Theory and Practice of Global Transformations

by

Alexandre Fernandez

A thesis submitted in partial fulfilment for the award of the degree
Doctor of Computer Science

to the department of Computer Science of
Université Paris Est

Committee composed of

Pawel Sobocinski	(Reviewer)
Enrico Formenti	(Reviewer)
Pablo Arrighi	(Chair)
Dominique Duval	(Examiner)
Daniela Petrisan	(Examiner)
Pierre Valarcher	(Supervisor)
Luidnel Maignan	(Advisor)
Antoine Spicher	(Advisor)

Defended on the 8th of December 2022

Contents

French summary

Les *transformations globales* sont un nouveau formalisme visant à décrire les calculs *synchrones*, *locaux* et *déterministes* sur un *espace dynamique*. Les *automates cellulaires* et *systèmes de Lindenmayer* sont deux exemples classiques de ces systèmes. Les transformations globales émergent naturellement de la tentative de les unifier et de les étendre à des espaces dynamiques.

En effet, bien que ces deux systèmes ne s'appliquent pas sur les mêmes structures, ils fonctionnent de la même manière. Les automates cellulaires sont des calculs sur des *configurations*, c'est-à-dire des grilles régulières décorées par des états. La particularité des automates cellulaires est que ceux-ci peuvent être décrits à travers un jeu de règles locales décrivant le comportement de l'automate sur des parties finies de configurations. Cette spécification locale permet de récupérer le calcul global en appliquant les règles de façon synchrone partout dans une configuration donnée. Les systèmes de Lindenmayer quant à eux sont des systèmes de réécriture de mots. Dans le cas le plus simple, un tel système est une simple fonction agissant sur l'ensemble des mots d'un alphabet donné. Il est aussi spécifié à l'aide d'une fonction locale, envoyant les lettres de l'alphabet vers des mots de l'alphabet. De la même manière que pour les automates cellulaires, le calcul global correspond à l'application de la fonction locale sur toutes les lettres d'un mot donné, puis de prendre la concaténation des résultats locaux obtenus.

Dans la littérature, de multiples extensions de ces systèmes à des espaces plus généraux et dynamiques ont été proposées. Cependant, parmi les approches génériques permettant de représenter une grande variété d'espaces et de dynamiques, on ne trouve pas de système pleinement synchrone. Certains de ces modèles de calcul sont complètement asynchrones comme les grammaires de graphes. D'autres méthodes s'appuient sur une stratégie maximal-parallel, synchrone, mais non-déterministe, comme dans le langage *MGS* par exemple. D'autre part, les approches pleinement synchrones permettant de représenter des espaces dynamiques ne sont pas très générales et sont contraintes à un type d'espace, comme pour les *dynamiques causales de graphes*.

La notion de transformation globale a été introduit en 2015 pour permettre la spécification locale du *raffinement de maillage triangulaire*. Ce calcul, qui prend un maillage avec des faces triangulaires en entrée, le raffine en subdivisant chacune de ses faces en quatre sous-faces triangulaires. Malgré le fait que ce processus semble intuitivement local, aucune spécification locale n'a pu être trouvée dans la littérature jusqu'alors. Cela provient du fait que les règles locales doivent spécifier le comportement du calcul non seulement sur les triangles, mais aussi sur les *recouvrements* entre ceux-ci. En effet, certains triangles partagent des arcs en commun, et il est nécessaire d'exprimer ce qu'il advient de cette relation au cours du calcul. Les transformations globales adressent ce problème en utilisant un ensemble de règles représentant les différents niveaux de localité à prendre en compte. De plus, un ensemble d'inclusions entre ces règles exprime le lien entre les relations entre les membres de gauche de règles et celles entre leurs membres de droite. L'apport des transformations globales est que la formalisation des structures manipulées, des règles locales, et de leur son extension au contexte global peuvent être décrit de façon unifiée dans le cadre de la *théorie des catégories*.

Cette recherche comporte deux axes. D'une part, les fondations théoriques sont

développées afin d'obtenir une méthode applicable à une grande variété de situations pratiques. D'autre part, la généricité de l'approche doit être démontrée. Il est alors utile de voir comment des systèmes connus peuvent être exprimés en tant que transformations globales, ou bien comment utiliser cette méthode pour concevoir de nouveaux systèmes. Dans ce but, chaque partie de ce travail utilise comme illustration un type de structure différent.

Le premier chapitre se concentre sur l'introduction de la méthode dans toute sa généralité. En effet, ce formalisme est paramétré par une description catégorique de l'espace manipulé. Les structures considérées sont représentées à travers une *catégorie* dont les *objets* sont des parties de ces structures et dont les *morphismes* sont des inclusions entres ces parties. Quant au calcul, il est représenté par un *foncteur*, c'est-à-dire un morphisme de cette catégorie vers elle-même, envoyant des parties sur des parties et leurs inclusions sur des inclusions, préservant ainsi la structure de la catégorie. Les règles et leurs inclusions mentionnées plus tôt peuvent elles aussi être exprimés dans ce cadre catégorique. On prend simplement une catégorie, décrivant les règles et les inclusions entre règles, et deux foncteurs, décrivant respectivement leurs membres de gauches et leurs membres de droites. Le calcul est alors spécifié de manière opérationnelle : la conception d'une transformation globale à travers un jeu de règles locales est d'abord présentée et son application sur un objet donné est décrite à l'aide de trois étapes de calculs. Tout d'abord, la décomposition. On cherche tous les sites d'applications des règles dans l'objet donné en entrée. Cette première étape est formalisée à l'aide du concept de *comma catégorie*. Puis l'application locale des règles. On obtient les résultats locaux à partir des règles. Et enfin la recomposition. On recolle tous les résultats locaux afin d'obtenir le résultat global. Cette dernière étape est obtenue à l'aide d'une construction catégorique particulière appelée une *colimite*.

Cette partie est illustrée par l'instanciation des systèmes de Lindenmayer avec et sans contexte. Les premiers sont des systèmes de réécriture de mots ou toutes les lettres d'un mot sont substituées simultanément vers un mot résultat correspondant. Dans les seconds, les règles s'appliquent sur des sous-mots pouvant se recouvrir les uns les autres. Malgré ces différences, une astuce consistant à représenter le bord des mots par un caractère spécial permet d'utiliser la méthode des transformations de la même manière pour représenter ces deux systèmes.

Le second chapitre se concentre sur l'étude des propriétés formelles des transformations globales, en particulier à travers leur relation avec le concept catégorique *d'extension de Kan*. Lors de l'étape de recomposition d'une transformation globale, la colimite choisit le résultat du calcul comme l'optimal parmi un ensemble de candidats. De la même manière, on peut caractériser le foncteur global d'une transformation comme une extension optimale de son jeu de règles, en exhibant les propriétés 2-catégoriques du formalisme. Cela mène directement aux extensions de Kan, et en particulier à celles qui peuvent être calculées objet par objet, à l'aide des trois étapes de calcul décrites plus tôt. Cette construction est illustrée à travers les automates cellulaires afin d'étudier la relation entre une fonction de transition locale et la fonction de transition globale correspondante. La fonction de transition globale d'un automate cellulaire agit sur des grilles infinies décorées appelées *configurations globales*. Dans les automates cellulaires, cette fonction peut être spécifiée à l'aide d'une fonction de transition locale, autrement dit, de par son comportement sur de petites parties finies de ces configurations. Lorsque l'on essaie d'étendre le

comportement d'un automate cellulaire aux configurations partielles, un choix se présente. Soit on étend la fonction locale en l'appliquant où on le peut dans une configuration partielle donnée, soit on restreint la fonction globale en regardant sur quelle partie de l'espace la donnée d'une configuration partielle la rend invariante. Ces deux approches peuvent être respectivement décrites comme des extensions de Kan à gauche et à droite. Deux cadres pour formuler ces extensions sont proposés. Le premier enrichit l'ensemble des configurations partielles d'une relation d'ordre partiel alignant les cellules sur leurs coordonnées absolue. Le second utilise autorise de décaler les configurations pour les mettre en relation à travers une catégorie de configurations. Les deux approches sont ensuite comparées, et la propriété d'une extension de Kan à gauche d'être calculable objet par objet est aussi exemplifiée à travers les automates cellulaires.

Dans le troisième chapitre, les propriétés calculatoires des transformations globales sont étudiées dans le but de leur implémentation concrète. La conception d'un algorithme générique est réalisée en formulant les étapes de calcul à l'aide de modules plus simples. Dans cette optique, ces modules sont recomposés afin de construire un algorithme de réécriture incrémental. Ces considérations algorithmiques sont réalisées à travers le prisme des transformations globales de graphes.

En effet, les calculs sur des graphes peuvent représenter une très grande variété de dynamiques intéressantes. Aussi, la notion usuelle de graphe en théorie des catégories, les *quivers*, possèdent beaucoup de bonnes propriétés de clotures, facilitant la définition de transformations globales de graphes. En particulier la catégorie des quivers est une catégorie de *préfaisceaux*, ce qui implique que la construction de colimite, utilisée dans l'étape de reconstruction, est toujours bien définie. On peut alors se concentrer sur les transformations globales de préfaisceaux comme une généralisation directe de la notion de transformation globale de graphes. Les catégories de préfaisceaux, en plus d'avoir toutes les colimites, partagent beaucoup de propriétés de la catégorie des ensembles. En particulier, les préfaisceaux peuvent être construits et décomposés uniquement à l'aide d'éléments appelés préfaiceaux représentables. Dans le cas des graphes, ces derniers sont simplement le graphe à un seul noeud et le graphe à un seul arc. De plus, les colimites dans les catégories de préfaisceaux sont calculées à partir des colimites d'ensembles. Ces différentes propriétés permettent d'exprimer un algorithme à la fois simple et générique.

Deux opérations nécéssaires pour spécifier correctement un algorithme de calcul sont alors dégagées. Pour la décomposition, on a besoin d'une opération pour trouver les occurrences d'un objet dans un autre, et d'une autre pour prolonger une occurrence d'un membre de gauche d'une règle, en une occurrence du membre de gauche d'une de ses sur-règles. Pour la reconstruction, on a besoin simplement d'une sous-classe particulière des colimites décrivant le recollement de deux objets le long d'un nombre arbitraire de sous-parties communes.

Puis on se concentre sur les morphismes *injectifs* de préfaisceaux. Ceux-ci permettent d'avoir des jeux de règles plus intuitifs. En effet, le membre de gauche d'une règle doit alors être compris comme une forme à identifier telle quelle dans un objet à transformer. Dans le cas où l'on considèrerait des morphismes non-injectifs, on doit aussi considérer tous les repliements possibles d'un membre de gauche d'une règle. Cependant, les catégories de préfaisceaux restreintes aux morphismes injectifs entre préfaisceaux n'ont pas les bonnes propriétés des catégories de préfaisceaux. En particulier elles n'ont pas les bonnes colimites. Pour contourner ce problème, on

décomposer l'entrée en uniquement avec des morphismes injectifs, mais recomposer le résultat en considérant n'importe quel type de morphismes. Par conséquent, on obtient un foncteur depuis la catégorie des préfaisceaux restreinte aux morphismes injectifs vers la catégorie des préfaisceaux générale. On se concentre alors sur les transformations globales qui préservent les morphismes injectifs, et on donne un critère suffisant sur les règles locales qui garantit cette propriété.

Le dernier chapitre se concentre sur l'étude du non-déterminisme dans les transformations globales. L'objectif n'est pas d'étendre le formalisme des transformations globales pour représenter le non-déterminisme, mais plutôt de montrer qu'il est déjà capable de le représenter. C'est en analogie avec les systèmes dynamiques discrets, un tel système étant simplement une fonction agissant sur un ensemble d'état. Dans ce cas, on peut représenter le non-déterminisme en prenant comme états des sous-ensembles d'un autre ensemble. On expérimente de la même manière une catégorie dont les objets seraient des ensembles d'objets d'une catégorie sous-jacente. Cependant, on n'a pas les bonnes colimites dans cette catégorie. Par conséquent, on n'est pas capable de représenter un système non-déterministe aussi simple que les systèmes de Lidnenmayer non-déterministes à l'aide de cette catégorie. Le point qui pose problème est exposé en termes de concaténation d'ensemble de mots. En effet lors de cette opération, on peut obtenir plusieurs fois le même mot, mais de différentes manières. Or cette donnée n'est pas préservée dans l'ensemble resultat. Pour résoudre ce problème on expérimente une autre catégorie ayant pour objets des familles indexées d'objets d'une catégorie sous-jacente. La construction obtenue envoie des objets vers des familles d'objets et ne peut donc pas être itérée en utilisant la composition usuelle. Ce problème est résolu à l'aide des aspects *monadiques* de cette construction.

Les aspects non-locaux de certains systèmes non-déterministes spatialisés sont aussi étudiés à travers la notion de corrélation spatiale. Il est montré que, malgré une caractérisation à travers des règles locales, certaines dépendances entre des choix non-déterministes au niveau local peuvent induire des dépendances globales, ce qui peut être interprété comme des effets non-locaux. De la même manière que pour la préservation des injections de graphes, l'absence de corrélations des transformations globales est montrée décidable seulement à l'aide des règles locales. En effet, une manière de garantir que les choix locaux sont indépendant est de s'assurer que pour chaque règle et chaque combinaison de choix pour ses sous-règles, il existe un choix correspondant pour cette première règle. Il est ensuite montré que ce critère fonctionne dès lors que la catégorie des structures considérées possède toutes les colimites. L'étude d'un critère s'appliquant à des catégories plus générales est laissé à plus tard.

La conclusion de ce document résume cette étude et y apporte de multiples ouvertures. Tout d'abord, dans les transformations globales, on représente les structures à l'aide d'une catégorie dont les objets sont les structures ou sous-structures et les morphismes décrivent les inclusions entre ces structures. À l'inverse d'autres approches comme les dynamiques causales de graphes où un système de coordonnées doit être encodé dans la structure des graphes manipulées, les transformations globales utilisent les morphismes pour relier les objets les uns aux autres. Cette approche est assez générique pour capturer les calculs locaux et pleinement synchrones. Ce document l'a démontré à travers les automates cellulaires, systèmes de Lindenmayer et les transformations globales de graphes.

Il y a cependant une différence majeure entre la catégorie des graphes et les catégories des mots et des configurations. En effet, comme c'est une catégorie de préfaisceaux, la catégorie des graphes est cocomplète, c'est-à-dire qu'elle a toutes les colimites. En d'autres termes, la catégorie des graphes peut être obtenue comme extension d'une catégorie très simple, décrivant simplement les notions de nœuds et d'arcs. D'autre part, la catégorie des configurations n'est pas cocomplète. En effet, n'importe quelle configuration peut-être décrite à l'aide d'une colimite de configurations, mais l'inverse ne tient pas. Si quelqu'un veut considérer des structures plus contraintes que les graphes, tout de même les voir comme extension d'une catégorie simple, il est intéressant de spécifier comment les objets de cette catégorie simple peuvent être combinés. Cette information supplémentaire peut être représentée par un *site*. C'est une catégorie à laquelle on ajoute une notion de *couverture*, pouvant indiquer où certains objets sont ouverts ou fermés. L'extension du formalisme des transformations globales sur ces objets pourrait donner une plus fine description des espaces manipulés et de leurs dynamiques et fait partie de futurs travaux.

La thématique principale de ce travail est le rapport entre le local et le global. L'étape de calcul d'une transformation globale consiste à appliquer de façon synchrone toutes les règles locales en un objet. Ce calcul a été formalisé d'abord par une formule utilisant la notion de colimite, puis par la notion d'extension de Kan décrivant comment un comportement global est l'extension optimale d'un jeu de règles. La propriété d'une extension de Kan de pouvoir être calculée objet par objet avec la formule de la colimite a été montrée comme liée à des considérations algorithmiques, car cela garantit que la réécriture d'un objet n'ait besoin de considérer que ses sous-objets.

Ces outils, développés pour décrire des systèmes existants, furent aussi capables de décrire de nouveaux systèmes. Pour les automates cellulaires, un simple changement dans les règles locales permet de représenter des automates cellulaires non-uniformes. De plus, il est possible de définir des transformations globales sur la catégorie des mots qui ne sont pas des systèmes de Lindenmayer. Par exemple, on peut construire une transformation globale qui renverse les mots, qui ne peut donc pas correspondre à un système de Lindenmayer. Dans ce cadre, on vise à garantir certaines propriétés globales de ces systèmes depuis le niveau local. C'est ce qui a été fait pour la préservation des injections dans les graphes et pour l'absence de corrélations pour le non-déterminisme. À travers ces deux exemples, une méthode générale pour garantir une propriété globale d'un tel système a été esquissée.

L'algorithme de calcul présenté dans ce travail nécessite aussi une étude plus approfondie. D'une part, sa complexité doit être discutée, mais comme il s'agit d'un algorithme très générique et en ligne, il doit être étudié en termes de complexité paramétrée et en ligne. D'autre part, même s'il s'agit d'un algorithme séquentialisant un processus massivement parallèle, sa parallélisation mérite d'être étudiée. Au lieu de lancer la réécriture depuis un point dans la structure en entrée, on peut très bien lancer en parallèle la réécriture depuis plusieurs points. Alors, une barrière de synchronisation devient nécessaire là où plusieurs de ces processus se rejoignent. De la même manière, d'autres dynamiques comme l'exécution asynchrone méritent d'être étudiée. En effet, le non-déterminisme permet déjà de représenter des processus asynchrones, en utilisant des règles spécifiant qu'un objet peut évoluer ou ne pas évoluer à chaque étape de calcul. C'est la première étape vers la formulation générale des transformations globales asynchrones.

Ce document se termine sur la comparaison des différentes définitions de la notion de transformation globale. En effet, chaque partie de ce travail a introduit une légère variante du concept original. Au final, la version la plus générale de ce concept est une extension 2-catégorique produisant un foncteur agissant sur une catégorie de structures. On recherche alors des propriétés locales qui garantissent le processus d'extention au cadre global.

Chapter 1

Introduction

Synchronous, *local* and *deterministic* spatialized discrete dynamical systems have many occurrences in computer science in fields ranging from fundamental computer science to modeling of real-world systems. Recently, *Global Transformations* (GT) were introduced in [Maignan and Spicher, 2015] as a general formalism to capture such systems and to describe dynamics over the topology of the space itself. The aim of this work is two-fold. On the one hand, the genericity of the approach is shown by expressing already known systems in terms of global transformations, and by designing new systems. On the other hand, the present work develops the theoretical foundations of global transformations in order to get a framework applicable to a wide variety of practical situations.

In this introduction, we present the difficulty of designing synchronous, local and deterministic systems where the space is also dynamic. We first describe some well-known paradigmatic but limited examples of such systems, *Cellular Automata* and *Lindenmayer Systems*. We then review attempts of extensions dealing with dynamic spaces, leading naturally to the proposition of global transformations.

1.1 Synchronous and Deterministic Systems

Cellular Automata and Lindenmayer Systems[1] are formal systems which both act over some decorated *space* where the *global dynamics* are specified through a set of *local evolution rules*. In contrast with other parallel systems, no *overlap* issue between the local rules can occur, so the process of selecting the sites of applications of the rules is completely *deterministic*. These spatialized discrete dynamical systems are thus referred here as local, deterministic and synchronous dynamical systems.

1.1.1 Cellular Automata

A typical Cellular Automaton (CA) is the evolution of some regular grid of objects, called *cells*, each associated with some state taken from a set called the *alphabet*. The regularity of the grid is usually captured by a group, called *space*, providing the positions of the cells in the grid and the ways to move from one cell to another. The data of the grid of cells, each labelled by some state is called a *configuration*.

[1]The discussion here and in the following chapters is restricted to deterministic Lindenmayer Systems only. Non-deterministic Lindenmayer Systems also exist. They are considered in chapter 5 about non-deterministic global transformations.

The dynamics are described by a set of local evolution rules specifying the cell state evolution as a function of the states of the cells in a given *neighborhood*. Thanks to the regularity of the space, the neighborhood is uniformly defined over the whole system, and the cells evolve altogether synchronously at each time step.

The most widely known of such systems is the Conway's *game of life* (figure 1.1), a cellular automaton acting over the infinite bi-dimensional grid decorated by states specifying whether a cell is dead or alive. This cellular automaton is described by the following rules over the eight-cell neighborhood (also named Moore neighborhood) of a cell: an alive cell remains alive if there are two or three alive cells in its neighborhood; a dead cell becomes alive if there are three alive cells in its neighborhood. In any other cases, the cell is dead at the next time step.

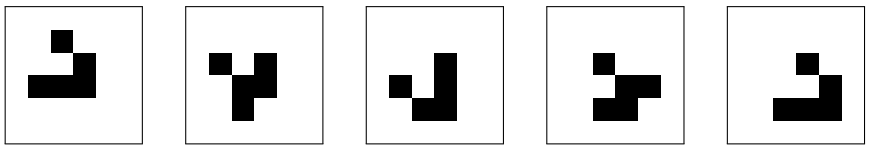
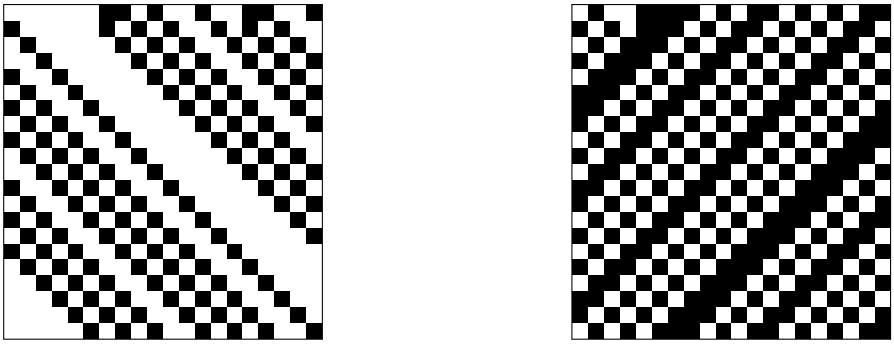


Figure 1.1: Four iterations of Conway's game of life

Cellular automata were introduced in 1966 by J. von Neumann to define self-replicating systems [Burks and Von Neumann, 1966]. He was originally attempting to describe such models using partial differential equations in $\mathbb{R}^3$. S. Ulam then suggested him to use a discrete model as a simpler model of self-replication, and he constructed a 29-state cellular automaton on $\mathbb{Z}^2$ exhibiting the expected property. Cellular automata were then studied as dynamical systems with the tools of symbolic dynamics. In 1969, the Curtis-Hedlund-Lyndon's theorem was proved. It characterizes cellular automata global transition functions as continuous endomorphisms on the shift space, a topological space over the set of configurations [Hedlund, 1969]. This gives a description of the global transition function of a cellular automaton that does not depend on its local presentation. In this sense, multiple local evolution rules can give rise the same cellular automaton.

In the 1980's, S. Wolfram systematically studied the so-called *elementary cellular automata*, *i.e.*, cellular automata over the group $\mathbb{Z}$ with alphabet $\{0, 1\}$ and with neighborhood $\{-1, 0, 1\}$, meaning that a local rule can look for the state of a cell, the state of its left neighbor, and the state of its right neighbor. As a simple example of such cellular automata, suppose one wants to model some road by some bi-infinite one-dimensional grids of cells, whose states 1 or 0 respectively specify whether or not there is a car in it. Assuming that the cars on this road are going to the right, one can let the cells evolve as follows. A 0-valued cell becomes 1-valued if the cell on its left is a 1-valued cell, a 1-valued cell becomes a 0-valued cell if the cell on its right is a 0-valued cell, and in any other case the cell keeps the same state. This elementary cellular automaton, called rule 184, exhibits some emergent features of actual traffic systems: clusters of freely moving cars separated by open road when traffic is light, and waves of stop-and-go traffic when it is heavy [Tadaki and Kikuchi, 1994] (figure 1.2). It also has been shown to model particle systems, such as deposition of particles over an irregular surface [Krug and Spohn, 1988] and ballistic annihilation of charged particles [Belitsky and Ferrari, 1995].

Cellular automata were shown to produce a wide variety of behaviors, ranging



(a) Light traffic (b) Heavy traffic

Figure 1.2: Space-time diagrams of rule 184 on a cycle of 20 cells, time goes down

from regular and predictable patterns to chaotic evolutions. In particular, the elementary cellular automata rule 30 (figure 1.3) introduced in [Wolfram, 1983] was later shown to be chaotic [Cattaneo et al., 2000]. They were then proved to be universal in [Smith III, 1971]. For instance, the game of life and the elementary cellular automaton rule 110 have been shown to be able to simulate a universal Turing machine [Berlekamp et al., 2004, Cook et al., 2004].

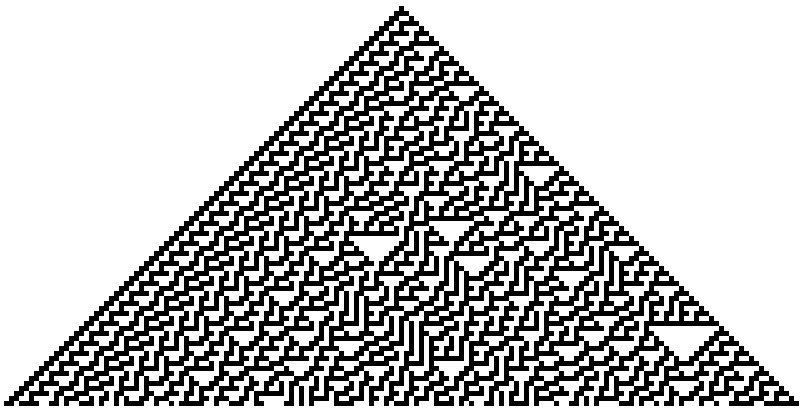


Figure 1.3: Space-time diagram of rule 30 starting from a single black-cell

These nice properties make cellular automata a rich tool to model a wide range of biological and physical systems. In particular, reversible cellular automata are used as a model of many physical systems ranging from quantum dynamics to fluid dynamics, the latter using lattice gas automata. To that end, stochastic and asynchronous extensions of cellular automata are also studied [Dobrushin et al., 1978, Cornforth et al., 2002]. Cellular automata can also be related to boolean networks, which act over directed graphs decorated by 0 and 1 using local rules that are functions of the incoming neighbors for each node, but these systems are not in the scope of this work due to the heterogeneous nature of the local rules.

1.1.2 Lindenmayer Systems

Lindenmayer systems, or L-systems, are parallel string rewriting systems that act over finite words of some alphabet. They are defined by means of local rules, called *productions*, sending letters to words over the alphabet, the result for an

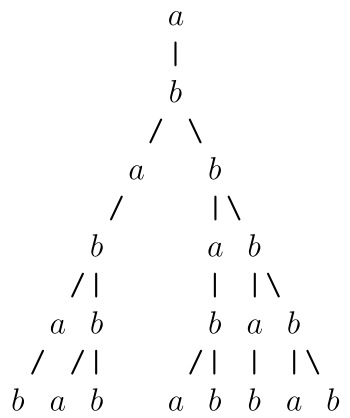


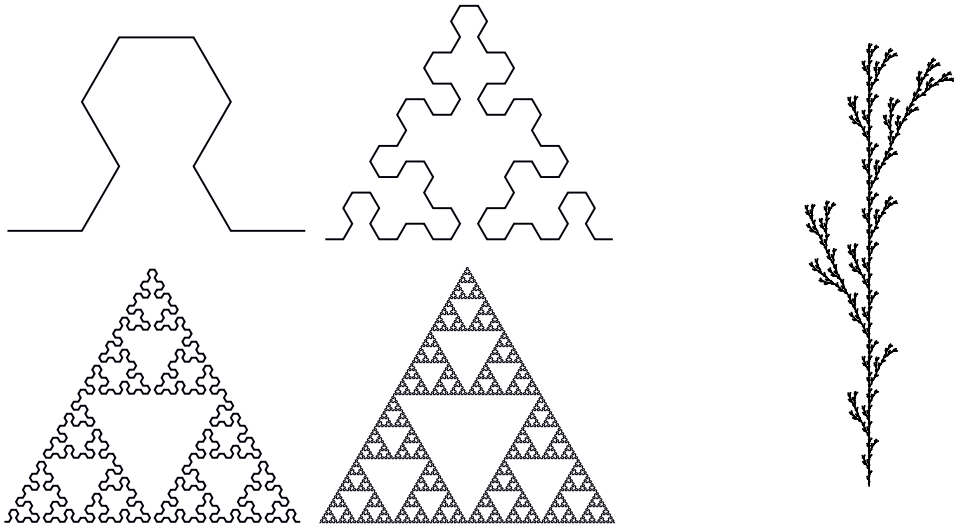
Figure 1.4: Application of filamentous bacteria Lindenmayer system on a

input word being the concatenation of the local results for each letter of that word. L-systems are named after A. Lindenmayer, a biologist who introduced them in [Lindenmayer, 1968] to model biological systems. The parallel and synchronous replacement of every letter in a given word is intended to model cell division in an organism, where multiple divisions can occur at the same time. The original instance of a L-System acts on finite words on the alphabet $\{a, b\}$ by mapping a to b and b to ab, and was used to describe the growth of the symbiotic bacteria *Anabaena catenula* which builds filamentous colonies (figure 1.4). In this model, the letters b and a respectively describe whether or not a cell is ready to divide.

The use of L-systems in biology has been investigated further. Notably, A. Lindenmayer and P. Prusinkiewicz published in 1990 a complete comprehensive book about the simulation of certain patterns in nature found in plant development [Prusinkiewicz and Lindenmayer, 2012]. L-systems have been recognized for their ability to encode tree-like structures. They can be used to draw structures exhibiting self-similarity, using *turtle graphics* and symbols manipulating a stack for branching structures. For instance, the Sierpinski triangle in figure 1.5a is drawn starting from the symbol A and using the rewriting rules $A \mapsto B-A-B$ and $B \mapsto A+B+A$, where $+$ and $-$ are constant. Here "A", "B" both mean the instruction "draw forward", and "$+$" and "$-$" mean "turn left/right by 60°" respectively. In figure 1.5b, a tree-like structure is drawn starting from symbol F and using the rule $F \mapsto F[+F]F[-F]F$ where $+$ and $-$ are constant. In this case, "F" means "draw forward", "$+$" and "$-$" mean "turn left/right by 25.7°" respectively, and symbols "[" and "]" mean "push/pop the coordinates to/from the stack".

In parallel to these usages in biological modeling and simulation, the computational power of L-systems as a formal system has been studied. L-systems provide a classification which differs from the classical Chomsky classification, where grammars productions are applied sequentially. As an example, there exist languages generated by L-systems which are not context-free in the sense of the Chomsky hierarchy [Herman and Rozenberg, 1975, Salomaa, 1987]. For example, consider the L-System on words over $\{a, b, c\}$ that maps a to aa, b to bb and c to cc. Iterated over the word abc it clearly generates the language $\{a^{2^n}b^{2^n}c^{2^n} \mid n > 0\}$, which is definitely not context-free similarly to $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

This last instance and all examples above are *context-free* L-Systems (also called D0L-System), where every letter has only one possible output and the productions do not use any context to decide the result for a letter. The L-System hierarchy



(a) Sierpinski triangle, iterations 2, 4, 6 and 8

(b) Simple plant, iteration 5, from [Prusinkiewicz and Lindenmayer, 2012]

Figure 1.5: Drawing self-similar structures using turtle graphics

considers context-sensitive L-Systems, or IL-Systems, allowing productions to use a context around a letter, *i.e.*, some letters on the left and some letters on the right, in a way that is similar to a one-dimensional cellular automaton. L-Systems also encompass non-determinism, using an arbitrary relation from letters to words of a given alphabet. Examples of context-free non-deterministic Lindenmayer Systems (0L-Systems) can be found in [Rozenberg and Salomaa, 1980].

Substitution dynamical system [Queffélec, 1987, Fogg et al., 2002] are similar systems studied in the field of symbolic dynamics. A simple substitution on words is a D0L-System with the additional property to be *non-erasing*, *i.e.*, it does not send a letter to the empty word. A substitution σ is defined as an application from an alphabet A into the set A^+ of finite non-empty words on A, which is extended to any words of $A^\star$ by concatenation, that is, $\sigma(w_1 w_2) = \sigma(w_1)\sigma(w_2)$. In this setting, the D0L-system considered above for Anabaena catenula is usually called the *Fibonacci substitution* ϕ, due to the fact that the length of the words generated by iterating this substitution starting from a follows the Fibonacci sequence. Indeed, at any step $i \geq 2$, we have $\phi^i(a) = \phi^{i-2}(a)\phi^{i-1}(a)$. The story of these systems began in 1906 with the discovering of the Thue-Morse sequence [Thue, 1906, Morse, 1921]. It is generated by the iteration of the substitutions $0 \mapsto 01$ and $1 \mapsto 10$ starting from 0. The study of those substitutions dynamical systems has been proved to be useful in many mathematical fields such as number theory, ergodic theory, spectral analysis and combinatorics [Queffélec, 2010, Fogg et al., 2002] and has connection to physics, notably through the study of tilings and quasicristals [Senechal, 1996]. Substitution dynamical system were also defined for tilings and multidimentional words. But some issues occur for these generalizations. For example, bi-dimensional words substitutions are easily defined when letters are sent to rectangular patterns of the same width and height, but need some "placement conditions" for other kind of patterns [Arnoux et al., 2004]. These conditions are specified using two levels of rules: a "smaller level", that sends letters to two-dimensional patterns of letters, and a "bigger level" that is used to relatively place the results of two adjacent letters.

1.1.3 Full-Synchronism and Determinism

Dynamical systems similar to cellular automata and L-systems, are referred as *full-synchronous* systems in [Arrighi and Dowek, 2012]. In these systems, the process of selecting the sites of application of the local rules is inherently deterministic. As the rules are applied at every possible position in a given object, the input is usually completely rewritten and no part of it is considered in the output.

Full-synchronism and determinism are two properties strongly related to each other, as we can see in the case of rule applications overlap. The overlap issue occurs when two (or more) local rules can be considered in the system so that they share a part of the locations of the applications. Considering these locations as resources, this means that the applications share a common resource. Definitively, there is no such overlap issue within cellular automata and L-systems as they have been presented here: each rule application focuses on a particular site (a cell or a letter) with no intersection with the other rule applications. However, in the general case, one may expect rules to express interaction between the system constituting components, and it may so happen that a single component can possibly be involved in many different interactions at the same time. In some cases, a mechanism has to be provided to choose from the set of all possible rule applications which ones are actually triggered. This mechanism is called the *rule application strategy*. Full-synchronism can be seen as a special strategy where there is no restriction and all applications have to be considered. This strategy does not comply with systems where the system components are resources to be consumed. In those cases, the strategy has to implement a *mutual exclusion* between overlapping rule applications so that at most one of them is triggered. These strategies range from sequential (only one rule is applied) to maximal-parallel (a maximal set of non-overlapping rule applications is chosen), but are all non-deterministic[2]. Maximal-parallel strategies are often considered in (bio)chemical computing systems, such as Paun Systems [Păun, 2000]. They are sometimes said synchronous, since the maximal-parallel strategy when used on systems where no overlap occurs (like cellular automata and L-systems) does coincide with full-synchronism.

