sigma'\colon (x, x') \mapsto x'$. The LCL-product would work just as well with other set products, but we would have to use a less convenient notation than (x, x').

Lemma 3.4.11. *Let $x \in \Sigma^k$, $x' \in \Sigma'^k$ and $\sigma \in \mathbb{S}_k$. We have*

$$\sigma(x \boxtimes x') = \sigma(x) \boxtimes \sigma(x').$$

Proof. We have

$$\begin{aligned}
\sigma(x \boxtimes x') &= \left((x \boxtimes x')_{\sigma^{-1}(1)}, \ldots, (x \boxtimes x')_{\sigma^{-1}(k)}\right) \\
&= \left((x_{\sigma^{-1}(1)}, x'_{\sigma^{-1}(1)}), \ldots, (x_{\sigma^{-1}(k)}, x'_{\sigma^{-1}(k)})\right) \\
&= \sigma(x) \boxtimes \sigma(x').
\end{aligned}$$

□

Lemma 3.4.12. $\underline{Y \boxtimes Y'} = \underline{Y} \boxtimes \underline{Y'}$

Proof. As $\overline{\underline{Y} \boxtimes \underline{Y'}} = Y \boxtimes Y'$, by preimages of $\bar{\cdot}$ we have $\underline{Y} \boxtimes \underline{Y'} \subseteq \underline{Y \boxtimes Y'}$. For the converse, consider an element of $\underline{Y \boxtimes Y'}$, which has to be of the form $\sigma(x \boxtimes x')$ for some permutation $\sigma \in \mathbb{S}_k$ and tuples $x \in \underline{Y}$ and $x' \in \underline{Y'}$. By Lemma 3.4.11

$$\sigma(x \boxtimes x') = \sigma(x) \boxtimes \sigma(x') \in \underline{Y} \boxtimes \underline{Y'},$$

since $\underline{Y}$ and $\underline{Y'}$ contain all permutations of their elements. □

Definition 3.4.13. Given two problems

$$\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}(d, \delta),$$

we denote

$$\Pi \times \Pi' = (\Sigma \times \Sigma', A \boxtimes A', P \boxtimes P') \in \mathrm{LCL}(d, \delta).$$

Furthermore, we denote

$$\pi_\Pi = \pi_\Sigma|_{\underline{A \boxtimes A'}} : \underline{A \boxtimes A'} \to \underline{A}$$

and

$$\pi_{\Pi'} = \pi_{\Sigma'}|_{\underline{A \boxtimes A'}} : \underline{A \boxtimes A'} \to \underline{A'},$$

where $\pi_\Sigma \colon \Sigma \times \Sigma' \to \Sigma$ and $\pi_{\Sigma'} \colon \Sigma \times \Sigma' \to \Sigma'$ are the projections of the set product $\Sigma \times \Sigma'$ extended to $(\Sigma \times \Sigma')^d$ through Definition 3.1.4.

Given a problem $\Pi'' = (A'', P'') \in \mathrm{LCL}(d, \delta)$ and LCP-maps $f_\Pi \colon \Pi'' \to \Pi$ and $f_{\Pi'} \colon \Pi'' \to \Pi'$, we denote by $f = f_\Pi \times f_{\Pi'} \colon \Pi'' \to \Pi \times \Pi'$ the map

$$a'' \mapsto f_\Pi(a'') \boxtimes f_{\Pi'}(a'').$$

Theorem 3.4.14. *$\Pi \times \Pi' \in \mathrm{LCL}(d, \delta)$, together with the LCP-maps $\pi_\Pi \colon \Pi \times \Pi' \to \Pi$ and $\pi_{\Pi'} \colon \Pi \times \Pi' \to \Pi'$, is the product of the problems $\Pi, \Pi' \in \mathrm{LCL}(d, \delta)$, and the map $f_\Pi \times f_{\Pi'}$ in Definition 3.4.13 is an LCP-map.*

Proof. The label map $\pi_\Sigma \colon \Sigma \times \Sigma' \to \Sigma$ induces by Lemma 3.4.5 an LCP-map $\Pi \times \Pi' \to (\Sigma, \pi_\Sigma(A \boxtimes A'), \pi_\Sigma(P \boxtimes P'))$. The maps $\pi_\Sigma \colon (\Sigma \times \Sigma')^k \to \Sigma^k$ induced through Definition 3.1.4 map

$$\overline{x \boxtimes x'} \mapsto [\pi_\Sigma(x_1, x'_1), \ldots, \pi_\Sigma(x_k, x'_k)] = [x_1, \ldots, x_k] = \overline{x},$$

so $\pi_\Sigma(A \boxtimes A') = A$ and $\pi_\Sigma(P \boxtimes P') = P$, giving us that the induced LCP-map is π_Π. Symmetrically, $\pi_{\Pi'}$ is also LCP.

To show that $f = f_\Pi \times f_{\Pi'}$ is LCP, let

$$\left(a''^{(1)}_{i_1}, \ldots, a''^{(\delta)}_{i_\delta}\right) \in \underline{P''}$$

for some $a''^{(1)}, \ldots, a''^{(\delta)} \in \underline{A''}$ and $i_1, \ldots, i_\delta \in \{1, \ldots, d\}$. By the LCP-rule for f_Π and $f_{\Pi'}$ we get

$$\left(f_\Pi\left(a''^{(1)}\right)_{i_1}, \ldots, f_\Pi\left(a''^{(\delta)}\right)_{i_\delta}\right) \in \underline{P}$$

and

$$\left(f_{\Pi'}\left(a''^{(1)}\right)_{i_1}, \ldots, f_{\Pi'}\left(a''^{(\delta)}\right)_{i_\delta}\right) \in \underline{P'},$$

giving us

$$\begin{aligned}
&\left(f\left(a''^{(1)}\right)_{i_1}, \ldots, f\left(a''^{(\delta)}\right)_{i_\delta}\right) \\
&= \left(\left(f_\Pi\left(a''^{(1)}\right) \boxtimes f_{\Pi'}\left(a''^{(1)}\right)\right)_{i_1}, \ldots, \left(f_\Pi\left(a''^{(\delta)}\right) \boxtimes f_{\Pi'}\left(a''^{(\delta)}\right)\right)_{i_\delta}\right) \\
&= \left(\left(f_\Pi\left(a''^{(1)}\right)_{i_1}, f_{\Pi'}\left(a''^{(1)}\right)_{i_1}\right), \ldots, \left(f_\Pi\left(a''^{(\delta)}\right)_{i_\delta}, f_{\Pi'}\left(a''^{(\delta)}\right)_{i_\delta}\right)\right) \\
&= \left(f_\Pi\left(a''^{(1)}\right)_{i_1}, \ldots, f_\Pi\left(a''^{(\delta)}\right)_{i_\delta}\right) \boxtimes \left(f_{\Pi'}\left(a''^{(1)}\right)_{i_1}, \ldots, f_{\Pi'}\left(a''^{(\delta)}\right)_{i_\delta}\right) \\
&\in \underline{P} \boxtimes \underline{P'} = \underline{P \boxtimes P'}.
\end{aligned}$$

In conclusion, the LCP-rule holds for f.

To show that f is unique, let f' be an LCP-map $\Pi'' \to \Pi \times \Pi'$, s.t. $\pi_\Pi \circ f' = f_\Pi$ and $\pi_{\Pi'} \circ f' = f_{\Pi'}$, and $a'' \in \underline{A''}$. We have $f'(a'') = a \boxtimes a'$ for some $a \in \underline{A}$ and $a' \in \underline{A'}$. Now

$$f_\Pi(a'') = \pi_\Pi \circ f'(a'') = (\pi_\Sigma(a_1, a'_1), \ldots, \pi_\Sigma(a_k, a'_k)) = (a_1, \ldots, a_k) = a,$$

and similarly $f_{\Pi'}(a'') = a'$. Hence $f'(a'') = f_\Pi(a'') \boxtimes f_{\Pi'}(a'') = f(a'')$, and thus $f'' = f$. □

3.5 Equivalence of Problems

For complexity results, the details of a 0-round reduction are usually not important, but rather we are interested whether one exists or not. We can reflect this in the category side by modifying the $\mathrm{LCL}(d, \delta)$ category into a thin counterpart, where morphisms indicate whether LCP-maps from one problem to another exist.

Definition 3.5.1. We denote by $\mathrm{TLCL}(d, \delta)$ the thin category, where the objects are the same as in $\mathrm{LCL}(d, \delta)$, and there is a unique morphism $\Pi \to \Pi'$ if and only if $\mathrm{LCP}(\Pi, \Pi') \neq \emptyset$. Furthermore, we denote by T the **thinning functor** $\mathrm{LCL}(d, \delta) \to \mathrm{TLCL}(d, \delta)$ that maps $\Pi \mapsto \Pi$ for objects and $(\varphi\colon \Pi \to \Pi') \mapsto (\Pi \to \Pi')$ for morphisms.

Proof. $\mathrm{TLCL}(d, \delta)$ is indeed a category. For every object Π, the identity morphism 1_Π in $\mathrm{LCL}(d, \delta)$ implies a morphism $\Pi \to \Pi$ in $\mathrm{TLCL}(d, \delta)$. Furthermore, for morphisms $\Pi \to \Pi'$ and $\Pi' \to \Pi''$ in $\mathrm{TLCL}(d, \delta)$ there are morphisms $\varphi\colon \Pi \to \Pi'$ and $\varphi'\colon \Pi' \to \Pi''$ in $\mathrm{LCL}(d, \delta)$, and thus $\varphi' \circ \varphi\colon \Pi \to \Pi''$ implies the composite $\Pi \to \Pi''$ of $\Pi \to \Pi' \to \Pi''$ in $\mathrm{TLCL}(d, \delta)$. By thinness the morphisms $\Pi \to \Pi$ are identities and composition associative.

In addition, T is a functor by thinness of $\mathrm{TLCL}(d, \delta)$, as the object and morphism maps are well defined. □

We define the equivalence of problems through the thin category.

Definition 3.5.2. We say that the problems $\Pi, \Pi' \in \mathrm{LCL}(d, \delta)$ are **equivalent**, denoted by $\Pi \simeq \Pi'$, if $\mathrm{T}(\Pi) \cong \mathrm{T}(\Pi')$.

In other words, two problems are equivalent if there exists LCP-maps in both directions between them. Two problems are the same from the complexity perspective, if they are equivalent, since they can be reduced to each other. Notice that the equivalence of problems is clearly reflexive, symmetric and transitive through the isomorphisms in $\mathrm{TLCL}(d, \delta)$, and therefore it is sensible to talk about equivalence classes. We want to study proof techniques like round elimination over these equivalence classes, which correspond to objects in the skeleton of the thin category.

Definition 3.5.3. We denote by $\mathrm{SkTLCL}(d,\delta)$ the skeleton of $\mathrm{TLCL}(d,\delta)$, and by $\mathrm{Sk}\colon \mathrm{TLCL}(d,\delta) \to \mathrm{SkTLCL}(d,\delta)$ the functor defined in Definition 2.1.13.

In conclusion, we now have tools to look at LCL-problems on three levels:

- $\mathrm{LCL}(d,\delta)$, where we can study specific LCP-maps in order to construct a 0-round reduction.
- $\mathrm{TLCL}(d,\delta)$, where we can study how a specific LCL-problem relates to others through reductions induced by LCP-maps in order to get complexity results.
- $\mathrm{SkTLCL}(d,\delta)$, where we can study proof techniques such as round elimination, which should not depend on the choice of representative in an equivalence class.

4 Capturing 0-round Reductions

4.1 Always Accepting Passives

In this section we discuss how well LCP-maps capture 0-round reductions. First we consider a scenario where some nodes can accept any configuration, and compare it to biregular trees, where round elimination is often studied. Then we define new categories, where morphisms are 0-round maps that only need to preserve global correctness, and see how they differ from the ones with LCP-maps. Finally, we investigate impossible labels, and show that they can be removed safely and functorially to simplify problems.

Let us start by considering which kind of reductions we want to capture. Firstly, we limit ourselves to reductions that add no additional round cost. This is not to say that reductions with round cost could not be useful. For example, if we found a function that maps n-neighbourhoods of solutions of $\mathrm{Re}^{n+1}(\Pi)$ to configurations of Π so that solutions are preserved, then we would get that Π is only n rounds harder than $\mathrm{Re}^{n+1}(\Pi)$ by gathering the n-neighbourhoods in n rounds and using the function, which would contradict $\mathrm{Re}^{n+1}(\Pi)$ being exactly $n+1$ rounds easier than Π with similar conclusions as with periodic points. However, such reductions can become exponentially more complicated and do not give equivalence classes of problems with the same exact complexity.

Secondly, we only consider reductions that are based on mapping solutions. It might be possible to have some other way of using an algorithm for Π as a subroutine for an algorithm for Π', but we only cover the case where Π is solved first and then the solution is transformed into a solution of Π'. In other words, reducing Π' to Π now becomes finding an algorithm for Π' given Π, so this is the simple way of doing reductions, and it might not even be possible to take into account all creative techniques someone might come up with, if we tried to cover a more general notion.

In conclusion, a 0-round reduction must be determined by a map, which must be executed for each node independently due to the 0-round constraint, must fulfill some solution-preserving conditions, and only has as input the data of the executing node, i.e. whether it is active and its edge labels if it is. For such maps $\Pi \to \Pi'$, we can ignore ones that swap active and passive nodes, as in order to produce the new active labels out of $(\bullet, \dots, \bullet)$ the problem Π' would have to be 0-round solvable, and ones involving randomness, as they have to work even for a specific choice of random bits. Thus they are determined by maps $\underline{A} \to \underline{A'}$ that preserve solutions.

Consider now the following setting. We are solving $\Pi = (A, P)$ and $\Pi' = (A', P')$ in some PN-network, but some predetermined subset of the passive nodes are allowed to accept any configuration. Now a 0-round reduction from Π' to Π corresponds to a map $\varphi\colon \underline{A} \to \underline{A'}$, such that for any PN-network, selection of always accepting

passive nodes and solution Π that is correct everywhere else, mapping the active outputs with φ gives a solution to Π' that is correct outside the same subset of passives. LCP-maps are clearly 0-round reductions in this situation, as the LCP-rule ensures that whenever a passive node has a valid configuration, the validity is preserved by the mapping. Conversely, setting the PN-network to be an infinite biregular tree and all but one passive nodes as always accepting gives us, that a valid labelling consists of a valid 1-neighbourhood of that one passive node and the active nodes outside that neighbourhood can output any active configuration. Hence preserving solutions corresponds to preserving that one neighbourhood i.e. the LCP-rule. In conclusion, in this setting the 0-round reductions correspond precisely to the LCP-maps.

By using the same idea of making almost all passive nodes in an infinite biregular tree always accepting, we can also simulate finite biregular trees whose leaves are passive. Namely, take a tree coloured with active and passive, where active nodes have degree d and passive nodes are either leaves or have degree δ, make the leaves always accepting and complete the tree into an infinite biregular tree where the added passive nodes are always accepting. These form a subset of cases in the previous setting, and by the same argument 0-round reductions here correspond precisely to LCP-maps. Trees can be particularly useful in studying round elimination since one of the assumptions of round elimination is that there are no small cycles. Therefore, a very similar setting is used in [1]. The differences are that we have defined LCL-problems with more mathematical rigour and that we do not allow active leaves since arbitrary active outputs do not mesh well with the functions. Nevertheless, we can equate that LCP-maps are the correct notion of 0-reductions in that setting.

4.2 GCP-categories

In the previous setting we made the solution-preserving conditions as strict as possible by essentially increasing the number of PN-networks, whose solutions must be preserved, as we introduced the always accepting passives. Now we will do the opposite by only considering global solutions in different graph families. With global solutions we can ignore labelling patterns that can only appear locally, since it is possible that a function breaks those, thus not being LCP, but still preserves solutions. The more we restrict the structure of the family, the more patterns become impossible and the easier it becomes for a function to qualify as LCP.

Definition 4.2.1. Let $\Pi = \Sigma, A, P \in \mathrm{LCL}(d, \delta)$ and N be a PN-network with an underlying (d, δ)-biregular graph $G = (V_A \sqcup V_P, E)$, where V_A are the active nodes and V_P the passive nodes. Recall, that by Definitions 2.2.13 and 2.2.14 a solution

of Π over N is of the form $f\colon V \to \Sigma_A \cup \Sigma_P$, where

$$\Sigma_A = \{(1,g) \mid g\colon \{1,\dots,d\} \to \Sigma, [g(1),\dots,g(d)] \in A\}\ ,$$
$$\Sigma_P = \{(0,g) \mid g\colon \{1,\dots,\delta\} \to \{\bullet\}\}\,.$$

We are only interested about the ones where the role of the nodes is predetermined, so we further assume $f(V_A) \subseteq \Sigma_A$ and $f(V_P) \subseteq \Sigma_P$. As Σ_P is a singleton, these functions are in bijective correspondence with functions $V_A \to \underline{A}$ by $f \mapsto (v \mapsto (f(v)_2(1),\dots,f(v)_2(d)))$, and hence we denote them by the corresponding $f\colon V_A \to \underline{A}$.

Since $f\colon V_A \to \underline{A}$ represents the output of the active nodes, it induces a function $h_f\colon V_P \to \Sigma^\delta$ that represents the labels seen by the passive nodes. Recall from Definition 2.2.2 that the port i of node v in N is connected to the port $p(v,i)_2$ of node $p(v,i)_1$. Hence we define

$$h_f\colon v \mapsto \left(f(p(v,1)_1)_{p(v,1)_2},\dots,f(p(v,\delta)_1)_{p(v,\delta)_2}\right).$$

Now Definition 2.2.14 implies that f is a solution if and only if h_f is $V_E \to \underline{P}$. Therefore by abuse of notation we denote the solutions by the set

$$\Pi(N) = \{f\colon V_A \to \underline{A} \mid h_f\colon V_P \to \underline{P}\}\,.$$

Definition 4.2.2. Let $\Pi = (A,P), \Pi' = (A',P') \in \mathrm{LCL}\,(d,\delta)$ and $\mathscr{G}$ be a family of possibly infinite (d,δ)-biregular graphs. We say that $\varphi\colon \underline{A} \to \underline{A'}$ is **global correctness preserving over the graph family** $\mathscr{G}$ ($\mathscr{G}$-GCP), if for all PN-networks N whose underlying graph is in $\mathscr{G}$, φ satisfies the implication

$$f \in \Pi(N) \Rightarrow \varphi \circ f \in \Pi'(N). \qquad \text{(GCP-rule)}$$

We denote the set of $\mathscr{G}$-GCP maps from Π to Π' by $\mathrm{GCP}_{\mathscr{G}}\,(\Pi,\Pi')$. If $\mathscr{G}$ is the family of all (d,δ)-biregular graphs, we say φ is GCP and denote the set $\mathrm{GCP}\,(\Pi,\Pi')$.