However, one can remark that full-synchronism and determinism are not equivalent properties. Full-synchronism still allows one special type of non-determinism: the *rule-based non-determinism* considers non-deterministic rules, *i.e.*, rules with multiple outputs. It has already been encountered in non-deterministic extensions of cellular automata and L-systems. This non-determinism is somewhat predictable and local, in contrast with the non-determinism of maximal-parallel systems. Chapter 5 of the present document is entirely devoted to the study of rule-based non-determinism.

1.2 Dynamical Systems with Dynamic Space

The main property that makes local, synchronous and deterministic systems difficult to express is considering a dynamic space. One main feature of L-systems in contrast with cellular automata is that they can be seen as some primitive instance of dynamical systems over dynamic space. Indeed, a word can shrink or grow under

[2] A deterministic mechanism would consider the system as a whole and prevents it from being local.

the action of a L-system accordingly to the rules that sends every letter to words of arbitrary size. The local rules send each "point" of the space, *i.e.*, each letter, to a possibly empty collection of points. Conversely, the local rules of cellular automata specify only updates of each cell state, which induces a static space. If one considers words as configurations over some subset of $\mathbb{Z}$, it is clear that L-systems act not only on the states but also on the topology of a word, whereas cellular automata act only on states distributed over some fixed topology.

However, the strategy of cellular automata to work on a fixed structure is easily applicable to a wide range of spaces. In contrast, the extension of L-systems to a different nature of space is not so obvious, as already evoked with substitution systems. In general, dynamical systems over less regular (such as graphs) and dynamic spaces have been considered many times in the literature. However, no general frame of work for describing full-synchronous systems over dynamic spaces has been developed. Depending on the use case, some have focused on asynchronous systems sometimes achieving genericity, others focused on full-synchronous systems over a specific space.

1.2.1 Asynchronous and Non-Deterministic Approaches

In this section, we focus on a strategy that consists in defining first how a rule application works locally independently of any other rule application, allowing a local modification of the spacial structure, then in iterating or composing this procedure to express simultaneous applications.

A first approach worth mentioning comes from the idea of rewriting graphs by applying some replacement rules as a natural generalization of term rewriting. In this field, *graph transformations* and *graph grammars* provide a categorical framework to rewrite a part of a graph [Rozenberg, 1997]. This proposition has been recognized as being algebraic and very generic, and has been generalized to any structures exhibiting the right algebraic properties, captured by the concept of adhesive categories. Graph transformations and their extensions are intensively used in software specification using decorated graphs, where algebraic operations are performed on labels [Löwe et al., 1993], but also in molecular biology with the κ-language [Danos and Laneve, 2004]. These methods focus on defining categorical operations, that given a rule, and a site of application in a given graph, computes the resulting graph. Consequently, they are usually used to rewrite sequentially parts of a graph. The different approaches, such as Double Pushout [Corradini et al., 1997] (DPO, depicted in figure 1.6), Single Pushout [Ehrig et al., 1997] (SPO) and Sesqui Pushout [Corradini et al., 2006] (SqPO), differ mainly on how the edges between the rewritten part of a graph and the rest of the graph are handled. The dynamics of such systems are studied for the sequential application of productions using the Church-Rosser property [Ehrig et al., 1973, Ehrig et al., 2006]. In contrast with the idea of iterating applications of rules, parallel [Ehrig, 1978], amalgamated [Boehm et al., 1987] and concurrent [Ehrig and Rosen, 1980] graph transformations propose frameworks for expressing simultaneous applications of rules by making explicit what happens at the overlap. This specification results in the design of a bigger rule integrating the initial concurrent rules. Whereas these approaches permit to deal with multiple rule applications, the synchronous triggering of all the possible rule applications over the whole system and the resulting global dynamics

are not examined.

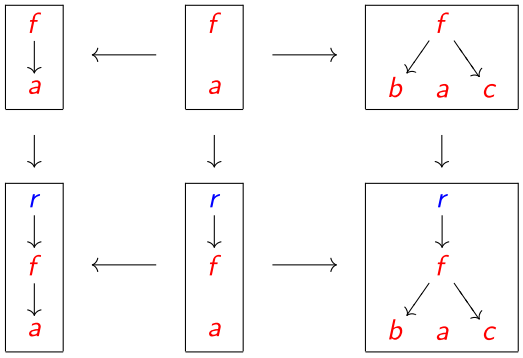


Figure 1.6: DPO rule (top), input (bottom left), output (bottom right) (source?)

In the field of graph rewriting, multiple works gave a description of some execution mechanism, but many did not achieve both synchronism and determinism. Determinism was achieved in [Salzberg et al., 2004] with a mechanism inspired by DNA transcription and folding. In this work, a sequential rewriting of a graph is specified by another graph in a machine-tape dichotomy. Practical approaches to synchronous graph rewriting were sometimes studied using a maximal-parallel execution scheme. For instance, [Sayama, 2007] specified some very general scheme to define such systems, by abstracting away the selection of sites of application and the replacement mechanisms. In this work, it is discussed that in general, synchronism is not achieved, but no specific solution is provided.

Some works used a maximal-parallel process along with some conflict resolution strategies. [Tomita et al., 2009] proposed some class of graph rewriting automata over dynamic graphs where each node has degree three. They achieve a simple presentation of their automata, by restricting their system to four types of structural rules, two for nodes and two for edges. Their update scheme uses two alternating steps of rewriting, one step for nodes and one for edges. Some of their rewrites rules are mutually exclusive, and these constraints must be taken carefully into account to ensure consistency. Similar approaches are found in [Klales et al., 2010] for the concrete problem of defining a lattice gas automata with dynamical geometry, in [Kreowski and Kuske, 2006, Mammen et al., 2010] for describing multi-agent systems, and in [Kurth et al., 2004] that defines a general graph grammar system using a maximal-parallel resolution strategy.

Even if it does not permit to reach full-synchronism, the strategy developing so far is very fruitful and very generic. It led to the recognition that the process of rule application can be parameterized by the nature of the underlying space. The MGS language develops this idea and proposes an original paradigm of programming, *interaction-based programming*, where a wide variety of structures can be rewritten by generic rules, leading to polytypism (property of programs being independent of the nature of the data structures) and allowing the expression of usual rule-based computing models (L-systems, cellular automata, mesh rewriting systems, multi-agent systems, etc.) [Spicher and Giavitto, 2017].

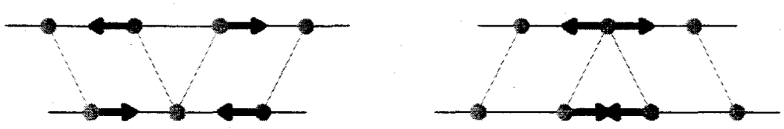


Figure 1.7: Hasslacher and Meyer steps modifying space, time goes up, from [Hasslacher and Meyer, 1998]

1.2.2 Full-Synchronous Extensions

Occurrences of full-synchronous dynamical systems with dynamic space in the literature were always tailored specifically to some fixed structure, such as graphs with some constraints. However, there is still a large variety of such systems, some designed to model some specific problem, other giving some general setting to capture a large class of systems.

Some very simple instance of such system occur in [Hasslacher and Meyer, 1998], as an extension of lattice gas automata to model the simplest non-trivial reversible particle system with dynamical space. The structure manipulated can be seen as cyclic graphs or cyclic words of varying period. Nodes are decorated by particles going left or right, both can also be found on a single node. The system is described using two steps of computations, the scattering phase, that moves particles according to their direction, and the advection phase, that create or remove space accordingly to particles going through each other. This system achieves to be full-synchronous, but it is so simple that it can be encoded in a trivial extension of Lindenmayer systems to periodic words. This system was later extended to a dynamical two-dimensional triangular lattice, but had to introduce a specific rule application strategy to preserve spatial consistency [Klales et al., 2010].

Cellular automata, originally defined on regular lattices, have also been extended to Cayley and Hyperbolic graphs [Róka, 1999, Herrmann and Margenstern, 2003, Tullio Ceccherini-Silberstein, 2010]. Yet these graphs are still regular and self-similar as they remain invariant under translations and the space is still static. Extensions to more general graphs were considered in [Papazian and Rémila, 2002, Derbel et al., 2008] with the topology remaining fixed, only the state being updated. [Smith et al., 2011] proposed a synchronous system over dynamical graphs, by fixing the set of nodes, he used local rules over the adjacency matrix of a graph to define a dynamic over the edges.

A general theory for describing full-synchronous systems over dynamical graphs came in 2012 when Arrighi and Dowek introduced causal graph dynamics, a general extension of cellular automata over port graphs, *i.e.*, bounded degree and connected graphs admitting a coordinate system provided by ports [Arrighi and Dowek, 2012]. Each node having a set of ports, on which edges connect. A causal graph dynamics is a function from port graphs to port graphs that is localizable. The properties of causal graph dynamics are shown to be similar to cellular automata. In particular, as for the Curtis Hedlund Lyndon theorem classifies cellular automata as shift-equivarient continuous mappings from configurations to configurations, causal graph dynamics are shown to be exactly the translation-equivarient uniform continuous bounded functions from port graphs to port graphs.

Another general method was developed in [Echahed and Maignan, 2017] for port graphs. It tackles the problem the other way around by specifying the systems from

the local level using overlapping rewriting rules. Two rule application can share some nodes and edges, remove some of them and add fresh ones. Rules need to be shown to be compatible with each other with respect to their overlap. When two rules overlaps, the resulting new node and edges for their overlap is duplicated, and the resulting structure is not a graph, a quotient operation is then applied to merge these elements and restore the graph structure. Subsequent works of T. Boy de la Tour and R. Echahed extends the graph transformation framework for attributed graphs using weak spans, *i.e.*, rules that specify the lack of effect on some part of its input [Boy de la Tour and Echahed, 2018]. This formalizes a notion of compatible overlapping rules, that are therefore parallelizable, which allow them to achieve full-synchronism. This notion of compatibility is named parallel coherence and is formulated in the same manner as amalgamation and parallelism in graph transformations [Boy de la Tour and Echahed, 2020].

1.3 The Global Transformation Proposition

Global transformations are a proposition of framework for expressing and studying synchronous, deterministic and local systems. Compared to previously considered works, we aim at providing

- full-synchronism: rule applications are all considered,
- dynamic space: rules can specify changes of the underlying structure,
- spatial genericity: the framework is independent of the underlying structure.

We first give an intuitive description of the global transformations proposition, then the agenda of the present document is developed. Let us now fix some vocabulary for the rest of this work.

As we want to achieve spacial genericity, we abstract away the structure by a collection C of *objects* over which a given system acts. For instance, to capture a D0L system over words on a and b, we set the collection of objects C to contain every such word.

We focus on systems specified locally by means of a collection $\mathbf{\Gamma}$ of *local rules* meant to be applied in parallel everywhere in an input object. A local rule γ in $\mathbf{\Gamma}$ is a pair $\gamma : l \mapsto r$, where l and r are objects in C called the *left-hand side* (l.h.s.) and the *right-hand side* (r.h.s.). Such a local rule specifies that any *occurrence* (or *matching*) of l in an input object o must be transformed to r. For instance, the D0L system of figure 1.4 is specified using two local rules, $\gamma_a : a \mapsto ab$ and $\gamma_b : b \mapsto a$. Given a rule $\gamma : l \mapsto r$, an occurrence of l in an object o is also called an *instance* of γ in o and its associated r.h.s. is referred as a *local result*.

The notion of *context* is not relevant in this work. Indeed, rather than considering rules with a transformed part and a context part and gathering all rule instances such that the transformed parts are mutually exclusive, we do not differentiate between the context part and the transformed part and allows rules instances to overlap in a given object. For instance, consider the rule 184 elementary cellular automaton (figure 1.2). Its rules, which are of the form $abc \mapsto d$ (for $a, b, c, d \in \{0, 1\}$), consider three cells in the l.h.s., where only the central 1-cell is meant to be transformed in a 0-cell and its neighborhood consisting of the 1-cell on its left and the 0-cell on

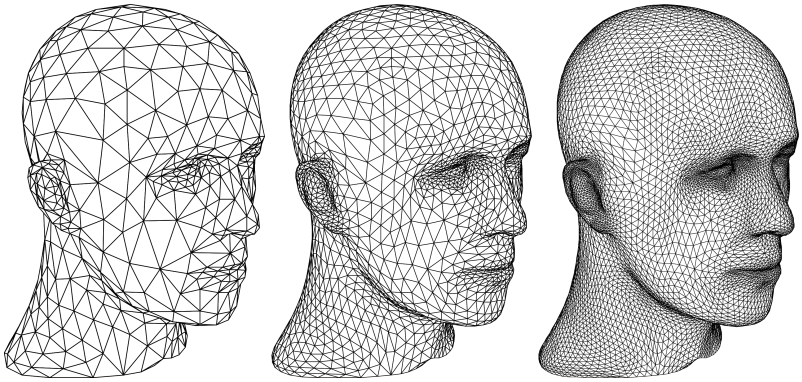


Figure 1.8: Triangular mesh refinement, from [Schröder et al., 1998]

its right is only considered as some context. In our work, as no such distinction is made, we consider the full triplet of cells 110 to be transformed in to 0. Indeed, we allow multiple overlapping rules to be applied in parallel over the same cells. For example, given a configuration containing 1100, we apply the two overlapping rules $110 \mapsto 0$ and $100 \mapsto 1$ that shares the central pair of cells 10. That pair of cells is considered to be transformed by the two rules at the same time.

1.3.1 Global Transformations

Global transformations were introduced in [Maignan and Spicher, 2015] to tackle the issue of a rule-based specification of triangular mesh refinement. Figure 1.8 taken from [Schröder et al., 1998] illustrates a common procedure consisting in replacing each triangle of a mesh by four finer triangles. Beyond the difficulty of dealing with the coordinates of newly inserted nodes (an important issue in the field of computational geometry that we simply skip from our considerations), the whole operation can be summarized as *applying in parallel everywhere in the input triangular mesh the rule pictured in figure 1.9.* The apparent simplicity of this infor-

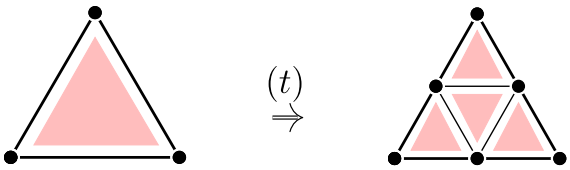


Figure 1.9: Triangle subdivision rule

mal specification hides the whole difficulty of expressing formally a full-synchronous transformation of an arbitrary object (here, a 3D mesh) involving a dynamic space. The question has been explicitly asked in [Smith et al., 2003] which proposes an iterative algorithm implementing the informal local rule of figure 1.9 over a carefully specified data-structure. This approach circumvents the difficulty of providing a computational solution from a rule-based specification. An executable rule-based specification was achieved later in [Spicher et al., 2010] where the operation has been expressed in the MGS language as the compound of two successive transformations (edge subdivisions followed by new edges insertions) to deal with the constraint of

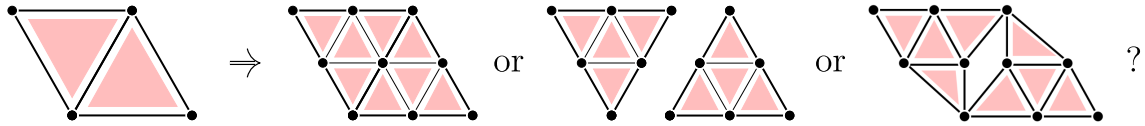


Figure 1.10: Multiple possible output for two glued triangles

using a maximal-parallel strategy. Global transformations provide a rule-based and full-synchronous solution to this problem.

The design of global transformations comes from the identification of a lack of specification in the sole data of the rule (t) of figure 1.9. Consider the mesh on the left of figure 1.10 composed of two triangles glued by a side. Observe now that any output on the right of figure 1.10 is compatible with the rule (t). Indeed, each of these outputs provides a subdivided version of the two initial triangles, the differences lying in how these refinements are combined to one another. Nothing in the rule (t) specifies this combination.

The intuition that the first output of figure 1.10 is the "right" one comes from the fact that the two applications of the rule (t) overlap and that the outputs have to overlap as well. A natural way for filling this lack of information is to express the becoming of the shared edge with an additional rule, say (e), specifying how an edge is itself subdivided. Moreover, since the issue also occurs when two triangles are glued by a vertex, a third rule, say (n), is also required to express the dynamics of this other kind of overlap. This leads to the collection of rules given in figure 1.11. Notice now that the edge rule (e) occurs six times in the triangle rule (t) (three times

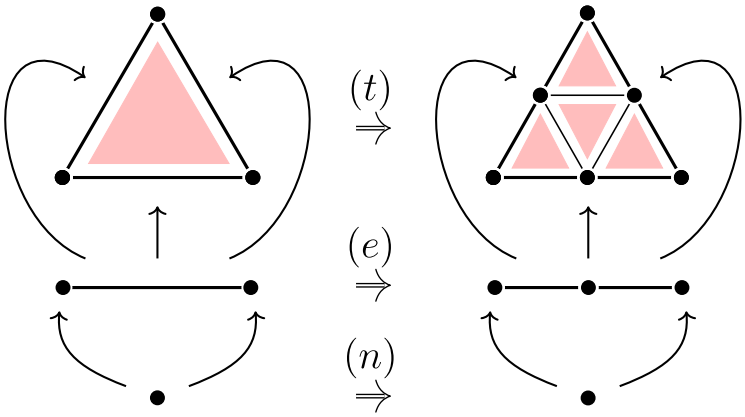


Figure 1.11: Triangular mesh refinement rule system

illustrated in the figure and the symmetric inclusions): with each occurrence of the l.h.s. of (e) in the l.h.s. of (t), an occurrence of the r.h.s. of (e) in the r.h.s. of (t) is associated. Such an association between an inclusion in the l.h.s. and an inclusion in the r.h.s. of two rules has to be seen as a new kind of rule, named *rule inclusion*: for each inclusion of a l.h.s. into another l.h.s., a *rule inclusion* specifies an associated inclusion between the two corresponding r.h.s. The same kind of rule inclusions are defined between (n) and (e), rule inclusions between (n) and (t) being obtained by transitivity. The data of the rules and the rule inclusions is called a *rule system*. As already mentioned at the end of section 1.1.2, similar rule systems were already used for the same purpose in the field of two-dimensional word substitutions.

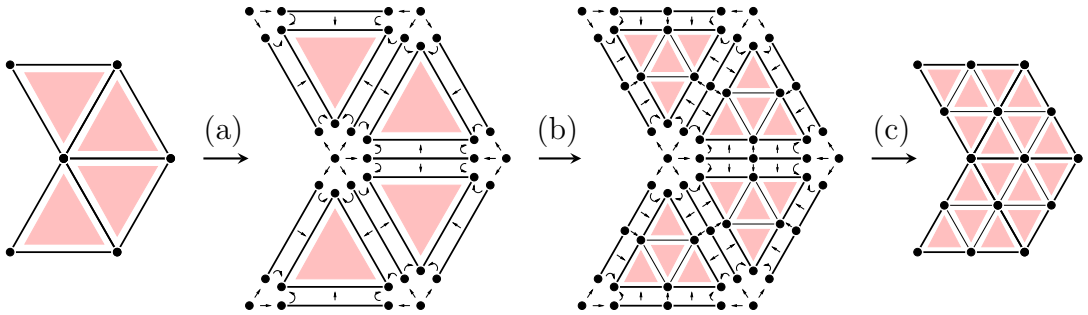


Figure 1.12: (a) Pattern matching (b) Application (c) Reconstruction

The rule system of figure 1.11 is now exhaustive, since it specifies all the missing behaviors on possible overlaps. Moreover, it is executable in the following sense. The rewriting of an input mesh consists of the rewriting of every matching of l.h.s. in the input, and it is specified by mean of three computations steps. Figure 1.12 depicts these steps for an example of input:

1. *Pattern matching step* (figure 1.12a): the input mesh is split into all the matchings of the rule l.h.s. together with all the inclusion information between these matchings. Any unmatched part of the mesh is lost.

2. *Local application step* (figure 1.12b): each found l.h.s. matching of the rules is replaced by the corresponding r.h.s.; inclusions between matchings are replaced as well with respect to the rule inclusions.

3. *Reconstruction step* (figure 1.12c): the output mesh is built by merging the r.h.s. instances using the tracked inclusion information.

Albeit being presented for the paradigmatic example of triangular mesh refinement, the given global transformation procedure is highly generic and permits to describe full-synchronous transformations over any kind of space.

1.3.2 Organization of the document

This document summarizes my research activities. Through these activities, I have participated in the development of global transformations at different levels, theoretical and practical. To support the claim of genericity of the global transformations approach, it is useful to see how common full-synchronous systems can be retrieved or original full-synchronous systems can be designed in the global transformation framework. So, for each contribution, a different kind of spatial structure has been considered as an illustration. The document is organized as follows.

Chapter 2: Formalisation of Global Transformations. Global transformations are introduced formally in their full generality. The proposed framework is indeed parameterized by the manipulated structure which is abstracted by the means of a category, and the computation steps (*pattern matching*, *local applications* and *reconstruction*) are defined using general categorical operations. The necessity of using category theory is introduced pedagogically. The definition of a global transformation is discussed in an operational fashion: the process of designing of a rule

system for a given goal is first discussed and the application of global transformations on a given object is described through the three computation steps.

The chapter exemplifies the global transformation construction by instantiating context-free and context-sensitive L-systems.

Chapter 3: "Local vs Global" Relationship. The chapter investigates deeper the formal properties of global transformations by exploring their strong relationship with the categorical concept of *Kan extension*. This permits a universal characterization of the global behavior of global transformations as an optimal extension of a rule system, and exhibits the 2-categorical features of the formalism.

The construction is exemplified with cellular automata to emphasize the relations between the local transition function and the global transition function. These relations are discussed in terms of two extensions of both local transition function and global transition function to every *partial* configurations. To capture these extensions, two settings are considered, the first one being sensible to coordinates, the second being shift invariant. The implications of the latter choice for non-commutative groups are also discussed.

Chapter 4: Computation of Global Transformations. Following a software realization, the computational features of global transformations are discussed. A *generic* algorithm is designed through the formulation of the three computation steps into simpler building blocks. In this perspective, these simpler building blocks are recomposed to give rise to an *online* rewriting algorithm.

These algorithmic considerations are done through the prism of graph rewriting. In particular, the chapter focuses on global transformations of graphs that preserve injective morphisms, and gives a local criterion on the rules that implies this global property. At a technical level, the construction focuses on categories of presheaves which are presented as a unified language to work with a wide range of structures including graph-like structures.

Chapter 5: Non-Deterministic Global Transformations. The document ends with the study of rule-based non-determinism in global transformations. The definition of non-deterministic global transformations using categories where objects are families of objects of an underlying category is experimented. The obtained construction maps objects to families of objects and therefore cannot be iterated using the usual composition. To solve this issue, the monadic features of the construction are investigated and composition is defined by shifting to another 2-categorical context, the associated 2-category of Kleisli, where the construction is shown to still correspond to global transformations.

Non-local features of some non-deterministic spatialized systems are also discussed through the notion of spatial correlation. It is shown that, albeit being characterized through local rules, dependencies in the non-deterministic choices may lead to some kind of incompatibilities, which induce non-local effects. Similarly to injection preservation in chapter 4, the correlation-freeness of a global transformation is shown to be decidable only by means of the local rules.

Chapter 2

Formalisation of Global Transformations

So far, global transformations have been described informally through the simple but paradigmatic example of triangular mesh refinement. In this chapter, the formal basic definition of global transformations is introduced in almost full generality. Indeed, the proposed framework is made to be generic over the nature of the objects to be transformed. So this framework can be used to describe any full-synchronous systems by capturing precisely the desired properties (locality, determinism, full-synchronism) which we advocate to be independent of the underlying space. The abstraction over the nature of the objects is achieved by using category theory. This general mathematical theory of structures and their relations was introduced in [Eilenberg and MacLane, 1945], and is particularly pertinent to this end. The general concept of a category simply consists of a collection of objects which are not manipulated *per se* but through the relationships they have with each other, captured by collections of morphisms between them. We make use of this algebraic definition to represent spatial structures by a category where objects are pieces of spaces and morphisms are inclusions between these pieces. In contrast with ordered sets, where the question "is an element smaller/included in another?" admits only a yes or no answer, an object in a category can be related to another object in multiple ways, which permits to express that some data can be found at multiple places in some space.

In section 2.1, we gently introduce both category theory and the global transformation formalism. The necessary categorical notions are motivated by the practical purpose of representing and rewriting a given structure. These notions are illustrated with the example of triangle mesh refinement given in the introduction (section 1.3.1) for an intuitive understanding. To support this claim of genericity, the second part of the chapter is devoted to revisit the computing model of deterministic Lindenmayer systems. In section 2.2, we instantiate the framework to describe context-free Lindenmayer systems to give a strong and formal insight on each ingredient of global transformations setting. Section 2.3 introduces context-sensitive Lindermayer systems and provides a global transformation translation. We conclude with a short discussion in section 2.4.

2.1 Global Transformations

This section is devoted to defining global transformations. Notions and notations, borrowed from category theory, are introduced by the necessity of formally expressing the basic concepts of global transformations. For helping the understanding, the reader is invited to get in mind the example of triangular mesh refinement given in section 1.3.1. However, the definitions are given in full generality relating each requirement to a categorical construction, based on the following correspondence:

- Categories *versus* spatial structure: the kind of objects we want to address are spatially extended, as are the triangular meshes. As such, they can be decomposed into pieces and the objects can be related to one another by identifying the pieces they have in common. Given a class of objects, we aim at formally capturing this spatial relationship with a category of *pieces of objects*.

- Functors *versus* rules systems and transformations: we aim at rewriting these objects by applying local rules specifying local evolution of identified sub-parts. This mechanism relies on the statement: the way the local output occurs in the global output depends on the way the local part occurs in the global input. This monotony principle is perfectly captured by a functor.

- Comma category *versus* pattern matching: the application of rules requires not only to identify all the occurrences of the rules l.h.s. in the input object, but also to track the inclusions between these occurrences. This procedure is directly described through the definition of a particular comma category.

- Colimit *versus* reconstruction: the output is obtained by merging all the r.h.s. of the rule applications by using the gluing information given by the inclusions between the rule occurrences. The colimit is a categorical construction expressing this idea of merge.

2.1.1 A Structure for Locality

Our first objective is to define a formal object able to describe the spatial extension of the objects to be transformed. We propose to rely on the notion of locality for this purpose, allowing to consider local spatial parts of these objects. Since we want to abstract away from the nature of these objects, let us call these spatial extended objects "things" for a while.

To describe locality, we need to formalize the notion of a thing being a part of another thing. Of course, when a thing a is a part of a thing b which is itself a part of a thing c, then a is a part of c. But this preorder structure is not enough. We also need to track *where* things occur in each other. So when a occurs at a place p in b and b occurs at a place q in c, we need a way to know where a consequently occurs in c. This is exactly formalized by the notion of category. In this formalism, we would talk of three *objects* a, b and c, and two *morphisms*[1] $p : a \to b$ and $q : b \to c$ being composed to give $q \circ p : a \to c$.

[1] In the literature, they are also often called *homomorphisms*, because they are indeed homomorphisms of some sort in many examples, or even *arrow*, to refer to an abstract pair with a source and destination.

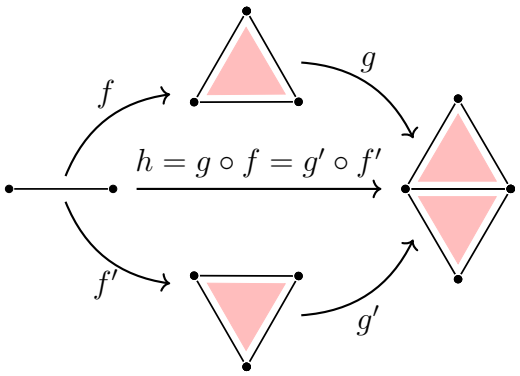


Figure 2.1: Two equal composition of morphisms for triangular meshes

Definition 2.1.1. A *category* $\mathbf{C}$ consists of:

- a collection[2] abusively denoted $\mathbf{C}$ whose elements are called *objects*;
- for any $a, b \in \mathbf{C}$, a collection $\mathrm{Hom}_{\mathbf{C}}(a, b)$ whose elements are called *morphisms*, the notations $f : a \to b \in \mathbf{C}$ and $a \xrightarrow{f} b \in \mathbf{C}$ meaning $f \in \mathrm{Hom}_{\mathbf{C}}(a, b)$[3], we omit to mention $\mathbf{C}$ when it does not lead to any confusion;
- for every three objects a, b, c, a composition law $\mathrm{Hom}_{\mathbf{C}}(b, c) \times \mathrm{Hom}_{\mathbf{C}}(a, b) \to \mathrm{Hom}_{\mathbf{C}}(a, c)$ denoted $\circ$, *i.e.*, sending any $f : a \to b$ and $g : b \to c$ to $g \circ f : a \to c$.

Moreover, the composition law has to be *associative* (for any $f : a \to b$, $g : b \to c$ and $h : c \to d$, $h \circ (g \circ f) = (h \circ g) \circ f$) with an *identity morphism* $\mathrm{id}_x : x \to x$ for each $x \in \mathbf{C}$ (for any $f : a \to x$ and $g : x \to b$, $\mathrm{id}_x \circ f = f$ and $g \circ \mathrm{id}_x = g$).

This notion of locality allows for example to relate triangular meshes to one another at a spatial level. In the associated category, say $\mathbf{M}$[4], objects are triangular meshes and morphisms are inclusions between these meshes. Figure 2.1 illustrates three of such meshes, an edge e, a triangle t (pictured twice), and a pair of adjacent triangles p, and five mesh inclusions $f, f' : e \to t$ identifying edges in t, $g, g' : t \to p$ identifying the two triangles of p, and $h : e \to p$ identifying the gluing edge in p. Morphisms f and g are defined so that the identified edge in t by f corresponds to the gluing edge in p with respect to g. Formally, this corresponds to the equation $h = g \circ f$. The same holds for f' and g' for the other triangle in p giving $h = g' \circ f'$.

2.1.2 Respecting Locality

Given a category $\mathbf{C}$ describing a locality structure, what does it mean for a transformation F on $\mathbf{C}$ to respect locality? Consider a local part a of a global input b by $p : a \to b$. It is expected from such a transformation to treat a locally, thus producing the output $F(a)$ for this part independently, so that the global output $F(b)$ should contain this local output. Of course, *the way the local output occurs in the*

[2]The cardinality of these categories is of less importance in our work; we will not distinguish between small and large categories.

[3]Given $f : a \to b$, we will sometimes use the term *domain* to refer to a and *codomain* for b

[4]The formal definition of category $\mathbf{M}$ is not addressed here, but a similar category will be investigated in example 4.1.3 of chapter 4.

global output depends on the way the local part occurs in the global input. Formally, this means that there should be an associated morphism $F(p) : F(a) \to F(b)$ for each such p in a coherent way. This is exactly captured by (homo)morphisms of categories, so-called *functors*.

Definition 2.1.2. A *functor* $F : \mathbf{C} \to \mathbf{D}$, with $\mathbf{C}$ and $\mathbf{D}$ two categories, consists of:

- a map abusively denoted $F : \mathbf{C} \to \mathbf{D}$ mapping the objects of $\mathbf{C}$ to objects of $\mathbf{D}$, and
- a map abusively denoted $F : \mathrm{Hom}_{\mathbf{C}}(a, b) \to \mathrm{Hom}_{\mathbf{D}}(F(a), F(b))$ for any $a, b \in \mathbf{C}$ mapping the morphisms of $\mathbf{C}$ to morphisms of $\mathbf{D}$ accordingly.

Moreover, these maps have to preserve identity morphisms ($\forall a \in \mathbf{C}, F(\mathrm{id}_a) = \mathrm{id}_{F(a)}$) and composition ($\forall a, b, c \in \mathbf{C}, \forall f : a \to b, \forall g : b \to c, F(g \circ f) = F(g) \circ F(f)$). We call these constraints the *functoriality conditions* of F.

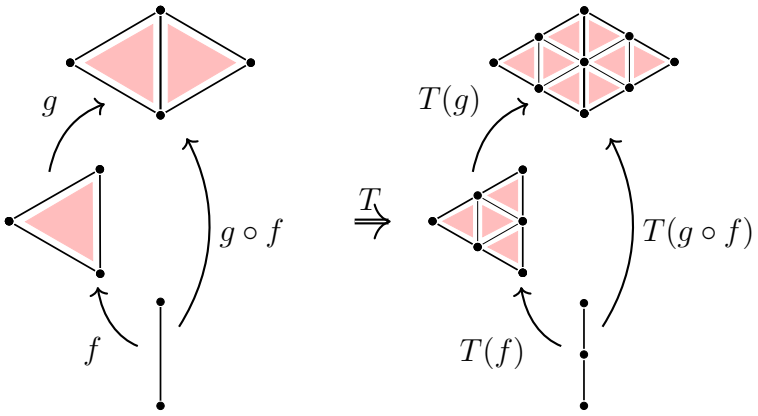


Figure 2.2: Triangular mesh refinement as a functor T

The principle of monotonicity of functors is well illustrated by the example of triangular mesh refinement: given a triangular mesh, the refinement can be locally tracked on any subpart of that mesh. Figure 2.2 depicts the functoriality of this transformation, say $T : \mathbf{M} \to \mathbf{M}$. This example considers morphisms f and g of figure 2.1. The figure illustrates how the refinement $T(e)$ of the isolated edge e (occurring in the triangle t by $f : e \to t$) is included in the refinement $T(t)$ of the triangle t. This inclusion of $T(e)$ in $T(t)$ is the image $T(f)$ of f by T. The same holds for the inclusion g identifying the left triangle of p, which is mapped to $T(g)$, and for the inclusion $g \circ f$ identifying the inner edge in p, which is mapped to $T(g \circ f)$. The triangular mesh refinement functor T not only gives an output $T(f)$, $T(g)$ and $T(g \circ f)$ for each of these morphisms, but also respects the composition. That is, $T(g \circ f)$ must send the subdivided edge to the overlap between the two neighboring subdivided triangles in p, *i.e.*, $T(g \circ f) = T(g) \circ T(f)$.

In general, we certainly want a global transformation to be a functor from $\mathbf{C}$ to $\mathbf{C}$. Moreover, since this global transformation is specified through local rules, we must specify things within these rules. Let us first describe these rules in the categorical setting given so far. For this purpose, consider a set $\mathbf{\Gamma}$ of rules, and two functions $l : \mathbf{\Gamma} \to \mathbf{C}$ and $r : \mathbf{\Gamma} \to \mathbf{C}$ identify for each rule $\gamma \in \mathbf{\Gamma}$, its l.h.s.

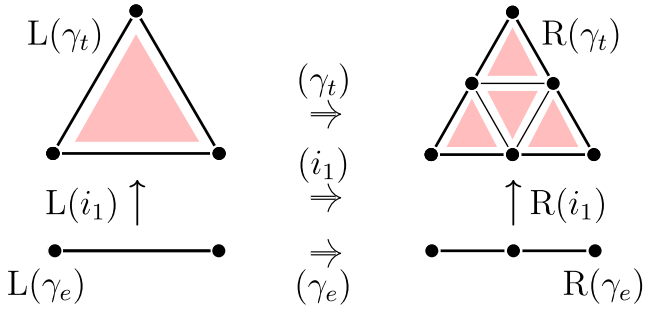


Figure 2.3: Rule inclusion between an edge and a triangle for mesh refinement

$l(\gamma)$ and its r.h.s. $r(\gamma)$. For simplicity, we consider l injective[5]. Inclusions between l.h.s. can be addressed backward through l^{-1} and then forward through r. Thus, for any two rules γ and γ', any morphism $p : l(\gamma) \to l(\gamma')$ in $\mathbf{C}$ is mapped to $r(l^{-1}(p)) : r(\gamma) \to r(\gamma')$ in $\mathbf{C}$. This naturally promotes $\mathbf{\Gamma}$ to a category $\mathbf{\Gamma}$ whose morphisms, called *rule inclusions*, are completely determined in terms of those of $\mathbf{C}$ by $l^{-1}(p) : \gamma \to \gamma'$. Consequently, l and r are lifted to functors of signature $\mathbf{\Gamma} \to \mathbf{C}$. Moreover, l is lifted to a so-called *fully faithful* functor, that is, respecting $\mathrm{Hom}_{\mathbf{\Gamma}}(\gamma, \gamma') \cong_l \mathrm{Hom}_{\mathbf{C}}(l(\gamma), l(\gamma'))$. This fully specifies the data describing a *rule system*.

Definition 2.1.3. A *rule system* T consists of:

- a category $\mathbf{C}_T$,
- a category $\mathbf{\Gamma}_T$ whose objects and morphisms are called *rules* and *rule inclusions* respectively,
- a fully faithful injective functor $\mathrm{L}_T : \mathbf{\Gamma}_T \to \mathbf{C}_T$ called the *l.h.s. functor*,
- a functor $\mathrm{R}_T : \mathbf{\Gamma}_T \to \mathbf{C}_T$ called the *r.h.s. functor*.

The indices are omitted when it does not lead to any confusion.

Let us illustrate this definition with the rule system for the triangular mesh refinement triangulation, given on figure 1.11 of section 1.3.1. Figure 2.3 focuses on one of its rule inclusion. Here, rule γ_e (resp. γ_t) expresses the refinement of an edge (resp. a triangle). The figure pictures the l.h.s. $\mathrm{L}(\gamma_e)$ (resp. $\mathrm{L}(\gamma_t)$) of rule γ_e (resp. γ_t) as the isolated edge e (resp. the simple triangle t). Similarly, the r.h.s. $\mathrm{R}(\gamma_e)$ (resp. $\mathrm{R}(\gamma_t)$) is illustrated as $T(e)$ (resp. $T(t)$) the corresponding refined version. The rule inclusion $i_1 : \gamma_e \to \gamma_t$ is one of the six morphisms of $\mathrm{Hom}_{\mathbf{\Gamma}}(\gamma_e, \gamma_t) \equiv \mathrm{Hom}_{\mathbf{M}}(e, t)$ (the three different edges of t and their symmetric inclusions). This rule inclusion is the rule that sends the inclusion $\mathrm{L}(i_1) : \mathrm{L}(\gamma_e) \to \mathrm{L}(\gamma_t)$ of the edge to the bottom edge of the triangle, to the inclusion $\mathrm{R}(i_1) : \mathrm{R}(\gamma_e) \to \mathrm{R}(\gamma_t)$ between their subdivided counterparts.

A last remark is that this definition of a rule system is equivalent to requiring $\mathbf{\Gamma}_T$ to be a *full subcategory* of $\mathbf{C}_T$, *i.e.*, a category whose object is a subcollection of objects of $\mathbf{C}_T$ and that includes all morphisms between these objects. In this view, R_T specifies the behavior of the transformation on $\mathbf{\Gamma}_T \subseteq \mathbf{C}_T$ only, thus being

[5]The non-injective case is discussed in chapter 3.

equivalent to a partial functor from $\mathbf{C}_T$ to $\mathbf{C}_T$. The functor L_T is then the inclusion functor of $\mathbf{\Gamma}_T$ in $\mathbf{C}_T$. This complies with the idea that the behavior of a global transformation on all possible inputs is generated by its behavior on some selected "small" inputs. Below, we will define a global transformation as a *well-defined rule system*, *i.e.*, a rule system that permits to consistently rewrite every object in the category $\mathbf{C}_T$.

2.1.3 Designing Locality

Designing a rule system T consists in choosing the appropriate category $\mathbf{C}_T$ and taking it into account in the choice of the rules.

The objects of $\mathbf{C}_T$ are usually clear: at least all possible inputs, outputs, l.h.s. and r.h.s.. They are often accompanied by their natural sub-parts. The real matter is to design the morphisms of $\mathbf{C}_T$. *A morphism specifies that the result of its source has to be found in the result of its target.* This main guide is illustrated in sections 2.2.2 and 2.3 for DL systems.

While inclusions between rules are rarely considered in traditional rewriting systems, they are of first importance in the design of rules in a rule system: they indicate how locality explains the construction of the result. The leitmotiv is indeed that *any local relation between r.h.s. in the result should be justified by a local relation between the l.h.s. in the input.* This relation between two l.h.s. $\mathrm{L}_T(\gamma_1)$ and $\mathrm{L}_T(\gamma_2)$ is not necessarily a direct inclusion, and a third l.h.s. $\mathrm{L}_T(\gamma_{12})$ might be needed. If the relation is an overlap, this relation has the form $\gamma_1 \xleftarrow{p} \gamma_{12} \xrightarrow{q} \gamma_2$. If it is an interaction in a common context, the form can be $\gamma_1 \xrightarrow{p} \gamma_{12} \xleftarrow{q} \gamma_2$. The former has been used for illustration in sections 2.2.2 and 2.3.

Once a global transformation specified by means of its local rules, it is possible to apply it on any given input. This is done by (1) collecting all the occurrences of l.h.s. in the input, then by (2) passing to the corresponding r.h.s., and (3) by constructing an object containing the required r.h.s. In this whole process, the morphisms need to be conserved, since they specify how the l.h.s. were found in the input, and thus how the r.h.s. must be recomposed. We give here a general description of the operations involved in the specification of the result. The important role of the morphisms in this process is only slightly highlighted. Their purpose is precised in the following section, where we proceed to formulate DL systems in this setting.

2.1.4 Pattern Matching

The process of finding all occurrences of the l.h.s. in an input object X is often called *pattern matching*. In this process, one considers every place p in X where the l.h.s. of a rule γ occurs, *i.e.*, every pair $\langle \gamma \in \mathbf{\Gamma}_T, p : \mathrm{L}_T(\gamma) \to X \rangle$. We call such a pair, a *rule instance* of γ. Note that for any rule inclusion $q : \gamma \to \gamma'$ in $\mathbf{\Gamma}_T$, an instance of γ' in X at some place $p' : \mathrm{L}_T(\gamma') \to X$ necessarily implies an instance of γ at the place $p = p' \circ \mathrm{L}_T(q)$. These data form a particular case L_T/X of *comma category*[6].

Definition 2.1.4. The category F/d, with $F : \mathbf{C} \to \mathbf{D}$ a functor and $d \in \mathbf{D}$, has:

- for objects, every pair $\langle c \in \mathbf{C}, f : F(c) \to d \rangle$,

[6] A comma category is normally defined between two functors to the same category, the case of a single object being represented by a constant functor.

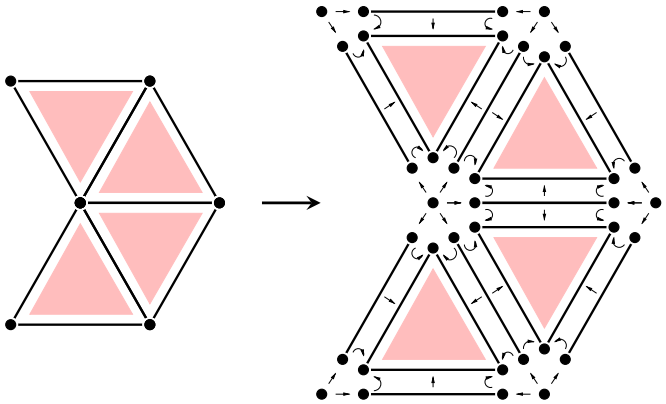


Figure 2.4: Pattern matching for mesh refinement

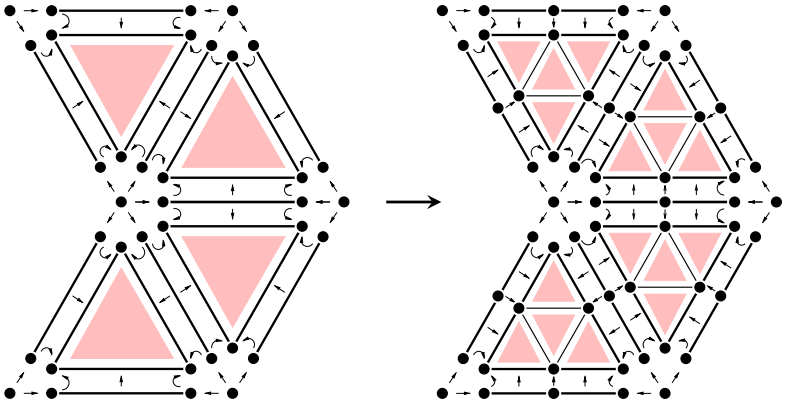


Figure 2.5: Application of local rules

- for morphisms from $\langle c_1, f_1\rangle$ to $\langle c_2, f_2\rangle$, pairs $\langle g : c_1 \to c_2, f_2\rangle$ with $f_1 = f_2 \circ F(g)$.

Given two morphisms $\langle g, f_2\rangle : \langle c_1, f_1\rangle \to \langle c_2, f_2\rangle$ and $\langle h, f_3\rangle : \langle c_2, f_2\rangle \to \langle c_3, f_3\rangle$ the composite $\langle h, f_3\rangle \circ \langle g, f_2\rangle$ is defined to be $\langle h \circ g, f_3\rangle$. This category comes naturally with a *projection functor* $\mathrm{Proj}[F/d] : F/d \to \mathbf{C}$ mapping each object $\langle c \in \mathbf{C}, f : F(c) \to d\rangle$ to c and each morphism $\langle g, f\rangle$ to g.

For L_T/X, the functor $\mathrm{Proj}[\mathrm{L}_T/X] : \mathrm{L}_T/X \to \mathbf{\Gamma}_T$ maps each rule instance found in the input X to the actual rule. Since R_T gives for each rule its r.h.s., we can compose these functors to obtain $\mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/X] : \mathrm{L}_T/a \to \mathbf{C}_T$ that gives the pieces that have to be patched together, and how they have to be patched to construct the result. Figure 2.4 depicts the pattern matching step for triangular mesh refinement. The left part of the figure corresponds to the functor $\mathrm{L} \circ \mathrm{Proj}[\mathrm{L}/X]$, so-called diagram of matchings, where X is the mesh on the left of the figure. The diagram of local results for theses matchings, obtained by replacing each rule and rule inclusions l.h.s. by their r.h.s., and formally defined as $\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/X]$, is depicted in the right part of figure 2.5.

2.1.5 Constructing The Result

As well exemplified on the right part of figure 2.5, the functor $\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/X]$ point out the instances of r.h.s. to merge to build the output of the operation. The inclusions

between those instances are to guide such a merge. The categorical concept of *colimit* and its various extensions are usually understood for this purpose:

> The intuitive general idea of a colimit is that it defines an object obtained by sewing together the objects of the diagram, according to the instructions given by the morphisms of the diagram. [7]

The *diagram* refers here to the functor $\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/X]$ and the *instructions* to the inclusions between the r.h.s. instances. We introduce the concept of colimit in the context of global transformations.

Taking the aforementioned simplistic view of a preorder, the result might be described as the smallest way to have all the r.h.s. of the rule instances patched together; we would talk of least upper bound. In the categorical setting, upper bounds are generalized as *cocones* and least upper bounds as *colimits*.

Upper bounds are usually defined for a *subset of elements*. For categories, morphisms and multiplicities must also be taken into account. For this reason, cocones are defined for a *multiset of objects and morphisms*. Such a collection is specified as an arbitrary functor into the category of interest, so-called *diagram*, the source category playing the role of an index set. In our case, a cocone can be interpreted as a candidate for being the output of the transformation: it is an object together with a set of morphisms showing where each instance r.h.s. appears. Moreover, anytime two instances r.h.s. are related by rule inclusion, it is required to also be the case in the candidate.

Definition 2.1.5. A *cocone* C for a diagram $F : \mathbf{I} \to \mathbf{C}$ consists of:

- an object of $x \in \mathbf{C}$ called the *apex* also abusively denoted C, and
- a morphism $C_i : F(i) \to x$ for each object $i \in \mathbf{I}$, called *component morphism* (or simply *component*) of the cocone

such that $F(f) \circ C_i = C_j$ for any $f : i \to j$ in $\mathbf{I}$.

The *colimit* is the "best" candidate, that is, the common factor appearing in all candidates. As such, it should logically appear in any cocone u at the unique exact place devised by the morphisms $\{C_i\}_i$ indicating where the parts must be found.

Definition 2.1.6. A *colimit* of a diagram $F : \mathbf{I} \to \mathbf{C}$ is a cocone C such that for any cocone D there is a unique *mediating morphism* $m : C \to D$, *i.e.*, a morphism such that $D_i = m \circ C_i$ for any $i \in \mathbf{I}$.

In general, there might be more than one colimit. However, any two colimits for the same diagram F are necessarily isomorphic. For this reason, one often talks about "the" colimit $\mathrm{Colim}(F)$ as if it is unique, even when it is not the case. This is either just a loose notation or the choice of one precise colimit in the collection of isomorphic colimits.

Let us illustrate this definition for triangular meshes. Figure 2.6 (based on figure 2.1) describes on the left a diagram. This diagram selects 3 objects (the isolated edge e and the triangle t twice) and two morphisms ($f, f' : e \to t$) identifying one of the edges of each triangle. On the right of figure 2.6, a cocone for this diagram

[7] `https://ncatlab.org/nlab/show/colimit`, July, 2022

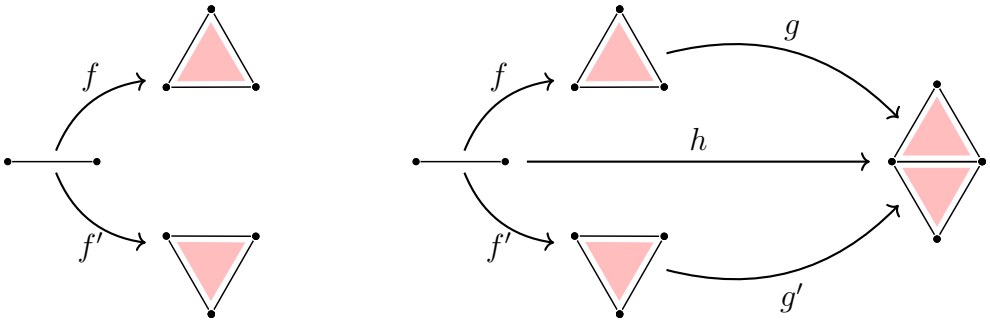


Figure 2.6: Illustration of colimit for triangular meshes.

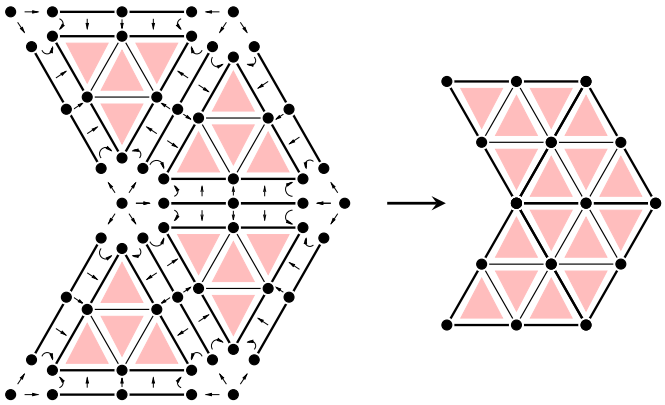


Figure 2.7: Reconstruction for mesh refinement

is pictured. The apex consists of p, the triangular mesh composed of two triangles glued by an edge. The cocone components are $g, g' : t \to p$ and $h : e \to p$, and express how each object of the diagram is found in p. These components are coherent with each other since $h = g \circ f = g' \circ f'$. This cocone is a colimit in the sense that any other cocone identifying the three objects of the diagram and respecting the inclusion f and f' must include an occurrence of p which is uniquely determined. In this example, the shape of the diagram expresses how to sew two objects (here, the two triangles) by a common sub-part (here, an edge). This basic construction is called a *pushout*.

Figure 2.7 depicts a more elaborated colimit than a simple pushout, corresponding to the reconstruction step of the local results obtained in figure 2.5 for triangular mesh refinement. The gluing of every mesh in the diagram of local results $R \circ \mathrm{Proj}[\mathrm{L}/X]$ consists of taking the colimit of this diagram. Altogether, the result of applying a rule system T on X is the best object containing all the r.h.s. of the rule instances found in X, arranged as required by rule inclusions.

2.1.6 Definition of Global Transformations

Gathering all the ingredients introduced so far, we almost get the definition of global transformations. Indeed, we have described the expected computation on an input object $X \in \mathbf{C}$, transformed into an object $T(X) \in \mathbf{C}$ defined by:

$$T(X) \cong \mathrm{Colim}(\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/X]). \tag{2.1}$$

When the colimit exists for any such X, this gives rise to a function from $\mathbf{C}$ to $\mathbf{C}$. However, we expect this function to respect the monotony principle of global transformation, so to extend to the morphisms of $\mathbf{C}$. Luckily, all the constructions of equations 2.1 are functorial leading the function to be indeed the exact functor meeting our expectations. Let us detail a bit this functor.

Consider two input objects $X, X' \in \mathbf{C}$ such that X occurs in X' by $p : X \to X'$. Our objective is to define the morphism $T(p) : T(X) \to T(X')$ expressing how $T(X)$ occurs in $T(X')$.

At first, notice that anything matched in X is also matched in X'. Indeed, any matching $\langle \gamma, \mu : \mathrm{L}(\gamma) \to X \rangle$ in X induces a matching $\langle \gamma, p \circ \mu : \mathrm{L}(\gamma) \to X' \rangle$ in X'. Similarly, any matching inclusion $\langle i : \gamma' \to \gamma, \mu : \mathrm{L}(\gamma) \to X \rangle$ of L/X induces a matching inclusion $\langle i : \gamma' \to \gamma, p \circ \mu : \mathrm{L}(\gamma) \to X' \rangle$ of L/X'. These remarks lead to the definition of a functor $\mathrm{L}/p : \mathrm{L}/X \to \mathrm{L}/X'$ expressing that L/X is included in L/X'.

The direct consequence of the inclusion of the comma categories, is that the r.h.s. diagram of X is included in the r.h.s. diagram of X', which is expressed formally by:

$$R \circ \mathrm{Proj}[\mathrm{L}/X'] \circ \mathrm{L}/p = R \circ \mathrm{Proj}[\mathrm{L}/X].$$

We are finally left to relate the two colimits $T(X)$ and $T(X')$. The main remark is that the restriction of the colimit $T(X')$ to the image of L/p is a cocone D of the diagram $R \circ \mathrm{Proj}[\mathrm{L}/X]$ defined by:

$$D_{\langle \gamma, \mu : \mathrm{L}(\gamma) \to X \rangle} := T(X')_{\langle \gamma, p \circ \mu \rangle}.$$

Since $T(X)$ is a colimit, by definition 2.1.6, there is a unique $p' : T(X) \to T(X')$ such that $D_{\langle \gamma, \mu \rangle} = p' \circ T(X)_{\langle \gamma, \mu \rangle}$ for any $\langle \gamma, \mu \rangle \in \mathrm{L}/X$. In other words, there is only one way, captured by the mediating morphism $T(p) := p'$, to relate $T(X)$ into $T(X')$ while respecting the way $T(X)$ and $T(X')$ have been built from a collection of r.h.s.

We finally get the definition of global transformation.

Definition 2.1.7. A global transformation is a rule system T such that the eponymous functor:

$$T(-) \cong \mathrm{Colim}(\mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/-])$$

is well defined.

2.1.7 Computation is up to isomorphism

The functor T is only characterized up to isomorphism. Indeed, the colimit were defined to be a cocone, *i.e.*, with a collection of morphisms from the objects of the diagram, with a unique mediating morphism into any other cocone. Such categorical approaches tend to only rely on the *external* relations between objects rather than on their *internal* structure. Consequently, objects having the same relations with other objects are considered the same. This is made precise by the notion of isomorphism, which generalize bijections between sets.

Definition 2.1.8. A morphism $f : a \to b$ in a category $\mathbf{C}$ is an *isomorphism* if it is *invertible*. More precisely, if it exists another morphism denote $f^{-1} : b \to a$ in $\mathbf{C}$ such that $f \circ f^{-1} = id_b$ and $f^{-1} \circ f = id_a$. When there is such *isomorphism* between two objects a and b, they are said to be *isomorphic*.

In the case of colimits, consider two colimits C and C' of the same diagram. The mediating morphisms $m : C \to C'$ and $n : C' \to C$ actually form an isomorphism. Indeed, observe that $n \circ m : C \to C$ and $m \circ n : C' \to C'$ are both mediating morphisms. But note that the identities morphisms id_C and $id_{C'}$ are also mediating morphisms, so by uniqueness of mediatings we have $n \circ m = id_C$ and $m \circ n = id_{C'}$ so m and n are inverses to each other and C and C' are isomorphic.

In category theory, isomorphic objects are considered the same. Indeed, functors cannot distinguish between such objects, as they must send isomorphic objects to isomorphic objects. This is a direct consequence of the fact that functors preserves identities and composition: given a functor $F : \mathbf{C} \to \mathbf{D}$ and an isomorphism $f : a \to b$ with inverse $f^{-1} : b \to a$ in $\mathbf{C}$, as $f \circ f^{-1} = id_b$, we must have $F(f) \circ F(f^{-1}) = id_{F(b)}$, and $F(f^{-1}) \circ F(f) = id_{F(a)}$ holds in the same manner.

For these reasons, we did not depict symmetries in the rule system of triangle mesh refinement (figure 1.11). There is many such symmetries: an undirected edge have two isomorphism, the first being the identity and the second consist in the flip along its vertices, and a triangle have six isomorphisms, the identity, two rotations, and three flips along one of its edges. Albeit being not depicted in the figures, the formalism consider each of these symmetries as rule isomorphisms, which prevents application of isomorphic rules to produce duplicated local results.

As L-systems acts on finite words and two finite words are isomorphic only when they are equal, these considerations are in fact not necessary for the moment. However, we will come back to these notions in the following chapters. The remainder of the chapter is devoted to illustrate exhaustively this definition by encoding Lindenmayer systems as global transformations.

2.2 Deterministic Context-free L-Systems

In this section, our goal is to build a global transformation which reproduces the exact same behavior of a given deterministic context-free Lindenmayer system $\mathcal{L}$. First, we introduce the original definition of these systems together with the required notations. We then define in all generality the category $\mathbf{W}_S$ enriching any set of words S to capture the notion of locality for words. Then, we define the global transformation $T\mathcal{L}$ associated to $\mathcal{L}$. Finally, the soundness of the translation of $\mathcal{L}$ into $T\mathcal{L}$ is shown.

2.2.1 L-System Original Definition

As introduced in chapter 1, Lindenmayer systems (L-system) are a formal system allowing the generation of formal languages by a parallel string rewriting mechanism. Traditionally, they are presented as a triplet giving (1) an alphabet used to build words, (2) a set of rewriting rules called *productions* associating to some sub-words a local evolution, and (3) an axiom[8], that is, an initial word. Derivations are then obtained by, starting from the axiom, applying all the productions synchronously wherever they appear. In this context, being deterministic means that all the productions always agree so that there is only one possible result. Well-known classes

[8]Since the axiom does not play any role in our study, it has been omitted from our definitions without any modification of the original spirit of L systems.

of such DL systems are *D0L systems* and *DIL systems*. We focus on the former class, the latter is studied in section 2.3.

Notations. For an alphabet Σ and $n \in \mathbb{N}$, Σ^*, Σ^n, $\Sigma^{<n}$, $\Sigma^{\leqslant n}$ and $\Sigma^{\geqslant n}$ express respectively the set of finite words, of words of length n, of words of length lower than n, of words of length at most n, and of words of length at least n. The symbol ε denotes the empty word and $|w|$ the length of some word w. Concatenation of a sequence of words $w_1, \ldots, w_n$ is denoted by juxtaposition $w_1 w_2 \ldots w_n$ or by the product $\prod_{i=1}^{n} w_i$. When some set of words are involved in a concatenation, the result is a set in the common way. For any two words u and v, and a non-negative integer p, $u \preceq_p v$ denotes that u is included in v at position p, *i.e.*, there exist $\alpha \in \Sigma^p$ and $\beta \in \Sigma^*$ such that $v = \alpha u \beta$.

D0L systems are the simplest and firstly defined class of L systems. The 0 of D0L means that the rewriting is done context-free: each letter is replaced independently of the others. This leads to the following definition, strongly inspired by [Rozenberg and Salomaa, 1980].

Definition 2.2.1. A *D0L system* $\mathcal{L}$ consists of:

- a set $\Sigma_{\mathcal{L}}$ called the *alphabet*, and
- a mapping $\mathrm{P}_{\mathcal{L}} : \Sigma_{\mathcal{L}} \to \Sigma_{\mathcal{L}}^*$.

The fact $\mathrm{P}_{\mathcal{L}}(a) = w$ is written $a \to_{\mathcal{L}} w$ and referred to as a *production* of $\mathcal{L}$. The *yield* function $\Rightarrow_{\mathcal{L}}$ is given by

$$w \Rightarrow_{\mathcal{L}} w' \quad :\Leftrightarrow \quad w = a_1 \ldots a_n \ \wedge \ w' = \mathrm{P}_{\mathcal{L}}(a_1) \ldots \mathrm{P}_{\mathcal{L}}(a_n).$$

Clearly, the yield function $\Rightarrow_{\mathcal{L}}$ is the monoid endomorphism of $\Sigma_{\mathcal{L}}^*$ generated by $\to_{\mathcal{L}}$. Conversely, it is easy to see that any endomorphism of the free monoid $\Sigma_{\mathcal{L}}^*$ defines a D0L system.

Example 2.2.1. Consider the D0L system π that was given in the introduction (see figure 1.4) as a model of growth of a filamentous organism. π defined on the alphabet $\Sigma_\pi = \{a, b\}$ by the productions: $a \to_\pi b$ and $b \to_\pi ab$. For example, the evolution starting from the string *abbab* is given by the iteration of the associated yield function:

$$abbab \Rightarrow_\pi bababbab \Rightarrow_\pi abbabbababbab \Rightarrow_\pi bababbababbabbababbab \Rightarrow_\pi \ldots$$

2.2.2 The Category of Words $\mathbf{W}_S$

As previously mentioned, the category over which a global transformation act formalizes the notion of locality for the objects to be transformed and their associated results. In the case of D0L systems, these objects are finite words and the morphisms (meaning "being a part of") express how a word appears as a subword of another, also known as factors. This is coherent with the guideline indicated in section 2.1.3. Indeed, for any D0L system on any alphabet Σ, the result of a word always includes the result of its subwords. For the sake of brevity, we define the category for an arbitrary subset of words S on an arbitrary alphabet Σ.

Definition 2.2.2. Let $\mathbf{W}_S$, where $S \subseteq \Sigma^*$ for some Σ, be the category having:

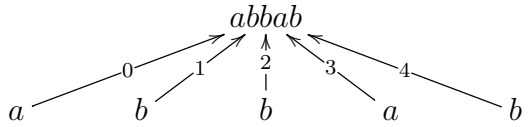
- for objects all the considered words of S, and
- for any two words $u, v \in S$, $\mathrm{Hom}_{\mathbf{W}_S}(u, v) := \{\, p \in \mathbb{N} \mid u \preceq_p v \,\}$,
- for any two $p : u \to v$ and $q : v \to w$, $q \circ p := (p + q : u \to w)$.

It is trivial to check that $\mathbf{W}_S$ is a category. Particularly, the composition says that if u is included in v at position p which is itself included in w at position q, then u is included in w at position $p + q$. This operation is definitely associative, and the identity inclusion is 0: any word is included in itself at position 0. Notice that if $S \subseteq S'$, $\mathbf{W}_S$ is a full subcategory of $\mathbf{W}_{S'}$.

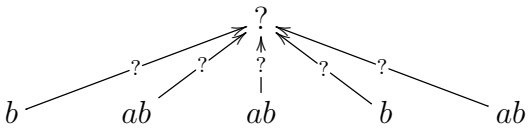
2.2.3 Designing the Rules of $T\mathcal{L}$

In a global transformation, while rules specify, as usual, how a sub-part locally evolves, rule inclusions play the important role of tracking during the rewriting how the input is decomposed and how the output is recomposed. Let us illustrate how it works with the transition $abbab \Rightarrow_\pi bababbab$ of example 2.2.1.

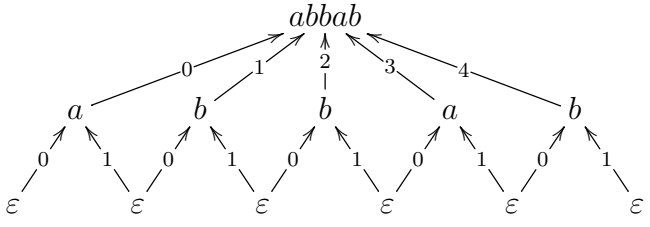
The following diagram presents how $abbab$ is decomposed by pattern matching with the l.h.s. of the productions: each occurrence of each letter is identified by its position through a morphism of $\mathbf{W}_{\Sigma^*}$.



The local application of the productions on each occurrence exhibits all the subwords composing the result.

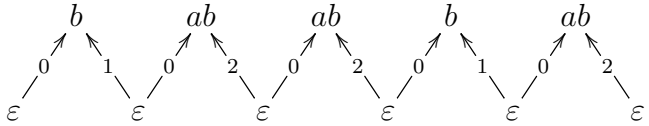


However, without any more relationship between them, there is no mean to determine in which order they have to be assembled to compose the expected result. As prescribed in section 2.1.3, the leitmotiv is to track local relations from l.h.s. to r.h.s. Here, two r.h.s. should be consecutive exactly when their l.h.s. are in the input. This can be captured by realizing that the empty word ε is included between any two adjacent letters so that it can play the role of a common part between l.h.s. that must be mapped to a common part between r.h.s. For, let us consider an additional rule with ε as l.h.s. The following diagram is obtained by tracking the occurrences of the l.h.s. (including now ε) in $abbab$ together with the relations (*i.e.*, morphisms) between these occurrences.

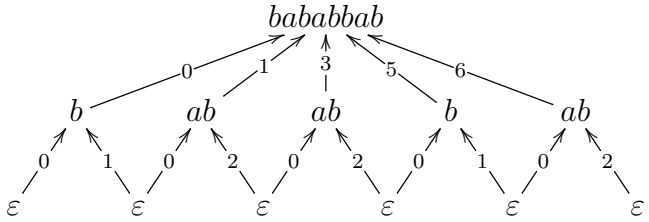


Let us illustrate how to read this diagram by considering the second ε on the left. It represents the occurrence of ε between the first two letters a and b of $abbab$. From this ε, the morphism $1 : \varepsilon \to a$ represents how it is also a sub-occurrence of the first occurrence of a, precisely the empty word occurring on the right of that $a = a\varepsilon$. Similarly, the other morphism $0 : \varepsilon \to b$ shows how it also appears as a sub-occurrence of the occurrence of $b = \varepsilon b$. As such, this diagram provides a representation of the comma category $\mathrm{L}_{T\mathcal{L}}/abbab$ defined in section 2.1.4.

Since the yield function of a D0L system is a monoid endomorphism, the additional rule is naturally completed with ε as r.h.s. to get $\varepsilon \Rightarrow_\pi \varepsilon$. The associated inclusion rules are designed as follows: the left-sided occurrence $0 : \varepsilon \to a$ of ε in a one-letter word a is sent to the leftmost occurrence $0 : \varepsilon \to \mathrm{P}_\pi(a)$ of ε in the associated r.h.s. $\mathrm{P}_\pi(a)$, and similarly for the right-sided occurrences. In our example, these rules transform the previous pattern matching situation into the following diagram.



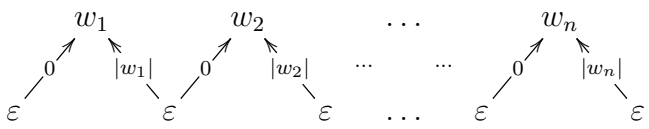
The previous question marks can now be resolved. In categorical terms, the cocone consisting of the word *bababbab* together with the morphisms from the r.h.s. turns out to be the colimit in $\mathbf{W}_{\Sigma^*_\pi}$ of that diagram, that is, the smallest word containing exactly all the r.h.s. in the right order. Diagrammatically:



2.2.4 The Global Transformation $T\mathcal{L}$

In definition 2.2.1, the concatenation definitely appears as the underlying engine of the yield function, and the previous picture illustrates how it is captured as a colimit in $\mathbf{W}_{\Sigma^*}$. In fact, this situation reproduces exactly in all generality, and even holds when restricting to some subsets of words $S \subseteq \Sigma^*$. So theorem 2.2.1 serves as the underlying engine in the definition of $T\mathcal{L}$.

Theorem 2.2.1. *Let $w_1, \ldots, w_n$ be a non-empty sequence of words of $S \subseteq \Sigma^*$. The word $w_1 \ldots w_n$, if present in S together with ε, is the colimit in $\mathbf{W}_S$ of:*



the morphisms $0 : \varepsilon \to w_1$ and $|w_n| : \varepsilon \to w_n$ being optional.

Proof. The given diagram can be formally described by the functor $F : \mathbf{I} \to \mathbf{W}_S$ where:

- the set of objects of $\mathbf{I}$ is $\{\frac{1}{2}, 1, 1+\frac{1}{2}, \ldots, n, n+\frac{1}{2}\}$,

- $F(i+\frac{1}{2}) := \varepsilon$ for all $i \in \{0,\dots,n\}$, and
- $F(i) := w_i$ for all $i \in \{1,\dots,n\}$,

for the object parts of $\mathbf{I}$ and F and, for their morphism parts:

- $\mathrm{Hom}_{\mathbf{I}}(i,i') := \{(i,i')\}$ if $i' \in \{1,\dots,n\}$ and $|i'-i| = \frac{1}{2}$,
- $\mathrm{Hom}_{\mathbf{I}}(i,i') := \emptyset$ if $i' \in \{1,\dots,n\}$ and $|i'-i| > \frac{1}{2}$,
- $F((i-\frac{1}{2},i)) = 0 \quad : \varepsilon \to w_i$ for all $i \in \{1,\dots,n\}$,
- $F((i+\frac{1}{2},i)) = |w_i| : \varepsilon \to w_i$ for all $i \in \{1,\dots,n\}$.