We can observe that if $\mathscr{G}' \subseteq \mathscr{G}$, then $\mathscr{G}$-GCP-maps are $\mathscr{G}'$-GCP, since the GCP-rule holds over all $\mathscr{G}' \subseteq \mathscr{G}$. Thus being GCP is a sufficient condition for being $\mathscr{G}$-GCP. Furthermore, LCP-maps are GCP, since the LCP-rule gives that if $h_f(v) \in \underline{P}$ then $h_{\varphi\circ f}(v) \in \underline{P'}$, and hence in particular 1_Π is GCP. Because the composition of $\mathscr{G}$-GCP-maps $\varphi \in \mathrm{GCP}_{\mathscr{G}}\,(\Pi,\Pi')$ and $\varphi' \in \mathrm{GCP}_{\mathscr{G}}\,(\Pi',\Pi'')$ is also $\mathscr{G}$-GCP by $f \in \Pi(N)$ implying $\varphi\circ f \in \Pi'(N)$ which in turn implies $\varphi'\circ\varphi\circ f \in \Pi''(N)$, we have that $\mathscr{G}$-GCP-maps and LCL-problems form categories.

Definition 4.2.3. Let $\mathscr{G}$ be a family of (d,δ)-biregular graphs. We denote by $\mathrm{LCL}_{\mathscr{G}}\,(d,\delta)$ the category that has $\mathrm{LCL}\,(d,\delta)$ as objects and $\mathscr{G}$-GCP-maps as morphisms. In case of GCP-maps, we denote $\mathrm{LCL}_\forall\,(d,\delta)$.

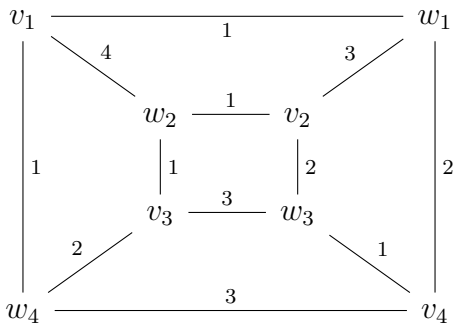
By the previous remarks, we have as subcategories $\mathrm{LCL}(d,\delta) \subseteq \mathrm{LCL}_\forall(d,\delta) \subseteq \mathrm{LCL}_{\mathscr{G}}(d,\delta) \subseteq \mathrm{LCL}_{\mathscr{G}'}(d,\delta)$. The following examples show that we can have $\mathrm{LCL}(d,\delta) \subsetneq \mathrm{LCL}_\forall(d,\delta)$ and for strong enough restrictions on $\mathscr{G}$ we can have $\mathrm{LCL}_\forall(d,\delta) \subsetneq \mathrm{LCL}_{\mathscr{G}}(d,\delta)$.

Example 4.2.4. Consider the problems $\Pi = (\{[1,2,3],[1,1,4]\},\{[1,2,3]\})$ and $\Pi' = (\{[1,2,3]\},\{[1,2,3]\})$, which correspond to 3-edge-colouring a $(3,3)$-biregular graph. The first one has an additional active configuration $[1,1,4]$, which clearly cannot appear in a globally correct labelling as there is no passive configuration that accepts label 4. Thus the map $\varphi\colon \underline{A} \to \underline{A'}$ defined by

$$a \mapsto \begin{cases} a, & \overline{a} = [1,2,3] \\ (1,2,3), & \overline{a} = [1,1,4] \end{cases},$$

is GCP, since applying it to a valid solution would be identical to applying $1_{\Pi'}$.

However, we can show that $\mathrm{LCP}(\Pi,\Pi') = \emptyset$. Consider the following labelling in a cube graph with active nodes $\{v_1,\dots,v_4\}$ and passive nodes $\{w_1,\dots,w_4\}$:



We can choose the port numbering so that the nodes v_1 and w_2 output or see $(1,1,4)$ and the others $(1,2,3)$. If a $\psi \in \mathrm{LCP}(\Pi,\Pi')$ existed, it would have to map $(1,2,3)$ to one of its permutations, and therefore applying it to the labelling would correspond to a bijective relabelling of edges not incident to v_1. By the LCP-rule w_1 and w_4 must accept the new labelling, so edges v_1w_1 and v_1w_4 end up having the same label, which has to appear in $\psi((1,1,4))$ twice. However, as Π' has no active configurations that repeat labels, we get that $\mathrm{LCP}(\Pi,\Pi') = \emptyset$.

In conclusion, $\mathrm{LCL}(3,3)$ and $\mathrm{LCL}_\forall(3,3)$ are different, even after taking the thin versions of the categories.

Example 4.2.5. Let $\mathscr{G}$ be the family of cycle graphs of length $2n$, where n is divisible by 3, and $\mathscr{G}'$ the subfamily where n is 3 modulo 6. Consider the problems $\Pi'' = (\{[1,1],[2,2]\},\{[1,2]\})$ and $\Pi' = (\{[3,3],[4,5]\},\{[3,4],[5,5]\})$. Π'' corresponds to 2-colouring a cycle of length n by considering the passive nodes

as edges and the passive condition as the two neighbouring nodes having different colours. Π' on the other hand corresponds to labelling the nodes of the cycle of length n so that runs of two outputs $[4,5]$ are always separated by one $[3,3]$. In particular, Π'' has valid solutions only if n is even and Π' only if n is divisible by 3. Let $\Pi = \Pi' \oplus \Pi''$, which corresponds to solving either Π' or Π''.

Consider the function $\varphi\colon \underline{A} \to \underline{A'}$ defined by

$$a \mapsto \begin{cases} a, & \overline{a} = [4,5] \ , \\ (3,3), & \overline{a} \neq [4,5] \ . \end{cases}$$

Since Π'' is unsolvable over $\mathscr{G}'$, a valid solution solution to Π over $\mathscr{G}'$ is given by a solution of Π', on which φ acts by $1_{\Pi'}$, and hence φ is $\mathscr{G}'$-GCP.

Conversely, $\mathscr{G}$ includes cycles where n is even, and therefore the solution to Π can be given by a solution of Π''. However, $\mathrm{GCP}_{\mathscr{G}}(\Pi'', \Pi') = \emptyset$ since such map would need to map two adjacent active nodes to the same configuration $[4,5]$, but then all active nodes from the 2-colouring would map to $[4,5]$ and none to $[3,3]$. Thus the GCP-rule for $\underline{A} \to \underline{A'}$ breaks for 2-colourings, giving us also $\mathrm{GCP}_{\mathscr{G}}(\Pi, \Pi') = \emptyset$.

In conclusion, by restricting the graph family from $\mathscr{G}$ to $\mathscr{G}'$, we cut off the possibility that the pattern of 2-colouring appears in a solution for Π, and this made some maps $\underline{A} \to \underline{A'}$ to be $\mathscr{G}'$-GCP.

In Example 4.2.4 GCP-maps get separated from LCP-maps because of the impossible label 4, which does not appear in the passive side and can thus never appear in a global solution, but LCP-maps still have to take into account what happens to the other labels in its configuration. In the following section we will see that impossible labels are indeed the only thing that can cause such separation. However, the separation does not always happen, since there has to also be possible labels in the same configuration that get affected, and in particular $\mathrm{LCL}(1,\delta) = \mathrm{LCL}_{\forall}(1,\delta)$.

On the other hand, the consequence of Example 4.2.5 is that we can find more 0-round reductions and thus prove more things about the problems if we know more about the graphs we are working with. As an extreme case, if we choose $\mathscr{G}$ to be empty or only contain the empty graph, all functions $\underline{A} \to \underline{A'}$ become $\mathscr{G}$-GCP, and we can for example conclude that over an empty network all non-empty problems have the same exact complexity.

4.3 Impossible Labels

As we observed in Example 4.2.4, labels that appear in active configurations but not in passive ones can cause separation of LCP- and GCP-maps, since in GCP-maps their configurations can freely be mapped to any target configuration. We will now

show that without them GCP-maps reduce to LCP-maps, in particular because infinite trees have no long range dependencies that could cause some configuration not appear in any solution, unlike in Example 4.2.5. We start with notation that helps us detect and remove impossible labels.

Definition 4.3.1. Let $X \subseteq \Sigma^d / \sim_\sigma$ and $Y \subseteq \Sigma^\delta / \sim_\sigma$ for some set Σ. We denote

$$\Sigma_X = \bigcup_{x \in X} \{x_1, \ldots, x_d\}$$

and

$$Y_X = Y \cap \Sigma_X^\delta / \sim_\sigma .$$

When we derive the alphabet of a problem (A, P) implicitly, we use Σ_A and Σ_P symbolically as above, but without referring to a particular set Σ.

Removal of labels that do not appear on the active side is trivial.

Lemma 4.3.2. *Let* $(A, P) \in \mathrm{LCL}(d, \delta)$. *Now* $1_{\underline{A}} \colon (A, P) \cong (A, P_A)$.

Proof. $1_{\underline{A}}$ induces the relation $=_{\Sigma_A}$ defined by

$$l =_{\Sigma_A} l' \iff l = l' \in \Sigma_A,$$

with which the pairs (P, P_A) and (P_A, P) are compatible. □

We introduce two notions of having impossible labels removed.

Definition 4.3.3. Let $\Pi = (A, P) \in \mathrm{LCL}(d, \delta)$. We say that Π **has no impossible active labels** if all labels in Σ_A appear in P_A, i.e.

$$\Sigma_A \subseteq \Sigma_{P_A}.$$

Definition 4.3.4. Let $\Pi = (A, P) \in \mathrm{LCL}(d, \delta)$. We say that Π **has no impossible labels** if the same set of labels appears in the active and passive configurations, i.e.

$$\Sigma_A = \Sigma_P.$$

We denote the full subcategory of $\mathrm{LCL}(d, \delta)$ induced by such problems by $\mathrm{LCL}^*(d, \delta)$.

The intuition behind Definition 4.3.3 is that we want no labels that appear in active configurations but cannot appear in any solution. We do not mind if a label appears in a passive configuration but not any active one, since it does not affect the correctness-preserving conditions. However, we have to restrict the check to P_A, since otherwise we can get a chain of active configurations that are impossible, because they have a label that only appears in impossible passive configurations, and passive configurations that that are impossible, because they have a label that only appears in impossible active configurations, as illustrated by the following example.

Example 4.3.5. Consider the problem

$$\Pi = (\{[1,1],[1,2],[3,4]\},\{[1,1],[2,3],[4,5]\}).$$

- The passive configuration $[4,5]$ cannot appear in a solution since there are no active configurations with the label 5.
- The active configuration $[3,4]$ can never appear in a solution since there are no passive configurations with label 4 that appear in a solution.
- The passive configuration $[2,3]$ cannot appear in a solution since there are no active configurations with the label 3 that appear in a solution.
- The active configuration $[1,2]$ can never appear in a solution since there are no passive configurations with label 2 that appear in a solution.

In conclusion, a solution of Π can only have configurations $[1,1]$.

Moving between the two notions is easy, since the latter implies the former, and and for the other direction the remaining impossible passive labels are removed when Lemma 4.3.2 is applied.

Lemma 4.3.6. *Let $\Pi = (A,P) \in \mathrm{LCL}(d,\delta)$ have no impossible labels. Now $P = P_A$, and Π has no impossible active labels.*

Proof. We have

$$P_A = P \cap \Sigma_A^\delta / \sim_\sigma = P \cap \Sigma_P^\delta / \sim_\sigma = P,$$

and therefore $\Sigma_A = \Sigma_P = \Sigma_{P_A}$. □

Lemma 4.3.7. *Let $(A,P) \in \mathrm{LCL}(d,\delta)$ have no impossible active labels. Now (A,P_A) has no impossible labels.*

Proof. We have

$$P_A = P \cap \Sigma_A^\delta / \sim_\sigma \subseteq \Sigma_A^\delta / \sim_\sigma,$$

and therefore $\Sigma_A \subseteq \Sigma_{P_A} \subseteq \Sigma_A$. □

The following theorem shows us that removing the impossible active labels is enough to make GCP-maps LCP.

Theorem 4.3.8. *Let $\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}(d,\delta)$. If Π has no impossible active labels, we have $\mathrm{LCP}(\Pi,\Pi') = \mathrm{GCP}(\Pi,\Pi')$.*

Proof. As $\mathrm{LCP}(\Pi, \Pi') \subseteq \mathrm{GCP}(\Pi, \Pi')$, it is enough to show $\mathrm{GCP}(\Pi, \Pi') \subseteq \mathrm{LCP}(\Pi, \Pi')$. Assume that Π has no impossible active labels and let $\varphi \in \mathrm{GCP}(\Pi, \Pi')$. Let

$$\left(a_{i_1}^{(1)}, \ldots, a_{i_\delta}^{(\delta)}\right) \in \underline{P}$$

for some $a^{(1)}, \ldots, a^{(\delta)} \in \underline{A}$ and $i_1, \ldots, i_\delta \in \{1, \ldots, d\}$. We want to show

$$\left(\varphi\left(a^{(1)}\right)_{i_1}, \ldots, \varphi\left(a^{(\delta)}\right)_{i_\delta}\right) \in \underline{P'}$$

by constructing a solution to Π with a neighbourhood corresponding to $\left(a_{i_1}^{(1)}, \ldots, a_{i_\delta}^{(\delta)}\right)$.

Consider a network N consisting of a passive node w, active nodes $v_1, \ldots, v_\delta$ connected to its ports $1, \ldots, \delta$ through their ports $i_1, \ldots, i_\delta$ respectively, and alternating layers of passive and active nodes that are connected to the previous layer through their port 1 so that the underlying graph forms a biregular tree, that is infinite if $d \neq 1 \neq \delta$. Denote the set of active nodes by V_A and passive nodes by V_P. We construct an output $f\colon V_A \to \underline{A}$ and determine its corresponding $h_f\colon V_P \to \Sigma^\delta$ recursively as follows. For $v_1, \ldots, v_\delta$ let $f(v_i) = a^{(i)}$. We get

$$h_f = \left(a_{i_1}^{(1)}, \ldots, a_{i_\delta}^{(\delta)}\right) \in \underline{P}.$$

Now for $k \in \mathbb{Z}_+$, the nodes at distance $2k$ from w are passive and nodes from distance $2k+1$ are active. Each node w' at distance $2k$ sees at their port 1 an output $l \in \Sigma_A$ given by an active node at distance $2k-1$. As Π has no impossible active labels, l appears in P_A, i.e. there is some $p \in \underline{P_A}$ s.t. $p_1 = l$. Since $p_2, \ldots, p_\delta \in \Sigma_A$, we can select for each of the ports $i \in \{2, \ldots, \delta\}$ of w' some $a \in \underline{A}$ with $a_1 = p_i$, and set $f(v') = a$ for the node v' in the port i. This defines f for the layer at distance $2k+1$ and gives $h_f(w') = p \in \underline{P}$.

As f gets defined for every active layer and $h_f(w') \in \underline{P}$ for every passive node w', we have $f \in \Pi(N)$. Now by the GCP-rule $\varphi \circ f \in \Pi'(N)$, giving us

$$\left(\varphi\left(a^{(1)}\right)_{i_1}, \ldots, \varphi\left(a^{(\delta)}\right)_{i_\delta}\right) = h_{\varphi \circ f}(w) \in \underline{P'}.$$

In conclusion, the LCP-rule holds for φ, and thus $\mathrm{GCP}(\Pi, \Pi') = \mathrm{LCP}(\Pi, \Pi')$. □

We have now seen, that when there are no impossible active labels, GCP-maps are just LCP-maps that are easier to work with since the conditions are local. Although removing the impossible active labels is enough, we can conveniently get rid of all impossible labels using the following simple algorithm that does so safely.

Definition 4.3.9. Let $\Pi \in \mathrm{LCL}(d, \delta)$. We define $\mathrm{Ir}(\Pi) \in \mathrm{LCL}(d, \delta)$ as the output of of the following algorithm:

- Let $\Pi^{(0)} = (A^{(0)}, P^{(0)}) = \Pi$
- For $i \in \mathbb{Z}_+$, let $\Pi^{(i)} = (A^{(i)}, P^{(i)})$, where $P^{(i)} = P_{A^{(i-1)}}$ and $A^{(i)} = A_{P^{(i)}}$.
- If $\Pi^{(i)} = \Pi^{(i-1)}$, return $\mathrm{Ir}(\Pi) = \Pi^{(i)}$

The following lemma shows that Ir is well defined, and the subsequent theorem that it does what we are expecting.

Lemma 4.3.10. *The algorithm terminates, and for $i \in \mathbb{Z}_+$ we have $P^{(i)} = P^{(i-1)}_{A^{(i-1)}}$ and $A^{(i)} = A^{(i-1)}_{P^{(i)}}$.*

Proof. We have

$$P^{(1)} = P_{A^{(0)}} = P^{(0)}_{A^{(0)}}.$$

For $i \in \mathbb{Z}_+$, assume that we have

$$P^{(i)} = P^{(i-1)}_{A^{(i-1)}}.$$

Now $P^{(i)} \subseteq P^{(i-1)}$, so

$$A^{(i)} = A_{P^{(i)}} \subseteq A_{P^{(i-1)}} = A^{(i-1)},$$

giving us

$$A^{(i)} = A^{(i-1)} \cap A_{P^{(i)}} = A^{(i-1)}_{P^{(i)}}.$$

Furthermore, now $A^{(i)} \subseteq A^{(i-1)}$ so

$$P^{(i+1)} = P_{A^{(i)}} \subseteq P_{A^{(i-1)}} = P^{(i)},$$

giving us

$$P^{(i+1)} = P^{(i)} \cap P_{A^{(i)}} = P^{(i)}_{A^{(i)}}.$$

Hence by induction $P^{(i)} = P^{(i-1)}_{A^{(i-1)}}$ and $A^{(i)} = A^{(i-1)}_{P^{(i)}}$ for all $i \in \mathbb{Z}_+$. Furthermore, since $A^{(i)} \subseteq A^{(i-1)}$ and $P^{(i)} \subseteq P^{(i-1)}$, on each iteration $A^{(i)}$ or $P^{(i)}$ gets strictly smaller, and as A and P are finite, the algorithm terminates. □

Theorem 4.3.11. *Let $\Pi = (A, P) \in \mathrm{LCL}(d, \delta)$. Now $\mathrm{Ir}(\Pi) \in \mathrm{LCL}^*(d, \delta)$ and $\mathrm{Ir}(\Pi) \subseteq (A, P_A) \subseteq \Pi$. If $\mathrm{Ir}(\Pi) \neq \Pi_\emptyset$, then there is $\varphi \in \mathrm{GCP}(\Pi, \mathrm{Ir}(\Pi))$ s.t. $\varphi|_{\mathrm{Ir}(\Pi)} = 1_{\mathrm{Ir}(\Pi)}$. If $\mathrm{Ir}(\Pi) = \Pi_\emptyset$, Π is unsolvable over all non-empty networks.*

Proof. For each $i \in \mathbb{Z}_+$ we have $A^{(i)} \subseteq A^{(i-1)}$ and $P^{(i)} \subseteq P^{(i-1)}$, so $\Pi^{(i)} \subseteq \Pi^{(i-1)}$. Now by induction

$$\mathrm{Ir}(\Pi) \subseteq \Pi^{(1)} \subseteq \left(A^{(0)}, P^{(1)}\right) = (A, P_A) \subseteq \Pi.$$

Observe also, that $A^{(i)} = A_{P^{(i)}} \subseteq \Sigma^d_{P^{(i)}}/\sim_\sigma$ and $P^{(i)} = P_{A^{(i-1)}} \subseteq \Sigma^\delta_{A^{(i-1)}}/\sim_\sigma$, so therefore $\Sigma_{A^{(i)}} \subseteq \Sigma_{P^{(i)}} \subseteq \Sigma_{A^{(i-1)}}$. For some $j \in \mathbb{Z}_+$ we have we have $\mathrm{Ir}(\Pi) = \Pi^{(j)} = \Pi^{(j-1)}$, giving us $\Sigma_{A^{(j)}} = \Sigma_{P^{(j)}}$ by $\Sigma_{A^{(j-1)}} = \Sigma_{A^{(j)}}$, and thus $\mathrm{Ir}(\Pi) \in \mathrm{LCL}^*(d, \delta)$.