By definition 2.1.6, a colimit is a particular cocone. For the considered functor, a cocone is given by a word u and a collection of morphisms $\{\eta_{u,i} : F(i) \to u\}_{i\in\mathbf{I}}$ respecting the conditions of definition 2.1.5 for each morphism in $\mathbf{I}$. For any $i \in \{0,\dots,n-1\}$, the morphism $(i+\frac{1}{2},i) : (i+\frac{1}{2}) \to i$ imposes the equality $\eta_{u,(i+\frac{1}{2})} = \eta_{u,i} \circ F((i+\frac{1}{2},i)) = \eta_{u,i} + |w_i|$ and the morphism $(i+\frac{1}{2},i+1) : (i+\frac{1}{2}) \to (i+1)$ the equality $\eta_{u,(i+\frac{1}{2})} = \eta_{u,(i+1)} \circ F((i+\frac{1}{2},(i+1))) = \eta_{u,(i+1)} + 0$. Altogether, this means that $\eta_{u,(i+1)} = \eta_{u,i} + |w_i|$. By definition of $\mathbf{W}_S$, $\eta_{u,i} : w_i \to u$ is the position of w_i in u so the previous equality means that w_i and w_{i+1} appears consecutively in u. So u contains the word $w_1 \dots w_n$ at position $\eta_{u,1}$. This immediately shows that $w_1 \dots w_n$ together with the obvious morphisms is a cocone. Moreover, it is the excepted colimit since for any cocone u, we have $g_{w,u} = \eta_{u,1} : w_1 \dots w_n \to u$ as required by definition 2.1.6. Finally, the two mentioned morphisms are optional in the sense that they do not contribute to the concatenation operation. They simply state that the empty word has to appear at the borders of the colimit, which is in fact trivially true for any word. □

The empty word ε might seem too minimalistic as a common subpart, and one might consider this as not natural. This is an extreme case, the case where there is no context. Bigger "common parts" are considered for proper DIL systems. In all cases, the important locality relation transported from l.h.s. to r.h.s. here should seem natural: the relation of subwords being consecutive in a word.

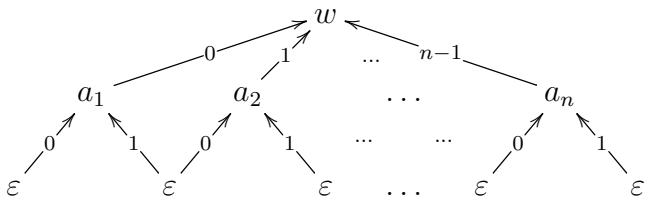
Definition 2.2.3. The global transformation $T\mathcal{L}$, where $\mathcal{L}$ is a D0L system, is the one where: $\mathbf{C}_{T\mathcal{L}}$ is $\mathbf{W}_{\Sigma_{\mathcal{L}}^*}$, $\mathbf{\Gamma}_{T\mathcal{L}} = \mathbf{W}_{\Sigma_{\mathcal{L}}\cup\{\varepsilon\}}$ is the full subcategory of $\mathbf{W}_{\Sigma_{\mathcal{L}}^*}$ with objects $\Sigma_{\mathcal{L}} \cup \{\varepsilon\}$, $\mathrm{L}_{T\mathcal{L}}$ is the associated inclusion functor, and $\mathrm{R}_{T\mathcal{L}}$ is given by:

- $\mathrm{R}_{T\mathcal{L}}(\varepsilon) := \varepsilon$,
- for any object $a \in \Sigma_{\mathcal{L}}$, $\mathrm{R}_{T\mathcal{L}}(a) := \mathrm{P}_{\mathcal{L}}(a)$,
- for any $a \in \Sigma_{\mathcal{L}}$, $\mathrm{R}_{T\mathcal{L}}$ maps the inclusion $0 : \varepsilon \to a$ to $0 : \varepsilon \to \mathrm{P}_{\mathcal{L}}(a)$,
- for any $a \in \Sigma_{\mathcal{L}}$, $\mathrm{R}_{T\mathcal{L}}$ maps the inclusion $1 : \varepsilon \to a$ to $|\mathrm{P}_{\mathcal{L}}(a)| : \varepsilon \to \mathrm{P}_{\mathcal{L}}(a)$.

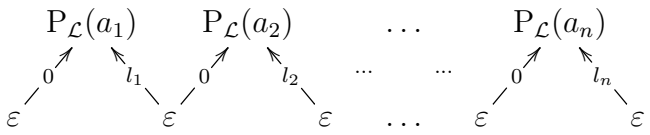
This definition is sound in the sense that $T\mathcal{L}$ mimics $\mathcal{L}$ step by step.

Theorem 2.2.2. *For any $w \in \Sigma_{\mathcal{L}}^*$, $w \Rightarrow_{\mathcal{L}} T\mathcal{L}(w)$.*

Proof. Suppose that $w = a_1 \dots a_n$ with $n = |w|$. We want to show that $T\mathcal{L}(w) = \mathrm{P}_\mathcal{L}(a_1) \dots \mathrm{P}_\mathcal{L}(a_n)$. The pattern matching $\mathrm{L}_{T\mathcal{L}}/w$ of section 2.1.4 takes the following form.



$T\mathcal{L}(w)$ is defined as the colimit of the r.h.s. diagram $\mathrm{R}_{T\mathcal{L}} \circ \mathrm{Proj}[\mathrm{L}_{T\mathcal{L}}/w]$:



where l_i denotes the length of $\mathrm{P}_\mathcal{L}(a_i)$. This diagram is similar to the one of theorem 2.2.1 which states that the colimit, a.k.a. $T\mathcal{L}(w)$, is the concatenation of the r.h.s. in the right order $\mathrm{P}_\mathcal{L}(a_1) \dots \mathrm{P}_\mathcal{L}(a_n)$ as expected. Notice finally that the result still holds when $n = 0$. In this case, the diagram reduces to a unique object ε. The colimit of such a one-object diagram is the object itself, that is ε, so that $T\mathcal{L}(\varepsilon) = \varepsilon$. □

2.3 Deterministic Context-sensitive L-Systems

While D0L systems correspond to context-free rewriting, DIL systems provide a context-dependent rewriting where the fate of a letter is influenced by its neighbors. In this section, we propose to capture DIL systems as global transformations to show that global transformations, without any more elaboration, are able to deal with contexts. We first define DIL systems, then we translate that definition in terms of global transformations.

2.3.1 DIL Systems

DIL systems are (the deterministic) part of a larger class of L systems called, *L systems with interactions* (IL systems), which were introduced to model biological organisms where cells can interact with each other.

More specifically, a DIL system comes with a signature which consists of two non-negative integers, say p and q, so that, the evolution of a letter depends on its p left neighbors and its q right neighbors. This leads to the following definition, strongly inspired by [Rozenberg and Salomaa, 1980].

Definition 2.3.1. A *DIL system* $\mathcal{L}$ of *signature* (p, q) (or shortly a *$D(p, q)L$ system*), with $p, q \in \mathbb{N}$, consists of:

- a set $\Sigma_\mathcal{L}$ called the *alphabet*, and
- a mapping $\mathrm{P}_\mathcal{L} : \Sigma_\mathcal{L}^{\leqslant p} \times \Sigma_\mathcal{L} \times \Sigma_\mathcal{L}^{\leqslant q} \to \Sigma_\mathcal{L}^*$.

The fact $\mathrm{P}_{\mathcal{L}}(\alpha, a, \beta) = w$ is written $(\alpha, a, \beta) \to_{\mathcal{L}} w$ and referred to as a *production* of $\mathcal{L}$. The *yield* function $\Rightarrow_{\mathcal{L}}$ is given by

$$w \Rightarrow_{\mathcal{L}} w' \quad :\Leftrightarrow \quad \begin{cases} w = a_1 \dots a_n \\ w' = w_1 \dots w_n \\ (a_{i-p} \dots a_{i-1}, a_i, a_{i+1} \dots a_{i+q}) \to_{\mathcal{L}} w_i \quad \forall i \in [1, n] \end{cases}$$

where $a_j = \varepsilon$ for any invalid position $j \notin [1, n]$.

This definition is very intuitive and extends naturally definition 2.2.1 of D0L systems, identified to D(0, 0)L systems. Notice that, in a D(p, q)L system, thanks to the context, a letter is able to evaluate its distance to the left (resp. right) border within the range of its context, that is, by a maximum of p (resp. q).

Example 2.3.1. Let us update the D0L system π of Example 2.2.1 to forbid any b to divide when it only has a's around, modeling for example an over-consuming of resources by young cells preventing the division of older cells. The resulting system σ turns out to be of type D(1, 1)L for checking neighbor states with productions:

$$(a, b, a) \to_\sigma b \quad (\varepsilon, b, a) \to_\sigma b \quad (a, b, \varepsilon) \to_\sigma b \quad (\_, a, \_) \to_\sigma b \quad (\_, b, \_) \to_\sigma ab$$

where the underscores stand for "any other context". Starting from the same initial string *abbab*, the system σ gives rise to the derivation:

$$abbab \Rightarrow_\sigma bababbb \Rightarrow_\sigma bbbbababab \Rightarrow_\sigma abababababbbbbb \Rightarrow_\sigma \dots$$

Adding a b at the end of the starting string gives the derivation:

$$abbabb \Rightarrow_\sigma bababbabab \Rightarrow_\sigma bbbbabababbbb \Rightarrow_\sigma abababababbbbabababababab \Rightarrow_\sigma \dots$$

2.3.2 A Category for Words with Borders

The translation of a DIL system into a global transformation is an exercise very similar to the previous one. Definition 2.2.3 only needs to be updated to deal with contexts, particularly at the borders. For this, we introduce an extra symbol $\star \notin \Sigma$ to represent the border, to be able to identify if a subword lies near the border of a word. For any $\alpha \in \Sigma^*$, $\star\alpha\star$ denotes the *word* α with its two borders, $\star\alpha$ (resp. $\alpha\star$) is the *subword* α stuck to the left (resp. right) border, and α is a proper *subword* without any border constraints. These situations are exactly captured by the category $\mathbf{W}_{\{\star,\varepsilon\}\Sigma^*\{\star,\varepsilon\}}$. Notice how words and subwords are distinguished in this setting. Particularly, a word $\star\alpha\star$ has no morphism to any other word $\star\beta\star$ because the symbol $\star$ can not be found in the middle of these words. This lack of morphisms is coherent with DIL systems, since the result of a word $\star\beta\star$ is not required to include the result of $\star\alpha\star$ anytime α is a subword of β. This is because the letters near the borders can evolve in different ways, as it is illustrated in the two derivations of example 2.3.1. When considering a D(p, q)L system, the most that can be said about a word $\star u\star$ containing a subword $vwx \in \Sigma^p\Sigma^*\Sigma^q$ is that the result for u includes the result of w. The morphisms of the category $\mathbf{W}_{\{\star,\varepsilon\}\Sigma^*\{\star,\varepsilon\}}$ are thus completely coherent with this situation.

2.3.3 DIL Systems as Global Transformations

Now that we have defined the locality structure for words with borders, we are able to define the global transformation. Following the program presented in section 2.1.3, we first notice that, in contrast with DOL systems, a minimum amount of information is required to express a DIL system as a global transformation: not proper words of size less than $p+q$ cannot be transformed since the required context is not known. This leads us to the following definition.

Definition 2.3.2. The global transformation $T\mathcal{L}$, where $\mathcal{L}$ is a D(p,q)L system for some $p, q \in \mathbb{N}$, is the one where $\mathbf{C}_{T\mathcal{L}} = \mathbf{W}_S$ and $\mathbf{\Gamma}_{\mathcal{L}} = \mathbf{W}_{S'}$ are both full subcategories of $\mathbf{W}_{\{\star,\varepsilon\}\Sigma_{\mathcal{L}}^*\{\star,\varepsilon\}}$ with:

$$S := \star{\Sigma_{\mathcal{L}}}^{<(p+q)}\star \ \cup \ \{\star,\varepsilon\}{\Sigma_{\mathcal{L}}}^{\geqslant(p+q)}\{\star,\varepsilon\},$$

and

$$S' := \star{\Sigma_{\mathcal{L}}}^{<(p+q)}\star \ \cup \ \star{\Sigma_{\mathcal{L}}}^{p+q} \ \cup \ {\Sigma_{\mathcal{L}}}^{p+q}\star \ \cup \ {\Sigma_{\mathcal{L}}}^{p+q} \ \cup \ {\Sigma_{\mathcal{L}}}^{p+q+1},$$

$\mathrm{L}_{\mathcal{L}}$ is the inclusion functor induced by $S' \subseteq S$, and $\mathrm{R}_{\mathcal{L}}$ is given by:

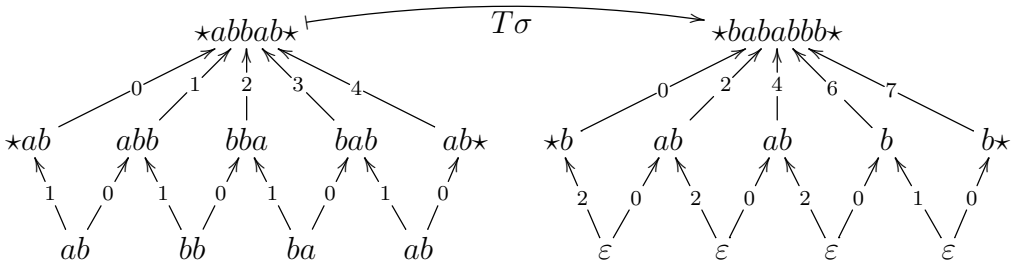
1. For any $w \in {\Sigma_{\mathcal{L}}}^{<(p+q)}$, $\mathrm{R}_{\mathcal{L}}(\star w\star) := \star w'\star$ with $w \Rightarrow_{\mathcal{L}} w'$.
2. For any $a_1, \ldots, a_{p+q} \in \Sigma_{\mathcal{L}}$,

$$\mathrm{R}_{\mathcal{L}}(\star a_1 \ldots a_{p+q}) := \star \left(\prod_{i=1}^{p} \mathrm{P}_{\mathcal{L}}(a_1 \ldots a_{i-1}, a_i, a_{i+1} \ldots a_{i+q}) \right),$$

$$\mathrm{R}_{\mathcal{L}}(a_{p+q} \ldots a_1\star) := \left(\prod_{i=q}^{1} \mathrm{P}_{\mathcal{L}}(a_{i+p} \ldots a_{i+1}, a_i, a_{i-1} \ldots a_1) \right) \star .$$

3. For any $(\alpha, a, \beta) \in {\Sigma_{\mathcal{L}}}^p \times \Sigma_{\mathcal{L}} \times {\Sigma_{\mathcal{L}}}^q$, $\mathrm{R}_{\mathcal{L}}(\alpha a \beta) := \mathrm{P}_{\mathcal{L}}(\alpha, a, \beta)$.
4. For any $w \in {\Sigma_{\mathcal{L}}}^{p+q}$, $\mathrm{R}_{\mathcal{L}}(w) := \varepsilon$.
5. For any $\alpha \in {\Sigma_{\mathcal{L}}}^{p+q}$ and $a \in \Sigma_{\mathcal{L}} \cup \{\star\}$, $\mathrm{R}_{\mathcal{L}}$ maps the inclusion $0 : \alpha \to \alpha a$ to $0 : \varepsilon \to R_{\mathcal{L}}(\alpha a)$ and the inclusion $1 : \alpha \to a\alpha$ to $|R_{\mathcal{L}}(\alpha a)| : \varepsilon \to \mathrm{R}_{\mathcal{L}}(\alpha a)$.

In this definition, case (1) gives a particular attention to the finite set of words with length lower than $p + q$ which are explicitly transformed. Longer words are managed by cases (2-5). Let us illustrate these cases with the following picture describing the derivation of $abbab \Rightarrow_\sigma bababbb$ of example 2.3.1 as computed by its categorical counterpart $T\sigma$.



Case (2) deals with left and right borders respectively: a unique rule transforms all letters at a distance lower than p (resp. q) of the left (resp. right) border. Here,

the two such border matchings are $0 : \star ab \to \star abbab\star$ and $4 : ab\star \to \star abbab\star$. These rules are particularly responsible for the presence of the left and right symbols $\star$ in the result. For example, $\mathrm{R}_{T\sigma}(\star ab) = \star \mathrm{P}_\sigma(\varepsilon, a, b) = \star b$. Case (3) is in charge of proper internal subwords using $\mathrm{P}_\mathcal{L}$ directly. The matching $2 : bba \to \star abbab\star$ gives rise to the r.h.s. $\mathrm{R}_{T\sigma}(bba) = \mathrm{P}_\sigma(b, b, a) = ab$. Finally, case (4) deals with the common part of the previous rules to prepare the concatenation. Each such common part, *e.g.*, $bba \overset{1}{\leftarrow} ba \overset{0}{\rightarrow} bab$, is mapped to ε with respect to the inclusion rules of case (5). In our example, it is mapped to $ab \overset{2}{\leftarrow} \varepsilon \overset{0}{\rightarrow} b$ which specifies that ab and b have to be consecutive in the result: abb at position 4. As easily observed, the whole r.h.s. situation coincides with the expectations of theorem 2.2.1 (without the optional morphisms) so that the output is the concatenation of the r.h.s. in the right order. The soundness of definition 2.3.2 holds in the exact same way as for theorem 2.2.2; the proof is not reproduced here.

2.4 Final Discussion

In this chapter, we have introduced global transformations as a categorical formalism capturing full synchronous rewriting systems. The inherent issue of dealing with overlapping rule applications is turned into the main feature of the setting to specify parallel rewriting. The core concept behind this feature is rule inclusions that specify how the result of any overlapping rules should have their result patched together. Consequently, rules inclusions describe how the resulting object is structured from the topology of the input object. The other feature of global transformations is the use of generic operations that are pattern-matching, local application and reconstruction to compute the rewriting of any object.

The genericity of the approach comes from the use of the language of categories that permit to define rewriting in any category. This chapter, together with the introduction, formalized a categorical setting for word rewriting, and an informal picture for mesh rewriting. This shows how customizable are global transformations. Indeed, any spatialized data, with some notion of parts and inclusions between these parts, can be described with a category. Using morphisms as a way to place objects *relatively* to each other, permits to describe any dynamical space without the need of a global addressing. This approach contrasts with Causal Graph Dynamics [Arrighi and Dowek, 2012], where some addressing is necessary. Consequently, in such systems it must be shown that a computation is translation-equivarient.

The step of computation is generic too, as the three operations in the computation of a global transformation are defined for any category. Pattern matching has already been seen as a generic operation [Spicher and Giavitto, 2017]. On the contrary, the reconstruction part is often an *ad hoc* operation coming with the underlying data structure (*e.g.*, the concatenation for strings). In global transformations, the genericity of reconstruction is captured by colimits, a very general notion of quotient/amalgamation. Such a use of colimits has already been investigated in SPO and DPO techniques for sequential and parallel graph rewriting [Grzegorz et al., 1999]. Global transformations can be compared to these techniques and their extensions [Maignan and Spicher, 2015].

The strength of global transformations is then to point out the parallel between pattern matching and reconstruction as a kind of monotony property based on the

structure of locality. This is closely related to continuity, and a part of future works is devoted to understand this relation with topology. In particular, it would be interesting to extend to global transformations the topological characterization of cellular automata [Hedlund, 1969], as it has been done for causal graph dynamics [Arrighi and Dowek, 2012], two other sources of inspiration of global transformations.

Definition 2.1.7 is a basic and straight implementation of the spirit of global transformations informally introduced in section 1.3. Two particular points have been avoided previously but deserve discussions: requiring the l.h.s. functor L to be a fully faithful embedding, and choosing the colimit for implementing the reconstruction. About the l.h.s. functor, the constraint to be a fully faithful embedding permit to simplify the presentation since a rule system can be seen as a partial functor from $\mathbf{C}$ to $\mathbf{C}$, given by $\mathrm{R} \circ \mathrm{L}^{-1}$. However, there is no real need for the functor to be full, that is surjective for the morphisms between the l.h.s.: any missing morphism would be managed as any other morphism by definition 2.1.7. In other words, the rule system does need to be exhaustive as soon as the missing cases can be recovered from the given ones. Similarly, being a faithful embedding asks the rule system to deal with l.h.s. and l.h.s. inclusions injectively. But, it is not an issue as soon as the multiple rules with same l.h.s. have coherent r.h.s. with each other. This allows to design rule systems with finer semantics on the rules. We consider examples where L is an arbitrary functor in chapter 3.

For the second constraint, using colimits for the reconstruction step is natural, but could be too restrictive when the desired colimits do not exist in the category at work. While it is not a problem for our proposition of translation of L systems as global transformations, the formal definition of the mesh refinement illustration is not so simple. There are many ways to solve the issue, all being more or less related to the definition of colimits, but all based on the construction of a cocone. In chapter 4, we investigate the idea of using the colimits of another category. More accurately, we consider global transformations on presheaves (including the case of the trianguar meshes) with only injective morphisms, and we *report* the colimit computation in the category of presheaves with arbitrary morphisms, where colimits always exist.

Chapter 3

"Local vs Global" Relationship

The global behavior of a global transformation was presented in chapter 2 in an *operational* fashion: the three computational steps of *pattern-matching*, *local application* and *reconstruction* are used to rewrite a given object. This chapter investigates the *relation* between this global behavior and the local presentation, through the question of "How is the global transformation functor an extension of the rule system?". To this end, we emphasize that global transformations happen to be connected to a well-known universal construction named *Kan extensions*. Kan extensions [MacLane, 2013] were originally defined in [Kan, 1958] as a way to extend functors to a "broader" domain. This notion is ubiquitous in category theory, as many universal constructions such as adjoint functors, limits and colimits can be expressed as Kan extensions.

In particular, this gives a new insight about the local and global relationship in cellular automata. Indeed, these systems are usually presented either as a local behavior extended to a global and uniform behavior, or as a continuous uniform global behavior for the appropriate topology [Ceccherini-Silberstein and Coornaert, 2010, Hedlund, 1969]. In this chapter, we offer a third point of view tailored for global transformations and instantiated on cellular automata. While categories are generalizations of partially ordered sets (posets), cellular automata can be fully treated using Kan extensions in terms of posets only. The corresponding construction permit to make clear the involved structures and to introduce Kan extensions in a very simple and non-categorical way. In a second time, the setting is enriched to take into account the shift operations, allowing to go from an absolute positioning to a relative positioning. It requires a transition to the more general setting of categories, which appears to be closer to the case of global transformations.

The rest of the chapter is organized as follows. In section 3.1, we motivate the general definition of Kan extensions through the study of the relationship between the global behavior and the rule system in global transformations. Then section 3.2 focuses on the case of cellular automata. We recall the direct definitions of cellular automata on groups, local transition function, global transition function, shift action, and also consider the counterparts of these functions on arbitrary partial configurations. Two such counterparts are considered, one minimal extension to partial configurations, the *coarse transition function* that only applies the local rules wherever it can, and a maximal extension, the *fine transition function*, that tries to deduce the local results even where the local rules cannot be applied. This bigger picture permit to show that the various local/global relations between these

objects are all captured by left and right Kan extensions, and provides an alternative definition of these objects in section 3.3 for the coarse function and section 3.4 for the fine function. In these sections, the set of partial configurations is first lifted to a poset and the functions are characterized by Kan extensions of monotonous functions. Then the poset of partial configurations is extended into a category using the shift operation to relate partial configurations. All the proofs are provided in details to show how all the addressed concepts can be easily manipulated once understood. In the final section, we discuss on the contributions of this work. We also comment the link with Curtis-Hedlund theorem and discuss how the use of category allows a smooth transition from the cellular automata case to the more general case of global transformations.

3.1 Global Transformations as Kan extensions

Global transformations were defined in chapter 2 as a general framework for extending some local behavior. A rule system T is a local presentation of a global transformation over some category $\mathbf{C}$. It consists of two functors L and R from a category of rules $\mathbf{\Gamma}$ into $\mathbf{C}$. The l.h.s. functor L specifies the subcategory of $\mathbf{C}$ over which the local behavior is defined, and the r.h.s. functor R specifies their local results. Then the extended behavior T is defined in a *pointwise* manner, *i.e.*, by its value over each object $X \in \mathbf{C}$, using the following formula:

$$T(X) \cong \mathrm{Colim}(\mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/X]) \tag{3.1}$$

In chapter 2 we have studied this *pointwise* definition in an "operational" fashion with the help of three operations: pattern-matching, local application and reconstruction. These steps explain how the behavior $T(c)$ on an arbitrary object $c \in \mathbf{C}$ is obtained from the behavior of the rules as the amalgamation of the local results given by the instances of the rules in c.

But there is another way to characterize how T is an extension, by observing that Equation (3.1) corresponds to some special case of a ubiquitous categorical concept: *Kan extensions*. The concept of Kan extension emerges from the quest of expressing how some functor is an extension of another one only by the means of a *universal property*.

3.1.1 Global Transformations and Universal Properties

Universal properties are a categorical way to identify an object (up to isomorphism) without referring to the internal structure of those objects. Definition using a universal property[1] follows the following pattern.

1. *Category*: one considers the category where the object to identify lies; the morphisms are used to compare it with other objects of the same nature.

2. *Property*: a characterization of the object to identify is chosen; we call *candidates* those objects of the category for which the property holds.

[1] A formal definition of universal properties exists but it is not reproduced here since we only need to refer to this notion informally.

3. *Universality*: among these candidates, the "best" object is characterized up to isomorphism through a universal morphism, *i.e.*, a unique morphism from (resp. to) this object to (resp. from) every other candidate, which preserve the property in some sense.

For example, we have seen in chapter 2 that the colimit is a cocone over a diagram determined up to isomorphism by the fact that it has a unique mediating morphism to every other cocone over the same diagram.

The goal here is to define a universal property to characterize the functor $T : \mathbf{C} \to \mathbf{C}$ only by the means of its relationship with L and R (the property), and with any other functors $G : \mathbf{C} \to \mathbf{C}$ having the same relationship with L and R (the universality).

The Category. Since we want to characterize the functor $T : \mathbf{C} \to \mathbf{C}$, we need to consider a category providing a way to compare functors from $\mathbf{C}$ to $\mathbf{C}$. The usual definition of (homo)morphisms of functors is called *natural transformations*.

Definition 3.1.1. Given two functors $\mathrm{F}, \mathrm{G} : \mathbf{C} \to \mathbf{D}$, a *natural transformation* $\eta : \mathrm{F} \Rightarrow \mathrm{G}$ is given by a *component morphism* $\eta_x : \mathrm{F}(x) \to \mathrm{G}(x)$ in $\mathbf{D}$ for each object x in $\mathbf{C}$ such that $\eta_y \circ F(f) = G(f) \circ \eta_x$ for each $f : x \to y$ in $\mathbf{C}$. This last condition is called the *naturality condition*.

When every component η_x is an isomorphism, we say that η is a *natural isomorphism*, in other words F and G are isomorphic.

The composition of natural transformations between functors of the same signature is called *vertical composition* and is defined as follows.

Definition 3.1.2. Given three functors $F, G, H : \mathbf{C} \to \mathbf{D}$ and two natural transformations $\eta : F \Rightarrow G$ and $\rho : G \Rightarrow H$, their *vertical composition* is $\rho \circ \eta : F \Rightarrow$, defined component-wise, *i.e.*, by $(\rho \circ \eta)_c = \rho_c \circ \eta_c$ for any $c \in \mathbf{C}$.

This composition provides us with the expected category, with functors as objects and natural transformations as morphisms (proof in Section II.4 of [MacLane, 2013]).

The Property. Since the functor T is defined from a rule system, one expects that T respects the rules where the input is the data of the l.h.s. of a given rule.

Proposition 3.1.1. *For a global transformation T, we have $\mathrm{R} \cong T \circ \mathrm{L}$, i.e., there exists a natural isomorphism $\iota : \mathrm{R} \Rightarrow T \circ \mathrm{L}$.*

Proof. Notice that, as L is fully faithful, objects in $\mathrm{L}/\mathrm{L}(\gamma)$ are of the form $\langle \gamma', f \rangle$ where $f = \mathrm{L}(i)$ for some unique $i : \gamma' \to \gamma$. Then, observe that $\mathrm{L}/\mathrm{L}(\gamma)$ contains an object $\langle \gamma, \mathrm{L}(id_\gamma) \rangle$ and that every other object $\langle \gamma', \mathrm{L}(i) \rangle$ is a subobject of $\langle \gamma, \mathrm{L}(id_\gamma) \rangle$ by the morphism $\langle i, \mathrm{L}(id_\gamma) \rangle : \langle \gamma', \mathrm{L}(i) \rangle \to \langle \gamma, \mathrm{L}(id_\gamma) \rangle$.

Then, for any rule $\gamma \in \mathbf{\Gamma}$, we must find an isomorphism $\mathrm{R}(\gamma) \cong_\iota T(\mathrm{L}(\gamma))$. We proceed by showing that $\mathrm{R}(\gamma)$ forms a colimit cocone over $D = \mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/\mathrm{L}(\gamma)]$ whose components are $\mathrm{R}(\gamma)_{\langle \gamma', \mathrm{L}(i) \rangle} = \mathrm{R}(i)$ for any object $\langle \gamma', \mathrm{L}(i) \rangle$ in $\mathrm{L}/\mathrm{L}(\gamma)$. We first show that it forms a cocone, *i.e.*, it respects $\mathrm{R}(\gamma)_{\langle \gamma'', \mathrm{L}(j) \rangle} = \mathrm{R}(\gamma)_{\langle \gamma', \mathrm{L}(i) \rangle} \circ D(\langle k, \mathrm{L}(i) \rangle)$ for any morphism $\langle k, \mathrm{L}(i) \rangle : \langle \gamma'', \mathrm{L}(j) \rangle \to \langle \gamma', \mathrm{L}(i) \rangle$ of D. This directly follows from $j = i \circ k$ as $\mathrm{L}(j) = \mathrm{L}(i) \circ \mathrm{L}(k)$ by definition of $\mathrm{L}/\mathrm{L}(\gamma)$ and L is fully faithful. To check that $\mathrm{R}(\gamma)$ is the colimit, we show that there is a unique mediating $m : \mathrm{R}(\gamma) \to c$ for

any other cocone C with apex $c \in \mathbf{C}$. In particular, such m respects $C_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle} = m \circ \mathrm{R}(\gamma)_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle}$, but as $\mathrm{R}(\gamma)_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle} = \mathrm{R}(\mathrm{L}(id_\gamma)) = id_{\mathrm{R}(\gamma)}$, it uniquely defines m to be equal to $C_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle}$. Then as $\mathrm{R}(\gamma)$ and $T(\mathrm{L}(\gamma))$ both form a colimit cocone over D and as colimits are defined up to isomorphism, let $\iota_\gamma : \mathrm{R}(\gamma) \to T(\mathrm{L}(\gamma))$ be the corresponding isomorphism, *i.e.*, the mediating $T(\mathrm{L}(\gamma))_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle}$.

Finally, we show the naturality of ι, *i.e.*, for any rule inclusion $i : \gamma \to \gamma' \in \mathbf{\Gamma}$ we must have $T(\mathrm{L}(i)) \circ \iota_\gamma = \iota_{\gamma'} \circ \mathrm{R}(i)$. In particular, notice that the mediating m from $\mathrm{R}(\gamma)$ to $\mathrm{R}(\gamma')$ must respect $m \circ \mathrm{R}(\gamma)_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle} = \mathrm{R}(\gamma')_{\langle \gamma, \mathrm{L}(i) \circ \mathrm{L}(id_\gamma)\rangle}$ which gives $m = \mathrm{R}(i)$. Notice that as $T(\mathrm{L}(i)) \circ \iota_\gamma$ and $\iota_{\gamma'} \circ \mathrm{R}(i)$ are both composite of mediatings they are also mediatings from $\mathrm{R}(\gamma) \to \mathrm{R}(\gamma')$, so by uniqueness we have $T(\mathrm{L}(i)) \circ \iota_\gamma = \iota_{\gamma'} \circ \mathrm{R}(i)$. □

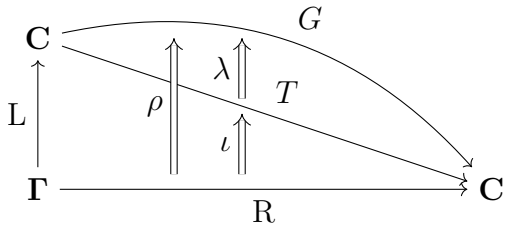
This fact, together with the functoriality of T gives that for each instance $f : \mathrm{L}(\gamma) \to c$ of a l.h.s. in an object $c \in C$ there is an occurrence $\overline{T}(f) : \mathrm{R}(\gamma) \to T(c)$ of the associated r.h.s. $\mathrm{R}(\gamma)$ in the result $T(c)$. Furthermore, for any rule inclusion $i : \gamma \to \gamma'$ between two such instances $f : \mathrm{L}(\gamma) \to c$ and $g : \mathrm{L}(\gamma') \to c$, *i.e.*, such that $g = f \circ \mathrm{L}(i)$, we have $T(g) \cong T(f) \circ \mathrm{R}(i)$. This is a concise description of how the global behavior of a global transformation is an extension of its rule system.

One can note that ι being a natural isomorphism comes from the fact that the l.h.s. functor L is a fully faithful embedding. Without this strong constraint, the result still holds in a weak form, where ι is a simple natural transformation.

In general, this property (with or without the fully faithful constraint of the l.h.s. functor) is not sufficient to characterize the functor T. Indeed, there can be multiple functors that respect this specification. The collection of candidates is then defined as all the functors $G : \mathbf{C} \to \mathbf{C}$ providing a natural transformation $\rho : \mathrm{R} \Rightarrow G \circ \mathrm{L}$.

Universality. We now have every building block to characterize $\langle T, \iota\rangle$ as the "minimal extension" of the rules. Indeed, for any other candidate $\langle G, \rho\rangle$, there exists some unique natural transformation from T to G that sends ι to ρ.

Proposition 3.1.2. *Given any functor $G : \mathbf{C} \to \mathbf{C}$ together with a natural transformation $\rho : \mathrm{R} \Rightarrow G \circ \mathrm{L}$, there exists some unique natural transformation $\lambda : T \Rightarrow G$ such that $\lambda_{\mathrm{L}(\gamma)} \circ \iota_\gamma = \rho_\gamma$ for any $\gamma \in \mathbf{\Gamma}$.*



Proof. Take any $c \in \mathbf{C}$, we first have to define a morphism $\lambda_c : T(c) \to G(c)$. For any $\langle \gamma, f : \mathrm{L}(\gamma) \to c\rangle$ in L/c, take the composite $G(f) \circ \rho_\gamma : \mathrm{R}(\gamma) \to G(\mathrm{L}(\gamma)) \to G(c)$. Observe that this defines a cocone $G(c)$ over $D = \mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/\mathrm{L}(\gamma)]$ given by $G(c)_{\langle \gamma, f\rangle} = G(f) \circ \rho_\gamma$. Indeed, take any morphism $\langle h, f\rangle : \langle \gamma', g\rangle \to \langle \gamma, f\rangle$, in L/c, by definition we have $g = f \circ \mathrm{L}(h)$. Then we have $G(c)_{\langle \gamma, f\rangle} \circ D(\langle h, f\rangle) = G(f) \circ \rho_\gamma \circ R(h) = G(f) \circ G(\mathrm{L}(h)) \circ \rho_{\gamma'} = G(g) \circ \rho_{\gamma'} = G(c)_{\langle \gamma', g\rangle}$.

So let λ_c to be the unique mediating from the colimit $T(c)$ to the cocone $G(c)$, *i.e.*, such that $\lambda_c \circ T(c)_{\langle \gamma, f\rangle} = G(c)_{\langle \gamma, f\rangle}$ for any $\langle \gamma, f\rangle$ in L/c. For any $\gamma \in \mathbf{\Gamma}$

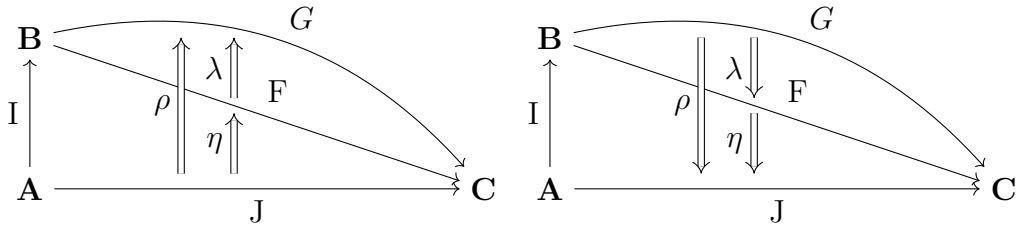
consider the isomorphism $\iota_\gamma : \mathrm{R}(\gamma) \to (T \circ \mathrm{L})(\gamma)$. By proposition 3.1.1 we have $\iota_\gamma = T(L(\gamma))_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle}$. It gives $\lambda_{\mathrm{L}(\gamma)} \circ \iota_\gamma = G(c)_{\langle \gamma, \mathrm{L}(id_\gamma)\rangle} = G(\mathrm{L}(id_\gamma)) \circ \rho_\gamma$ so we get $\lambda_{\mathrm{L}(\gamma)} \circ \iota_\gamma = \rho_\gamma$ as required.

We then have to show that λ is natural, *i.e.*, for any $m : c \to c' \in \mathbf{C}$ we have $\lambda_{c'} \circ T(m) = G(m) \circ \lambda_c$. To do this, we show that $\lambda_{c'} \circ T(m)$ and $G(m) \circ \lambda_c$ are both mediating morphisms from $T(c)$ to the cocone G' with apex $G(c')$ over $D_c = \mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/c]$, obtained by letting $G'_{\langle \gamma, f\rangle} = G(c')_{\langle \gamma, m\circ f\rangle}$ for any $\langle \gamma, f\rangle$ in L/c. Take any such $\langle \gamma, f\rangle$ in L/c. We have $\lambda_{c'} \circ T(m) \circ T(c)_{\langle \gamma, f\rangle} = \lambda_{c'} \circ T(c')_{\langle \gamma, m\circ f\rangle} = G(c')_{\langle \gamma, m\circ f\rangle}$ as $T(m)$ and λ_c are mediating morphisms so $\lambda_{c'} \circ T(m)$ is also mediating. We also have that $G(m) \circ \lambda_c \circ T(c)_{\langle \gamma, f\rangle} = G(m) \circ G(f) \circ \rho_\gamma$ as λ_c is mediating, and then it follows that $G(m) \circ G(f) \circ \rho_\gamma = G(m \circ f) \circ \rho_\gamma = G(c')_{\langle \gamma, m\circ f\rangle}$ so $G(m) \circ \lambda_c$ is also mediating. By uniqueness of mediatings we get that $\lambda_{c'} \circ T(m) = G(m) \circ \lambda_c$. □

3.1.2 Kan Extensions

Proposition 3.1.2 gives a universal property for the pair $\langle T, \iota\rangle$: it is a pair $\langle F : \mathbf{C} \to \mathbf{C}, \eta : \mathrm{R} \Rightarrow F \circ \mathrm{L}\rangle$ such that for any other candidate pair $\langle G : \mathbf{C} \to \mathbf{C}, \rho : \mathrm{R} \Rightarrow G \circ \mathrm{L}\rangle$ there exists a unique natural transformation $\lambda : F \to G$ such that $\lambda_{\mathrm{L}} \circ \eta = \rho$. In fact, this universal property corresponds precisely to the notion of being the left Kan extension of R along L.

Definition 3.1.3. Given three categories $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, three functors $\mathrm{I} : \mathbf{A} \to \mathbf{B}$, $\mathrm{J} : \mathbf{A} \to \mathbf{C}$ and $F : \mathbf{B} \to \mathbf{C}$, and a natural transformation $\eta : J \Rightarrow F \circ I$ (resp. $\eta : F \circ I \Rightarrow J$), the pair $\langle F, \eta\rangle$ is the *left (resp. right) Kan extension of J along I* if for any other pair $\langle G : \mathbf{B} \to \mathbf{C}, \rho : J \Rightarrow G \circ I\rangle$ (resp. $\rho : G \circ I \Rightarrow J$), there is a unique natural transformation $\lambda : F \to G$ (resp. $\lambda : G \to F$) such that we have $\lambda_{I(a)} \circ \eta_a = \rho_a$ (resp. $\eta_a \circ \lambda_{I(a)} = \rho_a$) for any $a \in \mathbf{A}$.



(Left) (Right)

For the moment, we focus on left Kan extensions and their connection to global transformations, we come back to right Kan extensions later in this chapter.

Remark 3.1.1. We already have seen a simpler example of left Kan extension in chapter 2. Take any functor $I : \mathbf{A} \to \mathbf{B}$, and the constant functor $c : \mathbf{A} \to \mathbf{1}$, where $\mathbf{1}$ is the category with one object and one morphism. The left Kan extension $\langle F : \mathbf{1} \to \mathbf{B}, \eta : I \Rightarrow F \circ c\rangle$ of I along c, if it exists, corresponds to the colimit of I. Indeed, $\langle F, \eta\rangle$ specifies a cocone where F selects the apex in $\mathbf{B}$ and η gives the cocone components. This cocone is the colimit as, in this setting, the universal property of being a left Kan extension coincides with the universal property of being a colimit.

In category theory, universal properties characterize an object uniquely up-to-isomorphism, this is also called a universal construction. The property to be a left

Kan extension gives a concise characterization of our extended functor T. Indeed, left Kan extensions are unique up to isomorphism.

Remark 3.1.2. As for any universal construction in category theory, the left Kan extension $\langle F, \eta \rangle$ can be shown to be unique up to isomorphism. This is shown in the same manner as for colimits in Section 2.1.7. Indeed, by considering another left Kan extensions $\langle G, \rho \rangle$ of the J along I, we get two unique natural transformations $\lambda : F \Rightarrow G$ and $\mu : G \Rightarrow F$ that respects the needed equalities. By noticing that id_F is the unique natural transformation from F to F such that $id_{F\mathrm{I}} \circ \eta = \eta$ and that $\mu \circ \lambda$ respects the same property, we get that $\mu \circ \lambda = id_F$. By the same argument, we get $\lambda \circ \mu = id_G$, so F and G are isomorphic.

The left Kan extension $\langle T, \iota \rangle$ have some specific properties. First, it can be obtained by computing the colimits of equation (3.1). This is not always the case, and such left Kan extensions are called *pointwise*. A pointwise left Kan extensions $F : \mathbf{B} \to \mathbf{C}$ of $J : \mathbf{A} \to \mathbf{C}$ along $I : \mathbf{A} \to \mathbf{B}$ is such that its values $F(b)$ for each object $b \in B$ can be computed independently, only by the means of objects in I/b, *i.e.*, using only morphisms $m : I(a) \to b$ from objects of I. Secondly, the natural transformation ι is a natural isomorphism, *i.e.*, $T \circ \mathrm{L}$ does exactly the same work as R. This is directly linked with the definition of rule systems. Indeed, by remarking that the proof of proposition 3.1.1 does not depend on L and R having the same category as destination, we can rephrase it as follows.

> Given a pointwise left Kan extension $\langle F, \eta \rangle$ of a functor J along a functor I, then if I is fully faithful, we have that η is a natural isomorphism.

This is a slight generalization of Proposition 3.7.3 of [Borceux, 1994] (or Corollary X.3.3 of [MacLane, 2013]) that considers the destination of J to have all (small) colimits. Indeed, this fact implies that the obtained extension is pointwise (Theorem 3.7.2 of [Borceux, 1994]).

To summarize, we have seen that the definition of global transformations from chapter 2, corresponds to a pointwise left Kan extension with the additional property that L is fully faithful. This gently introduced the concept of Kan extension, which describes how a functor is an extension of another. In our setting, they are considered for relating the local presentation of a system to its global behavior.

In the rest of the chapter, we apply the concept of Kan extension to the case of cellular automata, where the study of the relationship between the local transition function and the global transition function was thoroughly studied using topological tools. This allows us not only to give multiple instances of left and right Kan extensions, extending our knowledge about Kan extensions and global transformations, but also to give a new setting for studying locality in cellular automata.

3.2 Cellular Automata on Partial Configurations

A cellular automaton is a dynamical system over some spaces called *configurations*. A common example of such spaces is the two-dimensional grid $G = \mathbb{Z} \times \mathbb{Z}$ decorated with the set of states $Q = \{0, 1\}$. A *global configuration* is then some function $f : G \to Q$ and the set of global configurations is denoted by Q^G. The behavior of a cellular automaton is defined by its *global transition function* $\Delta : Q^G \to Q^G$ over global configurations that can be localized, *i.e.*, defined by the means of a *local*

transition function $\delta : Q^N \to Q$ over *local configurations* $c : N \to Q$, where N is some subset of G called the neighborhood.

In this section, we aim at extending the behavior of a cellular automaton to every partial configurations, *i.e.*, functions $p : Q^S \to Q$ for any subset $S \subseteq G$. It is indeed an extension as global configurations are also partial configurations, and the behavior of our extension should be the same as Δ over global configurations.

3.2.1 Cellular Automata

Let us give some basic definitions to fix the notations. We also note small caveats early on, to avoid having to deal with many unrelated details at the same time in a single proof or construction later on.

Definition 3.2.1. A *group* is a set G with a binary operation $- \cdot - : G \times G \to G$ which is associative, which has a neutral element 1 and for which any $g \in G$ has inverse g^{-1}. A right action of the group on a set X is a binary operation $- \blacktriangleleft - : X \times G \to X$ such that $x \blacktriangleleft 1 = x$ and $(x \blacktriangleleft g) \blacktriangleleft h = x \blacktriangleleft (g \cdot h)$.

In cellular automata, the group G represents the space, each element $g \in G$ being at the same time an absolute and a relative position. This space is decorated with states that evolve through local interactions only. The classical formal definitions go as follows and work with the entire, often infinite, space.

Definition 3.2.2. A *cellular automaton* on a group G is given by a finite subset $N \subseteq G$ called the *neighborhood*, a finite set of *states* Q, and a *local transition function* $\delta : Q^N \to Q$. The elements of the set Q^N are called *local configurations*. The elements of the set Q^G are called *global configurations*, and a right action $- \blacktriangleleft - : Q^G \times G \to Q^G$ is defined on Q^G by $(c \blacktriangleleft g)(h) = c(g \cdot h)$ for all $h \in G$. The *global transition function* $\Delta : Q^G \to Q^G$ of such a cellular automaton is defined as $\Delta(c)(g) = \delta((c \blacktriangleleft g) \restriction N)$.

Notice that in this definition, we refer explicitly to *global* and *local* configurations; the term *configuration* with no qualification will be introduced in definition 3.2.3.

Proposition 3.2.1. *The function* $- \blacktriangleleft - : Q^G \times G \to Q^G$ *of definition 3.2.2 is indeed a right action.*

Proof. For any $g, h \in G$, we have $((c \blacktriangleleft g) \blacktriangleleft h)(i) = (c \blacktriangleleft g)(h \cdot i) = c(g \cdot h \cdot i) = (c \blacktriangleleft (g \cdot h))(i)$ for any $i \in G$, so $((c \blacktriangleleft g) \blacktriangleleft h) = (c \blacktriangleleft (g \cdot h))$ and also $(c \blacktriangleleft 1)(i) = c(1 \cdot i) = c(i)$ as required by definition 3.2.1 of right actions. □

This choice of definition and right notation for the so-called shift action has two advantages. Firstly, the definition of the action is a simple associativity. Secondly, when instantiated with $G = \mathbb{Z}$ with sum, the content of $c \blacktriangleleft 5$ is the content of c shifted to the left, as the symbols indicates. Indeed, for $c' = c \blacktriangleleft 5$, $c'(-5) = c(0)$ and $c'(0) = c(5)$.

Proposition 3.2.2. *For all* $c \in Q^G$ *and* $g \in G$, $\Delta(c)(g)$ *is determined by* $c \restriction g \cdot N$.

Proof. Indeed, $\Delta(c)(g) = \delta((c \blacktriangleleft g) \restriction N)$ so the value is determined by $(c \blacktriangleleft g) \restriction N$. But for any $n \in N$, $(c \blacktriangleleft g)(n) = c(g \cdot n)$ by definition of $\blacktriangleleft$. □

In common cellular automata terms, this proposition means that the *neighborhood of* g is $g \cdot N$, in this order. Let us informally call *shifted local configurations* the objects of the form $c \restriction g \cdot N \in \bigcup_{g \in G} Q^{g \cdot N}$. Note that, at our level of generality, two different positions $g \neq g' \in G$ might have the same neighborhood $g \cdot N = g' \cdot N$. Although the injectivity of the function $(- \cdot N)$ could be a useful constraint to add, which is often verified in practice, we do not impose it, so the reader should keep this in mind.

Proposition 3.2.3. *The function* $- \cdot N : G \to \mathbf{2}^G$ *is not necessarily injective.*

Proof. Considering the group $G = \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}$ and $N = \{(0,0),(1,0)\}$, we have $(0,0) + N = (1,0) + N = \{(0,0),(1,0)\}$ because of the torsion. □

Because of this, it is useful to replace the shifted local configurations, *i.e.*, the union $\bigcup_{g \in G} Q^{g \cdot N}$, by the disjoint union $\bigcup_{g \in G} (\{g\} \times Q^{g \cdot N})$. The elements of the latter are of the form $(g \in G, c \in Q^{g \cdot N})$ and keep track of the considered "center" of the neighborhood. More explicitly, two elements $(g_0, c \restriction g_0 \cdot N), (g_1, c \restriction g_1 \cdot N) \in \bigcup_{g \in G} (\{g\} \times Q^{g \cdot N})$ are different as soon as, $g_0 \neq g_1$ even if $g_0 \cdot N = g_1 \cdot N$. This encodes things according to the intuition of a centered neighborhood.

3.2.2 Partial configurations

In the previous formal statements, one sees different kinds of configurations, explicitly or implicitly: global configurations $c \in Q^G$, local configurations $(c \blacktriangleleft g) \restriction N \in Q^N$, shifted local configurations $c \restriction g \cdot N \in Q^{g \cdot N}$, and their resulting "placed states" $(g \mapsto \Delta(c)(g)) \in Q^{\{g\}}$. In the cellular automata literature, one often considers configurations defined on other subsets of the space, *e.g.*, finite connected subsets. More generally, we are interested in all configurations Q^S for arbitrary subsets $S \subseteq G$.

Definition 3.2.3. A *configuration* c is a partial function from G to Q. Its domain of definition is denoted $|c|$ and called its *support*. The set of all configurations is denoted $\mathrm{Conf} = \bigcup_{S \subseteq G} Q^S$. We extend the previous right action $\blacktriangleleft$ and define it to map each $c \in \mathrm{Conf}$ to $c \blacktriangleleft g$ having support $|c \blacktriangleleft g| = \{h \in G \mid g \cdot h \in |c|\}$ and states $(c \blacktriangleleft g)(h) = c(g \cdot h)$.

Proposition 3.2.4. *The latter action is well-defined and is a right action.*

Proof. The configuration $c \blacktriangleleft g$ is well-defined on all of its support. Indeed, for any $h \in |c \blacktriangleleft g|$, $(c \blacktriangleleft g)(h) = c(g \cdot h)$ and $g \cdot h \in |c|$ by definition of h. The right action property is verified as in the proof of proposition 3.2.1. □

Let us restate proposition 3.2.2 more precisely using definition 3.2.3.

Proposition 3.2.5. *For* $c \in \mathrm{Conf}$ *and* $g \in G$ *s.t.* $g \cdot N \subseteq |c|$, $(c \blacktriangleleft g) \restriction N = (c \restriction g \cdot N) \blacktriangleleft g$.

Proof. Indeed, $|(c \restriction g \cdot N) \blacktriangleleft g| = \{h \in G \mid g \cdot h \in |(c \restriction g \cdot N)|\} = \{h \in G \mid g \cdot h \in g \cdot N\} = N = |(c \blacktriangleleft g) \restriction N|$. Also for any $n \in N$, $((c \blacktriangleleft g) \restriction N)(n) = (c \blacktriangleleft g)(n) = c(g \cdot n)$ and $((c \restriction g \cdot N) \blacktriangleleft g)(n) = (c \restriction g \cdot N)(g \cdot n) = c(g \cdot n)$. □

3.2.3 The Coarse Transition Function

A first way to extend the global transition function to the whole set of partial configurations is as follows. Taking a configuration c, look for all places g where the whole neighborhood $g \cdot N$ is defined and take the local transition result of these places only.

Definition 3.2.4. The *interior* of a subset $S \subseteq G$ is $\text{int}(S) = \{g \in G \mid g \cdot N \subseteq S\}$.

Definition 3.2.5. The *coarse transition function* $\Phi : \text{Conf} \to \text{Conf}$ is defined for all $c \in \text{Conf}$ as $|\Phi(c)| = \text{int}(|c|)$ and $\Phi(c)(g) = \delta((c \blacktriangleleft g) \upharpoonright N)$.

Proposition 3.2.6. *For any $c \in \text{Conf}$ and $g \in G$, the statements $g \in \text{int}(|c|)$, $g \cdot N \subseteq |c|$, and $N \subseteq |c \blacktriangleleft g|$ are equivalent. (So Φ is well-defined in definition 3.2.5.)*

Proof. The first and second statements are equivalent by definition 3.2.4 of int. The second and third statements are equivalent by definition 3.2.3 of $\blacktriangleleft$. □

Remember proposition 3.2.3. If we do have injectivity of neighborhoods, we have $\text{int}(g \cdot N) = \{g\}$. But since we do not assume it, we only have the following.

Proposition 3.2.7. *Let $S \subseteq G$. It is always the case that $S \subseteq \text{int}(S \cdot N)$ but we do not necessarily have equality, even when $S = \{g\}$ for some $g \in G$.*

Proof. Consider any $s \in S$. Clearly, $s \cdot N \subseteq S \cdot N$, so by definition 3.2.4, $s \in \text{int}(S \cdot N)$. However, we do not have equality in the example of the proof of proposition 3.2.3 with $S = \{(0,0)\}$. Indeed, in this case, $\text{int}(S \cdot N) = \{(0,0),(1,0)\}$. □

Of course, the coarse transition function shares with the global transition function the fact that it commutes with the shift.

Proposition 3.2.8. *For any $c \in \text{Conf}$ and $g \in G$ we have $\Phi(c \blacktriangleleft g) = \Phi(c) \blacktriangleleft g$.*

Proof. We must check that $|\Phi(c \blacktriangleleft g)| = |\Phi(c) \blacktriangleleft g|$ and that for any $h \in |\Phi(c \blacktriangleleft g)|$ we have $\Phi(c \blacktriangleleft g)(h) = (\Phi(c) \blacktriangleleft g)(h)$. For the first part, we have :

$$\begin{aligned}
& h \in |\Phi(c \blacktriangleleft g)| \iff h \in |\Phi(c) \blacktriangleleft g| \\
\iff \quad & h \in |\Phi(c \blacktriangleleft g)| \iff g \cdot h \in |\Phi(c)| \text{ (def. 3.2.3)} \\
\iff \quad & h \in \text{int}(|c \blacktriangleleft g|) \iff g \cdot h \in \text{int}(|c|) \text{ (def. 3.2.5 of } \Phi) \\
\iff \quad & N \subseteq |(c \blacktriangleleft g) \blacktriangleleft h| \iff N \subseteq |c \blacktriangleleft g \cdot h| \text{ (prop. 3.2.6)}
\end{aligned}$$

The last proposition is true by the proposition 3.2.4. For the second part, we have:

$$\begin{aligned}
\Phi(c \blacktriangleleft g)(h) &= \delta(((c \blacktriangleleft g) \blacktriangleleft h) \upharpoonright N) \text{ (def. 3.2.5 of } \Phi) \\
&= \delta((c \blacktriangleleft g \cdot h) \upharpoonright N) \text{ (prop. 3.2.4)} \\
&= \Phi(c)(g \cdot h) \text{ (def. 3.2.5 of } \Phi) \\
&= (\Phi(c) \blacktriangleleft g)(h) \text{ (def. 3.2.3)}
\end{aligned}$$

□

Another useful remark to which we come back below is the following.

Proposition 3.2.9. *For any $g \in G$, any $M \subsetneq N$, and any $c \in Q^{g \cdot M}$, $|\Phi(c)| = \emptyset$. Also, for any $c \in Q^G$, $|\Phi(c)| = |\Delta(c)|$.*

Proof. By definition 3.2.5 of Φ. □

3.2.4 The Fine Transition Function

For some applications, the previous definitions are not satisfactory. For example, it is common to consider two cellular automata to be essentially the same if they generate the same global transition functions. However, here, two such cellular automata give different coarse transition functions if they have a different neighborhood.

To refine the previous definitions, a second approach is to take a configuration c, and look at all places for which the result is already determined by the partial data defined in c. For a position g, we define now c_g the part of c that is in the neighborhood of g. We can consider c_g as a partial function from $g \cdot N$ to Q. If this partial function is enough for the result at g to be already determined, *i.e.*, if any extension of c_g into a function c' from $g \cdot N$ to Q gives the same result $\delta(c' \blacktriangleleft g)$, we say that g is in the *determined subset* of c and call $q_{c,g}$ the determined result.

Definition 3.2.6. For any $c \in \mathrm{Conf}$ and $g \in G$, let $c_g = c \restriction (g \cdot N \cap |c|)$.

Definition 3.2.7. Given a configuration $c \in \mathrm{Conf}$, its *determined subset* is $\det(c) = \{g \in G \mid \exists q \in Q, \forall c' \in Q^{g \cdot N}, c' \restriction |c_g| = c_g \implies \delta(c' \blacktriangleleft g) = q\}$. For any $g \in \det(c)$, we denote $q_{c,g} \in Q$ the unique state q having the mentioned property.

Note that this definition depends on the cellular automaton local transition function δ and on the data of the configuration c, contrary to definition 3.2.4 of interior that only depends on its neighborhood N and on the support of the configuration.

Definition 3.2.8. Given a cellular automaton, its *fine transition function* $\phi : \mathrm{Conf} \to \mathrm{Conf}$ is defined as $|\phi(c)| = \det(c)$ and $\phi(c)(g) = q_{c,g}$, *i.e.*, $\phi(c)(g) = \delta(c' \blacktriangleleft g)$ for any $c' \in Q^{g \cdot N}$ such that $c' \restriction |c_g| = c_g$.

Proposition 3.2.10. *The fine transition function ϕ is well-defined.*

Proof. This is the case precisely because we restrict the support of $\phi(c)$ to the determined subset of the c. □

Proposition 3.2.11. *Consider the constant cellular automaton $\delta(c) = q$, $\forall c \in Q^N$, for a specific $q \in Q$ and regardless of the neighborhood N chosen to represent it. We have $|\phi(c)| = G$ for any $c \in \mathrm{Conf}$.*

Proof. Indeed, even with no data at all, *i.e.*, for c such that $|c| = \emptyset$, the result at all position is determined and is q. □

Note that, contrary to proposition 3.2.9 of the coarse transition function, the fine transition function definition is explicitly about considering non-empty results for some configurations of support $M \subsetneq N$ (see definition 3.2.6). When there is no such proper "sub-local" configuration with determined result, the two transition functions (definitions 3.2.5 and 3.2.8) are actually equal since $\mathrm{int}(c) = \det(c)$ for any $c \in \mathrm{Conf}$ (definitions 3.2.7 and 3.2.4). But let us not insist on this point.

Of course, the fine transition function shares with the global transition function the property of commuting with the shift.

Proposition 3.2.12. *For any $c \in \mathrm{Conf}$, we have $\det(c \blacktriangleleft g) = g^{-1} \cdot \det(c)$ and for any $h \in \det(c \blacktriangleleft g)$ we have $q_{c \blacktriangleleft g, h} = q_{c, g \cdot h}$.*

Proof. First, observe that definition 3.2.7, $h \in \det(c \blacktriangleleft g)$ means that there exists $q_{c \blacktriangleleft g,h} \in Q$ such that for any $c' \in Q^{h \cdot N}$ we have:

$$c' \upharpoonright |(c \blacktriangleleft g)_h| = (c \blacktriangleleft g)_h \implies \delta(c' \blacktriangleleft h) = q_{c \blacktriangleleft g,h} \tag{3.2}$$

Then observe that $(c \blacktriangleleft g)_h = c_{g \cdot h} \blacktriangleleft g$:

$$\begin{aligned}
(c \blacktriangleleft g)_h &= c \blacktriangleleft g \upharpoonright (h \cdot N \cap |c \blacktriangleleft g|) && \text{(def. 3.2.6)} \\
&= c \blacktriangleleft g \upharpoonright (g^{-1} \cdot g \cdot h \cdot N \cap g^{-1} \cdot |c|) && \text{(def. 3.2.3)} \\
&= (c \upharpoonright (g \cdot h \cdot N \cap |c|)) \blacktriangleleft g && \text{(prop. 3.2.4)} \\
&= c_{g \cdot h} \blacktriangleleft g && \text{(def. 3.2.6)}
\end{aligned}$$

Using this equality and proposition 3.2.4 on the l.h.s. of equation (3.2) we get:

$$c' \blacktriangleleft g^{-1} \upharpoonright |c_{g \cdot h}| = c_{g \cdot h} \implies \delta(c' \blacktriangleleft h) = q_{c \blacktriangleleft g,h} \tag{3.3}$$

$$\iff c' \blacktriangleleft g^{-1} \upharpoonright |c_{g \cdot h}| = c_{g \cdot h} \implies \delta(c' \blacktriangleleft g^{-1} \upharpoonright g \cdot h) = q_{c \blacktriangleleft g,h} \quad \text{(prop. 3.2.4)} \tag{3.4}$$

Moreover, as $(- \blacktriangleleft g^{-1})$ from a bijection froms $Q^{h \cdot N}$ to $Q^{g \cdot h \cdot N}$, definition 3.2.7 gives $h \in \det(c \blacktriangleleft g) \iff g \cdot h \in \det(c)$ so $\det(c \blacktriangleleft g) = g^{-1} \cdot \det(c)$. In this setting, we must have $\delta(c' \blacktriangleleft g^{-1} \upharpoonright g \cdot h) = q_{c,g \cdot h}$ so the implication (3.4) gives $q_{c \blacktriangleleft g,h} = q_{c,g \cdot h}$. □

Proposition 3.2.13. *For any $c \in \mathrm{Conf}$ and $g \in G$ we have $\phi(c \blacktriangleleft g) = \phi(c) \blacktriangleleft g$.*

Proof. We must check that $|\phi(c \blacktriangleleft g)| = |\phi(c) \blacktriangleleft g|$ and that for any $h \in |\phi(c \blacktriangleleft g)|$ we have $\phi(c \blacktriangleleft g)(h) = (\phi(c) \blacktriangleleft g)(h)$. For the first part, we have :

$$\begin{aligned}
& h \in |\phi(c \blacktriangleleft g)| \iff h \in |\phi(c) \blacktriangleleft g| && \\
\iff \quad & h \in |\phi(c \blacktriangleleft g)| \iff g \cdot h \in |\phi(c)| && \text{(def. 3.2.3)} \\
\iff \quad & h \in \det(c \blacktriangleleft g) \iff g \cdot h \in \det(c) && \text{(def. 3.2.8 of } \phi)
\end{aligned}$$

It is true by proposition 3.2.12. This proposition also gives:

$$\begin{aligned}
\phi(c \blacktriangleleft g)(h) &= q_{c \blacktriangleleft g,h} && \text{(def. 3.2.8)} \\
&= q_{c,g \cdot h} && \text{(prop. 3.2.12)} \\
&= \phi(c)(g \cdot h) && \text{(def. 3.2.8)} \\
&= (\phi(c) \blacktriangleleft g)(h) && \text{(def. 3.2.3)}
\end{aligned}$$

□

We have defined two possible extensions of the global transition function Δ to all partial configurations, the coarse transition function Φ and the fine transition function ϕ. The next sections are dedicated to the study of the link between these definitions and the general concept of Kan extensions.

3.3 Coarse Transition as Kan Extensions

The coarse transition function Φ is the direct extension of the local transition function δ to every partial configurations, it just applies the local transition function

everywhere in a partial configuration. In this sense, it is clear that if we take two partial configurations c and c', where c' is "bigger" than c the local transition function sends c' to a bigger partial configuration $\Phi(c')$ than $\Phi(c)$. Consequently, the coarse transition function appears clearly related to global transformations, so to left Kan extensions in the light of section 3.1.

This section is devoted to establish the relationship between the coarse transition function and the concept of left Kan extension. This idea is first studied by the means of posets and monotonic functions, by lifting Conf to a poset and Φ to a monotonic function. Posets are a particular kind of category where there is at most one morphism between any two objects. In this context, functors are simple monotonic functions and natural transformations are simple comparisons of monotic functions. In this simpler setting, Φ is characterized as a poset left Kan extension.

The downside of this first characterization is that the shift operation is not taken into account in the proposed structure of poset for Conf. As a consequence, the "local rules" should be specified for every shifted local configuration, *i.e.*, every local configuration at every position $g \in G$, to get the shift equivariance of the cellular automaton. To solve this issue, we lift the set Conf of configurations to a category **Conf** taking into account the shift operation. The monotonic transition function Φ is then lifted to a functor and is characterized as a Left Kan extension. This category is expressive enough so that local rules have not to be specified for every shifted local configuration. We then discuss the implications of working up to isomorphism in the category of configurations **Conf**.

3.3.1 The Coarse Monotonic Transition Function

In this section, we first lift the set of partial configurations to a poset Conf, where the restriction operation $(- \restriction -)$ gives an order relation. In this setting it is clear that Φ is a monotonic function, as for two configurations $c \preceq c'$ there are necessary more places to apply the local transition function δ in c' than in c.

The Poset of Partial Configurations

Definition 3.3.1. A *partial order* on a set X is a binary relation $- \preceq - \subseteq X \times X$ which is reflexive, transitive, and antisymmetric. A set endowed with a partial order is called a *partially ordered set*, or *poset* for short.

Definition 3.3.2. Given any two configurations $c, c' \in \mathrm{Conf}$, we set $c \preceq c'$ if and only if $\forall g \in |c|, g \in |c'| \wedge c(g) = c'(g)$. This is read "$c$ is a subconfiguration of c'" or "c' is a superconfiguration of c".

Proposition 3.3.1. *The set* Conf *with this binary relation is a poset. In this poset, the shifted local configurations $c \in \bigcup_{g \in G} Q^{g \cdot N}$ are subconfigurations of the (appropriate) global configurations $c' \in Q^G$. Shifted local configurations form an antichain. Global configurations form an antichain.*

Proof. As can be readily seen, since each global configuration restricts to many shifted local configurations, and recalling that an antichain is a subset S of the poset such that neither $x \preceq x'$ nor $x' \preceq x$ hold for any two different $x, x' \in S$. □

Kan Extensions for Posets

As introduced in section 3.1, Kan extensions are formally defined in all generality for categories, functors and natural transformations. We revisit here the concept of Kan extension in the simpler case where the category is a poset.

A pre-ordered set (poset where there is no constraint of reflexivity) is precisely a category where there is either zero or one morphism between any two objects (short proof in Section I.2 of [MacLane, 2013]). In this case, functors are monotonic functions

Definition 3.3.3. Given two posets $(X, \preceq_X)$ and $(Y, \preceq_Y)$, a function $f : X \to Y$ is said to be *monotonic* if for all $x, x' \in X$, $x \preceq_X x'$ implies $f(x) \preceq_Y f(x')$.

Proposition 3.3.2. *For any $g \in G$, the function $(- \blacktriangleleft g) : \text{Conf} \to \text{Conf}$ is monotonic.*

Proof. Given any $c, c' \in \text{Conf}$ such that $c \preceq c'$, this claim is equivalent to:

$$\begin{aligned} & (c \blacktriangleleft g) \preceq (c' \blacktriangleleft g) \text{ (by def 3.3.3)} \\ \iff & \forall h \in |c \blacktriangleleft g|; h \in |c' \blacktriangleleft g| \wedge (c \blacktriangleleft g)(h) = (c' \blacktriangleleft g)(h) \text{ (def 3.3.2)} \\ \iff & \forall h \in G \text{ s.t. } g \cdot h \in |c|; g \cdot h \in |c'| \wedge c(g \cdot h) = c'(g \cdot h) \text{ (def 3.2.3)} \end{aligned}$$

which is true by the application of definition 3.3.2 of $c \preceq c'$ on $g \cdot h$. □

The definition of natural transformations when the functors are monotonic functions coincides with a pointwise comparison of such functions.

Definition 3.3.4. Given two posets $(X, \preceq_X)$ and $(Y, \preceq_Y)$, we define the binary relation $- \Rightarrow -$ on the set of all monotonic functions from X to Y by $f \Rightarrow f' \iff \forall x \in X, f(x) \preceq_Y f'(x)$.

Proposition 3.3.3. *Given two posets $(X, \preceq_X)$ and $(Y, \preceq_Y)$, the set of monotonic functions between them together with this binary relation forms a poset.*

Proof. As one can easily check. □

All the ingredients are now given to revist the notions of left and right Kan extension for the very specific case of posets.

Definition 3.3.5. In this setting, given three posets A, B and C, and three monotonic functions $i : A \to B$, $f : A \to C$ and $g : B \to C$, g is said to be the *left (resp. right) Kan extension of f along i* if g is the $\Rightarrow$-minimum (resp. $\Rightarrow$-maximum) element in the set of monotonic functions $\{h : B \to C \mid f \Rightarrow h \circ i\}$ (resp. $\{h : B \to C \mid h \circ i \Rightarrow f\}$).