Assume that $A^{(i)} \neq \emptyset$. Now there is some $a^{(i)} \in \underline{A^{(i)}}$. Define

$$\psi_i \colon \underline{A^{(i-1)}} \to \underline{A^{(i)}} \colon a \mapsto \begin{cases} a, & a \in \underline{A^{(i)}}, \\ a^{(i)}, & a \notin \underline{A^{(i)}}. \end{cases}$$

Clearly $\psi_i|_{\underline{A^{(i)}}} = 1_{\Pi^{(i)}}$. Furthermore, if

$$a \in \underline{A^{(i-1)}} \setminus \underline{A^{(i)}} = \underline{A^{(i-1)}} \setminus \Sigma^\delta_{P^{(i)}},$$

at least one of the labels in a does not appear in $P^{(i)}$, and therefore a does not appear in a valid solution to $(A^{(i-1)}, P^{(i)})$, allowing us to map a freely to $a^{(i)}$ without breaking the GCP-rule. Hence $\psi_i \in \mathrm{GCP}\left((A^{(i-1)}, P^{(i)}), \Pi^{(i)}\right)$ and by Lemma 4.3.2 with $P^{(i)} = P_{A^{(i-1)}} = (P^{(i-1)})_{A^{(i-1)}}$ we have $1_{\underline{A^{(i-1)}}} \in \mathrm{LCP}\left(\Pi^{(i-1)}, (A^{(i-1)}, P^{(i)})\right)$, giving us $\psi_i \in \mathrm{GCP}\left(\Pi^{(i-1)}, \Pi^{(i)}\right)$. Now define $\varphi_i = \psi_i \circ \cdots \circ \psi_1 \in \mathrm{GCP}\left(\Pi, \Pi^{(i)}\right)$. We have $\varphi_1|_{\Pi^{(1)}} = \psi_1|_{\underline{A^{(1)}}} = 1_{\Pi^{(1)}}$, and if $\varphi_{i-1}|_{\Pi^{(i-1)}} = 1_{\Pi^{(i-1)}}$, we have $\varphi_i|_{\Pi^{(i)}} = \psi_i \circ \varphi_{i-1}|_{\Pi^{(i)}} = \psi_i \circ 1_{\Pi^{(i-1)}}|_{\Pi^{(i)}} = \psi_i|_{\underline{A^{(i)}}} = 1_{\Pi^{(i)}}$, so by induction $\varphi_i|_{\Pi^{(i)}} = 1_{\Pi^{(i)}}$. In conclusion, if $A^{(j)} \neq \emptyset$, we have $\varphi = \varphi_j \in \mathrm{GCP}(\Pi, \mathrm{Ir}(\Pi))$ and $\varphi|_{\mathrm{Ir}(\Pi)} = 1_{\mathrm{Ir}(\Pi)}$.

Assume now that there is $k \in \mathbb{Z}_{\geq 0}$ s.t. $A^{(k)} = \emptyset$. Now $P_{A^{(k)}} = \emptyset$, and therefore $\mathrm{Ir}(\Pi) = \Pi^{(k+1)} = \Pi_\emptyset$. Hence if $\mathrm{Ir}(\Pi) \neq \Pi_\emptyset$, we have $A^{(j)} \neq \emptyset$ and we are covered by the previous case. If $A^{(0)} = \emptyset$, $\Pi = (A^{(0)}, P^{(0)})$ has no solutions over non-empty networks. On the other hand if $A^{(0)} \neq \emptyset$, let k be the first index of an empty $A^{(i)}$, i.e. $A^{(k-1)} \neq \emptyset$. Now

$$A^{(k-1)} \cap \Sigma^\delta_{P^{(k)}}/\sim_\sigma = A^{(k)} = \emptyset,$$

and thus every $a \in \underline{A^{(k-1)}}$ has a label that does not appear in $P^{(k)}$ so a does not appear in any solution to $(A^{(k-1)}, P^{(k)})$. Hence $(A^{(k-1)}, P^{(k)})$ is unsolvable over all non-empty networks, and since by Lemma 4.3.2 we have $(A^{(k-1)}, P^{(k)}) \cong \Pi^{(k-1)}$ and there is $\varphi^{(k-1)} \in \mathrm{GCP}\left(\Pi, \Pi^{(k-1)}\right)$, Π is also unsolvable over all non-empty networks. □

Furthermore, Ir can be used to check whether a problem is in $\mathrm{LCL}^*(d, \delta)$.

Lemma 4.3.12. *Let $\Pi = (A, P) \subseteq \Pi' = (A', P') \in \mathrm{LCL}(d, \delta)$. If Π has no impossible active labels, then $(A, P_A) \subseteq \mathrm{Ir}(\Pi')$.*

Proof. For $i \in \mathbb{Z}_{\geq 0}$ let $\Pi'^{(i)} = (A'^{(i)}, P'^{(i)})$ as in Definition 4.3.9, and $(A'', P'') = \mathrm{Ir}(\Pi')$. If $A \subseteq A''$, we have $P_A \subseteq P_{A''} \subseteq P'_{A''} = P''$, giving us $(A, P_A) \subseteq \mathrm{Ir}(\Pi)$. Hence it is enough to show $A \subseteq A''$.

For a contradiction, assume $A \not\subseteq A''$. Now there is some $k \in \mathbb{Z}_{\geq 0}$ for which $A \subseteq A'^{(k)}$ but $A \not\subseteq A'^{(k+1)}$. In particular, there is some $a \in \underline{A}$ s.t. $a \notin \underline{A'^{(k+1)}}$. Therefore, as $\underline{A'^{(k+1)}} = \underline{A'^{(k)}} \cap \Sigma^d_{P'_{A'^{(k)}}} \supseteq \underline{A'^{(k)}} \cap \Sigma^d_{P_A}$, we have $a \notin \Sigma^d_{P_A}$, giving us $\Sigma_A \not\subseteq \Sigma_{P_A}$, which contradicts Π having no impossible active labels. In conclusion, $A \subseteq A''$ and thus $(A, P_A) \subseteq \mathrm{Ir}(\Pi')$. □

Theorem 4.3.13. *Let* $\Pi = (A, P) \in \mathrm{LCL}(d, \delta)$. *Now* $\Pi \in \mathrm{LCL}^*(d, \delta)$ *if and only if* $\Pi = \mathrm{Ir}(\Pi)$.

Proof. "⇐": $\Pi = \mathrm{Ir}(\Pi) \in \mathrm{LCL}^*(d, \delta)$ by Theorem 4.3.11.

"⇒": By Lemmas 4.3.6 and 4.3.12 and Theorem 4.3.11, we have

$$\Pi = (A, P_A) \subseteq \mathrm{Ir}(\Pi) \subseteq \Pi.$$ □

The following lemma will allow us to define Ir as a functor.

Lemma 4.3.14. *Let* $\Pi = (A, P), \Pi' = (A', P') \in \mathrm{LCL}(d, \delta)$ *and* $\varphi \in \mathrm{GCP}(\Pi, \Pi')$. *Now* $\varphi|_{\mathrm{Ir}(\Pi)} \in \mathrm{LCP}(\mathrm{Ir}(\Pi), \mathrm{Ir}(\Pi'))$.

Proof. As $\mathrm{Ir}(\Pi) \subseteq \Pi$, we have $\varphi|_{\mathrm{Ir}(\Pi)} \in \mathrm{GCP}(\mathrm{Ir}(\Pi), \Pi')$. Let $\mathrm{Ir}(\Pi) = (A'', P'')$ and $\mathrm{Ir}(\Pi') = (A''', P''')$. We want to show $\varphi(A'') \subseteq A'''$ through Lemma 4.3.12 by showing that $(\varphi(A''), P')$ has no impossible active labels, after which we can use Theorem 4.3.11 to restrict the codomain of $\varphi|_{\mathrm{Ir}(\Pi)}$. Let $a \in \underline{A''}$. We show that $\varphi(a)$ has no impossible active labels by constructing a solution to $\mathrm{Ir}(A)$ around a.

Consider a network N consisting of a central active node v, and alternating layers of passive and active nodes that are connected to the previous layer through their port 1 so that the underlying graph forms a (d, δ)-biregular tree, that is infinite if $d \neq 1 \neq \delta$. Denote the set of active nodes by V_A and passive nodes by V_P. We construct an output $f\colon V_A \to \underline{A''}$ and determine its corresponding $h_f\colon V_P \to \Sigma''^{\delta}$ recursively as follows.

Let $f(v) = a$. For $k \in \mathbb{Z}_+$, the nodes at distance $2k-1$ from v are passive and nodes at distance $2k$ are active. Each passive node w' at distance $2k-1$ sees at their port 1 an output $l \in \Sigma_{A''} = \Sigma_{P''}$ given by an active node at distance $2k-2$. There is now some $p \in \underline{P''}$ s.t. $p_1 = l$. Since $p_2, \ldots, p_\delta \in \Sigma_{P''} = \Sigma_{A''}$, we can select for each of the ports $i \in \{2, \ldots, \delta\}$ of w' some $a' \in \underline{A''}$ with $a'_1 = p_i$, and set $f(v') = a'$ for the active node v' in the port i. This defines f for the layer at distance $2k$ and gives $h_f(w') = p \in \underline{P''}$.

As f gets defined for every active layer and $h_f(w') \in \underline{P''}$ for every passive node w', we have $f \in \mathrm{Ir}(\Pi)(N)$. Now by the GCP-rule $\varphi|_{A''} \circ f \in \Pi'(N)$, giving us that for the neighbours w of v we have $h_{\varphi|_{A''} \circ f}(w) \in \underline{P'}$, and as the outputs of their neighbours are in the image of $\varphi|_{A''}$, we have $h_{\varphi|_{A''} \circ f}(w) \in \Sigma^{\delta}_{\varphi(A'')}$. In conclusion, $h_{\varphi|_{A''} \circ f}(w) \in \underline{P'_{\varphi(A'')}}$ and thus $\varphi(a) \in \Sigma^d_{P'_{\varphi(A'')}}$,

When we apply this to all $a \in A''$, we get $\Sigma_{\varphi(A'')} \subseteq \Sigma_{P'_{\varphi(A'')}}$, i.e. that $(\varphi(A''), P')$ has no impossible active labels. Now $\varphi(A'') \subseteq A'''$ by Lemma 4.3.12. If $A'' \neq \emptyset$, we have by Theorem 4.3.11 some $\psi \in \mathrm{GCP}(\Pi', \mathrm{Ir}(\Pi'))$ for which $\psi|_{A'''} = 1_{A'''}$, and therefore

$$\psi \circ \varphi|_{A''} = \psi|_{\varphi(A'')} \circ \varphi|_{A''} = 1_{A'''} \circ \varphi|_{A''} = \varphi|_{A''} \in \mathrm{GCP}(\mathrm{Ir}(\Pi), \mathrm{Ir}(\Pi')).$$

Now as $\mathrm{Ir}(\Pi) \in \mathrm{LCL}^*(d, \delta)$, by Theorem 4.3.8 we get

$$\varphi|_{\mathrm{Ir}(\Pi)} = \varphi|_{A''} \in \mathrm{LCP}(\mathrm{Ir}(\Pi), \mathrm{Ir}(\Pi')).$$

On the other hand if $A'' = \emptyset$, then

$$\varphi|_{\mathrm{Ir}(\Pi)} = \varphi_\emptyset \in \mathrm{LCP}(\mathrm{Ir}(\Pi), \mathrm{Ir}(\Pi')). \qquad \square$$

The following is now well defined.

Definition 4.3.15. Let $\Pi, \Pi' \in \mathrm{LCL}_\forall(d, \delta)$ and $\varphi \in \mathrm{GCP}(\Pi, \Pi')$. We define $\mathrm{Ir}(\varphi) = \varphi|_{\mathrm{Ir}(\Pi)} \in \mathrm{LCP}(\mathrm{Ir}(\Pi), \mathrm{Ir}(\Pi'))$.

Theorem 4.3.16. Ir *is a functor* $\mathrm{LCL}_\forall(d, \delta) \to \mathrm{LCL}^*(d, \delta)$

Proof. Let $\Pi, \Pi', \Pi'' \in \mathrm{LCL}_\forall(d, \delta)$, $\varphi \in \mathrm{GCP}(\Pi, \Pi')$ and $\varphi' \in \mathrm{GCP}(\Pi', \Pi'')$. Now $\mathrm{Ir}(1_\Pi) = 1_\Pi|_{\mathrm{Ir}(\Pi)} = 1_{\mathrm{Ir}(\Pi)}$ and

$$\mathrm{Ir}(\varphi' \circ \varphi) = \varphi' \circ \varphi|_{\mathrm{Ir}(\Pi)} = \varphi'|_{\mathrm{Ir}(\Pi')} \circ \varphi|_{\mathrm{Ir}(\Pi)} = \mathrm{Ir}(\varphi') \circ \mathrm{Ir}(\varphi). \qquad \square$$

Corollary 4.3.16.1. Ir *is a functor* $\mathrm{LCL}(d, \delta) \to \mathrm{LCL}^*(d, \delta)$, $\mathrm{LCL}^*(d, \delta) \to \mathrm{LCL}^*(d, \delta)$ *and* $\mathrm{LCL}(d, \delta) \to \mathrm{LCL}(d, \delta)$.

Proof. $\mathrm{LCL}^*(d, \delta)$ is a subcategory of $\mathrm{LCL}(d, \delta)$, which is a subcategory of $\mathrm{LCL}_\forall(d, \delta)$. $\square$

In conclusion, Ir provides us a way to transform $\mathrm{LCL}_\forall(d, \delta)$, which is the most general setting we would want to work in without further restricting the graph family and having knowledge about its properties, into $\mathrm{LCL}^*(d, \delta)$, where the problems and morphisms are simpler to work with, while preserving the morphism structure and providing by Theorem 4.3.11 a way to lift the structure of $\mathrm{LCL}^*(d, \delta)$ back to $\mathrm{LCL}_\forall(d, \delta)$ apart from problems unsolvable over all non-empty networks. In the remaining text we will continue giving results in $\mathrm{LCL}(d, \delta)$, but for practical applications it will most likely make sense to restrict to $\mathrm{LCL}^*(d, \delta)$.

5 Round Elimination as a Functor

5.1 Goals

In this section our goal is to define round elimination as a functor between categories. We start off by defining the object map that corresponds to the traditional round elimination, and checking that different simplification choices give equivalent output problems. Then we define how round elimination works on morphisms by defining a map on LCP-maps, which is regrettably not strong enough to give a functor $\mathrm{LCL}(d,\delta) \to \mathrm{LCL}(\delta,d)$, but can be thinned to a functor $\mathrm{TLCL}(d,\delta) \to \mathrm{TLCL}(\delta,d)$. Finally, we prove the result that the maximum period of periodic points of round elimination is 2.

We will define round elimination for the whole $\mathrm{LCL}(d,\delta)$. However, in order to avoid some artefacts that would arise from impossible passive labels, we define it so that round elimination on $(A,P) \in \mathrm{LCL}(d,\delta)$ corresponds to doing traditional round elimination according to Definition 2.2.15 on the problem $(A,P_A) \cong (A,P)$. Furthermore, as we want round elimination to restrict to $\mathrm{LCL}^*(d,\delta) \to \mathrm{LCL}^*(\delta,d)$, we apply the same isomorphism to the output in order to remove the impossible passive labels that could otherwise appear. If $(A,P) \in \mathrm{LCL}^*(d,\delta)$, we have $(A,P_A) = (A,P)$ by Lemma 4.3.6, and thus over $\mathrm{LCL}^*(d,\delta)$ the definition matches the traditional one with the added step of removing labels that do not appear on the active side.

Definition 5.1.1. For $(A,P) \in \mathrm{LCL}(d,\delta)$, we call Σ_A the **restricted alphabet** and P_A the **restricted passive configurations**.

Definition 5.1.2. We define the object map $\mathrm{Re}\colon \mathrm{LCL}(d,\delta) \to \mathrm{LCL}(\delta,d)$ to be

$$(\Sigma, A, P) \mapsto (S, B, Q_B),$$

where S, B and Q are defined as follows:

- The new alphabet is the set of non-empty subsets of the restricted alphabet:
$$S = 2^{\Sigma_A} \setminus \{\emptyset\}.$$
- The new active configurations are formed by subsets where taking any element from each subset gives a tuple corresponding to an old (restricted) passive configuration:
$$\underline{B} = \left\{ b \in S^\delta \mid b_1 \times \cdots \times b_\delta \subseteq \underline{P} \right\}.$$
- The new passive configurations are formed by subsets where an element can be taken from each subset, so that the resulting tuple corresponds to an old active configuration:
$$\underline{Q} = \left\{ q \in S^d \mid q_1 \times \cdots \times q_d \cap \underline{A} \neq \emptyset \right\}.$$

Observe, that when defining S we use Σ_A, which only has labels present in (A, P_A). In particular, this implies that $b_1 \times \cdots \times b_\delta$ in the definition of $\underline{B}$ cannot contain a value in $\underline{P} \setminus \underline{P_A}$. Hence $(A, P_A) \mapsto (B, Q)$ corresponds to the traditional definition, and $\mathrm{Re}\left((A, P)\right) = \mathrm{Re}\left((A, P_A)\right) \cong (B, Q)$.

In practice, we are usually interested in problems that have impossible labels removed. For example, the Round Eliminator tool [5] requires this for inputs, and performing round elimination with the maximality simplification of Definition 2.2.16 on such problem removes them from the output. Hence applying the isomorphisms $(A, P) \cong (A, P_A)$ and $(B, Q) \cong (B, Q_B)$ as pre- and postprocessing steps is reasonable. Furthermore, following example illustrates how the preprocessing step is necessary for functoriality of $\mathrm{Re}\colon \mathrm{TLCL}\,(d, \delta) \to \mathrm{TLCL}\,(\delta, d)$.

Example 5.1.3. Consider a $(1, 1)$-biregular problem

$$\Pi = (\Sigma, A, P) = (\{\bullet\}, \emptyset, \{[\bullet]\}),$$

which is isomorphic to $(\Sigma_A, A, P_A) = \Pi_\emptyset$. Using Definition 5.1.2 we get $\mathrm{Re}\,(\Pi) = \Pi_\emptyset$. However, Definition 2.2.15 would give us $\Pi' = (\{\{\bullet\}\}, \{[\{\bullet\}]\}, \emptyset)$ as output for Π, and $\Pi_\emptyset$ as output for $\Pi_\emptyset$. As there is no functions $\{(\{\bullet\})\} \to \emptyset$, there can be no LCP-maps $\Pi' \to \Pi_\emptyset$, and thus no morphism in the thin category either. In other words, the traditional round elimination makes impossible labels appear in active configurations, and these configurations that would never appear in a valid solution would have to be safely mapped somewhere, making existence of LCP-maps less likely. Furthermore, the traditional round elimination would map isomorphic problems to non-equivalent ones, and therefore it would be impossible to make it a functor, even over the thin categories.

With the postprocessing step, round elimination indeed restricts to $\mathrm{LCL}^*\,(d, \delta) \to \mathrm{LCL}^*\,(\delta, d)$.

Theorem 5.1.4. *If* $\Pi \in \mathrm{LCL}^*\,(d, \delta)$*, then* $\mathrm{Re}(\Pi) \in \mathrm{LCL}^*\,(\delta, d)$*, i.e.*

$$\mathrm{Re}|_{\mathrm{LCL}^*(d,\delta)}\,(\Pi)\colon \mathrm{LCL}^*\,(d, \delta) \to \mathrm{LCL}^*\,(\delta, d).$$

Proof. Let $\Pi = (\Sigma, A, P) \in \mathrm{LCL}^*\,(d, \delta)$ and $(S, B, Q_B) = \mathrm{Re}(\Pi)$ as in Definition 5.1.2. Let $X \in S_B$. In particular $\emptyset \neq X \subseteq \Sigma_A$, so we can pick $x \in X$ and there exists some $a = (x, a_2, \ldots, a_d) \in \underline{A}$. As $\Sigma_A = \Sigma_P$, for each a_i there is some $p = (a_i, p_2, \ldots, p_\delta) \in \underline{P}$, and thus $\{a_i\} \times \{p_2\} \times \cdots \times \{p_\delta\} = \{p\} \subseteq \underline{P}$, giving us $(\{a_i\}, \{p_2\}, \ldots, \{p_\delta\}) \in \underline{B}$ and hence $\{a_i\} \in S_B$. Therefore as $X \times \{a_2\} \times \cdots \times \{a_d\} \supseteq \{a\} \subseteq \underline{A}$, we have $(X, \{a_2\}, \ldots, \{a_d\}) \in Q_B$ and thus $X \in S_{Q_B}$, giving us $S_B \subseteq S_{Q_B}$. In conclusion, (S, B, Q) has no impossible active labels and hence (S, B, Q_B) has no impossible labels by Lemma 4.3.7. □

In order to confirm that the postprocessing steps in Definition 2.2.16 would give an equivalent output, we generalise the maximality in Definition 2.2.16 to find maximality equivalences.

Lemma 5.1.5. *Let* $(\Sigma, A, P) \in \mathrm{LCL}\,(d, \delta)$ *with a preorder* $\leq \subseteq \Sigma^2$ *that is compatible with* (P, P)*. Let* $A' \subseteq A$ *s.t.*

$$\forall a \in \underline{A} : \exists a' \in \underline{A'} : a \leq^d a'.$$

Now $(A, P) \simeq (A', P)$ *by any* $\varphi \colon \underline{A} \to \underline{A'}$ *that maps* $a \mapsto a'$ *for each* a*.*

Proof. The induced relation $\sim_\varphi$ is a subrelation of $\leq$, and therefore by Lemma 3.2.5 compatible with (P, P). Hence φ is LCP by Lemma 3.2.4. The converse of the equivalence is given by $(A', P) \subseteq (A, P)$. □

Notice, that if $\leq$ is a partial order, A' is valid if and only if it contains all maximal elements of A w.r.t. $\leq^d$. In particular, if the alphabet consists of sets, $\leq$ can be chosen to be $\subseteq$. Thus if $\mathrm{Re}(\Pi) = (S, B, Q_B)$, we can choose B' to be the maximal elements of B with respect to inclusion, and combined with Lemma 4.3.2 get the equivalence $\mathrm{Re}(\Pi) \cong (S, B, Q) \simeq (S, B', Q) \cong (S_{B'}, B', Q_{B'})$, where $(S_{B'}, B', Q_{B'})$ corresponds to the simplification in Definition 2.2.16.

5.2 Functoriality

In order to extend Re to a functor $\mathrm{TLCL}\,(d, \delta) \to \mathrm{TLCL}\,(\delta, d)$, we have to show that if there exists an LCP-map $\Pi \to \Pi'$, then there also exists one $\mathrm{Re}\,(\Pi) \to \mathrm{Re}\,(\Pi')$. We do this by constructing a function $\mathrm{LCP}\,(\Pi, \Pi') \to \mathrm{LCP}\,(\mathrm{Re}(\Pi), \mathrm{Re}(\Pi'))$. To that end, we define some auxiliary functions.

Definition 5.2.1. Let $(\Sigma, A, P) \in \mathrm{LCL}\,(d, \delta)$. For $i \in \{1, \ldots, d\}$, we define the coordinate projections

$$\pi_i \colon \underline{A} \to \Sigma_A \colon a \mapsto a_i.$$

Notice, that π_i is surjective for each i, since a label in Σ_A has to appear in some configuration of A and $\underline{A}$ contains all permutations of each configuration. In particular, this gives us that each non-empty subset of Σ_A has a non-empty preimage in $\underline{A}$, enabling the following definition.

Definition 5.2.2. We define by preimages of π_i

$$\tau_i \colon S \to 2^{\underline{A}} \setminus \{\emptyset\} : s \mapsto \pi_i^{-1}(s).$$

We can also extend other functions to maps of non-empty subsets instead of individual elements.

Definition 5.2.3. Consider $\varphi \in \mathrm{LCP}(\Pi, \Pi')$ and π_i. By abuse of notation, we write

$$\varphi\colon 2^{\underline{A}} \setminus \{\emptyset\} \to 2^{\underline{A'}} \setminus \{\emptyset\} : X \mapsto \{\varphi(a) \mid a \in X\}$$

and

$$\pi_i\colon 2^{\underline{A}} \setminus \{\emptyset\} \to S\colon X \mapsto \{\pi_i(a) \mid a \in X\}.$$

Remark. We have $\pi_i \circ \tau_i = 1_S$ by surjectivity of π_i, since

$$l \in s \in S \iff \exists a \in \tau_i(s) : \pi_i(a) = l \iff l \in \pi_i \circ \tau_i(s).$$

However, $\tau_i \circ \pi_i$ is not necessarily $1_{2^{\underline{A}} \setminus \{\emptyset\}}$. For example, if $A = \{[1,1], [1,2]\}$, then

$$\tau_1 \circ \pi_1(\{(1,1)\}) = \tau_1(\{1\}) = \{(1,1),(1,2)\}.$$

This is one obstacle for Re being a functor $\mathrm{LCL}(d,\delta) \to \mathrm{LCL}(\delta,d)$.

We combine the auxiliary functions to construct the LCP-map.

Definition 5.2.4. Let $\Pi, \Pi' \in \mathrm{LCL}(d,\delta)$, $\varphi \in \mathrm{LCP}(\Pi,\Pi')$, τ_i over Π and π'_i over Π'. We define

$$\alpha_i = \pi'_i \circ \varphi \circ \tau_i\colon S \to S',$$

$$\alpha\colon S \to S'\colon s \mapsto \bigcup_{i=1}^{d} \alpha_i(s),$$

and finally

$$\psi\colon \underline{B} \to S'^{\delta}\colon b \mapsto (\alpha(b_1), \ldots, \alpha(b_\delta)).$$

Theorem 5.2.5. *Let* $\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}(d,\delta)$, $\varphi \in \mathrm{LCP}(\Pi,\Pi')$, $\mathrm{Re}(\Pi) = (S, B, Q_B)$, $\mathrm{Re}(\Pi') = (S', B', Q'_{B'})$, *and* $\psi\colon \underline{B} \to S'^{\delta}$ *as in Definition 5.2.4. Then*

$$\psi \in \mathrm{LCP}(\mathrm{Re}(\Pi), \mathrm{Re}(\Pi')).$$

Proof. We need to show that $\psi(\underline{B}) \subseteq \underline{B'}$ and that ψ is LCP.

Let $b \in \underline{B}$. We want to show that $\alpha(b_1) \times \cdots \times \alpha(b_\delta) \subseteq \underline{P'}$. Let $p' \in \alpha(b_1) \times \cdots \times \alpha(b_\delta)$. For each $j \in \{1, \ldots, \delta\}$, we have $p'_j \in \alpha(b_j)$, and in particular $p'_j \in \alpha_i(b_j)$ for some $i \in \{1, \ldots, d\}$. Thus there exists some $l_j \in b_j$ and $a^{(j)} \in \underline{A}$, for which $\varphi(a^{(j)})_i = p'_j$ and $a_i^{(j)} = l_j$. Hence as $(l_1, \ldots, l_\delta) \in b_1 \times \cdots \times b_\delta \subseteq \underline{P}$, we have by the LCP-rule of φ that $p' \in \underline{P'}$. In conclusion, $\alpha(b_1) \times \cdots \times \alpha(b_\delta) \subseteq \underline{P'}$ and thus $\psi(b) \in \underline{B'}$, giving us that ψ is indeed $\underline{B} \to \underline{B'}$.

To show that ψ is LCP, we want to show that $(Q_B, Q'_{B'})$ is compatible with $\sim_\psi$. $\sim_\psi$ is a subrelation of $\sim_\alpha \subseteq S \times S'$, which in turn is defined by

$$s \sim_\alpha s' \iff s' = \alpha(s).$$

Let now $q \in \underline{Q_B}$ and $q \sim_\alpha^d q'$ for some $q' \in S'^d$. The relation forces $q' = (\alpha(q_1), \dots, \alpha(q_d))$. As $\psi(\underline{B}) \subseteq \underline{B'}$, we have $\alpha(S_B) \subseteq S'_{B'}$, and therefore $q' \in S'^d_{B'}$. Furthermore, we know that $q_1 \times \cdots \times q_d \cap \underline{A} \neq \emptyset$ and therefore there is some $a \in q_1 \times \cdots \times q_d \cap \underline{A}$. For each $i \in \{1, \dots, d\}$, we have $a_i \in q_i$, and hence $\varphi(a)_i \in \alpha(q_i)$, giving us $\varphi(a) \in q'_1 \times \cdots \times q'_d$, and therefore $q'_1 \times \cdots \times q'_d \cap \underline{A'} \neq \emptyset$, i.e. $q' \in \underline{Q'_{B'}}$. In conclusion, $(Q_B, Q'_{B'})$ is compatible with $\sim_\alpha$, and thus by Lemmas 3.2.5 and 3.2.4 ψ is LCP. □

Definition 5.2.6. Using Theorem 5.2.5, we define

$$\mathrm{Re}\colon \mathrm{LCP}\,(\Pi, \Pi') \to \mathrm{LCP}\,(\mathrm{Re}(\Pi), \mathrm{Re}(\Pi')) : \varphi \mapsto \psi.$$

The intuition behind the induced map can be seen through the simulation idea of round elimination. In $\mathrm{Re}(\Pi)$ the active nodes simulate possible configurations that their neighbours could have from A and collect the labels on their incident edges to form a configuration in B. On a simulation instance of Π, on each neighbour the configuration $a \in \underline{A}$ can be mapped to $\varphi(a) \in \underline{A'}$, and we get a simulation instance of Π', from which the labels can be collected to get a configuration in B'. ψ moves from sets of labels in $b \in \underline{B}$ to all instances that give those labels using the τ_i, then mapping those instances with φ and finally collecting the results with the corresponding π_i. Every choice of labels p' from a configuration $\psi(b)$ is in $\underline{P'}$, since it is the result of mapping an instance corresponding to a choice $p \in \underline{P}$ from b while respecting the LCP-rule. Conversely, ψ is LCP as the choice $a \in \underline{A}$ from $q \in \underline{Q_B}$ corresponds to an instance agreed upon by all neighbours, which is mapped to an instance corresponding to the choice $\varphi(a) \in \underline{A'}$ from $(\alpha(q_1), \dots, \alpha(q_d))$.

The existence of the morphism map is enough to induce a functor for the thin categories. That in turn shows that round elimination respects equivalence, which allows us to define it over the equivalence classes. The latter should be considered the ultimate definition as it is independent of any additional pre- or postprocessing equivalences, thus unifying the different traditional definitions.

Corollary 5.2.6.1. $\mathrm{Re}\colon \mathrm{TLCL}\,(d, \delta) \to \mathrm{TLCL}\,(\delta, d)$, *defined on objects through* $\mathrm{Re}\colon \mathrm{LCL}\,(d, \delta) \to \mathrm{LCL}\,(\delta, d)$ *and on morphisms by* $(\Pi \to \Pi') \mapsto (\mathrm{Re}(\Pi) \to \mathrm{Re}(\Pi'))$, *is a functor.*

Proof. The morphism map is well-defined, because the existence of an LCP-map $\varphi\colon \Pi \to \Pi'$ implies by Theorem 5.2.5 the existence of $\mathrm{Re}(\varphi)\colon \mathrm{Re}(\Pi) \to \mathrm{Re}(\Pi')$. Functoriality is trivial, as $\mathrm{TLCL}\,(\delta, d)$ is thin. □

Corollary 5.2.6.2. $\mathrm{Re} = \mathrm{Sk} \circ \mathrm{Re} \circ \iota\colon \mathrm{SkTLCL}\,(d, \delta) \to \mathrm{SkTLCL}\,(\delta, d)$, *defined as the composition of functors* $\iota\colon \mathrm{SkTLCL}\,(d, \delta) \subseteq \mathrm{TLCL}\,(d, \delta)$, $\mathrm{Re}\colon \mathrm{TLCL}\,(d, \delta) \to \mathrm{TLCL}\,(\delta, d)$ *and* $\mathrm{Sk}\colon \mathrm{TLCL}\,(\delta, d) \to \mathrm{SkTLCL}\,(\delta, d)$, *is a functor.*

Proof. The composition of functors is a functor. □

5.3 Short Periods

Next we provide an answer to the previously open question of whether there are periodic points of round elimination with period longer than two. We show that if a problem obtained by repeatedly applying round elimination is equivalent to the original problem, so is the one obtained after two iterations. Because of trouble that would arise from impossible labels, we will be working over $\mathrm{LCL}^*(d,\delta)$, and the key to the result is to observe that in $\mathrm{LCL}^*(d,\delta)$ an LCP-map $\Pi \to \mathrm{Re}^2(\Pi)$ always exists.

As round elimination moves from an alphabet to its power set, we will find the following isomorphism that wraps labels in singletons useful.

Definition 5.3.1. Let $\Pi = (\Sigma, A, P) \in \mathrm{LCL}(d,\delta)$. We denote by $s\colon \Sigma \to \{\{l\} \mid l \in \Sigma\}$ the bijection $l \mapsto \{l\}$, and by $s(\Pi)$ the problem $(s(A), s(P)) \cong \Pi$ and $s\colon \Pi \cong s(\Pi)$ the isomorphism given by Lemma 3.4.5.

Round elimination also switches the roles of active and passive nodes, so it will be useful to analyse problems where active and passive configurations are flipped.

Definition 5.3.2. Let $\Pi = (A, P) \in \mathrm{LCL}(d,\delta)$. We denote by $\mathrm{Refl}(\Pi)$ the **reflected problem** $(P, A) \in \mathrm{LCL}(\delta, d)$.

Our new tools have some convenient properties.

Lemma 5.3.3. *The operation* Refl *commutes with* s*, i.e. for all* $\Pi = (A,P) \in \mathrm{LCL}(d,\delta)$ *we have* $s\,\mathrm{Refl}(\Pi) = \mathrm{Refl}\,s(\Pi)$.

Proof. We have

$$s\,\mathrm{Refl}(\Pi) = s((P,A)) = (s(P), s(A)) = \mathrm{Refl}((s(A), s(P))) = \mathrm{Refl}\,s(\Pi). \qquad \square$$

Lemma 5.3.4. *The operations* s *and* Refl *preserve inclusions i.e. for all* $\Pi = (A,P) \subseteq \Pi' = (A',P') \in \mathrm{LCL}(d,\delta)$ *we have* $s(\Pi) \subseteq s(\Pi')$ *and* $\mathrm{Refl}(\Pi) \subseteq \mathrm{Refl}(\Pi')$.

Proof. We have $A \subseteq A'$ and $P \subseteq P'$. Therefore

$$s(A) = \{s(a) \mid a \in A\} \subseteq \{s(a) \mid a \in A'\} = s(A')$$

and similarly $s(P) \subseteq s(P')$, giving us $s(\Pi) \subseteq s(\Pi')$. On the other hand

$$\mathrm{Refl}(\Pi) = (P,A) \subseteq (P',A') = \mathrm{Refl}(\Pi'). \qquad \square$$

Lemma 5.3.5. *The operations* s *and* Refl *do not introduce impossible labels, i.e. for all* $\Pi \in \mathrm{LCL}^*(d,\delta)$ *we have* $s(\Pi) \in \mathrm{LCL}^*(d,\delta)$ *and* $\mathrm{Refl}(\Pi) \in \mathrm{LCL}^*(\delta,d)$.

Proof. Let $\Pi = (A, P) \in \mathrm{LCL}^*(d, \delta)$. Now $\Sigma_A = \Sigma_P$, so for $s(\Pi)$ we have $\Sigma_{s(A)} = s(\Sigma_A) = s(\Sigma_P) = \Sigma_{s(P)}$ and for $\mathrm{Refl}(\Pi) = (P, A)$ we have $\Sigma_P = \Sigma_A$. □

We will find the LCP-map $\Pi \to \mathrm{Re}^2(\Pi)$ in $\mathrm{LCL}^*(d, \delta)$ through an intermediate step in the reflected category.