A specificity of Kan extensions for posets is that they are a complete characterization as stated in the following proposition (not only up to isomorphism, as we have already seen in remark 3.1.2).

Proposition 3.3.4. *The left (resp. right) Kan extension for posets is unique when it exists.*

Proof. It is defined as the minimum of a set, and as any minimum, it may not exist, but when it does, it is always unique. □

Characterization as Poset Left Kan Extension

The coarse transition function Φ is defined on the set of all configurations Conf and we claim that it is generated, in the Kan extension sense, by the local transition function δ shifted everywhere. We define the latter, with proposition 3.2.3 in mind.

Definition 3.3.6. We define Loc to be the poset $\text{Loc} = \bigcup_{g\in G}(\{g\} \times Q^{g\cdot N})$ with trivial partial order $(g, c) \preceq (g', c') \iff (g, c) = (g', c')$. The "fully shifted local transition function" $\overline{\delta} : \text{Loc} \to \text{Conf}$ is defined, for any $(g, c) \in \text{Loc}$ as $|\overline{\delta}(g, c)| = \{g\}$ and $\overline{\delta}(g, c)(g) = \delta(c \blacktriangleleft g)$. The second projection of Loc is the monotonic function $\pi_2 : \text{Loc} \to \text{Conf}$ defined as $\pi_2(g, c) = c$.

Proposition 3.3.5. Loc *is a poset and* $\overline{\delta}$ *and* π_2 *are monotonic functions.*

Proof. Indeed, the identity relation is an order relation and any function respects the identity relation. □

Proposition 3.3.6. *The coarse transition function* Φ *is monotonic.*

Proof. Indeed, take $c, c' \in \text{Conf}$ such that $c \preceq c'$. We want to prove that $\Phi(c) \preceq \Phi(c')$ and this is equivalent to:

$$\begin{aligned}
&\forall g \in |\Phi(c)|, g \in |\Phi(c')| \wedge \Phi(c)(g) = \Phi(c')(g)\\
\iff &\forall g \in \text{int}(|c|), g \in \text{int}(|c'|) \wedge \delta(c \blacktriangleleft g \restriction N) = \delta(c' \blacktriangleleft g \restriction N)\\
\iff &\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge \delta(c \blacktriangleleft g \restriction N) = \delta(c' \blacktriangleleft g \restriction N),
\end{aligned}$$

by definition 3.2.3 of $\preceq$, definition 3.2.5 of Φ, and definition 3.2.4 of int. The final statement is implied by:

$$\begin{aligned}
&\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge c \blacktriangleleft g \restriction N = c' \blacktriangleleft g \restriction N\\
\iff &\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge \forall n \in N, (c \blacktriangleleft g)(n) = (c' \blacktriangleleft g)(n)\\
\iff &\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge \forall n \in N, c(g \cdot n) = c'(g \cdot n),
\end{aligned}$$

the last equivalence being by definition 3.2.3. To prove it, take $g \in G$ such that $g \cdot N \subseteq |c|$, and $n \in N$. By definition 3.3.2, since $c \preceq c'$ and $g \cdot n \in |c|$, we have $g \cdot n \in |c'|$, and $c(g \cdot n) = c'(g \cdot n)$ as wanted. □

Proposition 3.3.7. Φ *is the poset left Kan extension of* $\overline{\delta}$ *along* $\pi_2 : \text{Loc} \to \text{Conf}$.

Proof. By definition 3.3.5 of left Kan extensions, we need to prove firstly that Φ is such that $\overline{\delta} \Rightarrow \Phi \circ \pi_2$, and secondly that it is smaller than any other such monotonic function.

For the first part, $\overline{\delta} \Rightarrow \Phi \circ \pi_2$ is equivalent to:

$$\begin{aligned}
&\forall (g, c) \in \text{Loc}, \overline{\delta}(g, c) \preceq \Phi(c) \text{ (defs. 3.3.4 and 3.3.6 of } \Rightarrow \text{ and } \pi_2)\\
\iff &\forall (g, c) \in \text{Loc}, \forall h \in |\overline{\delta}(g, c)|, h \in |\Phi(c)| \wedge \overline{\delta}(g, c)(h) = \Phi(c)(h) \text{ (d. 3.3.2 of } \preceq)\\
\iff &\forall (g, c) \in \text{Loc}, g \in |\Phi(c)| \wedge \delta(c \blacktriangleleft g) = \Phi(c)(g) \text{ (def. 3.3.6 of } \overline{\delta})\\
\iff &\forall (g, c) \in \text{Loc}, g \cdot N \in |c| \wedge \delta(c \blacktriangleleft g) = \delta((c \blacktriangleleft g) \restriction N) \text{ (defs 3.2.4, 3.2.5 of } \Phi).
\end{aligned}$$

This last statement is true by definition 3.3.6 of Loc, *i.e.*, since $c \in Q^{g\cdot N}$, $c \blacktriangleleft g = (c \blacktriangleleft g) \restriction N$.

For the second part, let $F : \mathrm{Conf} \to \mathrm{Conf}$ be a monotonic function such that $\overline{\delta} \Rightarrow F \circ \pi_2$. We want to show that $\Phi \Rightarrow F$, which is equivalent to:

$$\begin{aligned}
&\forall c \in \mathrm{Conf}, \Phi(c) \preceq F(c) \text{ (def. 3.3.4 of } \Rightarrow) \\
\iff &\forall c \in \mathrm{Conf}, \forall g \in |\Phi(c)|, g \in |F(c)| \wedge \Phi(c)(g) = F(c)(g) \text{ (def. 3.3.2 of } \preceq) \\
\iff &\forall c \in \mathrm{Conf}, \forall g \in \mathrm{int}(|c|), g \in |F(c)| \wedge F(c)(g) = \delta((c \blacktriangleleft g) \upharpoonright N) \text{ (def. 3.2.5)}
\end{aligned}$$

So take $c \in \mathrm{Conf}$ and $g \in \mathrm{int}(|c|)$, and consider $d_g = c \upharpoonright g \cdot N$. Since $d_g \preceq c$ and F is monotonic, we have $F(d_g) \preceq F(c)$. Moreover, $\overline{\delta} \Rightarrow F \circ \pi_2$ and $(g, d_g) \in \{g\} \times Q^{g \cdot N} \subseteq \mathrm{Loc}$, so $\overline{\delta}(g, d_g) \preceq F(d_g)$ by definitions 3.3.4 and 3.3.6 of $\Rightarrow$ and π_2. By transitivity $\overline{\delta}(g, d_g) \preceq F(c)$. By definition 3.3.6 of $\overline{\delta}$ and definition 3.3.2 of $\preceq$, we obtain $g \in |F(c)|$, and $F(c)(g) = \overline{\delta}(g, d_g)(g) = \delta((c \blacktriangleleft g) \upharpoonright N)$, as wanted. □

As a sidenote, remark that in order to have the equality $\overline{\delta} = \Phi \circ \pi_2$, one needs to have the injectivity of neighborhood function. Indeed, without injectivity, we have two different $g, g' \in G$ having the same neighborhood M, *i.e.*, $M = g \cdot N = g' \cdot N$. This means that, given any local configuration $c \in Q^M$ on this neighborhood, each pair $(g, c), (g', c) \in \mathrm{Loc}$ have different results $\overline{\delta}(g, c) \in Q^{\{g\}}$ and $\overline{\delta}(g', c) \in Q^{\{g'\}}$ with different support $\{g\}$ and $\{g'\}$. However, their common projection $\pi_2(g, c) = \pi_2(g', c) = c$ have a unique result $\Phi(c)$ with a support such that $\{g, g'\} \subseteq |\Phi(c)|$. So we have a strict comparison $\overline{\delta} \Rightarrow \Phi \circ \pi_2$. When the neighborhood function is injective, π_2 is also injective and the previous situation can not occur so we have equality $\overline{\delta} = \Phi \circ \pi_2$.

3.3.2 The Coarse Transition Functor

We have just introduced the "fully shifted" local transition function, which describes the local behavior at each position independently. Usually, the transition table is only defined on those local configurations $c \in Q^N$ centered at 1. In this section, we visit this idea by considering that these local configurations can "occur" at any position in another configuration $c' \in \mathrm{Conf}$. In other word, for each pair of configurations $c, c' \in \mathrm{Conf}$, we manipulate the *set* of occurrences of c in c' rather than the binary answer of the proposition $c \preceq c'$. This makes the transition from posets to categories.

The Category of Configurations

Previously, the set of configurations has been equipped with a structure of poset considering a morphism from c to c' if and only if $c \preceq c'$. We now consider additional morphisms between c and c' addressing that c may be found at different positions in c', *i.e.*, a morphism for each $g \in G$ such that $c \preceq c' \blacktriangleleft g$. So we equip the set of configurations with a structure of category.

Definition 3.3.7. Let **Conf** be the category whose objects are the configurations of Conf, and for any configurations c and c', the set of morphisms $\mathrm{Hom}_{\mathbf{Conf}}(c, c')$ is $\{g \in G \mid c \preceq (c' \blacktriangleleft g)\}$. Morphisms are thus identified with elements of the group. Given two morphisms $g : c \to c'$ and $g' : c' \to c''$, their composite $g' \circ g$ is given by $g' \cdot g$. For each object c, the identity morphism $id_c : c \to c$ is given by the neutral element 1 of the group.

Proposition 3.3.8. *This is a well-defined category.*

Proof. By definition 3.3.7, given two morphisms $g : c \to c'$ and $g' : c' \to c''$, we have $c \preceq (c' \blacktriangleleft g)$ and $c' \preceq (c'' \blacktriangleleft g')$. Then $(c' \blacktriangleleft g) \preceq ((c'' \blacktriangleleft g') \blacktriangleleft g)$ (prop. 3.3.2), and finally $c \preceq ((c'' \blacktriangleleft g') \blacktriangleleft g)$ which rewrites to $c \preceq (c'' \blacktriangleleft (g' \cdot g))$ and proves that the group element $g' \cdot g$ is a morphism from c to c''. Finally, the composition is associative by associativity of the group operation and the identity morphism $id_c : c \to c$ is the neutral morphism for the composition as it is the neutral element for the group operation. □

In order to avoid unnecessary technical details, we do not consider the behavior of the functor on configurations having no result.

Definition 3.3.8. Let $\mathbf{Conf}^\star$ be the full subcategory of **Conf** that contains only *super local configurations*, *i.e.*, $\mathrm{Ob}_{\mathbf{Conf}^\star} = \{c \in \mathrm{Conf} \mid \exists g \in G \text{ s.t. } g \cdot N \subseteq |c|\}$ is the set of configurations that contain at least one local configuration.

Proposition 3.3.9. *This is a well-defined category*

Proof. As a full subcategory $\mathbf{Conf}^\star$ have for objects a subset of the objects of **Conf** and for any two objects $c, d \in \mathrm{Ob}_{\mathbf{Conf}^\star}$ the set $\mathrm{Hom}_{\mathbf{Conf}^\star}(c, d)$ is equal to $\mathrm{Hom}_{\mathbf{Conf}}(c, d)$, the composition is well-defined and respects associativity and identity laws. □

Notice that given any configuration c and any $g \in G$, we can consider the configuration $c \blacktriangleleft g$. Interestingly, the simple relations $c \preceq (c \blacktriangleleft g) \blacktriangleleft g^{-1}$ and $(c \blacktriangleleft g) \preceq c \blacktriangleleft g$ respectively imply the existence of morphisms $g^{-1} : c \to (c \blacktriangleleft g)$ and $g : (c \blacktriangleleft g) \to c$ by definition 3.3.7. These morphisms are inverse morphisms in the sense that we have $g^{-1} \circ g = id_{(c \blacktriangleleft g)}$ and $g \circ g^{-1} = id_c$. So c and $c \blacktriangleleft g$ are isomorphic.

Another remark is that the poset Conf is found in the category **Conf** as expected, in the form of morphisms $1 : c \to c'$ since $c \preceq c' \blacktriangleleft 1 = c'$. Moreover, such a morphism $1 : c \to c'$ has no inverse morphism from c' to c in general. In fact, as a group element $1 \in G$ is its own inverse, but as a morphism from c to c', its inverse exists only if $c = c'$. So the reader should be careful to check whether a group element is manipulated as a morphism or not, and when it is a morphism, the reader should keep in mind the associated pair of configurations.

To work with the category **Conf**, we need to lift monotonic functions, comparisons of monotonic functions and Kan extensions given earlier into the categorical setting. First, monotonic functions become functors (definition 2.1.2). While a monotonic function m sends each $x \preceq y$ to $m(x) \preceq m(y)$, a functor F needs to send each possible morphism $f : x \to y$ to a specific morphism $F(f) : F(x) \to F(y)$ since there are many possible choices. Moreover, F needs to respect compositions and identities.

Comparison of two monotonic functions m and n simply asks to check if it is true or false that, for each x, that $m(x) \preceq n(x)$ (definition 3.3.4). So the collection of monotonic functions is a poset (proposition 3.3.3). Lifting this idea for comparing two given functors F and G consists in choosing a specific $\eta_x : F(x) \to G(x)$ for each object x of the category. Moreover, these choices need to be coherent with image morphisms of F and G. Such a specific choice of a morphism η_x for each x is a natural transformation (definition 3.1.1). There may exist zero, one or more ways to define natural transformations between two functors F and G.

Characterization as a Category Left Kan Extension

The coarse monotonic transition function Φ given in section 3.2.3 is defined on the poset Conf. We lift its definition to categories $\mathbf{Conf}$ / $\mathbf{Conf}^\star$ and get the coarse transition functor. In the same manner, the coarse transition functor can also be characterized as a left Kan extension.

Definition 3.3.9. The *coarse transition functor* $\widetilde{\Phi} : \mathbf{Conf}^\star \to \mathbf{Conf}$ is defined for any $c \in \mathbf{Conf}^\star$ as $\widetilde{\Phi}(c) = \Phi(c)$ (given in def. 3.2.5) and for any morphism $g : c \to c'$ as $\widetilde{\Phi}(g) = g : \Phi(c) \to \Phi(c')$.

Proposition 3.3.10. *The* coarse transition functor $\widetilde{\Phi}$ *is well-defined.*

Proof. First, any object c in $\mathbf{Conf}^\star$ is sent to an object $\widetilde{\Phi}(c) \in \mathbf{Conf}$. We now check that, for any morphism $g : c \to c'$ in $\mathbf{Conf}^\star$, $\widetilde{\Phi}(g)$ is a valid morphism from $\widetilde{\Phi}(c)$ to $\widetilde{\Phi}(c')$. By definition 3.3.7 of morphisms of $\mathbf{Conf}$, this amounts to prove that for any $g \in G$, $c \preceq (c' \blacktriangleleft g)$ implies $\widetilde{\Phi}(c) \preceq (\widetilde{\Phi}(c') \blacktriangleleft g)$, that we obtain as follows.

$$\begin{aligned} c \preceq (c' \blacktriangleleft g) &\implies \Phi(c) \preceq \Phi(c' \blacktriangleleft g) \text{ (by prop 3.3.6 of monotonicity of } \Phi) \\ &\implies \Phi(c) \preceq \Phi(c') \blacktriangleleft g \text{ (by prop 3.2.8)} \\ &\implies \widetilde{\Phi}(c) \preceq \widetilde{\Phi}(c') \blacktriangleleft g \text{ (by def 3.3.9 of } \widetilde{\Phi}) \end{aligned}$$

Finally, we have $\widetilde{\Phi}(g \circ h) = g \circ h = \widetilde{\Phi}(g) \circ \widetilde{\Phi}(h)$ and $\widetilde{\Phi}(id_c) = id_c = 1 = id_{\widetilde{\Phi}(c)}$, which concludes the proof that $\widetilde{\Phi}$ is indeed a functor. □

The idea that we want to visit is to replace the fully shifted local transition function by only the usual local transition function to generate the coarse transition functor as a left Kan extension. Unfortunately, any transition function outputting a single state from each local configuration is doomed to failure. Indeed, a singleton configuration defined at any position $g \in G$ is isomorphic to the singleton configuration with the same state at any other position $g' \in G$. It follows that, categorically speaking, such a local transition function does not provide any positioning information of the resulting states. It is however possible to save the relative positioning between the states by relating pairs of resulting states from pairs of local configurations. Indeed, when two local configurations are separated by some element $b \in G$, so is their pair of results. Moreover, it is not necessary to take all pairs of configurations. It is enough to consider a generating set of the group, all other relative positioning being obtained transitively from them.

Let us make these ideas precise and fix once and for all a generating set B of G, that is, a subset of G such that any element of G can be expressed as a finite combination of elements of B and their inverses under the group operation. For each a relative position $b \in B$, we consider the two positions 1 and b, that we call *centers*, and take all configurations defined on the union of the neighborhood of the centers, that is, with support $1.N \cup b.N$. We also consider $1 \in B$. Indeed, with $b = 1$, the two centers equalize and local configurations are automatically included. The interplay between local configurations and pairs of local configurations is essential and arise from morphisms between them. A morphism between two such configurations is a morphism from $\mathbf{Conf}$ satisfying that centers are sent to centers.

Definition 3.3.10. We define **Loc** to be the category with as objects the configurations $\mathrm{Ob}_{\mathbf{Loc}} = \bigcup_{b\in B}\{b\} \times Q^{N\cup b\cdot N} = \bigcup_{b\in B}\{b\} \times Q^{\{1,b\}\cdot N}$ defined on pairs of neighborhoods, and, for any two such configurations (b, c) and (b', c'), a morphism $g : (b, c) \to (b', c')$ is the data of $g : c \to c' \in \mathbf{Conf}$ such that $g \cdot \{1, b\} \subseteq \{1, b'\}$.

Composition in **Loc** is inherited from **Conf**. For any object $(b, c) \in \mathbf{Loc}$, the morphism $1 : (b, c) \to (b, c)$ acts as the identity.

The relation $g \cdot \{1, b\} \subseteq \{1, b'\}$ encodes the intuition of centers being sent to centers. Indeed, consider a center $p \in \{1, b\}$ of the configuration (b, c). Since, the morphism $g : c \to c'$ means that $c \preceq c' \blacktriangleleft g$, we have $c(p) = (c' \blacktriangleleft g)(p) = c'(g \cdot p)$. So we expect $g \cdot p$ to be a center of (b', c'): either 1 or b'. Now, there are not many possible triples $b, b', g \in G$ satisfying $g \cdot \{1, b\} \subseteq \{1, b'\}$ and it is useful to enumerate the different cases:

1. Identities ($g.1 = 1$ and $g.b = b'$): in this case, we have $g = 1$ and $b = b'$ implying that $c = c'$ and $g = id_c = id_{(b,c)}$ is the identity morphism.

2. Other isomorphisms ($g.1 = b'$ and $g.b = 1$): this case, for which we have $g = b' = b^{-1}$, holds when B contains b and its inverse. The two configurations c and c' describe the exact same local states, but where centers are reversed. The two objects (b, c) and (b', c') in **Loc** are then isomorphic. This case is not useful and may be dropped by considering B without inverse.

3. Local configuration around center 1 ($g.1 = 1$ and $g.b = 1$): in this case, $b = 1$ which means that c is a local configuration. Moreover, we have $g = 1 : c \to c'$ which identifies the occurrence of c on the first center 1 of c'.

4. Local configuration around center b ($g.1 = b'$ and $g.b = b'$): in this case, $b = 1$ which also means that c is a local configuration. But now, we have $g = b' : c \to c'$ which identifies the occurrence of c on the second center b' of c'.

Proposition 3.3.11. Loc *is indeed a category.*

Proof. For any $(b, c) \in \mathbf{Loc}$ the morphism $id_{(b,c)} = 1$ is well-defined as $1 \cdot \{1, b\} = \{1, b\}$. The composition operation is well-defined. Indeed, given $f : (b, c) \to (b', c')$ and $g : (b', c') \to (b'', c'')$, we have:

$$\begin{aligned} & f \cdot \{1, b\} \subseteq \{1, b'\} \wedge g \cdot \{1, b'\} \subseteq \{1, b''\} \\ \iff & (g \cdot f) \cdot \{1, b\} \subseteq g \cdot \{1, b'\} \wedge g \cdot \{1, b'\} \subseteq \{1, b''\} \\ \implies & (g \cdot f) \cdot \{1, b\} \subseteq \{1, b''\}, \end{aligned}$$

which implies that $g \circ f = g \cdot f$ is a morphism of **Loc**. Associativity of the composition and neutrality of the identities are inherited from **Conf** and already proved in proposition 3.3.8. □

Definition 3.3.11. We consider the functor $\pi_2 : \mathbf{Loc} \to \mathbf{Conf}^\star$ defined by $\pi_2(b, c) = c$ on objects and $\pi_2(g) = g$ on morphisms.

Proposition 3.3.12. π_2 *is indeed a functor.*

Proof. For any object (b, c) of **Loc**, c is by definition an object of $\mathbf{Conf}^\star$ as $1 \cdot N \subseteq |c|$ and $b \cdot N \subseteq |c|$. Given any morphism $g : (b, c) \to (b', c')$ observe that $g : c \to c'$ is a morphism of **Conf** and c and c' are object of $\mathbf{Conf}^\star$. Then by the fact that $\mathbf{Conf}^\star$ is a full subcategory of **Conf** it follows that g is a morphism of $\mathbf{Conf}^\star$. As the mapping π_2 is the identity on morphisms, compositions and identities are trivially respected. □

Definition 3.3.12. The *local transition functor* $\dot{\delta} : \mathbf{Loc} \to \mathbf{Conf}$ is defined on any objects (b, c) by $|\dot{\delta}(b, c)| = \{1, b\}$ and $\dot{\delta}(b, c)(p) = \delta(c \blacktriangleleft p \restriction N)$, and on any morphisms $g : (b, c) \to (b', c')$ by $\dot{\delta}(g) = g$.

Proposition 3.3.13. *The local transition functor $\dot{\delta}$ is indeed a functor.*

Proof. First $\dot{\delta}$ is well-defined on objects since for any $(b, c) \in \mathbf{Loc}$, $\dot{\delta}(b, c)$ is in $Q^{\{1,b\}}$ which is a subset of $\mathrm{Ob}_{\mathbf{Conf}}$.

Then, for any morphism $g : (b, c) \to (b', c') \in \mathbf{Loc}$, we need to show that $\dot{\delta}(g)$ is a morphism of **Conf** from $\dot{\delta}(b, c)$ to $\dot{\delta}(b', c')$. By definition 3.3.7 of **Conf**, it rewrites to $\dot{\delta}(b, c) \preceq \dot{\delta}(b', c') \blacktriangleleft g$. We first show that $|\dot{\delta}(b, c)| \subseteq |\dot{\delta}(b', c') \blacktriangleleft g|$ by

$$\begin{aligned}
& |\dot{\delta}(b, c)| \subseteq |\dot{\delta}(b', c') \blacktriangleleft g| \\
\iff & \{1, b\} \subseteq \{g^{-1}, g^{-1} \cdot b'\} && \text{(by def. 3.2.3 of } \blacktriangleleft) \\
\iff & \{1, b\} \subseteq g^{-1} \cdot \{1, b'\} \\
\iff & g \cdot \{1, b\} \subseteq \{1, b'\}
\end{aligned}$$

which trivially holds by conditions on g in Definition 3.3.10. Also, we have for any $p \in \{1, b\}$,

$$\begin{aligned}
\dot{\delta}(b, c)(p) &= \delta(c \blacktriangleleft p \restriction N) \\
&= \delta((c' \blacktriangleleft g) \blacktriangleleft p \restriction N) && \text{(by def. 3.3.7 with } g : c \to c') \\
&= \delta(c' \blacktriangleleft g \cdot p \restriction N) && \text{(by prop. 3.2.4)} \\
&= \dot{\delta}(b', c')(g \cdot p) \\
&= (\dot{\delta}(b', c') \blacktriangleleft g)(p) && \text{(by def. 3.2.3 of } \blacktriangleleft)
\end{aligned}$$

which concludes that $\dot{\delta}(b, c) \preceq \dot{\delta}(b', c') \blacktriangleleft g$ as expected.

Finally, $\dot{\delta}$ respects compositions (for any two morphisms f, g in **Loc**, $\dot{\delta}(g \circ f) = g \circ f = \dot{\delta}(g) \circ \dot{\delta}(f)$) and identities (for any (b, c) in **Loc**, $\dot{\delta}(id_{(b,c)}) = id_c$), which ends the proof that $\dot{\delta}$ is a well-defined functor. □

Definition 3.3.13. We consider the natural transformation $\eta : \dot{\delta} \Rightarrow \widetilde{\Phi} \circ \pi_2$ defined by $\eta_{(b,c)} = 1 : \dot{\delta}(b, c) \to (\widetilde{\Phi} \circ \pi_2)(b, c)$, for any (b, c) in **Loc**.

Proposition 3.3.14. *η is indeed a natural transformation.*

Proof. We start by showing that morphisms are well-defined in **Conf** which means that $\dot{\delta}(b, c) \preceq (\widetilde{\Phi} \circ \pi_2)(b, c)$ by definition 3.3.7. To do so, we show that for any (b, c) in **Loc**, $|\dot{\delta}(b, c)| \subseteq |(\widetilde{\Phi} \circ \pi_2)(b, c)|$ and that $\dot{\delta}(b, c)$ and $(\widetilde{\Phi} \circ \pi_2)(b, c)$ coincide on

$|\dot{\delta}(b,c)|$. For the first part:

$$\begin{aligned}
|(\widetilde{\Phi} \circ \pi_2)(b,c)| &= |\widetilde{\Phi}(c)| && \text{(by def. 3.3.11 of } \pi_2) \\
&= \mathrm{int}(|c|) && \text{(by def. 3.2.5 of } \widetilde{\Phi}) \\
&= \{g \in G \mid g \cdot N \subseteq |c|\} && \text{(by def. 3.2.4 of int)} \\
&= \{g \in G \mid g \cdot N \subseteq \{1,b\} \cdot N\} && \text{(by def. 3.3.10 of } \mathbf{Loc}) \\
&\supseteq \{1,b\} \\
&= |\dot{\delta}(b,c)| && \text{(by def. 3.3.12 of } \dot{\delta})
\end{aligned}$$

For coincidence, consider $g \in |\dot{\delta}(b,c)|$:

$$\dot{\delta}(b,c)(g) = \delta(c \blacktriangleleft g \restriction N) = \widetilde{\Phi}(c)(g) = (\widetilde{\Phi} \circ \pi_2)(b,c)(g).$$

It remains to show the naturality condition on η, which means to check that for any $g : (b,c) \to (b',c') \in \mathbf{Loc}$, we have $\eta_{(b',c')} \circ \dot{\delta}(g) = (\widetilde{\Phi} \circ \pi_2)(g) \circ \eta_{(b,c)}$. It is simply given by $\eta_{(b',c')} \circ \dot{\delta}(g) = 1 \cdot g = g$ and $(\widetilde{\Phi} \circ \pi_2)(g) \circ \eta_{(b,c)} = g \cdot 1 = g$. □

The following lemma will be used in the proof of theorem 3.3.16.

Lemma 3.3.15. *Given a set D and map $f : B \times G \to D$. If for any $(b,g) \in B \times G$, we have $f(1,g) = f(b,g) = f(1, g \cdot b)$ then f is constant.*

Proof. Since for any b and g, $f(b,g) = f(1,g)$, it is enough to show that for any g, $f(1,g) = f(1,1)$ to show that f is a constant function. Since B is a generating set of G, $g = b_1 \cdot \ldots \cdot b_k$ for some natural integer k and $b_i \in B \cup B^{-1}$. In this setting, we define $h_i = b_1 \cdot \ldots \cdot b_i$ for $0 \leq i \leq k$. We now show that $f(1, h_{i+1}) = f(1, h_i)$ for $0 < i \leq k$. For $b_{i+1} \in B$, $f(1, h_{i+1}) = f(1, h_i \cdot b_{i+1}) = f(1, h_i)$ by assumptions on f. If $b_{i+1} \in B^{-1}$, $f(1, h_{i+1}) = f(b_{i+1}^{-1}, h_{i+1}) = f(1, h_{i+1} \cdot b_{i+1}^{-1})$ by using the assumptions on f. Then we get $f(1, h_{i+1} \cdot b_{i+1}^{-1}) = f(1, h_i \cdot b_{i+1} \cdot b_{i+1}^{-1}) = f(1, h_i)$. By transitivity, we obtain $f(1,g) = f(1,h_k) = f(1,h_0) = f(1,1)$. □

Theorem 3.3.16. *$(\widetilde{\Phi}, \eta)$ is the left Kan extension of $\dot{\delta}$ along π_2.*

Proof. Taking any other $\widetilde{\Psi} : \mathbf{Conf}^\star \to \mathbf{Conf}$ and $\rho : \dot{\delta} \Rightarrow \widetilde{\Psi} \circ \pi_2$ we need to show that there exists a unique natural transformation $\lambda : \widetilde{\Phi} \Rightarrow \widetilde{\Psi}$ such that for any $(b,l) \in \mathbf{Loc}$ we have $\lambda_l \circ \eta_{(b,l)} = \rho_{(b,l)}$. Given some $c \in \mathbf{Conf}^\star$, observe that for any $(b,l) \in \mathbf{Loc}$ and $g : l \to c$ in $\mathbf{Conf}^\star$, λ_c must be such that:

$$\begin{aligned}
\lambda_c \circ \widetilde{\Phi}(g) &= \widetilde{\Psi}(g) \circ \lambda_l && \text{(naturality of } \lambda) \\
\lambda_c \circ \widetilde{\Phi}(g) \circ \eta_{(b,l)} &= \widetilde{\Psi}(g) \circ \lambda_l \circ \eta_{(b,l)} \\
\lambda_c \circ \widetilde{\Phi}(g) \circ \eta_{(b,l)} &= \widetilde{\Psi}(g) \circ \rho_{(b,l)} && \text{(constraints on } \lambda) \qquad (3.5)
\end{aligned}$$

Under these constraints we let $\lambda_c^{(b,l,g)}$ to be the unique group element that satisfies equation (3.5), *i.e.*, $\lambda_c^{(b,l,g)} := \widetilde{\Psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(g)^{-1}$, for any $(b,l) \in \mathbf{Loc}$ and $g : l \to c$ in $\mathbf{Conf}^\star$. Observe that there must be at least one such (b,l). Indeed, as $c \in \mathbf{Conf}^\star$ there must be some s such that $s \cdot N \subseteq |c|$ so one can take $b = 1$ and $l = c \blacktriangleleft s \restriction N$.

Let us show that the value of $\lambda_c^{(b,l,g)}$ does not depend on the choice of (b,l) and g. We start with the special case where $c \in Q^G$ is a *global* configuration. Notice

that the triples (b, l, g) with $(b, l) \in \mathbf{Loc}$ and $g : l \to c$, are in bijection with the pair $(b, g) \in B \times G$, since for such a given pair the unique candidate is $l := c \blacktriangleleft g \restriction N \cup b \cdot N$. We are then able to define a function of $B \times G \to G$ mapping any (b, g) to $\lambda_c^{(b,l,g)}$. We show that all the $\lambda_c^{(b,l,g)}$ are equal by showing that this function is constant. For this, we use lemma 3.3.15 which simply requires that for any $(b, g) \in B \times G$:

$$\lambda_c^{(1,l\restriction N, g\circ h_1)} = \lambda_c^{(b,l,g)} = \lambda_c^{(1,l\blacktriangleleft b\restriction N, g\circ h_b)}$$

where $h_1 := 1 : (1, l \restriction N) \to (b, l)$ and $h_b := b : (1, l \blacktriangleleft b \restriction N) \to (b, l)$. The first required equality is derived as follows.

$$\begin{aligned}
\lambda_c^{(1,l\restriction N, g\circ h_1)} &= \widetilde{\Psi}(g \circ h_1) \cdot \rho_{(1,l\restriction N)} \cdot {\eta_{(1,l\restriction N)}}^{-1} \cdot \widetilde{\Phi}(g \circ h_1)^{-1} \\
&= \widetilde{\Psi}(g) \cdot \widetilde{\Psi}(h_1) \cdot \rho_{(1,l\restriction N)} \cdot {\eta_{(1,l\restriction N)}}^{-1} \cdot \widetilde{\Phi}(h_1)^{-1} \cdot \widetilde{\Phi}(g)^{-1} \\
&= \widetilde{\Psi}(g) \cdot \rho_{(b,l)} \cdot \dot{\delta}(h_1) \cdot \dot{\delta}(h_1)^{-1} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(g)^{-1} \\
&= \widetilde{\Psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(g)^{-1} \\
&= \lambda_c^{(b,l,g)}
\end{aligned}$$

where we use the naturality of η and ρ (definition 3.1.1 of natural transformations) and that $\widetilde{\Psi}$ and $\widetilde{\Phi}$ respect the composition (definition 2.1.2 of functors). The second equality is derived in the same manner.