Lemma 5.3.6. *Let $\Pi = (A, P) \in \mathrm{LCL}^*(d, \delta)$. We have $\mathrm{Refl}\, s(\Pi) \subseteq \mathrm{Re}(\Pi)$.*

Proof. Let $\mathrm{Re}(\Pi) = (S, B, Q_B)$. We have $\mathrm{Refl}\, s(\Pi) = (s(P), s(A))$. A configuration $b \in \underline{s(P)}$ is of the form $s(p)$ with $p \in \underline{P}$. Thus $b_1 \times \cdots \times b_\delta = \{p_1\} \times \cdots \times \{p_\delta\} = \{p\} \subseteq \underline{P}$, and as $p_1, \ldots, p_\delta \in \Sigma_P = \Sigma_A$, we have $b_1, \ldots, b_\delta \in S$. Therefore $b \in \underline{B}$, i.e. $s(P) \subseteq B$. Furthermore, a configuration $q \in \underline{s(A)}$ is of the form $s(a)$ with $a \in \underline{A}$. Thus $q_1 \times \cdots \times q_d = \{a_1\} \times \cdots \times \{a_d\} = \{a\} \subseteq \underline{A}$, and therefore $q \in \underline{Q}$, i.e. $s(A) \subseteq Q$. Hence by Lemma 3.4.6 we have $(s(P), s(A)) \subseteq (B, Q)$, and thus by Lemmas 5.3.5, 4.3.6, 4.3.12 and Theorem 4.3.11 we have

$$(s(P), s(A)) = \left(s(P), s(A)_{s(P)}\right) \subseteq \mathrm{Ir}\left((B, Q)\right) \subseteq (B, Q_B). \qquad \square$$

Lemma 5.3.7. *Let $\Pi = (A, P) \in \mathrm{LCL}^*(d, \delta)$. Now $s^2 \colon \Pi \to \mathrm{Re}^2(\Pi)$ is LCP.*

Proof. The map s^2 is an isomorphism $\Pi \to s^2(\Pi) = \mathrm{Refl}^2 s^2(\Pi) = (\mathrm{Refl}\, s)^2(\Pi)$. By Lemmas 5.3.4 and 5.3.6 we have $(\mathrm{Refl}\, s)^2(\Pi) \subseteq \mathrm{Refl}\, s\, \mathrm{Re}(\Pi)$, and now as $\mathrm{Re}\,\Pi \in \mathrm{LCL}^*(\delta, d)$ by Theorem 5.1.4, we can apply Lemma 5.3.6 again to get $\mathrm{Refl}\, s\, \mathrm{Re}(\Pi) \subseteq \mathrm{Re}^2(\Pi)$. Composing the isomorphism with these inclusions gives us the LCP-map $s^2 \colon \Pi \to \mathrm{Re}^2(\Pi)$ □

Whenever there are LCP-maps between different iterations of round elimination, we can use the morphism map of Re to extend them to cover a multiple of the iteration difference.

Lemma 5.3.8. *Let $\Pi \in \mathrm{LCL}(d, \delta)$. If there is an LCP-map $\varphi \colon \Pi \to \mathrm{Re}^n(\Pi)$ for some $n \in \mathbb{Z}_+$, then there are LCP-maps*

$$\mathrm{Re}^{(k-1)n}(\varphi) \circ \cdots \circ \mathrm{Re}^n(\varphi) \circ \varphi \colon \Pi \to \mathrm{Re}^{kn}(\Pi)$$

for all $k \in \mathbb{Z}_{\geq 0}$. Conversely, if there is an LCP-map $\psi \colon \mathrm{Re}^m(\Pi) \to \Pi$ for some $m \in \mathbb{Z}_+$, then there are LCP-maps

$$\psi \circ \mathrm{Re}^m(\psi) \circ \cdots \circ \mathrm{Re}^{(k-1)m}(\psi) \colon \mathrm{Re}^{km}(\Pi) \to \Pi$$

for all $k \in \mathbb{Z}_{\geq 0}$.

Proof. In the case $k = 0$ we have the LCP-map $1_\Pi \colon \Pi \to \Pi$ for both statements. Assume now that φ implies existence of an LCP-map

$$\varphi' = \mathrm{Re}^{(k-2)n}(\varphi) \circ \cdots \circ \mathrm{Re}^{n}(\varphi) \circ \varphi \colon \Pi \to \mathrm{Re}^{(k-1)n}(\Pi)$$

for some $k \in \mathbb{Z}_+$. Now $\mathrm{Re}^{(k-1)n}(\varphi)$ is an LCP-map $\mathrm{Re}^{(k-1)n}(\Pi) \to \mathrm{Re}^{kn}(\Pi)$, and we get an LCP-map $\mathrm{Re}^{(k-1)n}(\varphi) \circ \varphi' \colon \Pi \to \mathrm{Re}^{kn}(\Pi)$. Hence the first claim holds by induction. Similarly, if ψ implies

$$\psi' = \psi \circ \mathrm{Re}^{m}(\psi) \circ \cdots \circ \mathrm{Re}^{(k-2)m}(\psi) \colon \mathrm{Re}^{(k-1)m}(\Pi) \to \Pi,$$

we have

$$\mathrm{Re}^{(k-1)m}(\psi) \colon \mathrm{Re}^{km}(\Pi) \to \mathrm{Re}^{(k-1)m}(\Pi),$$

giving us

$$\psi' \circ \mathrm{Re}^{(k-1)m}(\psi) \colon \mathrm{Re}^{km}(\Pi) \to \Pi$$

and therefore by induction the second claim. □

We can apply Lemma 5.3.8 to s^2, or alternatively we could chain $s^2 \colon \mathrm{Re}^{n}(\Pi) \to \mathrm{Re}^{n+2}(\Pi)$ to get for example $s^4 \colon \Pi \to \mathrm{Re}^{2}(\Pi) \to \mathrm{Re}^{4}(\Pi)$. However, the next lemma shows that this is just a matter of intuition as both method give the same result.

Lemma 5.3.9. *Let $\Pi \in \mathrm{LCL}^*(d, \delta)$. Now the following morphisms are equal:*

$$\left(\mathrm{Re}(s^2) \colon \mathrm{Re}(\Pi) \to \mathrm{Re}^3(\Pi)\right) = \left(s^2 \colon \mathrm{Re}(\Pi) \to \mathrm{Re}^3(\Pi)\right).$$

Proof. Let $b \in \underline{B}$ and $k \in \{1, \ldots, \delta\}$. As s^2 is induced by a label map, it commutes with coordinate projection. Hence we have

$$\begin{aligned}
\mathrm{Re}(s^2)(b)_k &= \bigcup_{i=1}^{d} \pi_i' \circ s^2 \circ \tau_i(b_k) \\
&= \bigcup_{i=1}^{d} s^2 \circ \pi_i \circ \tau_i(b_k) \\
&= \bigcup_{i=1}^{d} s^2(b_k) = s^2(b)_k
\end{aligned}$$

In conclusion $\mathrm{Re}(s^2) = s^2$. □

Lemma 5.3.10. *Let $\Pi \in \mathrm{LCL}^*(d, \delta)$. There is an LCP-map $s^{2n} \colon \Pi \to \mathrm{Re}^{2n}(\Pi)$ for every $n \in \mathbb{Z}_{\geq 0}$.*

Proof. If we get $s^2 \colon \Pi \to \mathrm{Re}^2(\Pi)$ from Lemma 5.3.7, apply Lemma 5.3.8 to them to get the LCP-map $\Pi \to \mathrm{Re}^{2n}(\Pi)$, and simplify using Lemma 5.3.9, this map is the result. □

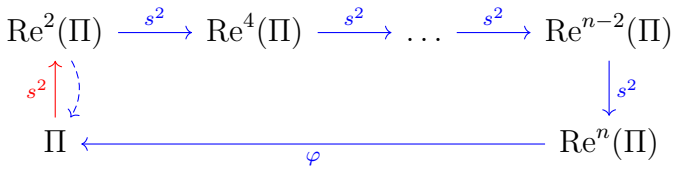


Figure 2: the even case

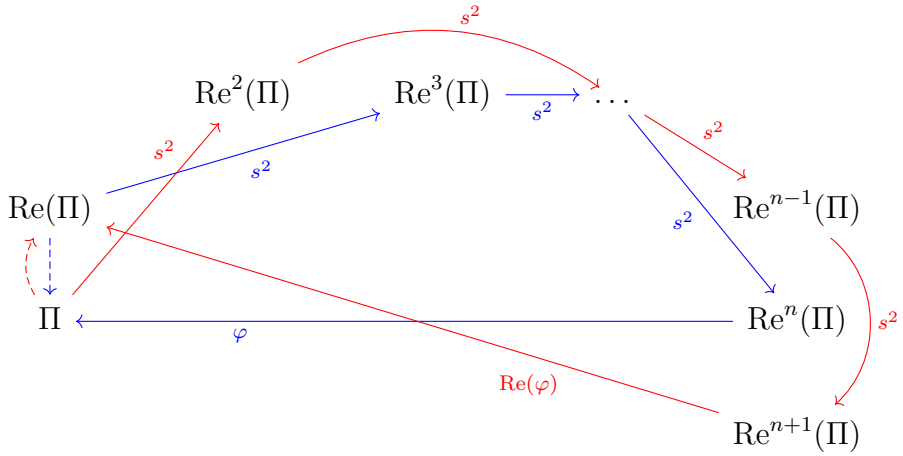


Figure 3: the odd case

We are now ready to prove our main theorem.

Theorem 5.3.11 (short period theorem)**.** *Let* $\Pi \in \mathrm{LCL}^*(d, \delta)$*. If there is an LCP-map* $\varphi\colon \mathrm{Re}^n(\Pi) \to \Pi$ *for some* $n \in \mathbb{Z}_+$*, then* $\mathrm{Re}^2(\Pi) \simeq \Pi$*, i.e.* Π *is a periodic point of period at most 2. In particular, if* n *is odd, then* $\mathrm{Re}(\Pi) \simeq \Pi$*, i.e.* Π *is a periodic point of period 1.*

Proof. Assume that n is even. The LCP-maps that give the equivalence in this case are illustrated in Figure 2. We have an LCP-map $s^2\colon \Pi \to \mathrm{Re}^2(\Pi)$ by Lemma 5.3.7. By Lemma 5.3.10 there is an LCP-map $s^{n-2}\colon \mathrm{Re}^2(\Pi) \to \mathrm{Re}^n(\Pi) = \mathrm{Re}^{n-2}\,\mathrm{Re}^2(\Pi)$. Now $\varphi \circ s^{n-2}$ is an LCP-map $\mathrm{Re}^2(\Pi) \to \Pi$. In conclusion $\mathrm{Re}^2(\Pi) \simeq \Pi$.

Assume now that n is odd. The LCP-maps that give the equivalence in this case are illustrated in Figure 3. By Lemma 5.3.10 there are LCP-maps $s^{n+1}\colon \Pi \to \mathrm{Re}^{n+1}(\Pi)$ and $s^{n-1}\colon \mathrm{Re}(\Pi) \to \mathrm{Re}^n(\Pi)$. Now we have LCP-maps $\mathrm{Re}(\varphi) \circ s^{n+1}\colon \Pi \to \mathrm{Re}(\Pi)$ and $\varphi \circ s^{n-1}\colon \mathrm{Re}(\Pi) \to \Pi$. In conclusion $\mathrm{Re}(\Pi) \simeq \Pi$, which also gives by Lemma 5.3.8 $\mathrm{Re}^2(\Pi) \simeq \Pi$.

□

In order to apply Theorem 5.3.11 to periodic points, we need to be a bit stricter about what we mean by a periodic point. In general periodicity can be given by

any type of 0-round reduction, but there is a chance that such a reduction could be something very esoteric, making it hard to state anything about periodic points in full generality. Instead, we would likely find it more useful to make statements in the context of our easy-to-work-with sandbox of LCL-categories. Therefore we consider periodic points given by LCP-maps.

Definition 5.3.12. We say that a problem $\Pi \in \mathrm{LCL}^*(d, \delta)$ is an **LCP-periodic point of round elimination** if it is a periodic point for which the defining 0-round reduction can be chosen to be induced by an LCP-map, i.e. there is an LCP-map $\mathrm{Re}^n(\Pi) \to \Pi$ for some $n \in \mathbb{Z}_+$.

Corollary 5.3.12.1. *The period of an LCP-periodic point of round elimination is at most 2.*

Proof. Follows directly from Theorem 5.3.11. □

By the next corollary checking LCP-periodicity is equivalent to finding LCP-maps $\mathrm{Re}^2(\Pi) \to \Pi$, which can be done by computer by checking the LCP-rule for finitely many function candidates. We explore this idea further in the next section.

Corollary 5.3.12.2. $\Pi \in \mathrm{LCL}^*(d, \delta)$ *is an LCP-periodic point of round elimination if and only if there is an LCP-map* $\mathrm{Re}^2(\Pi) \to \Pi$.

Proof. If there is an LCP-map $\mathrm{Re}^2(\Pi) \to \Pi$, Π is LCP-periodic, and conversely if Π is LCP-periodic, by Theorem 5.3.11 we have an LCP-map $\mathrm{Re}^2(\Pi) \to \Pi$. □

While we define the LCP-periodic points only for $\mathrm{LCL}^*(d, \delta)$, the following result shows that a corresponding definition outside $\mathrm{LCL}^*(d, \delta)$ would serve little practical purpose.

Theorem 5.3.13. *Let* $\Pi \in \mathrm{LCL}(d, \delta)$ *and assume that there is an GCP-map* $\mathrm{Re}^n(\Pi) \to \Pi$ *for some* $n \in \mathbb{Z}_+$. *Now* $\mathrm{Ir}(\Pi)$ *is LCP-periodic.*

Proof. We have $\mathrm{Ir}(\Pi) \subseteq \Pi$ by Theorem 4.3.11, and applying Re^n on the inclusion map gives us an LCP-map $\mathrm{Re}^n \mathrm{Ir}(\Pi) \to \mathrm{Re}^n(\Pi)$. If $\mathrm{Ir}(\Pi) = \Pi_\emptyset$, it is LCP-periodic as $\mathrm{Re}^2(\Pi_\emptyset) = \Pi_\emptyset$. Otherwise, there is a GCP-map $\Pi \to \mathrm{Ir}(\Pi)$ and the composition $\mathrm{Re}^n \mathrm{Ir}(\Pi) \to \mathrm{Re}^n(\Pi) \to \Pi \to \mathrm{Ir}(\Pi)$ is LCP by Theorems 4.3.8 and 5.1.4, thus making $\mathrm{Ir}(\Pi)$ LCP-periodic. □

6 Finding LCP-maps

6.1 Configuration-respecting and Nondeterministic Maps

In this section we discuss how to check the existence of LCP-maps from a problem to another. This is useful for example for determining whether two problems are in the same equivalence class, or through Corollary 5.3.12.2 for checking whether a problem is an LCP-periodic point of round elimination. Our end goal is to provide an implementable algorithm for finding an LCP-map. We will start with simplifying assumptions that reduce the number of functions the algorithm has to search through.

With the current definitions, it is possible for an LCP-map to map two permutations of the same configuration to two completely different configurations. This means that when we are determining which of the functions $\underline{A} \to \underline{A'}$ are LCP, we have to consider all $|\underline{A'}|^{|\underline{A}|}$ of them, which in the worst case can be $(d!\,|A'|)^{d!|A|}$. However, as we are interested in the existence of LCP-maps $\Pi \to \Pi'$ instead of finding every single one of them, we would ideally like to just describe where a representative of each $a \in A$ lands in $\underline{A'}$, reducing the number of cases to at most $(d!\,|A'|)^{|A|}$ and the space required to describe an LCP-map by $d!$-fold. In other words, we would like to restrict the search to LCP-maps that respect configurations and can be defined through representatives by deducing the rest of the behaviour from permutations.

Definition 6.1.1. Let $\Pi = (A, P), \Pi' = (A', P') \in \mathrm{LCL}\,(d, \delta)$. We say that an LCP-map $\varphi\colon \Pi \to \Pi'$ is **configuration-respecting (CR)** if there is a function $\overline{\varphi}\colon A \to A'$ s.t. the diagram

$$\begin{array}{ccc} \underline{A} & \xrightarrow{\varphi} & \underline{A'} \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ A & \xrightarrow{\overline{\varphi}} & A' \end{array}$$

commutes, i.e. for all $a \sim_\sigma a' \in \underline{A}$ we have $\varphi(a) \sim_\sigma \varphi(a')$.

Remark. Restricting to configuration-respecting LCP-maps gives a subcategory of $\mathrm{LCL}\,(d, \delta)$, since identity maps are CR and if $\varphi\colon \Pi \to \Pi'$ and $\varphi'\colon \Pi' \to \Pi''$ are CR, then $\varphi' \circ \varphi$ is CR by $\overline{\varphi'} \circ \overline{\varphi}$. We denote this subcategory by $\mathrm{CRLCL}\,(d, \delta)$.

Remark. The inverse of a CR-isomorphism is not necessarily CR. Consider the problems

$$\begin{aligned} \Pi &= (\{[a,a]\,,[b,b]\}\,,\{[a,a]\,,[b,b]\,,[a,b]\})\ , \\ \Pi' &= (\{[a,b]\}\,,\{[a,a]\,,[b,b]\,,[a,b]\})\ . \end{aligned}$$

Now the isomorphism $\varphi\colon \Pi \cong \Pi'$ defined by $(a,a) \mapsto (a,b), (b,b) \mapsto (b,a)$ is CR but its inverse is not.

Our goal is to show that when there are LCP-maps $\Pi \to \Pi'$, at least some of them are CR. Given an LCP-map $\varphi\colon \Pi \to \Pi'$, we would like to construct a CR-map by fixing a representative a for each configuration $\overline{a} \in A$ and then define the map on their permutations by $\sigma a \mapsto \sigma\varphi(a)$. However, if the stabiliser of $\varphi(a)$ does not contain the stabiliser of a, this is not well-defined and we would have to add arbitrary choices in addition to the choice of representatives. We will approach the problem by abstracting away the choices, showing a more general result, and then narrowing down to the existence of CR-maps.

Definition 6.1.2. A function $\varphi\colon \underline{A} \to 2^{\underline{A'}} \setminus \{\emptyset\}$ induces a label relation $\sim_\varphi$ on $\Sigma \times \Sigma'$ by

$$l \sim_\varphi l' \iff \exists a \in \underline{A}, i \in \{1,\dots,d\} : l' \in \pi_i(\varphi(a)), a_i = l.$$

Definition 6.1.3. Let $\Pi = (A,P), \Pi' = (A',P') \in \mathrm{LCL}\,(d,\delta)$ and $\varphi\colon \underline{A} \to 2^{\underline{A'}} \setminus \{\emptyset\}$. We call φ a **nondeterministic LCP-map** if (P,P') is compatible with $\sim_\varphi$.

The intuition here is that instead of a single output the function gives a set from which the output could be chosen at random.

Remark. The objects $\mathrm{LCL}\,(d,\delta)$ and nondeterministic LCP-maps form a category.

For nondeterministic LCP-maps $\varphi\colon \Pi \to \Pi'$ and $\varphi'\colon \Pi' \to \Pi''$ the composition $\varphi' \circ \varphi$ is given by

$$\underline{A} \to 2^{\underline{A''}} \setminus \{\emptyset\} : a \mapsto \bigcup_{a' \in \varphi(a)} \varphi'(a').$$

For all $p \in \underline{P}$ and $q'' \in \Sigma''^{\delta}$, with $p \sim^{\delta}_{\varphi' \circ \varphi} q''$ we have some $a^{(1)},\dots,a^{(\delta)} \in \underline{A}$, $i_1,\dots,i_\delta \in \{1,\dots,d\}$, for which $p = \left(a^{(1)}_{i_1},\dots,a^{(\delta)}_{i_\delta}\right)$ and

$$q'' \in \pi_{i_1}\left(\varphi' \circ \varphi\left(a^{(1)}\right)\right) \times \cdots \times \pi_{i_\delta}\left(\varphi' \circ \varphi\left(a^{(\delta)}\right)\right).$$

Since φ is nondeterministic LCP, we have

$$\pi_{i_1}\left(\varphi\left(a^{(1)}\right)\right) \times \cdots \times \pi_{i_\delta}\left(\varphi\left(a^{(\delta)}\right)\right) \subseteq \underline{P'}.$$

Thus as φ' is nondeterministic LCP, we have for each $a'^{(1)} \in \varphi\left(a^{(1)}\right), \dots, a'^{(\delta)} \in \varphi\left(a^{(\delta)}\right)$ that

$$\pi_{i_1}\left(\varphi'\left(a'^{(1)}\right)\right) \times \cdots \times \pi_{i_\delta}\left(\varphi'\left(a'^{(\delta)}\right)\right) \subseteq \underline{P''},$$

therefore giving

$$\pi_{i_1}\left(\varphi' \circ \varphi\left(a^{(1)}\right)\right) \times \cdots \times \pi_{i_\delta}\left(\varphi' \circ \varphi\left(a^{(\delta)}\right)\right) \subseteq \underline{P''}.$$

In conclusion, $q'' \in \underline{P''}$, and therefore (P, P'') is compatible with the induced label relation, making $\varphi' \circ \varphi$ a nondeterministic LCP-map.

Furthermore, 1_Π is given by $a \mapsto \{a\}$, and

$$\begin{aligned}
&(\varphi'' \circ \varphi') \circ \varphi \\
&= a \mapsto \bigcup_{a' \in \varphi(a)} \varphi'' \circ \varphi'(a') \\
&= a \mapsto \bigcup_{a' \in \varphi(a)} \bigcup_{a'' \in \varphi'(a')} \varphi''(a'') \\
&= a \mapsto \bigcup_{a'' \in \bigcup_{a' \in \varphi(a)} \varphi'(a')} \varphi''(a'') \\
&= a \mapsto \bigcup_{a'' \in \varphi' \circ \varphi(a)} \varphi''(a'') \\
&= \varphi'' \circ (\varphi' \circ \varphi).
\end{aligned}$$

We denote this category by $\mathrm{NDLCL}\,(d, \delta)$.

Given a nondeterministic LCP-map, we can reduce the sets of available outputs without losing the LCP-property. In particular, if we reduce them all the way to singletons, we end up with a deterministic LCP-map.

Definition 6.1.4. If $\varphi, \varphi' \colon \underline{A} \to 2^{\underline{A'}} \setminus \{\emptyset\}$ s.t. $\varphi'(a) \subseteq \varphi(a)$ for all $a \in \underline{A}$, we denote $\varphi' \subseteq \varphi$.

Lemma 6.1.5. *Let $\varphi \colon \Pi \to \Pi'$ be a nondeterministic LCP-map, $\varphi' \colon \underline{A} \to 2^{\underline{A'}} \setminus \{\emptyset\}$ and $\varphi'' \colon \underline{A} \to \underline{A'}$. If $\varphi' \subseteq \varphi$, then φ' is a nondeterministic LCP-map $\Pi \to \Pi'$. If $\varphi''(a) \in \varphi(a)$ for all $a \in \underline{A}$, then φ'' is an LCP-map $\Pi \to \Pi'$.*

Proof. $\sim_{\varphi'}$ and $\sim_{\varphi''}$ are subrelations of $\sim_\varphi$, and therefore by Lemma 3.2.5 compatible with the pair (P, P'), i.e. φ' is nondeterministic LCP by Definition 6.1.3 and φ'' is LCP by Lemma 3.2.4. □

Remark. Conversely, we can turn an LCP-map into a nondeterministic LCP-map by wrapping the output in a singleton, giving us a functor $\iota \colon \mathrm{LCL}\,(d, \delta) \to \mathrm{NDLCL}\,(d, \delta) : \varphi \mapsto s \circ \varphi$.

The intuition behind the following lemma is that LCP-maps must be robust against adversarial port numbering. Assume that an active node gets the configuration $a \in \underline{A}$ in a labelling that partially solves Π. Imagine that there was an adversary that temporarily gives the node fake port numbers according to a permutation σ, that are the worst-case scenario with respect to an LCP-map φ. With the fake port numbers the labels appear as σa and are mapped to $\varphi(\sigma a)$.

Afterwards the true port numbers are restored, giving the node the configuration $\sigma^{-1}\varphi(\sigma a)$ instead of $\varphi(a)$. As the fake port numbers do not affect the correctness of the labelling, local correctness is preserved in these instances.

Lemma 6.1.6. *Let* $\Pi, \Pi' \in \mathrm{LCL}(d, \delta)$ *and* $\varphi\colon \Pi \to \Pi'$ *an LCP-map. Now* $\varphi'\colon \underline{A} \to 2^{\underline{A'}} \setminus \{\emptyset\} : a \mapsto \{\sigma\varphi(\sigma^{-1}a) \mid \sigma \in \mathbb{S}_d\}$ *is a nondeterministic LCP-map.*

Proof. We want to show that $\sim_{\varphi'}$ is a subrelation of $\sim_{\varphi}$. Let $l \sim_{\varphi'} l'$. By Definition 6.1.2 there are $a \in \underline{A}$ and $i \in \{1, \dots, d\}$ s.t. $a_i = l$ and $l' \in \pi_i(\varphi'(a))$. In particular, $l' = (\sigma\varphi(\sigma^{-1}a))_i = (\varphi(\sigma^{-1}a))_{\sigma^{-1}(i)}$ for some $\sigma \in \mathbb{S}_d$. Now as $l = a_i = \sigma^{-1}(a)_{\sigma^{-1}(i)}$, we get $l \sim_{\varphi} l'$ by $\sigma^{-1}(a) \in \underline{A}$ and $\sigma^{-1}(i) \in \{1, \dots, d\}$. In conclusion, $\sim_{\varphi'}$ is a subrelation of $\sim_{\varphi}$ and therefore by Lemmas 3.2.4 and 3.2.5 compatible with the pair (P, P'), thus giving us that φ' is a nondeterministic LCP-map. □

It is now easy to see that we can modify an LCP-map into a CR-map by fixing representatives $a^{(i)}$ and choosing outputs for elements $a = \sigma a^{(i)}$ from appropriate sets $\{\sigma\varphi(\sigma^{-1}a) \mid \sigma \in \mathbb{S}_d\}$.

Theorem 6.1.7. *Let* $\Pi, \Pi' \in \mathrm{LCL}(d, \delta)$. *If there are LCP-maps* $\Pi \to \Pi'$, *there are CR-maps* $\Pi \to \Pi'$. *In particular, for any LCP-map* $\varphi\colon \Pi \to \Pi'$ *and choice of representatives* $a^{(1)}, \dots, a^{(|A|)} \in \underline{A}$ *such that* $\left\{\overline{a^{(1)}}, \dots, \overline{a^{(|A|)}}\right\} = A$, *we can construct a CR-map* $\psi\colon \Pi \to \Pi'$ *that only differs from* φ *on elements where* φ *does not respect the configurations determined by the representatives. In other words, for all* $i \in \{1, \dots, |A|\}$ *and* $a \sim_\sigma a^{(i)}$ *s.t.* $\varphi(a) \sim_\sigma \varphi\left(a^{(i)}\right)$, *we have* $\psi(a) = \varphi(a)$.

Proof. Let φ be an LCP-map $\Pi \to \Pi'$ and fix a choice of representatives $a^{(1)}, \dots, a^{(|A|)}$. We define $\psi\colon \underline{A} \to \underline{A'}$ as follows. For each $a \in \underline{A}$ there is exactly one $i \in \{1, \dots, |A|\}$ s.t. $a \sim_\sigma a^{(i)}$. Furthermore there is some $\sigma_a \in \mathbb{S}_d$ s.t. $a = \sigma_a\left(a^{(i)}\right)$, which we fix. Let now

$$\psi\colon a \mapsto \begin{cases} \varphi(a), & \varphi(a) \sim_\sigma \varphi\left(a^{(i)}\right) \\ \sigma_a\varphi\left(a^{(i)}\right), & \varphi(a) \not\sim_\sigma \varphi\left(a^{(i)}\right) \end{cases}.$$

Let φ' be the nondeterministic LCP-map in Lemma 6.1.6 derived from φ. Now in the $\varphi(a) \sim_\sigma \varphi\left(a^{(i)}\right)$ case we have $\psi(a) = \varphi(a) = \mathrm{id}\,\varphi(\mathrm{id}^{-1}\,a) \in \varphi'(a)$, and in the $\varphi(a) \not\sim_\sigma \varphi\left(a^{(i)}\right)$ case we have $\psi(a) = \sigma_a\varphi\left(a^{(i)}\right) = \sigma_a\varphi\left(\sigma_a^{-1}a\right) \in \varphi'(a)$. Therefore ψ is LCP by Lemma 6.1.5, and it is CR by $\overline{\psi}\colon \overline{a^{(i)}} \mapsto \overline{\varphi\left(a^{(i)}\right)}$. □

6.2 LCP-map Finding Algorithm

We will now present an algorithm for finding LCP-maps. More precisely, by the same ideas as in the previous results, it is enough to find a way to map one

representative from each active configuration so that the induced relation stays compatible with the passives, and this can be extended to all their permutations without changing the relation, giving us an LCP-map. We will build this map label by label, configuration by configuration, The additions to the relation are validated on each step to either move forward or reject and try other options. The algorithm will check all possible cases, so it also confirms when no LCP-map exists. While the algorithm is not completely naive brute force, we do not make any optimality claims, so finding more efficient algorithms remains a topic for further research.

When we are finding LCP-maps $(\Sigma, A, P) \to (\Sigma', A', P')$, we are keeping track of two growing things:

- A map from some arrays over Σ to arrays over Σ', which grows in number of arrays in the domain until we have array for each configuration in A, and in the size of the newest array, which grows until the array and its image correspond to representatives of active configurations.
- The induced relation, to which we add a new pair $(l, l') \in \Sigma \times \Sigma'$ for each new occurrence of mapping the label l to l', and for which we check the compatibility with (P, P') on each addition since otherwise the addition to the map can be rejected.

We start with the compatibility checks.

Definition 6.2.1. Let $(\Sigma, A, P), (\Sigma', A', P') \in \mathrm{LCL}\,(d, \delta)$, and assume that Σ and Σ' have ordered representations. We denote by CHECK the process defined as follows. CHECK takes as input a relation $R \subseteq \Sigma \times \Sigma'$, an addition $(l, l') \in \Sigma \times \Sigma'$ and the pair (P, P'), and produces a boolean output. We can assume that P and P' are represented by sorted lists of sorted arrays, i.e. $[p_1, \ldots, p_\delta] \in P$ with $p_1 \leq \cdots \leq p_\delta$ is represented by an array $(p_1, \ldots, p_\delta)$, and these arrays are sorted in lexicographic order. R is represented by a list of pairs in $\Sigma \times \Sigma'$ and in the context of the algorithm it is compatible with (P, P').

If $(l, l') \in R$, CHECK returns true. Otherwise let $R' = R \cup \{(l, l')\}$. We obtain a pair $(P_l, P'_{l'})$ by filtering the lists to contain only configurations where l or l' appear respectively, and removing one copy of l or l' respectively from each such configuration. These lists are then sorted. For each array $p \in P_l$, we find for each label p_i the set of labels p'_i for which $(p_i, p'_i) \in R'$. Now for each $p \in P_l$, for each choice $p' = (p'_1, \ldots, p'_\delta)$ of such labels, we sort p' and check whether it is in the list P'_l. If $p' \notin P'_l$ for some of them, return false, otherwise return true.

In other words, CHECK is supposed to check the compatibility of (P, P') with $R \cup (l, l')$, given that the pair is compatible with R. The idea is, that we only need to check pairs of configurations that involve l and l', and comparing

configurations is faster when we obtain unique representatives by sorting. Therefore, instead of naive check for which the running time can in the worst case be $\mathcal{O}\left(|P| \cdot |\Sigma'|^{\delta} \cdot |\underline{P'}| \cdot \delta\right) \leq \mathcal{O}\left(|P| \cdot |\Sigma'|^{\delta} \cdot |P'| \cdot \delta! \cdot \delta\right)$, the running time should be $\mathcal{O}\left(|P_l| \cdot |\Sigma'|^{\delta-1} \cdot \log\left(|P'_l|\right) \cdot \delta\right)$. One further optimisation that removes some duplicate checks is that when $p_i = p_{i+1}$, it is enough to check the p' where $p'_i \leq p'_{i+1}$.

The correctness of CHECK is given by the following lemma.

Lemma 6.2.2. *Let* $(\Sigma, A, P), (\Sigma', A', P') \in \mathrm{LCL}(d, \delta)$, $R \subseteq \Sigma \times \Sigma'$, $(l, l') \in \Sigma \times \Sigma' \setminus R$ *and the relation* R *be compatible with* (P, P'). *Denote*

$$P_l = \{[p_1, \ldots, p_{\delta-1}] \mid [p_1, \ldots, p_{\delta-1}, l] \in P\}$$

and similarly $P'_{l'}$. *Now* $R \cup \{(l, l')\}$ *is compatible with* (P, P') *if and only if it is compatible with* $(P_l, P'_{l'})$.

Proof. Denote $R \cup \{(l, l')\}$ by $\sim$.

"$\Rightarrow$": Assume that $\sim$ is compatible with (P, P'). Let $p \in \underline{P_l}$ and $p' \in \Sigma'^{\delta-1}$ s.t. $p \sim^{\delta-1} p'$. Now $(p_1, \ldots, p_{\delta-1}, l) \in \underline{P}$, and $(p_1, \ldots, p_{\delta-1}, l) \sim^{\delta} \left(p'_1, \ldots, p'_{\delta-1}, l'\right)$ since $l \sim l'$. Therefore by compatibility $\left(p'_1, \ldots, p'_{\delta-1}, l'\right) \in \underline{P'}$, giving us $p' \in \underline{P'_{l'}}$, and hence $\sim$ is compatible with $(P_l, P'_{l'})$.

"$\Leftarrow$": Assume that $\sim$ is compatible with $(P_l, P'_{l'})$. Let $p \in \underline{P}$ and $p' \in \Sigma'^{\delta}$ s.t. $p \sim^{\delta} p'$. If there is no $i \in \{1, \ldots, \delta\}$ for which $p_i = l$ and $p'_i = l'$, we have $p' \in \underline{P'}$ by compatibility of R and (P, P'). On the other hand if such i exists, we have $(p_1, \ldots, p_{i-1}, p_{i-1}, \ldots, p_{\delta}) \in \underline{P_l}$ and $(p_1, \ldots, p_{i-1}, p_{i-1}, \ldots, p_{\delta}) \sim^{\delta-1} \left(p'_1, \ldots, p'_{i-1}, p'_{i-1}, \ldots, p'_{\delta}\right)$. Therefore by compatibility $\left(p'_1, \ldots, p'_{i-1}, p'_{i-1}, \ldots, p'_{\delta}\right) \in \underline{P'_{l'}}$, and thus $p' = \left(p'_1, \ldots, p'_{i-1}, l', p'_{i-1}, \ldots, p'_{\delta}\right) \in \underline{P'}$. In conclusion, $\sim$ is compatible with (P, P'). □

Building the map candidate is done using two recursive functions, TRY_CONFIGURATIONS and TRY_PERMUTATIONS, that call each other. TRY_CONFIGURATIONS checks whether the map candidate can be extended to an LCP-map through selecting any of the output configurations for the next input configuration, and TRY_PERMUTATIONS checks whether the map candidate can be extended to an LCP-map through fixing any of the remaining permutations for that selected output configuration.

Definition 6.2.3. Let L and L' be lists or arrays. We denote by $L + L'$ their **concatenation**, i.e. the elements of L followed by elements of L' in their original order. We call the first element of a list or an array its **head** and the rest its **tail**, and denote them by HEAD and TAIL, so that $L = \mathrm{HEAD}(L) + \mathrm{TAIL}(L)$.

Definition 6.2.4. Let $(\Sigma, A, P), (\Sigma', A', P') \in \mathrm{LCL}(d, \delta)$. We denote by TRY_PERMUTATIONS the process defined as follows. TRY_PERMUTATIONS takes as input a relation $R \subseteq \Sigma \times \Sigma'$, the pair (P, P'), A' as a list of multisets where each output configuration is represented by a multiset of size d, a subset $B \subseteq A$ as a list of arrays of size d, and a multiset S of size $k \in \{0, \ldots, d\}$. It outputs either the value none or an extension to the map candidate, represented by the pair (φ, s), where φ is a function $\mathrm{TAIL}(B) \to \underline{A'}$ represented by a set $\{(a, \varphi(a)) \mid a \in \mathrm{TAIL}(B)\}$, and s is array of the elements of S in a specific order.

If $k = 0$ the process calls

$$\mathrm{TRY\_CONFIGURATIONS}\,(R, (P, P'), A', \mathrm{TAIL}(B))\,.$$

If the return value is none, it returns none, and if the return value is an extension φ, it returns $(\varphi, ())$.

If $k > 0$ the process iterates over the distinct values of S. Let $l = \mathrm{HEAD}(B)_k$ and $l' \in S$ the current iteration. If

$$\mathrm{CHECK}\,(R, (l, l'), (P, P'))$$

returns false, extending the candidate by the pair (l, l') would break it, so we move to the next iteration. Otherwise, we call

$$\mathrm{TRY\_PERMUTATIONS}\,(R \cup \{(l, l')\}\,, (P, P'), A', B, S \setminus [l'])\,,$$

fixing l' to be the current label in the output permutation. If the return value is none, no such extension works and we move to the next iteration. Conversely, if the return value is $(\varphi, (b, s))$, we have found a working extension, and we can break the iteration loop and return $(\varphi, s + (l'))$. If the iteration ends without finding an extension, return none.

Definition 6.2.5. Let $(\Sigma, A, P), (\Sigma', A', P') \in \mathrm{LCL}(d, \delta)$. We denote the following process by TRY_CONFIGURATIONS. TRY_CONFIGURATIONS takes as input a relation $R \subseteq \Sigma \times \Sigma'$, the pair (P, P'), A' as a list of multisets where each output configuration is represented by a multiset of size d and a subset $B \subseteq A$ as a list of arrays of size d. It outputs either the value none or an extension to the map candidate, represented by a function $\varphi\colon B \to \underline{A'}$ as a set $\{(a, \varphi(a)) \mid a \in B\}$.

If B is empty, there are no more input configurations that need to be matched to an output, and the process returns $\varphi = \emptyset$. Otherwise, the process tries to extend the candidate through selecting an output configuration for $\mathrm{HEAD}(B)$ by iterating over A'. Let $S \in A'$ be the current iteration. The process calls

$$\mathrm{TRY\_PERMUTATIONS}\,(R, (P, P'), A', B, S)\,,$$

fixing the output configuration. If the return value is (φ, s), we have found a working extension, so and we can break the iteration loop and return $\varphi \cup \{(\mathrm{HEAD}(B), s)\}$. Conversely, if the return value is none, we move to the next iteration as no valid extensions exist for that configuration, and if the iteration ends without finding an extension, return none.

These processes terminate, since on each call of TRY_PERMUTATIONS k decreases or TRY_CONFIGURATIONS is called with a smaller B. Note that by collecting all possible extensions instead of breaking the iteration loops, the algorithm could easily be modified to return all valid maps from the representatives, but for existence of LCP-maps we only need one.

Our algorithm now boils down to calling the TRY_CONFIGURATIONS process and interpreting the result.