We now consider the general case where c is not necessarily a global configuration. We show that for any $(b', l') \in \mathbf{Loc}$ and $g' : l' \to c$, we have $\lambda_c^{(b,l,g)} = \lambda_c^{(b',l',g')}$. We take some global configuration $d \in Q^G$ with $p : c \to d$. Using the special case of global configurations, we know that:

$$\begin{aligned}
\lambda_d^{(b,l,p\circ g)} &= \lambda_d^{(b',l',p\circ g')} \\
\widetilde{\Psi}(p \circ g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(p \circ g)^{-1} &= \widetilde{\Psi}(p \circ g') \cdot \rho_{(b',l')} \cdot {\eta_{(b',l')}}^{-1} \cdot \widetilde{\Phi}(p \circ g')^{-1} \\
\widetilde{\Psi}(p) \cdot \widetilde{\Psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(p \circ g)^{-1} &= \widetilde{\Psi}(p) \cdot \widetilde{\Psi}(g') \cdot \rho_{(b',l')} \cdot {\eta_{(b',l')}}^{-1} \cdot \widetilde{\Phi}(p \circ g')^{-1} \\
\widetilde{\Psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(g)^{-1} \cdot \widetilde{\Phi}(p)^{-1} &= \widetilde{\Psi}(g') \cdot \rho_{(b',l')} \cdot {\eta_{(b',l')}}^{-1} \cdot \widetilde{\Phi}(g')^{-1} \cdot \widetilde{\Phi}(p)^{-1} \\
\widetilde{\Psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(g)^{-1} &= \widetilde{\Psi}(g') \cdot \rho_{(b',l')} \cdot {\eta_{(b',l')}}^{-1} \cdot \widetilde{\Phi}(g')^{-1} \\
\lambda_c^{(b,l,g)} &= \lambda_c^{(b',l',g')}
\end{aligned}$$

which concludes the proof that the value of $\lambda_c^{(b,l,g)}$ does not depend on the choice of (b, l) and g. Consequently, we are able to define $\lambda_c := \lambda_c^{(b,l,g)}$. To finally get the component morphisms, it remains to show that $\lambda_c \in \mathrm{Hom}_{\mathbf{Conf}}(\widetilde{\Phi}(c), \widetilde{\Psi}(c))$, *i.e.*,, $\widetilde{\Phi}(c) \preceq \widetilde{\Psi}(c) \blacktriangleleft \lambda_c$. Take $g \in |\widetilde{\Phi}(c)|$ and consider $(1, l_g) \in \mathbf{Loc}$ with $l_g := c \blacktriangleleft g \restriction N$. We obviously have $l_g \preceq c \blacktriangleleft g$, which means that we also have a morphism $g : l_g \to c$. By the existence of the morphism $\widetilde{\Psi}(g) \circ \rho_{(1,l_g)} : \dot{\delta}(1, l_g) \to \widetilde{\Psi}(l_g) \to \widetilde{\Psi}(c)$, we have :

$$\begin{aligned}
&\dot{\delta}(1, l_g) \preceq (\widetilde{\Psi}(c) \blacktriangleleft \widetilde{\Psi}(g) \cdot \rho_{(1,l_g)}) \\
\iff &\dot{\delta}(1, l_g) \preceq (\widetilde{\Psi}(c) \blacktriangleleft \widetilde{\Psi}(g) \cdot \rho_{(1,l_g)} \cdot g^{-1} \cdot g) \\
\iff &\dot{\delta}(1, l_g) \preceq (\widetilde{\Psi}(c) \blacktriangleleft \widetilde{\Psi}(g) \cdot \rho_{(1,l_g)} \cdot g^{-1}) \blacktriangleleft g && \text{(by prop. 3.2.4 of } \blacktriangleleft\text{)} \\
\iff &\dot{\delta}(1, l_g) \preceq (\widetilde{\Psi}(c) \blacktriangleleft \widetilde{\Psi}(g) \cdot \rho_{(1,l_g)} \cdot {\eta_{(1,l_g)}}^{-1} \cdot \widetilde{\Phi}(g)^{-1}) \blacktriangleleft g && \text{(by def. 3.3.13 and 3.3.9)} \\
\iff &\dot{\delta}(1, l_g) \preceq (\widetilde{\Psi}(c) \blacktriangleleft \lambda_c^{(1,l_g,g)}) \blacktriangleleft g && \text{(by def. of } \lambda_c^{(1,l_g,g)}\text{)} \\
\iff &\dot{\delta}(1, l_g) \preceq (\widetilde{\Psi}(c) \blacktriangleleft \lambda_c) \blacktriangleleft g && \text{(by def. of } \lambda_c\text{)} \\
\implies &g \in |\widetilde{\Psi}(c) \blacktriangleleft \lambda_c| \wedge (\widetilde{\Psi}(c) \blacktriangleleft \lambda_c)(g) = \delta(l_g) && \text{(by def. 3.3.12)}
\end{aligned}$$

reminding that $1 \in |\dot{\delta}(1, l_g)|$. But $\widetilde{\Phi}(c)(g) = \Phi(c)(g) = \delta(l_g)$. We get $\widetilde{\Phi}(c)(g) = (\widetilde{\Psi}(c) \blacktriangleleft \lambda_c)(g)$ so λ_c is a well-defined morphism.

We then show that, for any $(b, l) \in \mathbf{Loc}$, we have $\lambda_l \circ \eta_{(b,l)} = \rho_{(b,l)}$. Taking the identity morphism $id_l : l \to l$, it gives:

$$\begin{aligned}
\lambda_l \cdot \eta_{(b,l)} &= \lambda_l^{(b,l,id_l)} \cdot \eta_{(b,l)} && \text{(by def. of } \lambda_l\text{)} \\
&= \widetilde{\Psi}(id_l) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\Phi}(id_l)^{-1} \cdot \eta_{(b,l)} && \text{(by def. of } \lambda_c^{(1,l,id_l)}\text{)} \\
&= \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \eta_{(b,l)} && (\widetilde{\Psi} \text{ and } \widetilde{\Phi} \text{ preserves identities}) \\
&= \rho_{(b,l)}
\end{aligned}$$

Finally we show the naturality of λ, *i.e.*, for any $g : c \to c' \in \mathbf{Conf}^\star$ we have $\widetilde{\Psi}(g) \circ \lambda_c = \lambda_{c'} \circ \widetilde{\Phi}(g)$. To see this, take any $(1, l) \in \mathbf{Loc}$ such that there is $i : l \to c$ and observe that:

$$\begin{aligned}
\lambda_{c'} \cdot \widetilde{\Phi}(g) &= \widetilde{\Psi}(g \circ i) \cdot \rho_{(1,l)} \cdot {\eta_{(1,l)}}^{-1} \cdot \widetilde{\Phi}(g \circ i)^{-1} \cdot \widetilde{\Phi}(g) \\
&= \widetilde{\Psi}(g) \cdot \widetilde{\Psi}(i) \cdot \rho_{(1,l)} \cdot {\eta_{(1,l)}}^{-1} \cdot \widetilde{\Phi}(i)^{-1} \cdot \widetilde{\Phi}(g)^{-1} \cdot \widetilde{\Phi}(g) \\
&= \widetilde{\Psi}(g) \cdot \widetilde{\Psi}(i) \cdot \rho_{(1,l)} \cdot {\eta_{(1,l)}}^{-1} \cdot \widetilde{\Phi}(i)^{-1} \\
&= \widetilde{\Psi}(g) \cdot \lambda_c
\end{aligned}$$

which shows that λ is natural.

Uniqueness comes from equation (3.5) which defines uniquely the component morphisms. □

3.3.3 Kan Extensions are "Up To Isomorphisms"

In the case of poset Kan extensions, such an extension is unique when it exists (see proposition 3.3.4). This is not the case for the category Kan extensions (see remark 3.1.2) that we consider, due to the degree of freedom of the absolute positioning of configurations. We now consider the implications of this formal statement in the concrete case of cellular automata. In particular, a cellular automaton and its shifted versions lead to isomorphic coarse transition functors, so they are Kan extensions of the same data as we now show.

Definition 3.3.14. For any $s \in G$, we call the *s-shifted cellular automaton* as the cellular automaton having neighborhood $N_s = s \cdot N$, the same set of states Q, and the local transition function $\delta_s : Q^{N_s} \to Q$ defined as $\delta_s(c) := \delta(c \blacktriangleleft s)$.

Proposition 3.3.17. *For any $c \in Q^{N_s}$, $\delta_s(c)$ is well-defined.*

Proof. Since $\delta : Q^N \to Q$, we simply need to show that $|c \blacktriangleleft s| = N$:

$$\begin{aligned}
|c \blacktriangleleft s| &= \{h \in G \mid s \cdot h \in |c|\} && \text{(def. 3.2.3 of } |c \blacktriangleleft s|\text{)} \\
&= \{h \in G \mid s \cdot h \in s \cdot N\} && \text{(Def. of } c\text{)} \\
&= N
\end{aligned}$$

□

Definition 3.3.15. Given any $s \in G$, the *s-shifted coarse transition function* $\Phi_s : \text{Conf} \to \text{Conf}$ is defined for all $c \in \text{Conf}$ as $|\Phi_s(c)| = \text{int}_s(|c|) = \{g \in G \mid g \cdot N_s \subseteq |c|\}$ and $\Phi_s(c)(g) = \delta_s(c \blacktriangleleft g \restriction N_s)$.

Proposition 3.3.18. *For any $s \in G$ such that $s \cdot g = g \cdot s$ for all $g \in G$, the shifted coarse transition functor $\widetilde{\Phi}_s$ is isomorphic to $\widetilde{\Phi}$, i.e., there exists a natural transformation $\iota : \widetilde{\Phi}_s \Rightarrow \widetilde{\Phi}$ such that for any $c \in$ **Conf** there is some morphism $g : \widetilde{\Phi}(c) \to \widetilde{\Phi}_s(c)$ such that $\iota_c \circ g = id_{\widetilde{\Phi}(c)}$ and $g \circ \iota_c = id_{\widetilde{\Phi}_s(c)}$.*

Proof. Let $\iota_c = s$ for any $c \in$ **Conf**. First we show that ι_c is a morphism from $\widetilde{\Phi}_s(c)$ to $\widetilde{\Phi}(c)$, *i.e.*, $\widetilde{\Phi}_s(c) \preceq \widetilde{\Phi}(c) \blacktriangleleft s$:

$$
\begin{aligned}
&\forall g \in |\widetilde{\Phi}_s(c)|, g \in |\widetilde{\Phi}(c) \blacktriangleleft s| \wedge \widetilde{\Phi}_s(c)(g) = (\widetilde{\Phi}(c) \blacktriangleleft s)(g) \\
\iff &\forall g \text{ s.t. } g \cdot N_s \subseteq |c|, s \cdot g \in |\widetilde{\Phi}(c)| \wedge \delta_s(c \blacktriangleleft g \restriction N_s) = \widetilde{\Phi}(c)(s \cdot g) \\
\iff &\forall g \text{ s.t. } g \cdot s \cdot N \subseteq |c|, s \cdot g \in \text{int}(|c|) \wedge \delta((c \blacktriangleleft g \restriction s \cdot N) \blacktriangleleft s) = \delta(c \blacktriangleleft s \cdot g \restriction N) \\
\iff &\forall g \text{ s.t. } g \cdot s \cdot N \subseteq |c|, s \cdot g \cdot N \subseteq |c| \wedge \delta(c \blacktriangleleft g \cdot s \restriction N) = \delta(c \blacktriangleleft s \cdot g \restriction N),
\end{aligned}
$$

line 2 is given by definition 3.3.15 and definition 3.2.3, line 3 by definitions 3.3.14 and 3.2.5 and line 4 by propositions 3.2.6, 3.2.5. The last line is true as $g \cdot s = s \cdot g$. In fact, as $g \in |\widetilde{\Phi}_s(c)|$ is equivalent to $g \cdot s \cdot N \subseteq |c|$ which is also equivalent to $g \in |\widetilde{\Phi}(c) \blacktriangleleft s|$ we have $\widetilde{\Phi}_s(c) = \widetilde{\Phi}(c) \blacktriangleleft s$. It follows that $s : \widetilde{\Phi}_s(c) \to \widetilde{\Phi}(c)$ is a valid morphism. But we can write equivalently $\widetilde{\Phi}_s(c) \blacktriangleleft s^{-1} = \widetilde{\Phi}(c)$ which gives that $s^{-1} : \widetilde{\Phi}(c) \to \widetilde{\Phi}_s(c)$ is a valid morphism. Then by $s \circ s^{-1} = s \cdot s^{-1} = 1$ and $s^{-1} \circ s = s^{-1} \cdot s = 1$ we get that s is an isomorphism. Finally, we show the naturality of ι, *i.e.*, $\widetilde{\Phi}(f) \circ \iota_c = \iota_{c'} \circ \widetilde{\Phi}_s(f)$ for any $f : c \to c'$:

$$
\begin{aligned}
\widetilde{\Phi}(f) \circ \iota_c &= f \cdot s \\
&= s \cdot f \\
&= \iota_{c'} \circ \widetilde{\Phi}(f)
\end{aligned}
$$

It is true as line 1 holds by definition 3.3.9 and by definition of ι, line 2 as $f \cdot s = s \cdot f$ and line 3 holds by definition of ι and definition 3.3.9. □

3.3.4 Coarse Functor and Global Transformations

As the coarse transition functor $\widetilde{\Phi}$ is characterized by a left Kan extension, it is worth to study its link with global transformations.

We first note that π_2 is not a fully faithful functor. As a concrete example, let $G = \mathbb{Z}$, $N = \{-1, 0, 1\}$, $B = \{0, 1\}$, and $(0, c)$ and $(1, c')$ where c, c' are constant functions to some state. Then there is only one function from $0 : (0, c) \to (1, c') \in \text{Loc}$ but two $0, 1 : c \to c'$ in **Conf**. It happens that π_2 does not need to be injective on objects either. As a concrete example, take $G = \mathbb{Z}/2\mathbb{Z}$ with $N = B = \{0, 1\}$, then take $(0, c)$ and $(1, c)$ with $|c| = \{0, 1\}$.

As we have seen in section 3.1, the fact that π_2 is not fully faithful is equivalent to the fact η is not a natural isomorphism. We can also wonder if, as a left Kan extension, $\langle \widetilde{\Phi}, \eta \rangle$ is pointwise or not.

In fact, it is clear that it is not the case. Indeed, consider two local configurations found at position g and h in two different connected components of π_2/c. Then

a cocone C over $\dot{\delta} \circ \mathrm{Proj}[\pi_2/c]$ can send their corresponding local results to any positions g', h' in G. In particular, take $g' = g$ and $h' \neq h$. Then, apart from some degenerate cases, there is no mediating morphism m from $\Phi(c)$ to the cocone C as such inclusion $m : \Phi(c) \to C$ must send g to g' and h to h'. This is reflected in the proof of theorem 3.3.16 by the use of a global configuration $c \in Q^G$ to prove that every $\lambda_c^{(b,l,g)}$ are equal.

All those facts imply that the left Kan extension $\langle \widetilde{\Phi}, \eta \rangle$ falls outside the usual definition of global transformations strictly speaking unless we restrict ourselves to (1) connected partial configurations and (2) L is fully faithful. However, these restrictions do not appear as strong bottlenecks. Indeed, the former is perfectly acceptable when the chosen neighborhood N and the generated set B coincide. For the latter, having a l.h.s. functor L not fully faithful can be turned into a rule system with a fully faithful l.h.s. functor: it suffices to replace $\mathbf{\Gamma}$ by $\mathbf{\Gamma}'$ the full image of L in $\mathbf{C}$, *i.e.*, the image of L completed with the missing morphisms, so that the corresponding inclusion $\mathrm{L}' : \mathbf{\Gamma}' \to \mathbf{C}$ is fully faithful. Then, to guarantee that this new rule system T' does the same computation as T, it suffices to replace R by $\mathrm{R}' = T \circ \mathrm{L}'$. The lifting of these two restrictions is left for future work.

3.4 Fine Transition as Kan Extensions

The study of the coarse transition function Φ as left Kan extensions emphasized how this function can be seen as the "minimal" extension of a cellular automaton to partial configurations. In contrast, the fine transition function ϕ can be seen as the "maximal" extension of a cellular automaton to partial configurations. Indeed, it is defined over the determined subset of a configuration which contains every position where the result of a cell can be deduced, whether or not the neighborhood is known at this position.

We proceed here as for the coarse transition function: we study the proposition in the context of the simpler setting of the poset of subconfigurations. We then explore the extension of the proposition to the case of the category of subconfigurations.

3.4.1 The Fine Monotonic Transition Function

Once again, we restrict ourselves to the simpler setting of Kan extension for poset. As for the coarse transition function, the fine transition function ϕ is defined on the set of all configurations Conf and we claim that it is generated, in the poset Kan extension sense. To do this, we first observe that ϕ is a monotonic function. Then we consider two ways to generate it and start by the simplest one. The second one is considered in the following using sub-local configurations in order to be closer to the direct definition and to be a "from local to global" characterization.

Proposition 3.4.1. *For any $g \in G$, the function $-_g : \mathrm{Conf} \to \mathrm{Conf}$ of definition 3.2.6 is monotonic.*

Proof. As one can easily check. □

Proposition 3.4.2. *The fine transition function ϕ is monotonic.*

Proof. Indeed, take $c_0, c_1 \in \text{Conf}$ such that $c_0 \preceq c_1$. We want to prove that $\phi(c_0) \preceq \phi(c_1)$ and this is equivalent to:

$$\begin{aligned} & \forall g \in |\phi(c_0)|, g \in |\phi(c_1)| \wedge \phi(c_0)(g) = \phi(c_1)(g) \text{ (def 3.3.2 of } \preceq) \\ \iff & \forall g \in \det(c_0), g \in \det(c_1) \wedge q_{c_0,g} = q_{c_1,g} \text{ (def 3.2.8 of } \phi) \end{aligned}$$

Take $g \in \det(c_0)$. We want to prove that $g \in \det(c_1)$, which means by definition 3.2.7 of $\det(c_1)$:

$$\exists q \in Q, \forall c_2 \in Q^{g \cdot N}, c_2 \upharpoonright |(c_1)_g| = (c_1)_g \implies \delta(c_2 \blacktriangleleft g) = q$$

We claim that the property is verified with $q = q_{c_0,g}$. Indeed, take any $c_2 \in Q^{g \cdot N}$ such that $c_2 \upharpoonright |(c_1)_g| = (c_1)_g$. We also have that $c_2 \upharpoonright |(c_0)_g| = (c_0)_g$ since the hypothesis $c_0 \preceq c_1$ implies $(c_0)_g \preceq (c_1)_g$ by proposition 3.4.1. By definition 3.2.7 of $\det(c_0)$, we obtain that $\delta(c_2 \blacktriangleleft g) = q_{c_0,g}$, so $q = q_{c_0,g}$ has the wanted property, which implies that $g \in \det(c_1)$ as wanted. But the above property of q set it to be precisely what we denote by $q_{c_1,g}$ (def. 3.2.7 of $q_{c_1,g}$), so $q_{c_0,g} = q_{c_1,g}$. □

Characterization as a Poset Right Kan Extension

Recall that determined subset $\det(c)$ of a configuration c is computed from the results of bigger configurations. In particular, an element $g \in G$ is in $\det(c)$ if the result of the global transition function Δ at this position is the same for any global configuration $c' \succeq c$, *i.e.*, there exists some state q such that for any such c' $\Delta(c')(g) = q$. Following this idea, we then characterize the fine monotonic transition function ϕ as a right Kan extension of the global transition function Δ.

Proposition 3.4.3. *The fine transition function ϕ is the right Kan extension of the global transition function Δ along the inclusion $i : Q^G \to \text{Conf}$.*

Proof. By definition 3.3.5 of right Kan extensions, we need to prove firstly that ϕ is such that $\phi \circ i \Rightarrow \Delta$, and secondly that it is greater than any other such monotonic functions.

For the first part, we actually have $\phi \circ i = \Delta$ since for any $c \in Q^G$, $|\phi(c)| = \det(c) = G = |\Delta(c)|$ and for any $g \in G$, we have $\phi(c)(g) = q_{c,g} = \delta(c_g \blacktriangleleft g) = \delta((c \upharpoonright g \cdot N) \blacktriangleleft g) = \delta((c \blacktriangleleft g) \upharpoonright N) = \Delta(c)(g)$ by defs. 3.2.8, 3.2.7, 3.2.6, 3.2.2 of ϕ, det, c_g and Δ and prop. 3.2.5.

For the second part, let $f : \text{Conf} \to \text{Conf}$ be a monotonic function such that $f \circ i \Rightarrow \Delta$. We want to show that $f \Rightarrow \phi$, which is equivalent to:

$$\begin{aligned} & \forall c \in \text{Conf}, f(c) \preceq \phi(c) \text{ (def. 3.3.4 of } \Rightarrow) \\ \iff & \forall c \in \text{Conf}, \forall g \in |f(c)|, g \in |\phi(c)| \wedge f(c)(g) = \phi(c)(g) \text{ (def. 3.3.2 of } \preceq) \\ \iff & \forall c \in \text{Conf}, \forall g \in |f(c)|, g \in \det(c) \wedge f(c)(g) = q_{c,g} \text{ (def. 3.2.8 of } \phi) \\ \iff & \forall c \in \text{Conf}, \forall g \in |f(c)|, \forall c' \in Q^{g \cdot N}, c \preceq c' \implies f(c)(g) = \delta(c' \blacktriangleleft g) \text{ (d. 3.2.7)} \end{aligned}$$

So take $c \in \text{Conf}$ and $g \in |f(c)|$ and $c' \in Q^{g \cdot N}$ such that $c \preceq c'$. Consider any $c'' \in Q^G$ such that $c' \preceq c''$ (or equivalently $c'' \upharpoonright g \cdot N = c'$). Since f is monotonic, we have $f(c) \preceq f(c'')$, which means that $f(c)(g) = f(c'')(g)$ by def. 3.3.2. But since $f \circ i \Rightarrow \Delta$, we have $f(c)(g) = \Delta(c'')(g) = \delta((c'' \blacktriangleleft g) \upharpoonright N)$ by def. 3.3.4 of $\Rightarrow$ and def. 3.2.2 of Δ. But by prop. 3.2.5, $(c'' \blacktriangleleft g) \upharpoonright N = (c'' \upharpoonright g \cdot N) \blacktriangleleft g = c' \blacktriangleleft g$. □

Proposition 3.4.4. *Let us consider another cellular automaton having neighborhood $N' \subseteq G$ and local transition function $\delta' : Q^{N'} \to Q$. Consider its corresponding global transition function $\Delta' : Q^N \to Q^N$ and fine transition function $\phi' : \mathrm{Conf} \to \mathrm{Conf}$. Then if $\Delta' = \Delta$, then $\phi' = \phi$.*

Proof. By propositions 3.4.3 and 3.3.4, ϕ is determined by Δ, and ϕ' by Δ'. So $\Delta' = \Delta$ gives $\phi = \phi'$. □

The Sub-Local Transition Function

The direct definition of the fine transition function is explicitly about assigning a result for a configuration c at a given $g \in G$ even when the whole neighborhood $g \cdot N$ is not complete. By isolating these "shifted sub-local configurations" in the poset of configurations, we can (right-)extend the local transition to them and show that, in the same way as the coarse transition function is the left Kan extension of the local transition function, the fine transition function is the left Kan extension of the sub-local transition function.

Definition 3.4.1. We define $\mathrm{Sub} = \bigcup_{g \in G, M \subseteq N} (\{g\} \times Q^{g \cdot M})$ with partial order defined as $(g, c) \preceq (g', c')$ if and only if $g = g'$ and $c \preceq c'$. The "fully shifted sub-local transition function" $\underline{\delta} : \mathrm{Sub} \to \mathrm{Conf}$ is defined, for any $g \in G$, any $M \subseteq N$ and any $c \in Q^{g \cdot M}$, as $|\underline{\delta}(g, c)| = \{g\} \cap \mathrm{det}(c)$ and, if $g \in \mathrm{det}(c)$, $\underline{\delta}(g, c)(g) = q_{c',g}$, *i.e.*, $\underline{\delta}(g, c)(g) = \delta(c' \blacktriangleleft g)$ for any $c' \in Q^{g \cdot N}$ such that $c = c' \restriction |c|$. The second projection of Sub is the function $\pi_2 : \mathrm{Sub} \to \mathrm{Conf}$ defined as $\pi_2(g, c) = c$.

In this definition, a given sub-local configuration can result either in an empty configuration when the transition is not determined, or in a configuration with only singleton support when the transition is determined.

Note that for a given cellular automaton, it is possible to restrict the poset Sub to an antichain. Indeed, any time a result is determined by a sub-local configuration (g, c), all bigger sub-local configuration (g, c') with $c \preceq c'$ does not contribute anything new. We do not elaborate on this because this antichain would be different for each cellular automaton, blurring the global picture presented below.

Proposition 3.4.5. *The fully shifted sub-local transition function $\underline{\delta}$ is monotonic*

Proof. As usual, take $(g, c), (g', c') \in \mathrm{Sub}$ such that $(g, c) \preceq (g', c')$. First, note that $g = g'$ by definition 3.4.1. We want to prove that $\underline{\delta}(g, c) \preceq \underline{\delta}(g, c')$ and this is equivalent to:

$$\begin{aligned} & \forall h \in |\underline{\delta}(c)|, h \in |\underline{\delta}(c')| \wedge \underline{\delta}(c)(h) = \underline{\delta}(c')(h) \\ \iff & \forall h \in \{g\} \cap \mathrm{det}(c), h \in \{g\} \cap \mathrm{det}(c') \wedge q_{c,g} = q_{c',g} \\ \iff & g \in \mathrm{det}(c) \implies g \in \mathrm{det}(c') \wedge q_{c,g} = q_{c',g'}, \end{aligned}$$

by definition 3.2.3 of $\preceq$ and definition 3.4.1 of $\underline{\delta}$. The end of this proof is similar to the one of proposition 3.4.2. □

Proposition 3.4.6. *The fully shifted sub-local transition function $\underline{\delta}$ is the right Kan extension of the fully shifted local transition function $\overline{\delta}$ along the inclusion $i : \mathrm{Loc} \to \mathrm{Sub}$.*

Proof. By definition 3.3.5 of right Kan extensions, we need to prove firstly that $\underline{\delta}$ is such that $\underline{\delta} \circ i \Rightarrow \overline{\delta}$, and secondly that it is greater than any other such monotonic functions.

For the first part, $\underline{\delta} \circ i \Rightarrow \overline{\delta}$ is equivalent to:

$$\begin{aligned}
&\forall (g,c) \in \mathrm{Loc}, \underline{\delta}(g,c) \preceq \overline{\delta}(g,c) \text{ (def. 3.3.4 of } \Rightarrow) \\
\iff &\forall (g,c) \in \mathrm{Loc}, \forall h \in |\underline{\delta}(g,c)|, h \in |\overline{\delta}(g,c)| \wedge \underline{\delta}(g,c)(h) = \overline{\delta}(g,c)(h) \text{ (d. 3.3.2 } \preceq) \\
\iff &\forall (g,c) \in \mathrm{Loc}, g \in \det(c) \implies g \in |\overline{\delta}(g,c)| \wedge q_{c,g} = \overline{\delta}(g,c)(g) \text{ (def. 3.4.1 of } \underline{\delta}) \\
\iff &\forall (g,c) \in \mathrm{Loc}, g \in \det(c) \implies g \in \{g\} \wedge q_{c,g} = \delta(c \blacktriangleleft g) \text{ (def. 3.3.6 of } \overline{\delta}).
\end{aligned}$$

This last statement is true by def. 3.2.7 of $q_{c,g}$.

For the second part, let $f : \mathrm{Sub} \to \mathrm{Conf}$ be a monotonic function such that $f \circ i \Rightarrow \overline{\delta}$. We want to show that $f \Rightarrow \underline{\delta}$, which is equivalent to:

$$\begin{aligned}
&\forall (g,c) \in \mathrm{Sub}, f(g,c) \preceq \underline{\delta}(g,c) \text{ (def. 3.3.4 of } \Rightarrow) \\
\iff &\forall (g,c) \in \mathrm{Sub}, \forall h \in |f(g,c)|, h \in |\underline{\delta}(g,c)| \wedge f(g,c)(h) = \underline{\delta}(g,c)(h) \text{ (def. 3.3.2)} \\
\iff &\forall (g,c) \in \mathrm{Sub}, \forall h \in |f(g,c)|, h \in \{g\} \cap \det(c) \wedge f(g,c)(h) = q_{c,g} \text{ (def. 3.4.1)}
\end{aligned}$$

So take $(g,c) \in \mathrm{Sub}$ and $h \in |f(g,c)|$. Consider any $c' \in \mathrm{Loc}$ such that $c \preceq c'$. Since f is monotonic, we have $f(g,c) \preceq f(g,c')$, which means that $h \in |f(g,c')|$ and $f(g,c)(h) = f(g,c')(h)$ by def. 3.3.2. But since $f \circ i \Rightarrow \overline{\delta}$, we have $h \in |\overline{\delta}(g,c')| = \{g\}$ and $f(g,c')(h) = \overline{\delta}(g,c')(h) = \delta(c' \blacktriangleleft g)$ by def. 3.3.4 of $\Rightarrow$ and def. 3.3.6 of $\overline{\delta}$. Since this is true for any c', this establishes exactly the defining property of $\det(c)$ by def. 3.2.7. □

Characterization as a Poset Left Kan Extension

We aim at obtaining the fine transition function as an extension only of the local transition function to get a "from local to global" characterization. As we have just seen, $\underline{\delta}$ is the right Kan extension of the fully shifted transition function $\dot{\delta}$. It remains to observe that $\underline{\delta}$ characterizes the fine transition function ϕ as left Kan extension.

Proposition 3.4.7. *The projection function* $\pi_2 : \mathrm{Sub} \to \mathrm{Conf}$ *is monotonic.*

Proof. As can be readily checked in definition 3.4.1. □

Proposition 3.4.8. ϕ *is the left Kan extension of* $\underline{\delta}$ *along* $\pi_2 : \mathrm{Sub} \to \mathrm{Conf}$.

Proof. By definition 3.3.5 of left Kan extensions, we need to prove firstly that ϕ is such that $\underline{\delta} \Rightarrow \phi \circ \pi_2$, and secondly that it is smaller than any other such monotonic functions.

For the first part, $\underline{\delta} \Rightarrow \phi \circ \pi_2$ is equivalent to:

$$\begin{aligned}
&\forall (g,c) \in \mathrm{Sub}, \underline{\delta}(g,c) \preceq \phi(c) \text{ (defs. 3.3.4 and 3.4.1 of } \Rightarrow \text{ and } \pi_2) \\
\iff &\forall (g,c) \in \mathrm{Sub}, \forall h \in |\underline{\delta}(g,c)|, h \in |\phi(c)| \wedge \underline{\delta}(g,c)(h) = \phi(c)(h) \text{ (def 3.3.2 of } \preceq) \\
\iff &\forall (g,c) \in \mathrm{Sub}, g \in \det(c) \implies g \in |\phi(c)| \wedge q_{c,g} = \phi(c)(g) \text{ (def. 3.4.1 of } \underline{\delta}) \\
\iff &\forall (g,c) \in \mathrm{Sub}, g \in \det(c) \implies g \in \det(c) \wedge q_{c,g} = q_{c,g} \text{ (def 3.2.8 of } \phi),
\end{aligned}$$

a most trivial statement.

For the second part, let $f : \mathrm{Conf} \to \mathrm{Conf}$ be a monotonic function such that $\underline{\delta} \Rightarrow f \circ \pi_2$. We want to show that $\phi \Rightarrow f$, which is equivalent to:

$$\begin{aligned} & \forall c \in \mathrm{Conf}, \phi(c) \preceq f(c) \text{ (def. 3.3.4 of } \Rightarrow) \\ \iff & \forall c \in \mathrm{Conf}, \forall g \in |\phi(c)|, g \in |f(c)| \wedge \phi(c)(g) = f(c)(g) \text{ (def. 3.3.2 of } \preceq) \\ \iff & \forall c \in \mathrm{Conf}, \forall g \in \det(c), g \in |f(c)| \wedge q_{c,g} = f(c)(g) \text{ (def. 3.2.8 of } \phi) \end{aligned}$$

So take $c \in \mathrm{Conf}$ and $g \in \det(c)$. Since $c_g \preceq c$ (def. 3.2.6) and f is monotonic, we have $f(c_g) \preceq f(c)$. Moreover $\underline{\delta} \Rightarrow f \circ \pi_2$ and $(g, c_g) \in \mathrm{Sub}$ so $\underline{\delta}(g, c_g) \preceq f(c_g)$ by definitions 3.3.4 and 3.3.6 of $\Rightarrow$ and π_2. By transitivity $\underline{\delta}(g, c_g) \preceq f(c)$. By definition 3.4.1 of $\underline{\delta}$ and definition 3.3.2 of $\preceq$, we therefore have $g \in |f(c)|$, and $f(c)(g) = \underline{\delta}(g, d_g)(g) = q_{c,g}$ as wanted. □

3.4.2 The Fine Transition Functor

The fine transition function ϕ given in section 3.2.4 is monotonic in the poset Conf. We lift its definition to the category **Conf** and get the fine transition functor. In the same manner as the coarse transition functor, the fine transition functor can also be characterized as a left Kan extension.

Definition 3.4.2. The *fine transition functor* $\widetilde{\phi} : \mathbf{Conf} \to \mathbf{Conf}$ is defined for any $c \in \mathbf{Conf}$ as $\widetilde{\phi}(c) = \phi(c)$ and for any morphism $g : c \to c'$ as $\widetilde{\phi}(g) = g : \phi(c) \to \phi(c')$.

Proposition 3.4.9. *The* fine transition functor $\widetilde{\phi}$ *is well-defined.*

Proof. First, any object c in **Conf** is sent to an object $\widetilde{\phi}(c) \in \mathbf{Conf}$. We now check that, for any morphism $g : c \to c'$ in **Conf**, $\widetilde{\phi}(g)$ is a valid morphism from $\widetilde{\phi}(c)$ to $\widetilde{\phi}(c')$. By definition 3.3.7 of morphisms of **Conf**, this amounts to prove that for any $g \in G$, $c \preceq (c' \blacktriangleleft g)$ implies $\widetilde{\phi}(c) \preceq (\widetilde{\phi}(c') \blacktriangleleft g)$, that we obtain as follows.

$$\begin{aligned} c \preceq (c' \blacktriangleleft g) & \implies \phi(c) \preceq \phi(c' \blacktriangleleft g) \text{ (by prop. 3.4.2 of monotonicity of } \phi) \\ & \implies \phi(c) \preceq \phi(c') \blacktriangleleft g \text{ (by prop. 3.2.13)} \\ & \implies \widetilde{\phi}(c) \preceq \widetilde{\phi}(c') \blacktriangleleft g \text{ (by def. 3.4.2 of } \widetilde{\phi}) \end{aligned}$$

Finally, we have $\widetilde{\phi}(g \circ h) = g \circ h = \widetilde{\phi}(g) \circ \widetilde{\phi}(h)$ and $\widetilde{\phi}(id_c) = id_c = 1 = id_{\widetilde{\phi}(c)}$, which concludes the proof that $\widetilde{\phi}$ is indeed a functor. □

Characterization as a Left Kan Extension

In the same way as for the coarse transition functor, we want to replace the fully shifted sub-local transition function by a functor taking only into account the relative positioning of sub-local configurations. For each relative position $b \in B$, we take all configurations defined on the union of the subsets of the neighborhood of the centers 1 and b, that is, with support $L \cup b \cdot R$ with $L, R \subseteq N$. This also describes pairs of sub-local configurations and when $b = 1$ sub-local configurations. Then, similarly to **Loc**, we define a category **Sub** having these configurations as objects and for morphisms the morphisms from **Conf** satisfying that centers are sent to centers.

Definition 3.4.3. We define **Sub** to be the category with as objects the configurations $\mathrm{Ob}_{\mathbf{BSub}} := \bigcup_{b\in B, L\subseteq N, R\subseteq N} \{b\} \times Q^{L\cup b\cdot R}$ defined on pairs of neighborhoods, and, for any two such configurations (b, c) and (b', c'), a morphism $g : (b, c) \to (b', c')$ is the data of $g : c \to c' \in \mathbf{Conf}$ such that $g \cdot \{1, b\} \subseteq \{1, b'\}$. Composition in **Sub** is inherited from **Conf**. For any object $(b, c) \in \mathbf{Sub}$, the morphism $1 : (b, c) \to (b, c)$ acts as the identity.