Theorem 6.2.6. *Let* $\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}\,(d, \delta)$. *If*

$$\mathrm{TRY\_CONFIGURATIONS}(\emptyset, (P, P'), A', A)$$

returns φ, *we can define an LCP-map* $\psi \colon \Pi \to \Pi'$ *as follows: For each* $a \in \underline{A}$, *there exists some* $\sigma \in \mathbb{S}_d$ *for which there is a unique* $(\sigma(a), a') \in \varphi$. *Fix a choice of such permutations and define* $\psi(a) = \sigma^{-1}(a')$. *Conversely, if the return value is none, there are no LCP-maps* $\Pi \to \Pi'$.

Proof. We start with the negative result. Assume that there is some $\varphi \colon \Pi \to \Pi'$. We can restrict it to the representatives of A given by the assumed order on Σ to get a map that corresponds to a potential output value. Assume that in the call stack, on each layer of TRY_CONFIGURATIONS calls all iterations before $S = \overline{\varphi(\mathrm{HEAD}(B))}$ have moved forward, and on each layer of TRY_PERMUTATIONS calls all iterations before $l' = \varphi(\mathrm{HEAD}(B))_k$ have moved forward. We now get a call stack where each call matches with φ. Since the relation R starts empty and only grows in TRY_PERMUTATIONS calls by $(\mathrm{HEAD}(B)_k, \varphi(\mathrm{HEAD}(B))_k)$, which are in $\sim_\varphi$, everywhere in the call stack R is a subrelation of $\sim_\varphi$ and thus compatible with (P, P'). Hence all CHECK calls in this stack return true, and as the only other way to return none is for the iterations to finish, the stack must reach the TRY_CONFIGURATIONS call with empty B, collapsing the stack with the return value corresponding to φ. In conclusion, if there is an LCP-map $\varphi \colon \Pi \to \Pi'$, the process call cannot return none, so by contraposition if the return value is none, there are no LCP-maps.

For the converse, assume a return value φ. This can only happen when the call stack is collapsed by a call to TRY_CONFIGURATIONS with empty B, and therefore for each $a \in A$ there is a TRY_CONFIGURATIONS call where the $\mathrm{HEAD}(B)$ of the $(\mathrm{HEAD}(B), s)$ added to φ is the sorted representative of a.

Furthermore, s is built by TRY_PERMUTATIONS calls to contain each element of $S \in A'$ exactly according to their multiplicity, so $s \in \underline{A'}$ and φ can thus be interpreted as a function from the representatives of A to $\underline{A'}$. In addition, R starts from empty relation, which is trivially compatible with (P, P') and on each addition (l, l') to R in the TRY_PERMUTATIONS calls, the CHECK calls ensure that compatibility is preserved, so in particular $R = \sim_\varphi$ in the collapsing call is compatible. Finally, when modifying φ to ψ, every $(a_i, \sigma^{-1}(a')_i) = (a_i, a'_{\sigma(i)}) = (\sigma(a)_{\sigma(i)}, a'_{\sigma(i)})$ in $\sim_\psi$ is already in $\sim_\varphi$, and hence $\sim_\psi$ is compatible with (P, P'), i.e. ψ is LCP. $\square$

7 Minimal Equivalent Problem

7.1 Definition

Consider the following exemplary lemma.

Lemma 7.1.1. *If* $\Pi \simeq \Pi' \in \mathrm{LCL}(d,\delta)$*, then* $\Pi \times \Pi' \simeq \Pi \simeq \Pi \oplus \Pi'$*.*

Proof. As $\Pi \simeq \Pi'$ there are some LCP-maps $\varphi\colon \Pi \to \Pi'$ and $\psi\colon \Pi' \to \Pi$. Now we have $1_\Pi \times \varphi\colon \Pi \to \Pi \times \Pi'$ and $1_\Pi \oplus \psi\colon \Pi \oplus \Pi' \to \Pi$, which together with $\pi_\Pi\colon \Pi \times \Pi' \to \Pi$ and $\iota_\Pi\colon \Pi \to \Pi \oplus \Pi'$ give us $\Pi \times \Pi' \simeq \Pi \simeq \Pi \oplus \Pi'$. □

It shows that an equivalence class can have arbitrarily complicated representatives, as we can construct such problems by repeatedly applying sums and products within the class. When we are studying the equivalence classes, it would be useful to be able to find a representative that is as simple as possible. It is not necessarily obvious how simplicity of representatives should be measured, but in this section we define one minimality condition, explore its implications and show how such minimal representative can be constructed.

Our minimality condition depends on epimorphisms, so we start by observing how they behave in $\mathrm{LCL}(d,\delta)$.

Definition 7.1.2. Let $\Pi = (\Sigma, A, P) \in \mathrm{LCL}(d,\delta)$. We denote

$$\Pi_T = (A, \Sigma^\delta / \sim_\sigma) \in \mathrm{LCL}(d,\delta)\,.$$

Turning Π to Π_T makes every non-empty problem is 0-round solvable, since in Π_T any output is accepted by the passives. While as problems Π_T are uninteresting, they can be useful in category theory due to the following fact.

Lemma 7.1.3. *Let* $\Pi = (\Sigma, A, P), \Pi' = (\Sigma', A', P') \in \mathrm{LCL}(d,\delta)$*. Every function* $\underline{A} \to \underline{A'}$ *is an LCP-map* $\Pi \to \Pi'_T$*.*

Proof. The relation $\Sigma \times \Sigma'$, i.e. $\sim$ where $l \sim l'$ for all $l \in \Sigma$, $l' \in \Sigma'$, is compatible with $(P, \Sigma'^\delta / \sim_\sigma)$. The induced label relation $\sim_\varphi$ for every function $\varphi\colon \underline{A} \to \underline{A'}$ is a subrelation of $\Sigma \times \Sigma'$, and therefore by Lemmas 3.2.5 and 3.2.4 all such functions are LCP $\Pi \to \Pi'_T$. □

Lemma 7.1.4. *An LCP-map is epic if and only if it is surjective as a set map.*

Proof. Recall Definition 2.1.8 and the fact that in Set epimorphisms are surjections. Let $\Pi = (A, P), \Pi' = (A', P') \in \mathrm{LCL}(d,\delta)$ and $\varphi\colon \Pi \to \Pi'$ be an LCP-map.

"⇒": Let φ be epic. If $A' = \emptyset$, φ is surjective. Assume that $A' \neq \emptyset$ and for a contradiction that φ is not surjective. There is now some $a' \in \underline{A'} \setminus \varphi(\underline{A})$. Let

$\Pi'' = (A'', P'') = (\Pi' \oplus \Pi')_T$. As $\underline{A''}$ consists of two disjoint copies of $\underline{A'}$, we have $|\underline{A''}| = 2\,|\underline{A'}| \geq 2$, and therefore there are distinct elements $b, b' \in \underline{A''}$. Consider the functions $\psi, \psi' \colon \underline{A'} \to \underline{A''}$ that are defined by $\psi \colon x \mapsto b$ and

$$\psi' \colon x \mapsto \begin{cases} b', & x = a' \,, \\ b, & x \neq a' \,. \end{cases}$$

By Lemma 7.1.3 ψ and ψ' are LCP $\Pi' \to \Pi''$. However, now $\psi \circ \varphi = (x \mapsto b) = \psi' \circ \varphi$, which contradicts φ being epic, and hence φ is surjective.

"$\Leftarrow$": Let $\varphi \colon \underline{A} \to \underline{A'}$ be surjective and $\psi, \psi' \colon (A', P') \to (A'', P'')$ be LCP-maps s.t. $\psi \circ \varphi = \psi' \circ \varphi$. By surjectivity, we have $\psi = \psi'$ as set maps. Therefore $\psi = \psi'$ also as LCP-maps and thus φ is epic. □

We use epimorphisms to define minimality in equivalence classes.

Definition 7.1.5. We say that a problem is **minimal in its equivalence class** if all its endomorphisms are epic.

This definition may at first seem strange, but it has the intuition of minimising the set of active configurations $\underline{A}$ behind it. Let us first consider what an analogous definition would mean in the category of finite sets. The equivalence classes would be the empty set, as no other set has functions to it, and the class of all non-empty finite sets, as between those we can always construct a function that maps all elements of a set to a single element of the other. If a finite set X has an endofunction $f \colon X \to X$ that is not surjective, its image is a strictly smaller subset $f(X) \subsetneq X$, that is equivalent to X by $f \colon X \to f(X) \subseteq X$. Thus by finding non-epic endofunctions and taking their images we can generate a sequence of sets in the equivalence class that strictly decrease in size, until we hit a singleton. Singletons are not unique, as we can freely choose the element in them, but they are unique up to unique isomorphism, which is the best we could hope for, and a singleton can certainly be considered a minimal non-empty set.

We would like to apply a similar iterative minimisation process to the LCL-problems. If we find an endomorphism that does not hit any representative of a configuration $\overline{a} \in A$, we can remove that configuration and stay in the same equivalence class similar to the finite set analogy. However, this iteration stops working in cases where an endomorphism covers some permutations of every configuration but is not necessarily epic, since then we cannot simply remove the uncovered permutations in the category LCL (d, δ). Our definition of minimality takes into account that such problems can be simplified further by replacing configurations with ones with more symmetry, thus reducing $|\underline{A}|$ even though $|A|$ stays the same, but now the existence of a minimal representative becomes non-trivial.

Our definition gives us uniqueness up to isomorphism.

Theorem 7.1.6. *The minimal problem of an equivalence class is unique up to isomorphism. In particular, all LCP-maps between minimal problems in the same equivalence class are isomorphisms.*

Proof. Let $\Pi = (A, P) \simeq \Pi' \in \mathrm{LCL}(d, \delta)$ be minimal problems in their equivalence class, with LCP-maps $\varphi\colon \Pi \to \Pi'$ and $\varphi'\colon \Pi' \to \Pi$, which exist by equivalence. The LCP-map $\varphi' \circ \varphi\colon \Pi \to \Pi$ is by minimality an epimorphism, and therefore by Lemma 7.1.4 a surjection $\underline{A} \to \underline{A}$. Hence $\varphi' \circ \varphi$ is by cardinality a bijection, i.e. a permutation of the elements of the set $\underline{A}$. As $\underline{A}$ is finite, for some $n \in \mathbb{Z}_+$ we get $(\varphi' \circ \varphi)^n = 1_{\underline{A}} = 1_\Pi$. Thus we have found an LCP-map $\psi = (\varphi' \circ \varphi)^{n-1} \circ \varphi'$ s.t. $\psi \circ \varphi = 1_\Pi$. By symmetry, there is $m \in \mathbb{Z}_+$ for which

$$1_{\Pi'} = (\varphi \circ \psi)^m = \varphi \circ (\psi \circ \varphi)^{m-1} \circ \psi = \varphi \circ (1_\Pi)^{m-1} \circ \psi = \varphi \circ \psi.$$

In conclusion, $\psi = \varphi^{-1}$ making φ an isomorphism. □

However, the definition does not give uniqueness up to unique isomorphism, as seen in the following example.

Example 7.1.7. Consider the problems

$$\Pi = (\{[a, a], [b, b]\}, \{[a, b]\})$$

and

$$\Pi' = (\{[a', a'], [b', b']\}, \{[a', b']\}).$$

A map $\underline{A} \to \underline{A}\colon x \mapsto (a, a)$ is not LCP, and therefore by symmetry all endomorphisms are epic for both problems. Thus both problems are minimal. However, there are two distinct isomorphisms $\Pi \cong \Pi'$, namely

$$\begin{cases} (a, a) \mapsto (a', a'), \\ (b, b) \mapsto (b', b') \end{cases}$$

and

$$\begin{cases} (a, a) \mapsto (b', b'), \\ (b, b) \mapsto (a', a'). \end{cases}$$

A less abstract way to minimise problems would have been through the cardinality of $\underline{A}$. However, finding the minimum of $|\underline{A}|$ might not be an easy task without our toolkit. It turns out that the equivalence of our minimality and minimum $|\underline{A}|$ boils down to proving the existence of minimal problems.

Lemma 7.1.8. *Let $\Pi = (A, P) \in \mathrm{LCL}(d, \delta)$ be minimal in its equivalence class. If $\Pi' = (A', P') \simeq \Pi$, we have $|\underline{A}| \leq |\underline{A'}|$, where $|\underline{A}| = |\underline{A'}|$ if and only if Π' is minimal.*

Proof. Let $\psi\colon \Pi' \to \Pi'$. By $\Pi \simeq \Pi'$ there are morphisms $\varphi\colon \Pi \to \Pi'$ and $\varphi'\colon \Pi' \to \Pi$. Now $\varphi' \circ \psi \circ \varphi\colon \Pi \to \Pi$ is by minimality epic and thus by Lemma 7.1.4 surjective, giving us $|\underline{A}| \leq |\psi(\underline{A'})| \leq |\underline{A'}|$. If $|\underline{A}| = |\underline{A'}|$, the inequality gives that every endomorphism $\psi\colon \Pi' \to \Pi'$ is surjective and by Lemma 7.1.4 epic. Conversely, if Π' is minimal, we get by symmetry the opposite inequality $|\underline{A'}| \leq |\underline{A}|$, and thus $|\underline{A}| = |\underline{A'}|$. □

In order to show existence, it would be easier if we could instead consider the seemingly weaker minimality condition that only CR-endomorphisms need to be epic. Conveniently, the following theorem shows that the conditions are equivalent.

Theorem 7.1.9. *A problem is minimal if and only if all its configuration-respecting endomorphisms are epic.*

Proof. The CR-endomorphisms are a subset of all endomorphisms, so it is trivial that all CR-endomorphisms of a minimal problem are epic. For the converse, let Π be a problem with an endomorphism $\varphi\colon \Pi \to \Pi$ that is not epic. In particular, there is some $a \in \underline{A} \setminus \varphi(\underline{A})$. For each configuration $\overline{a^{(i)}} \in A$, we have either $\varphi(a') \sim_\sigma a$ for all $a' \sim_\sigma a^{(i)}$, or we can choose the representative $a^{(i)}$ s.t. $\varphi\left(a^{(i)}\right) \not\sim_\sigma a$. With this choice of representatives $a^{(1)}, \ldots, a^{(|A|)}$, we can construct a CR-endomorphism $\psi\colon \Pi \to \Pi$ according to Theorem 6.1.7. Since the configurations that map to $\overline{a}$ do so according to φ, we have $a \notin \psi(\underline{A})$, and therefore by Lemma 7.1.4 ψ is not epic. In conclusion, we get by contraposition that if all CR-endomorphisms of Π are epic, all its endomorphisms are epic. □

7.2 Existence

Our goal now is to show by construction that a minimal problem exists in every equivalence class. In order to do this, we will first present a well-known way to construct simpler equivalent problems by merging labels. Then we will see how we can minimise problems with respect to configurations, and finally that performing a maximal merge on such problem gives us a minimal problem.

Definition 7.2.1. Let $\Pi = (\Sigma, A, P) \in \mathrm{LCL}(d, \delta)$. We denote by $\sim_P$ be the equivalence relation over Σ that relates labels that are **interchangeable in** P, determined by

$$l \sim_P l' \iff ([l, p_2, \ldots, p_\delta] \in P \iff [l', p_2, \ldots, p_\delta] \in P)\,.$$

Let $\sim$ be an equivalence relation that is a subrelation of $\sim_P$ and π the natural projection $\Sigma \to \Sigma/\sim$. We define the **merge with respect to** $\sim$ to be the problem $M_\sim(\Pi) = (\Sigma/\sim, \pi(A), \pi(P))$ given through Lemma 3.4.5, and the **maximal merge** $M(\Pi)$ to be $M_{\sim_P}(\Pi)$.

Lemma 7.2.2. *Merge preserves the equivalence class, i.e.* $M_\sim(\Pi) \simeq \Pi$.

Proof. We have $\pi\colon \Pi \to M_\sim(\Pi)$ by Lemma 3.4.5. As π is induced by a label map, it respects permutations and is therefore surjective $\underline{A} \to \underline{\pi(A)}$. Hence for each $a' \in \underline{\pi(A)}$ there is $a \in \underline{A}$ for which $\pi(a) = a'$, and we can define $\varphi\colon \underline{\pi(A)} \to 2^{\underline{A}} \setminus \{\emptyset\}$ by preimages $a' \mapsto \pi^{-1}(\{a'\})$. The label relation $\sim_\varphi$ on $(\Sigma/\sim) \times \Sigma$ is a subrelation of $\sim_{\pi^{-1}}$ given by

$$l' \sim_{\pi^{-1}} l \iff l' = \pi(l).$$

To show that $\sim_{\pi^{-1}}$ is compatible with $(\pi(P), P)$ let $p' \in \underline{\pi(P)}$ and $q \in \Sigma^\delta$ s.t. $p' \sim^\delta_{\pi^{-1}} q$. As the induced function $\pi\colon \underline{P} \to \underline{\pi(P)}$ is similarly surjective, there is some $p \in \underline{P}$ for which $p' = \pi(p)$. Now for each $i \in \{1, \dots, \delta\}$ we have $\pi(p_i) = p'_i = \pi(q_i)$, i.e. $p_i \sim q_i$. Therefore we have $p_i \sim_P q_i$ by Definition 7.2.1, giving us the implication

$$[q_1, \dots, q_{i-1}, p_i, \dots, p_\delta] \in P \Rightarrow [q_1, \dots, q_i, p_{i+1}, \dots, p_\delta] \in P,$$

which we can use as an induction step to get $q \in \underline{P}$.

In conclusion, $\sim_\varphi$ is compatible with $(\pi(P), P)$ by Lemma 3.2.5, and hence φ is a nondeterministic LCP-map $M_\sim(\Pi) \to \Pi$, from which we can choose an LCP-map $M_\sim(\Pi) \to \Pi$ by Lemma 6.1.5. □

Remark. As can be seen in appendix A, it is also possible to perform a corresponding operation in the opposite direction to distinguish between instances of a label that appears multiple times.

In order to construct a minimal problem, we introduce an intermediate minimality condition, which we can use to reduce the number of active configurations.

Definition 7.2.3. We say that a problem $\Pi \in \mathrm{LCL}(d, \delta)$ is **configuration-minimal** if for every CR-endomorphism $\varphi\colon \Pi \to \Pi$ the function $\overline{\varphi}$ is surjective.

Lemma 7.2.4. *For every* $\Pi \in \mathrm{LCL}(d, \delta)$*, there is a configuration-minimal problem* $\Pi' \subseteq \Pi$ *s.t.* $\Pi' \simeq \Pi$.

Proof. We will construct Π' by the following iterative process. Let $\Pi = (A, P)$ and $A_1 = A$. For $i \in \mathbb{Z}_+$, if there is a CR-map $\varphi_i\colon (A_i, P) \to (A_i, P)$ for which $\overline{\varphi_i}$ is not surjective, choose one such map and define $A_{i+1} = \overline{\varphi_i}(A_i)$. We now have $(A_{i+1}, P) \subseteq (A_i, P)$ and φ_i gives us an LCP-map $(A_i, P) \to (A_{i+1}, P)$, so $(A_{i+1}, P) \simeq (A_i, P) \simeq \Pi$. Furthermore, we have $|A_{i+1}| < |A_i|$, so as A is finite, the process terminates and we get some A_j for which no such map exists, and we can define $\Pi' = (A_j, P)$. □

A configuration-minimal problem is not yet minimal. If there are equivalent labels in an active configuration, elements mapping to permutations that differ only on those labels can be freely mapped to one of the permutations, allowing us to still construct non-epic endomorphisms. However, we can eliminate this case by merging the equivalent labels. It turns out that removing impossible passive configurations, which can unnecessarily block equivalences, and taking the maximal merge is enough to give us minimality.

Lemma 7.2.5. *If* $\Pi = (\Sigma, A, P) \in \mathrm{LCL}\,(d, \delta)$ *is configuration-minimal,* $M((A, P_A))$ *is minimal.*

Proof. Let $\Pi' = (\Sigma', A', P') = M((A, P_A))$ and φ be a CR-endomorphism $\Pi' \to \Pi'$. By Lemmas 4.3.2 and 7.2.2 we have $\Pi \cong (A, P_A) \simeq \Pi'$, and thus by Theorem 6.1.7 there are CR-maps $\pi\colon \Pi \to \Pi'$ and $\tau\colon \Pi' \to \Pi$. Now $\tau \circ \varphi \circ \pi\colon \Pi \to \Pi$ is CR, and thus by CR-minimality $\overline{\tau} \circ \overline{\varphi} \circ \overline{\pi}\colon A \to A$ is surjective, so as $|A'| \leq |A|$, $\overline{\varphi}$ is also surjective. Hence $\overline{\varphi}$ is a permutation on A, and there is some $n \in \mathbb{Z}_+$ for which $\overline{\varphi}^n = \mathrm{id}_A$.

Let now $a^{(1)}, \ldots, a^{(|A'|)} \in \underline{A'} : \left\{\overline{a^{(1)}}, \ldots, \overline{a^{(|A'|)}}\right\} = A'$ be a choice of representatives for the configurations. For each $i \in \{1, \ldots, |A'|\}$, let $A_i = \left\{\overline{a^{(i)}}\right\}$. As $\overline{\varphi}^n$ maps A_i to itself, φ^n also maps $\underline{A_i}$ to itself. Since $\left|\underline{A_i}\right| \leq d!$, there are by the pigeon hole principle $j_i, k_i \in \{0, \ldots, d!\}$ with $j_i < k_i$, for which $\varphi^{nj_i}\left(a^{(i)}\right) = \varphi^{nk_i}\left(a^{(i)}\right)$. In other words, there is $m_i = k_i - j_i \in \{1, \ldots, d!\}$ for which φ^{nm_i} fixes $\varphi^{nj_i}\left(a^{(i)}\right) \in \underline{A_i}$. Thus by setting $m = m_1 \cdot \cdots \cdot m_{|A'|}$, we get a CR-endomorphism $\psi = \varphi^{nm}$ that fixes a permutation from each configuration. Furthermore, as every label in Σ_A appears in A, every label of $\Sigma' = \Sigma_A / \sim_{P_A}$ appears in A', and therefore $l' \sim_\psi l'$ for all $l' \in \Sigma'$.

Assume now for a contradiction that φ is not epic, and thus not surjective by Lemma 7.1.4. Now ψ is not surjective either, giving $\psi \neq 1_{\Pi'}$. In consequence, there are some $a' \in \underline{A'}$ and $\sigma \in \mathbb{S}_d$, for which $\psi(a') = \sigma(a') \neq a'$. Hence for every $i \in \{1, \ldots, d\}$ we have $a'_{\sigma(i)} \sim_\psi a'_i$, but in particular for some $j \in \{1, \ldots, d\}$, $a'_{\sigma(j)} \neq a'_j$. As ψ is LCP, by Lemma 3.2.4 the pair (P', P') is compatible with $\sim_\psi$, which gives us for each i the implication

$$\left[a'_{\sigma(i)}, p'_2, \ldots, p'_\delta\right] \in P' \Rightarrow [a'_i, p'_2, \ldots, p'_\delta] \in P'.$$

Repeated application of the implication $d! - 1$ times also gives us

$$[a'_i, p'_2, \ldots, p'_\delta] = \left[a'_{\sigma^{d!}(i)}, p'_2, \ldots, p'_\delta\right] \in P' \Rightarrow \left[a'_{\sigma(i)}, p'_2, \ldots, p'_\delta\right] \in P',$$

and therefore in particular for j

$$\left[a'_{\sigma(j)}, p'_2, \ldots, p'_\delta\right] \in P' \iff \left[a'_j, p'_2, \ldots, p'_\delta\right] \in P'.$$

Fix now labels $l \in \pi^{-1}(a'_j)$ and $h \in \pi^{-1}(a'_{\sigma(j)})$ from the preimages with respect to the canonical projection $\pi\colon \Sigma_A/\sim_{P_A}$. If $[l, p_2, \ldots, p_\delta] \in P_A$, we have $\left[a'_j, \pi(p_2), \ldots, \pi(p_\delta)\right] \in P'$ and thus $\left[a'_{\sigma(j)}, \pi(p_2), \ldots, \pi(p_\delta)\right] \in P'$. Now there are some $\tilde{h} \sim_{P_A} h$, $\tilde{p_2} \sim_{P_A} p_2, \ldots, \tilde{p_\delta} \sim_{P_A} p_\delta$ for which $\left[\tilde{h}, \tilde{p_2}, \ldots, \tilde{p_\delta}\right] \in P_A$, and therefore by applying $\sim_{P_A}$ to each of them, we get $[h, p_2, \ldots, p_\delta] \in P_A$. Symmetrically, $[h, p_2, \ldots, p_\delta] \in P_A$ implies $[l, p_2, \ldots, p_\delta] \in P_A$, and therefore $l \sim_{P_A} h$, which contradicts the maximality of the merge as $\pi(l) \neq \pi(h)$.

In conclusion, all CR-endomorphisms of Π' are epic, and therefore Π' is minimal by Theorem 7.1.9. □

We get the existence of minimal problems in each equivalence class by combining the results.

Theorem 7.2.6. *For every $\Pi \in \mathrm{LCL}(d, \delta)$, there is a minimal problem $\Pi' \simeq \Pi$.*

Proof. By Lemma 7.2.4 there is configuration-minimal problem $(A'', P'') \subseteq \Pi$ s.t. $(A'', P'') \simeq \Pi$. Now the problem $\Pi' = M((A'', P''_{A''}))$ is minimal by Lemma 7.2.5 and by Lemmas 4.3.2 and 7.2.2 we have $\Pi' \simeq (A'', P''_{A''}) \cong (A'', P'') \simeq \Pi$. □

Corollary 7.2.6.1. *A problem $\Pi = (A, P) \in \mathrm{LCL}(d, \delta)$ is minimal in its equivalence class if and only if $|\underline{A}|$ is minimal over problems in the equivalence class.*

Proof. If Π is minimal, $|\underline{A}|$ is minimal by Lemma 7.1.8. Conversely, by Theorem 7.2.6 there is a minimal problem $\Pi' = (A', P') \simeq \Pi'$ and if $|\underline{A}|$ is minimal, we have $|\underline{A}| = |\underline{A'}|$, giving us by Lemma 7.1.8 that Π is minimal. □

8 Conclusions

In this thesis, I have provided a basis for studying 0-round reductions of biregular LCL-problems in the language of category theory. I found a suitable representation for the problems and defined LCP-maps between them to be 0-round maps that adhere to the local correctness preserving condition LCP-rule, which has multiple formulations. Selecting the problems as objects and LCP-maps as morphisms gave us the category $\mathrm{LCL}\,(d,\delta)$. The category-theoretical toolkit allowed us to import universal definitions, such as product and sum problems through limits and colimits, and helped us study 0-round equivalence of problems, corresponding to isomorphisms in the thin version $\mathrm{TLCL}\,(d,\delta)$ of $\mathrm{LCL}\,(d,\delta)$, and the relationships between the equivalence classes in the skeleton $\mathrm{SkTLCL}\,(d,\delta)$ of $\mathrm{TLCL}\,(d,\delta)$.

I justified the use of LCP-rule in the construction by considering categories $\mathrm{LCL}_{\mathscr{G}}\,(d,\delta)$ built around global correctness preserving conditions. It turned out that the most general such category, $\mathrm{LCL}_{\forall}\,(d,\delta)$, only differs from $\mathrm{LCL}\,(d,\delta)$ in presence of impossible active labels. Furthermore, I showed that we can safely move to the subcategory $\mathrm{LCL}^{*}\,(d,\delta)$, which has the impossible labels removed, by using the functor $\mathrm{Ir}\colon \mathrm{LCL}_{\forall}\,(d,\delta) \to \mathrm{LCL}^{*}\,(d,\delta)$.

The previous analysis was immediately useful as the impossible labels can cause inconvenience in round elimination. I defined round elimination as a map between categories $\mathrm{Re}\colon \mathrm{LCL}\,(d,\delta) \to \mathrm{LCL}\,(\delta,d)$ in such a way, that it corresponds to the traditional definition with some additional impossible passive label removal steps, thus making it restrict to a map $\mathrm{Re}\colon \mathrm{LCL}^{*}\,(d,\delta) \to \mathrm{LCL}^{*}\,(\delta,d)$. This did not quite give us a functor $\mathrm{LCL}\,(d,\delta) \to \mathrm{LCL}\,(\delta,d)$, but by constructing a morphism map we did get a functor $\mathrm{Re}\colon \mathrm{TLCL}\,(d,\delta) \to \mathrm{TLCL}\,(\delta,d)$, which is enough to study 0-round equivalence. Furthermore, a common alternative Definition 2.2.16 of round elimination corresponds to mapping to different equivalent problems, so the induced functor $\mathrm{Re}\colon \mathrm{SkTLCL}\,(d,\delta) \to \mathrm{SkTLCL}\,(\delta,d)$ unifies the different definitions.

The main result of my category-theoretical analysis was the short period theorem (Theorem 5.3.11). Its essential implication was that the longest period of a periodic point of round elimination is 2. The assumptions that we have to make to have the result hold are as follows. Firstly, we only consider periodicity over equivalence given by LCP-maps, which is reasonable since we want to include as many reductions as possible in the equivalence, but additional ones would require more specific knowledge about the problem or the underlying graph family. Secondly, we assume that the problem has no impossible labels, which is not an issue as by Theorem 5.3.13 the functor Ir can be applied to a periodic point with impossible labels to get one with a period at most 2.

The significance of the short period theorem is that instead of applying round elimination indefinitely and hoping to get back to the original problem, we can check the periodicity of a problem Π by checking whether an LCP-map $\mathrm{Re}^{2}(\Pi) \to \Pi$

exists. Therefore I provided an algorithm for determining the existence of an LCP-map $\Pi \to \Pi'$. I also showed that one simplifying assumption we can make when studying the existence of LCP-maps is that we can demand the maps to be configuration-respecting, i.e. mapping all representatives of a configuration to the same configuration, because by Theorem 6.1.7 such maps exist if LCP-maps exist. Indeed, a key observation that the algorithm uses was that we only need to be able to map one representative of each active configuration while respecting the LCP-rule, in order to construct a CR-map. The algorithm builds such candidate map label by label, configuration by configuration, and checks the compatibility of each addition to the induced relation.

Finally, I explored minimality in equivalence classes. I defined an endomorphism-based minimality condition, which can be checked without knowledge of other problems in the same equivalence class. I also provided a constructive proof that a minimal problem exists in each equivalence class. It turned out that minimal problems are unique up to isomorphism, and the condition is equivalent to a more intuitive one of minimising the available permutations of active configurations.

The most immediate topic of further research would be finding more efficient algorithms for determining the existence of an LCP-map $\Pi \to \Pi'$. Such algorithm should also be implemented and integrated with the round eliminator tool so that they can be used to check for periodicity of a problem. Furthermore, there is more to study with respect to equivalence classes and minimality. In particular, it would be useful to find an algorithm that minimises a problem efficiently, and to know how minimality behaves under round elimination, or how much the round elimination definition would have to be modified in order to make it preserve minimality.

References

[1] J. Hirvonen and J. Suomela, *Distributed Algorithms 2020*. Espoo, Finland: Aalto University, 2021.

[2] J. Suomela, "Landscape of Locality (Invited Talk)," in *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)* (S. Albers, ed.), vol. 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 2:1–2:1, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[3] M. Naor and L. Stockmeyer, "What can be computed locally?," *SIAM Journal on Computing*, vol. 24, no. 6, pp. 1259–1277, 1995.

[4] S. Brandt, "An automatic speedup theorem for distributed problems," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, (New York, NY, USA), p. 379–388, Association for Computing Machinery, 2019.

[5] D. Olivetti, "Round eliminator: a tool for automatic speedup simulation." URL: https://github.com/olidennis/round-eliminator, 2020.

[6] A. Balliu, D. Olivetti, and J. Suomela, "Periodic points in round elimination." URL: https://github.com/suomela/fixed-points, 2020.

[7] E. Riehl, *Category Theory in Context*. Courier Dover Publications, 2017.

[8] T. Leinster, *Basic Category Theory*. Cambridge Studies in Advanced Mathematics, Cambridge University Press, 2014.

A Unmerge

We can use the same principle as in Definition 7.2.1 in the other direction in order to separate labels that appear in different configurations or multiple times in the same configuration.

Definition A.0.1. Let $\Pi = (\Sigma, A, P) \in \mathrm{LCL}(d, \delta)$ and fix a choice of representatives $a^{(1)}, \dots, a^{(|A|)} \in \underline{A} : \left\{\overline{a^{(1)}}, \dots, \overline{a^{(|A|)}}\right\} = A$. Let

$$\begin{aligned}
\Sigma' &= \{1, \dots, |A|\} \times \{1, \dots, d\} \ , \\
A' &= \{[(i,1), \dots, (i,d)] \mid i \in \{1, \dots, |A|\}\} \subseteq \Sigma'^d / \sim_\sigma \ , \\
P' &= \left\{[(i_1, j_1), \dots, (i_\delta, j_\delta)] \mid \left[a_{j_1}^{(i_1)}, \dots, a_{j_\delta}^{(i_\delta)}\right] \in P\right\} \subseteq \Sigma'^\delta / \sim_\sigma \ .
\end{aligned}$$

We call the problem $U(\Pi) = (\Sigma', A', P') \in \mathrm{LCL}(d, \delta)$ the **unmerge** of Π.

The idea here is that we replace each appearance of a label in A with a unique label, and then modify P so that when a label appears multiple times resulting in unmerged labels l and l' we have $l \sim_{P'} l'$. While the definition depends on the choice of representatives, calling it *the* unmerge instead of *an* unmerge is justified, since unmerges are unique up to bijective relabelling.

Lemma A.0.2. *Let $\Pi = (\Sigma, A, P) \in \mathrm{LCL}(d, \delta)$, and $U_1(\Pi) = (\Sigma', A_1', P_1')$ and $U_2(\Pi) = (\Sigma', A_1', P_2')$ two unmerges of Π. There is now a permutation $\varphi\colon \Sigma' \to \Sigma'$ that induces an isomorphism $\varphi\colon U_1(\Pi) \cong U_2(\Pi)$.*

Proof. Let $a^{(1)_1}, \dots, a^{(|A|)_1}$ be the defining choice of representatives for $U_1(\Pi)$ and $a^{(1)_2}, \dots, a^{(|A|)_2}$ for $U_2(\Pi)$. As each configuration has exactly one representative in each case, there is a permutation $f \in \mathbb{S}_{|A|}$ and permutations $\sigma_1, \dots, \sigma_{|A|} \in \mathbb{S}_d$ s.t. $\sigma_i a^{(i)_1} = a^{(f(i))_2}$ for each $i \in \{1, \dots, |A|\}$. Conversely, we have $a^{(f^{-1}(i))_1} = \sigma_{f^{-1}(i)}^{-1} a^{(i)_2}$ for each $i \in \{1, \dots, |A|\}$. We define $\varphi\colon \Sigma' \to \Sigma'$ by $(i,j) \mapsto (f(i), \sigma_i(j))$. The function φ is a bijection by the inverse $(i,j) \mapsto (f^{-1}(i), \sigma_{f^{-1}(i)}^{-1}(j))$.

The function φ induces a map $\varphi\colon \underline{A_1'} \to \underline{A_2'}$, for which the induced label relation is

$$(i,j) \sim_\varphi (i',j') \iff (i',j') = (f(i), \sigma_i(j)).$$

For compatibility with (P_1, P_2), let $p' \in \underline{P_1}$ and $q' \in \Sigma'^\delta$ s.t. $p' \sim_\varphi^\delta q'$. Now $q_k' = (f(i_k), \sigma_{i_k}(j_k))$, where $(i_k, j_k) = p_k'$, for each $k \in \{1, \dots, \delta\}$. Furthermore,

$$\left(a_{\sigma_{i_1}(j_1)}^{f(i_1)_2}, \dots, a_{\sigma_{i_\delta}(j_\delta)}^{f(i_\delta)_2}\right) = \left(a_{j_1}^{(i_1)_1}, \dots, a_{j_\delta}^{(i_\delta)_1}\right) \in \underline{P},$$

and hence $q' \in \underline{P_2'}$. In conclusion, φ is LCP, and by symmetry its inverse is also LCP, so $\varphi\colon U_1(\Pi) \cong U_2(\Pi)$. □

Lemma A.0.3.

$$U(\Pi) \simeq \Pi$$

Proof. Let $a^{(1)}, \ldots, a^{(|A|)}$ be the defining choice of representatives for $U(\Pi)$. We define the equivalence relation $\sim$ over Σ' by

$$(i,j) \sim (i',j') \iff a_j^{(i)} = a_{j'}^{(i')},$$

in which case the the equivalence classes correspond to Σ_A (recall the Definition 5.1.1). Let π be the natural projection $\Sigma' \to (\Sigma'/\sim) \cong \Sigma_A$ defined by $(i,j) \mapsto a_j^{(i)}$

Now if $(i,j) \sim (i',j')$, we have

$$\begin{aligned}
&[(i,j), p_2, \ldots, p_\delta] \in P' \\
&\iff \left[a_j^{(i)}, \pi(p_2), \ldots, \pi(p_\delta)\right] \in P \\
&\iff \left[a_{j'}^{(i')}, \pi(p_2), \ldots, \pi(p_\delta)\right] \in P \\
&\iff [(i',j'), p_2, \ldots, p_\delta] \in P',
\end{aligned}$$

and therefore $\sim$ is a subrelation of $\sim_{P'}$. For every $\overline{a^{(i)}} \in A$ we have $\overline{a'} = [(i,1), \ldots, (i,\delta)] \in A'$ s.t. $\pi(\overline{a'}) = \overline{a^{(i)}}$, and for every $\overline{p} = \left[a_{j_1}^{(i_1)}, \ldots, a_{j_\delta}^{(i_\delta)}\right] \in P_A$ we have $\overline{p'} = [(i_1,j_1), \ldots, (i_\delta, j_\delta)] \in P'$ s.t. $\pi(\overline{p'}) = \overline{p}$. Hence $M_\sim(U(\Pi)) \cong (\Sigma_A, A, P_A)$, and thus by Lemmas 4.3.2 and 7.2.2 we get $U(\Pi) \sim \Pi$. □