The local configurations are also objects in **Sub** so every object and morphism of **Loc** are also objects and morphisms of **Sub**. The new morphisms in **Sub** are isomorphisms 1 and b^{-1} of sub-local configurations, defined in the same way that for local configurations, and inclusions $1, b$ or b^{-1} of sub-local configurations into "bigger" sub-local configurations. These inclusions can be of two sorts:

- Inclusions $1 : (1, l) \to (b, l')$ or $b : (1, l) \to (b, l')$ from sub-local configurations into pairs of sub-local configurations.
- Inclusions $1 : (b, l) \to (b, l')$ or $b^{-1} : (b, l) \to (b^{-1}, l')$ between pairs of sub-local configurations that act as restrictions of the isomorphisms of sub-local configurations.

Proposition 3.4.10. Sub *is indeed a category.*

Proof. The proof is the same as the proof that **Loc** is a category (proposition 3.3.11). □

Definition 3.4.4. The *sub-local transition functor* $\underline{\delta} : \mathbf{Sub} \to \mathbf{Conf}$ is defined on any object (b, c) by $|\underline{\delta}(b, c)| = \{1, b\} \cap \det(c)$ and $\underline{\delta}(b, c)(p) = q_{c,p}$, and on any morphism $g : (b, c) \to (b', c')$ by $\underline{\delta}(g) = g$.

Proposition 3.4.11. *The sub-local transition functor* $\underline{\delta}$ *is indeed a functor.*

Proof. First $\underline{\delta}$ is well-defined on objects, since for any $(b, c) \in \mathbf{Sub}$, $\underline{\delta}(b, c)$ is in $Q^{\{1,b\}\cap\det(c)}$ which is a subset of $\mathrm{Ob}_{\mathbf{Conf}}$.

Then, take any morphism $g : (b, c) \to (b', c') \in \mathbf{Sub}$. By definition 3.3.7 of **Conf**, it rewrites to $c \preceq c' \blacktriangleleft g$. Recall that as $c \preceq c' \blacktriangleleft g$, proposition 3.4.2 gives $\widetilde{\phi}(c) \preceq \widetilde{\phi}(c' \blacktriangleleft g)$, that is:

$$\det(c) \subseteq \det(c' \blacktriangleleft g), \tag{3.6}$$

and,

$$\forall p \in \det(c), q_{c,p} = q_{c'\blacktriangleleft g,p}. \tag{3.7}$$

We need to show that $\underline{\delta}(g)$ is a morphism of **Conf** from $\underline{\delta}(b, c)$ to $\underline{\delta}(b', c')$, *i.e.*, $\underline{\delta}(b, c) \preceq \underline{\delta}(b', c') \blacktriangleleft g$. First, we get $|\underline{\delta}(b, c)| \subseteq |\underline{\delta}(b', c') \blacktriangleleft g|$ by

$$\begin{aligned}
& |\underline{\delta}(b, c)| \subseteq |\underline{\delta}(b', c') \blacktriangleleft g| \\
\iff & |\underline{\delta}(b, c)| \subseteq g^{-1} \cdot |\underline{\delta}(b', c')| && \text{(by def. 3.2.3 of } \blacktriangleleft) \\
\iff & \{1, b\} \cap \det(c) \subseteq \{g^{-1}, g^{-1} \cdot b'\} \cap g^{-1} \cdot \det(c') && \text{(def. 3.4.4)} \\
\iff & \{1, b\} \cap \det(c) \subseteq \{g^{-1}, g^{-1} \cdot b'\} \cap \det(c' \blacktriangleleft g) && \text{(by prop. 3.2.12)} \\
\impliedby & \{1, b\} \subseteq \{g^{-1}, g^{-1} \cdot b'\} && \text{(by eq. (3.6))} \\
\iff & \{1, b\} \subseteq g^{-1} \cdot \{1, b'\} \\
\iff & g \cdot \{1, b\} \subseteq \{1, b'\}
\end{aligned}$$

which trivially holds by conditions on g in definition 3.4.3. Also, take any $p \in \{1, b\} \cap \det(c)$, that gives:

$$\begin{aligned}
\underline{\delta}(b,c)(p) &= q_{c,p} \\
&= q_{c' \blacktriangleleft g,p} && \text{(by eq. (3.7))} \\
&= q_{c',g\cdot p} && \text{(by prop. 3.2.12)} \\
&= \underline{\delta}(b',c')(g \cdot p) && \text{(by def. 3.2.3 of } \blacktriangleleft\text{)} \\
&= (\underline{\delta}(b',c') \blacktriangleleft g)(p)
\end{aligned}$$

which concludes that $\underline{\delta}(b,c) \preceq \underline{\delta}(b',c') \blacktriangleleft g$ as expected.

Finally, $\underline{\delta}$ respects compositions (for any two morphisms f, g in **Sub**, $\underline{\delta}(g \circ f) = g \circ f = \underline{\delta}(g) \circ \underline{\delta}(f)$) and identities (for any (b,c) in **Sub**, $\underline{\delta}(id_{(b,c)}) = id_c$), which ends the proof that $\underline{\delta}$ is a well-defined functor. □

Definition 3.4.5. We consider the functor $\pi_2 : \mathbf{Sub} \to \mathbf{Conf}$ defined by $\pi_2(b,c) = c$ on objects and $\pi_2(g) = g$ on morphisms.

Proposition 3.4.12. *π_2 is indeed a functor.*

Proof. For any $(b,c) \in \mathbf{Sub}$, c is by definition an object of **Conf**. Given any morphism $g : (b,c) \to (b',c')$ in **Sub**, observe that $g : c \to c'$ is a morphism of **Conf** and c an c' are object of **Conf**. As the mapping π_2 does the identity on morphisms, composition and identities are trivially respected. □

Definition 3.4.6. We consider the natural transformation $\eta : \underline{\delta} \Rightarrow \widetilde{\phi} \circ \pi_2$ defined by $\eta_{(b,c)} = 1 : \underline{\delta}(b,c) \to (\widetilde{\phi} \circ \pi_2)(b,c)$ for any $(b,c) \in \mathbf{Sub}$.

Proposition 3.4.13. *η is indeed a natural transformation.*

Proof. We start by showing that the morphisms are well-defined in **Conf** *i.e.*, $|\underline{\delta}(b,c)| \subseteq |(\widetilde{\phi} \circ \pi_2)(b,c)|$ and that $\underline{\delta}(b,c)$ and $(\widetilde{\phi} \circ \pi_2)(b,c)$ coincide on $|\underline{\delta}(b,c)|$. For the first part:

$$\begin{aligned}
|(\widetilde{\phi} \circ \pi_2)(b,c)| &= |\widetilde{\phi}(c)| && \text{(by Def of } \pi_2\text{)} \\
&= \det(c) && \text{(by def. 3.2.8)} \\
&\supseteq \{1, b\} \cap \det(c) \\
&= |\underline{\delta}(b,c)| && \text{(by def. 3.4.4)}
\end{aligned}$$

For the coincidence part, consider $g \in |\underline{\delta}(b,c)|$. Then $g \in \det(c)$ means:

$$\begin{aligned}
\underline{\delta}(b,c)(g) &= q_{c,g} \\
&= \widetilde{\phi}(c)(g) \\
&= (\widetilde{\phi} \circ \pi_2)(b,c)(g)
\end{aligned}$$

It remains to show the naturality condition on η, which means to check that for any $g : (b,c) \to (b',c')$ in **Sub** we have $\eta_{(b',c')} \circ \underline{\delta}(g) = (\widetilde{\phi} \circ \pi_2)(g) \circ \eta_{(b,c)}$. It is simply given by $\eta_{(b',c')} \circ \underline{\delta}(g) = 1 \cdot g = g$ and $(\widetilde{\phi} \circ \pi_2)(g) \circ \eta_{(b,c)} = g \cdot 1 = g$. □

Theorem 3.4.14. *$(\widetilde{\phi}, \eta)$ is the left Kan extension of $\underline{\delta}$ along π_2.*

Proof. Taking any other $\widetilde{\psi} : \mathbf{Conf} \to \mathbf{Conf}$ and $\rho : \underline{\delta} \Rightarrow \widetilde{\psi} \circ \underline{\pi}_2$ we need to show that there exists a unique natural transformation $\lambda : \widetilde{\phi} \Rightarrow \widetilde{\psi}$ such that for any $(b, l) \in \mathbf{Sub}$ we have $\lambda_l \circ \eta_{(b,l)} = \rho_{(b,l)}$. Given some $c \in \mathbf{Conf}$, observe that for any $(b, l) \in \mathbf{Sub}$ and $g : l \to c$ in $\mathbf{Conf}$, λ_c must be such that:

$$\begin{aligned}
\lambda_c \circ \widetilde{\phi}(g) &= \widetilde{\psi}(g) \circ \lambda_l && \text{(naturality of } \lambda\text{)} \\
\lambda_c \circ \widetilde{\phi}(g) \circ \eta_{(b,l)} &= \widetilde{\psi}(g) \circ \lambda_l \circ \eta_{(b,l)} \\
\lambda_c \circ \widetilde{\phi}(g) \circ \eta_{(b,l)} &= \widetilde{\psi}(g) \circ \rho_{(b,l)} && \text{(constraints on } \lambda\text{)}
\end{aligned} \tag{3.8}$$

Under these constraints we let $\lambda_c^{(b,l,g)}$ to be the unique group element that satisfies equation (3.8), *i.e.*, $\lambda_c^{(b,l,g)} := \widetilde{\psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\phi}(g)^{-1}$, for any $(b, l) \in \mathbf{Sub}$ and $g : l \to c$ in $\mathbf{Conf}$.

Let us show that the value of $\lambda_c^{(b,l,g)}$ does not depend on the choice of (b, l) and g. We denote by ε the empty configuration, *i.e.*, $|\varepsilon| = \emptyset$. First, note that for any $(b, l) \in \mathbf{Sub}$, there is a morphism $h_1 = 1 : (1, \varepsilon) \to (b, l) \in \mathbf{Sub}$ and a morphism $h_b = b : (1, \varepsilon) \to (b, l) \in \mathbf{Sub}$. Then for any $g : l \to c$ also observe that

$$\lambda_c^{(1,\varepsilon,g\circ h_1)} = \lambda_c^{(b,l,g)} = \lambda_c^{(1,\varepsilon,g\circ h_b)} \tag{3.9}$$

The first required equality is derived as follows:

$$\begin{aligned}
\lambda_c^{(1,\varepsilon,g\circ h_1)} &= \widetilde{\psi}(g \circ h_1) \cdot \rho_{(1,\varepsilon)} \cdot {\eta_{(1,\varepsilon)}}^{-1} \cdot \widetilde{\phi}(g \circ h_1)^{-1} \\
&= \widetilde{\psi}(g) \cdot \widetilde{\psi}(h_1) \cdot \rho_{(1,\varepsilon)} \cdot {\eta_{(1,\varepsilon)}}^{-1} \cdot \widetilde{\phi}(h_1)^{-1} \cdot \widetilde{\phi}(g)^{-1} \\
&= \widetilde{\psi}(g) \cdot \rho_{(b,l)} \cdot \underline{\delta}(h_1) \cdot \underline{\delta}(h_1)^{-1} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\phi}(g)^{-1} \\
&= \widetilde{\psi}(g) \cdot \rho_{(b,l)} \cdot {\eta_{(b,l)}}^{-1} \cdot \widetilde{\phi}(g)^{-1} \\
&= \lambda_c^{(b,l,g)}
\end{aligned}$$

where we use the naturality of η and ρ (definition 3.1.1 of natural transformations) and that $\widetilde{\psi}$ and $\widetilde{\phi}$ respect the composition (definition 2.1.2 of functors). The second equality is derived in the same manner.

Notice that the triples (b, ε, g) with $(b, \varepsilon) \in \mathbf{Sub}$ and $g : \varepsilon \to c$ are in bijection with $(b, g) \in B \times G$. It comes from the fact that for any $b \in B$, (b, ε) is a valid object of $\mathbf{Sub}$ and $g : \varepsilon \to c$ is always a morphism in $\mathbf{Conf}$.

It remains to show that, for any $b, b' \in \mathbf{Sub}$, $g, g' \in G$ we have $\lambda_c^{(b,\varepsilon,g)} = \lambda_c^{(b',\varepsilon,g')}$. Then we define a function $B \times G \to G$ mapping (b, g) to $\lambda_c^{(b,\varepsilon,g)}$. We show that all the $\lambda_c^{(b,\varepsilon,g)}$ are equal by showing that this function is constant. For this, we use lemma 3.3.15 which simply requires that for any $(b, g) \in B \times G$:

$$\lambda_c^{(1,\varepsilon,g\circ h_1)} = \lambda_c^{(b,\varepsilon,g)} = \lambda_c^{(1,\varepsilon,g\circ h_b)}$$

where $h_1 := 1 : (1, \varepsilon) \to (b, \varepsilon)$ and $h_b := b : (1, \varepsilon) \to (b, \varepsilon)$. This is simply a specific case of Equation (3.9) with $l = \varepsilon$.

This concludes the proof that the value of $\lambda_c^{(b,l,g)}$ does not depend on the choice of (b, l) and g. Consequently, we are able to define $\lambda_c := \lambda_c^{(b,l,g)}$. To finally get the component morphisms, it remains to show that $\lambda_c \in \mathrm{Hom}_{\mathbf{Conf}}(\widetilde{\phi}(c), \widetilde{\psi}(c))$, *i.e.*,, $\widetilde{\phi}(c) \preceq \widetilde{\psi}(c) \blacktriangleleft \lambda_c$. Take $g \in |\widetilde{\phi}(c)|$ and consider $(1, l_g) \in \mathbf{Sub}$ with $l_g := c_g \blacktriangleleft g$. We

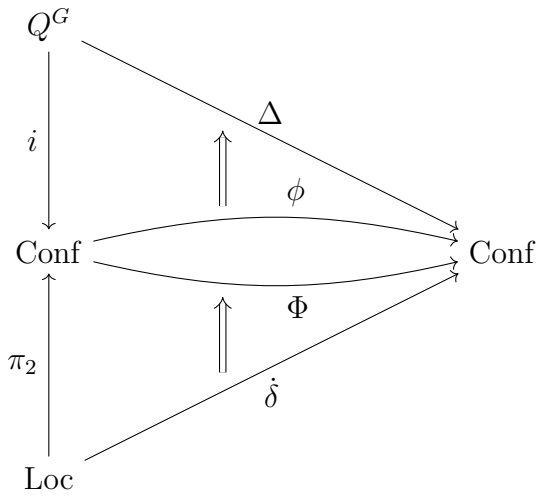


Figure 3.1: Bounds on poset extensions

obviously have $l_g \preceq c \blacktriangleleft g$, which means that we also have a morphism $g : l_g \to c$. By the existence of the morphism $\widetilde{\psi}(g) \circ \rho_{(1,l_g)} : \underline{\delta}(1, l_g) \to \widetilde{\psi}(l_g) \to \widetilde{\psi}(c)$, we have :

$$
\begin{aligned}
& \underline{\delta}(1, l_g) \preceq (\widetilde{\psi}(c) \blacktriangleleft \widetilde{\psi}(g) \cdot \rho_{(1,l_g)}) \\
\iff & \underline{\delta}(1, l_g) \preceq (\widetilde{\psi}(c) \blacktriangleleft \widetilde{\psi}(g) \cdot \rho_{(1,l_g)} \cdot g^{-1} \cdot g) \\
\iff & \underline{\delta}(1, l_g) \preceq (\widetilde{\psi}(c) \blacktriangleleft \widetilde{\psi}(g) \cdot \rho_{(1,l_g)} \cdot g^{-1}) \blacktriangleleft g && \text{(by prop. 3.2.4 of } \blacktriangleleft) \\
\iff & \underline{\delta}(1, l_g) \preceq (\widetilde{\psi}(c) \blacktriangleleft \widetilde{\psi}(g) \cdot \rho_{(1,l_g)} \cdot \eta_{(1,l_g)}{}^{-1} \cdot \widetilde{\phi}(g)^{-1}) \blacktriangleleft g && \text{(by def. 3.4.6, 3.4.2)} \\
\iff & \underline{\delta}(1, l_g) \preceq (\widetilde{\psi}(c) \blacktriangleleft \lambda_c^{(1,l_g,g)}) \blacktriangleleft g && \text{(by def. of } \lambda_c^{(1,l_g,g)}) \\
\iff & \underline{\delta}(1, l_g) \preceq (\widetilde{\psi}(c) \blacktriangleleft \lambda_c) \blacktriangleleft g && \text{(by def. of } \lambda_c) \\
\implies & g \in |\widetilde{\psi}(c) \blacktriangleleft \lambda_c| \wedge (\widetilde{\psi}(c) \blacktriangleleft \lambda_c)(g) = q_{l_g,1} && \text{(by def. 3.4.4)}
\end{aligned}
$$

reminding that as $g \in \det(c)$, then obviously $g \in \det(c_g)$ and $q_{c,g} = q_{c_g,g}$. By proposition 3.2.12 we have $1 \in \det(c_g \blacktriangleleft g)$, $1 \in |\underline{\delta}(1, l_g)|$ and $q_{c_g,g} = q_{c_g \blacktriangleleft g,1} = q_{l_g,1}$. But $\widetilde{\phi}(c)(g) = \phi(c)(g) = q_{c,g} = q_{l_g,1}$. We get $\widetilde{\phi}(c)(g) = (\widetilde{\psi}(c) \blacktriangleleft \lambda_c)(g)$ so λ_c is a well-defined morphism. The proofs that all these morphisms λ_l collectively form a natural transformation $\lambda : \widetilde{\phi} \Rightarrow \widetilde{\psi}$, and that the latter is the unique one respecting the required equality $\lambda_l \circ \eta_{(b,l)} = \rho_{(b,l)}$ are identical to the three last paragraphs of the proof of theorem 3.3.16. □

In contrast to the proof of theorem 3.3.16 where a global configuration $c \in Q^G$ was necessary to prove that the group elements $\lambda_c^{(b,c,g)}$ are equal for any (b, c) and g, it is not the case in this proof. This is a hint that this left Kan extension might be pointwise. Indeed, in this setting, we have that for any partial configuration p the comma category $\underline{\delta}$ is connected, at least through instances from the empty configuration ε.

3.5 Final Discussion

In this chapter, we observed that global transformations are captured by the abstract categorical notion of Kan extensions. This universal construction characterizes how

a functor can be an extension of a functor on a "smaller" domain. In particular, we have seen that the global transition functor of a global transformation is a particular case of a pointwise left Kan extension, where pointwise means it can be computed by the colimit formula given in the definition of global transformations.

At the end of chapter 2, we discussed that not requiring L to be fully faithful allows to design rule systems with finer semantics on the rules. We gave here a more formal statement: for pointwise left Kan extensions, the fact that L is fully faithful implies that the natural transformation of the extension is an isomorphism. In some sense, allowing any kind of natural transformation for the extensions does make the rules more expressive. Indeed, it specifies how the obtained functor relates to the rules, possibly doing more work than explicitly specified by rules and rule inclusions. Extending global transformations to rule systems where L is any functor would be an improvement in this sense. But as mentioned in section 3.3.4, it is always possible to construct a proper global transformation equivalent to some "non fully faithful global transformation". Consequently, we stick to the original definition from now on, and in-depth studies of the non fully faithful case are left for future work.

We also propose to revisit the relation local/global in cellular automata through the perspective of arbitrary partial configurations. The relation itself is shown to be expressed in terms of Kan extensions, extending the local behavior specified by the transition table to any partial configurations. This emphasizes the connection between cellular automata and global transformations, which we claimed to be a generalization of cellular automata.

Expressing cellular automata in terms of Kan extensions also permit to study other kind of Kan extensions, such as non-pointwise ones and right Kan extensions. We considered two ways of extending cellular automaton over partial configurations by filling the gap between the local transition function and the global transition function. The coarse transition function only uses the local results obtained from the local configurations included in a partial configuration. The fine transition function gather more information as it deduces the value for a cell in a partial configuration whenever every configuration containing this partial configuration agrees for the result for this cell. These two functions are then studied in section 3.3 and section 3.4 through the prism of two categorical contexts.

The first, encompasses the set of partial configuration with a partial order relation, that include a partial configuration in another only when they can be aligned using absolute positions. This construction shows that two particular extensions, the coarse and fine monotonic transition functions Φ and ϕ, correspond respectively to some left and right Kan extensions, resumed in the diagram of figure 3.1. There are additional simple structural facts to note about the monotonic functions considered. The first one is that the shift action on partial configurations, as given in definition 3.2.3, is the right Kan extension of the shift action on global configurations, as given in definition 3.2.2. Another one is that the coarse transition function Φ is always smaller than the fine transition function ϕ ($\Phi \Rightarrow \phi$), hence the names of these transition functions, coarse and fine. In fact, any transition function $f : \mathrm{Conf} \to \mathrm{Conf}$ such that $\overline{\delta} \Rightarrow f \circ \pi_2$ is necessarily such that $\Phi \Rightarrow f \Rightarrow \phi$. This means that any function respecting at least the local transition of the cellular automaton is in between the coarse and fine transition functions. These functions have a structure of poset with Φ and ϕ as extremal extensions. This shows, in some sense, the efficiency of the simple constraints of monotonicity and $\overline{\delta} \Rightarrow f \circ \pi_2$ for describing

extensions. In this first formal development, the single local behavior δ is explicitly “copied” on all $g \in G$ to obtain $\overline{\delta}$. It is readily possible to put a different behavior on each $g \in G$, with no real modification to the proofs. The statements are therefore valid for non-uniform cellular automata and automata networks. Notice that any constraint on the finiteness of the number of states is not mentioned, either.

A second construction is proposed to prevent the use of the highly redundant “fully shifted local transition function” $\overline{\delta}$. It consists in using the shift operation to relate configurations into a category of configurations instead of a poset of configurations. The former is very similar to the poset, except that the yes/no question “is this configuration a subconfiguration of this other one?” is replaced by the open-ended question “where does this configuration appear in this other one?” [Maignan and Spicher, 2015]. The constructions in the poset setting can then be revisited into this new setting. We then focused our study on lifting the case of the coarse and fine monotonic transition functions and get the coarse and fine transition functors. To explore the connection with global transformations, those two functors are characterized as left Kan extensions by lifting the work on posets.

We leave the characterization of a fine transition functor as a right Kan extension from the global transition function and the subsequent results for future works. Indeed, some important differences emerge and need to be taken carefully. They are essentially due to the degree of freedom gained by the possibility to relate previously not comparable configurations, thanks to the shift operation. For example, the reader might have noticed the use of $\mathbf{Conf}^\star$ instead of $\mathbf{Conf}$ which restricts the coarse transition functor to super local configurations only. Indeed, for configurations with an empty interior, the shift operation allows in some case many possibilities in positioning their results breaking the expected uniqueness property.

A main difference between the poset and the categorical constructions is that Kan extensions, as any categorical construction, are defined up to isomorphism, so that they do not allow to distinguish between isomorphic constructions. In our case, we have shown that, in some conditions, the coarse transition functors of two shifted automata are isomorphic. In other words, the given construction is not able to distinguish between some different cellular automata. Such limitation is not surprising. It is even expected in some way. The poset case is definitely the appropriate setting for describing cellular automata in their strict definitions, and the proposed result, with small modifications, is closely related to the Curtis-Hedlund theorem. Indeed, the “fully shifted local transition function” $\overline{\delta}$ is clearly an implementation of the required shift-equivariance condition, and if one restores the constraint of a finite set of states, one can see that the poset of finite support configurations is a “generating” part of the poset of open subsets of the product topology. In this case, the fine transition function ϕ can be viewed as encoding an important part of the topological behavior of the global transition function Δ [Ceccherini-Silberstein and Coornaert, 2010, Hedlund, 1969]. On the other hand, the transition to the categorical case has been thought to be compatible with generalizations of cellular automata to more arbitrary or dynamical spaces. In these cases, there is no convenient absolute positioning systems, and it is more natural to think in terms of occurrences of a configuration in another as we exactly did in $\mathbf{Conf}$. The direct consequence is the presence of isomorphisms making shifted configurations not distinguishable. This new setting goes out of the context of cellular automata strictly speaking, but is more suitable for working with cellular automata

modulo shift. From this perspective, the categorical construction is to be considered in the light of global transformations. The local/global relationship is described by a decomposition/recomposition process, which is efficiently presented as a Kan extension. The reader is invited to pay attention that the proofs given in the present work, especially for theorem 3.3.16, also follow the same decomposition/recomposition process involved in the local/global definition of cellular automata.

Notice that the coarse transition functor, is not obtained as a pointwise Kan extension, as, in the proof, it needs to use a connected super configuration to place the different connected components of the result of a non-connected partial configuration. But it can be made pointwise by restricting the extension to connected partial configurations only, *i.e.*, configurations that can be entirely described using objects and morphisms in **Loc**. Indeed, if we restricted this extension to the subcategory of **Conf** where all configurations are connected, then any resulting configuration would only be determined by its local results. As a consequence, there would be no need to consider global configurations in the proof of theorem 3.3.16. This questions the relevance of considering a notion of distance between two disconnected parts of a configuration when only relative positioning can be used in the computation. Indeed, similar structures could be represented using decorated graphs, and no notion of distance would make sense between unconnected parts. In contrast, the fine transition function is pointwise due to the nature of the local rules, allowing empty sub-local configuration. But the locality of the computation can be discussed, as, even when a partial configuration is finite, an infinite number of matchings from the empty sub-local configurations, representing the full space, occur in the computation.

Intuitively, pointwise left Kan extensions can be computed "algorithmically" using simple building blocks. This formulation in terms of building blocks is completely equivalent and is the one used in the other chapters, firstly because it is via these building blocks that we discovered these links between spatially-extended dynamical systems and category theory, and secondly because this formulation is closer to the software implementations of the considered models. In fact, it is possible to have an implementation completely generic over the considered particular kind of space. The following chapter focuses on the specification of such an algorithm for the considered large family of evolving graphs.

Chapter 4

Computation of Global Transformations

In chapter 2, the application of a global transformation on an input object were formalized using three operations: the *pattern-matching* step, that decomposes the input object in rule instances with respect to a rule system, the *local application* step, that transforms locally each rule instance independently, and finally the *reconstruction* step, that builds the global result from the local results using the inclusion's information.

This operational point of view already gives a general scheme for implementing a global transformation application algorithm for some fixed structure. However, it is unsatisfactory for practical purposes. Indeed, those three steps are somewhat global operations, as they require all information about the input object and do a synchronous computation on all of their inputs. This is an obstruction to produce a generic algorithm since one needs to provide for each kind of manipulated spaces, a global pattern matching and a global reconstruction algorithm. The generic part of the algorithm is reduced to the mere composition of the three steps of computations. In the algorithmic point of view, these operations are sub-specified as they only describe the desired output but not how to actually compute the output from the input. In practice, multiple implementations of these operations can be considered, and it is likely that they rely on simpler local operations.

In this work, we aim at providing a generic online algorithm using an *accretive* strategy: the output is computed by iteratively merging local results obtained by a pattern-matching procedure that follows a breadth-first-like traversal of the input. Moreover, we focus on computations such that, through this online algorithm, a new local result only accumulates into the actual intermediate result, the intermediate result only grows along the computation. On top of permitting online computations, we break down the global steps of computation into the simpler building blocks that are "pattern match one l.h.s. using inclusion information" and "merge two objects along a common part".

We investigate these ideas for rewriting many graph-like structures through global transformations. Those structures happen to be captured by the abstract notion of categories of presheaves. After establishing the particular properties of these categories, we focus on the computational aspects of rule systems based on such structures. To obtain an online algorithm, a complete study of the sub-steps within each synchronous step is done at the semantic level. To formalize accretive

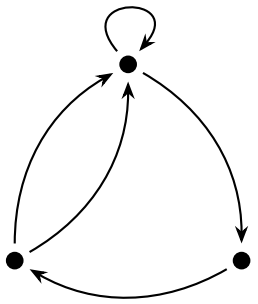


Figure 4.1: A directed multigraph or *graph*.

rule systems, we focus on the property of preserving a certain class of morphisms, that corresponds to the generalization of injective functions in arbitrary categories. Then we focus on the task of giving a criterion to characterize these systems at the local rule level. Finally, the online computation algorithm for these systems is given.

The chapter is organized as follows. section 4.1 first reminds general facts about presheaves in category theory. After adapting the definition of global transformations to presheaves in section 4.2, section 4.3 unfolds all implications of the online and accretive perspective at the semantic level, and gathers all necessary formal results. This leads to the presentation of the algorithm in section 4.4, followed by a discussion in section 4.5.

4.1 Background on Presheaves

Graphs have been well studied in category theory. In this work, we focus on an equivalence defining graphs as *contravariant* **Set**-*valued functors*, also called *presheaves*. More generally, categories of presheaves of some category **C** are interesting in the sense that they have a lot of common properties with the category **Set** of sets with functions between sets as morphisms. In this section, we first recall general knowledge about graphs in category theory. We then expose some of these properties, which gives us a general frame of work to define not only global transformations of graphs, but global transformations over any category of presheaves.

Graphs as Presheaves. We focus on directed multigraph, *i.e.*, graphs with directed edges, that allow self-loops and multiple edges between nodes. We will use the term *graph* to refer to this kind of graphs. Figure 4.1 depicts such a graph.

Definition 4.1.1. A *graph* G is a tuple $\langle V_G, E_G, s_G, t_G \rangle$ where V_G, E_G are sets containing respectively the *nodes* and the *edges* composing the G, and $s_G, t_G : E_G \to V_G$ are functions that assign the source and target nodes of each edge respectively.

Note that every item in the definition of a graph is an object or a morphism of the category **Set** of sets and functions. In other words, a graph can be viewed as data from **Set** attached to a category **G** having two objects and two morphisms arranged in the shape $\bullet \rightrightarrows \bullet$ in a functorial way. A graph is then a **Set**-valued functor from category **G**. By convention and without loss of generality, we express this idea in terms of *contravariant* functors, which are functors assigning morphisms of the domain to morphisms of the codomain in the reverse direction. This leads to the definition of the so-called presheaves.

Definition 4.1.2. A *presheaf* p over a category $\mathbf{C}$ is a **Set**-valued functor $p : \mathbf{C}^{op} \to \mathbf{Set}$ where $\mathbf{C}^{op}$ is the category having the same objects as $\mathbf{C}$ but as morphisms the reversed morphisms of $\mathbf{C}$, *i.e.*, a morphism $f : c \to d$ in $\mathbf{C}^{op}$ corresponds to a morphism $f : d \to c$ in $\mathbf{C}$. In other words, p sends any object $c \in \mathbf{C}$ to some set $p(c)$ and any morphism $f : d \to c \in \mathbf{C}$ to a morphism $p(f) : p(c) \to p(d)$.

The following properties could also be expressed in terms of classical (covariant) **Set**-valued functors. However, the use of presheaves here is useful for connecting with knowledge of category theory.

We now express the fact that presheaves can be used to encode the same structure as graphs. In particular, let us fix $\mathbf{G}$ to be the aforementioned category having the shape $\bullet \rightrightarrows \bullet$ in the following definition.

Definition 4.1.3. Let $\mathbf{G}$ be the category having two objects V and E and having only two morphisms $s, t : V \to E$ apart from identities.

As can be readily seen:

Proposition 4.1.1. *Presheaves over* $\mathbf{G}$ *correspond to graphs.*

Graph Morphisms as Natural Transformations. Our goal is to define global graph transformations. To do so, we need a category of graphs, and by extension an appropriate notion of a morphism $f : G \to H$ between two graphs. Let us take the classical notion of a homomorphism of graphs. Intuitively, such a morphism consists in two mappings, one mapping $f_V : V_G \to V_H$ from vertices to vertices and one mapping $f_E : E_G \to E_H$ from edges to edges, with the additional property that they respect the structure given by the sources and target functions. In particular, these mappings must send the source $s_G(e)$ and target $t_G(e)$ of an edge e in E_G to the source $s_H(e')$ and target $t_H(e')$ of the corresponding edge $e' = f_E(e)$ in E_H, *i.e.*, $f_V(s_G(e)) = s_H(f_E(e))$ and $f_V(t_G(e) = t_H(f_E(e))$ for any edge $e \in E_G$. Figure 4.2 depicts such a graph morphism which is defined as follows.

Definition 4.1.4. A *graph morphism* f from a graph G to a graph H denoted $f : G \to H$ is given by two mappings $f_V : V_G \to V_H$ and $f_E : E_G \to E_H$ such that $f_V \circ s_G = s_H \circ f_E$ and $f_V \circ t_G = t_H \circ f_E$.

$$\begin{array}{ccc} E_G & \xrightarrow{f_E} & E_H \\ \downarrow{\scriptstyle s_G} & & \downarrow{\scriptstyle s_H} \\ V_G & \xrightarrow{f_V} & V_H \end{array} \qquad \begin{array}{ccc} E_G & \xrightarrow{f_E} & E_H \\ \downarrow{\scriptstyle t_G} & & \downarrow{\scriptstyle t_H} \\ V_G & \xrightarrow{f_V} & V_H \end{array}$$

The category of presheaves over $\mathbf{C}$ is defined in the same manner. To do so, we need to use the usual definition of an "morphism of functors" *i.e.*, a natural transformation. Refining definition 3.1.1 of natural transformations to take account of the contravariance of presheaves leads to the following definition.

Definition 4.1.5. A natural transformation $\eta : p \Rightarrow q$ between two presheaves $p : \mathbf{C}^{op} \to \mathbf{Set}$ to $q : \mathbf{C}^{op} \to \mathbf{Set}$ is given by a function $\eta_c : p(c) \to q(c)$ for each $c \in \mathbf{C}$ such that for any morphism $f : c \to d \in \mathbf{C}$ we have $\eta_c \circ p(f) = q(f) \circ \eta_d$. For any $c \in \mathbf{C}$, the function η_c is called the *component* of η for c.

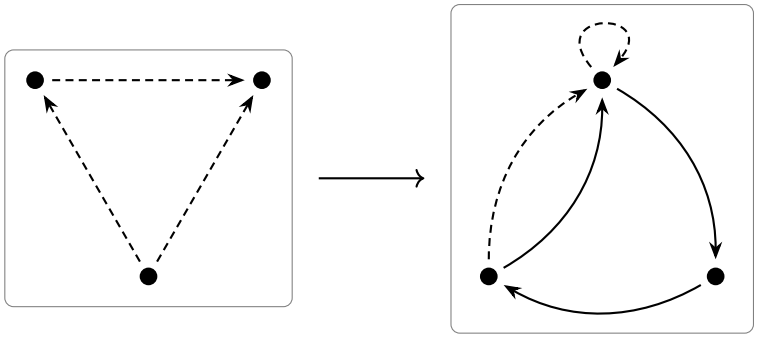


Figure 4.2: A *graph* (homo)morphism

$$\begin{array}{ccc} p(d) & \xrightarrow{\eta_d} & q(d) \\ \downarrow{\scriptstyle p(f)} & & \downarrow{\scriptstyle q(f)} \\ p(c) & \xrightarrow{\eta_c} & q(c) \end{array}$$

As can be readily seen:

Proposition 4.1.2. *Natural transformations between presheaves over* $\mathbf{G}$ *correspond to graph morphisms.*

Graphs and graph morphisms are endowed with the structure of category by simply using the composition of functions.

Definition 4.1.6. The category of graphs **Graph** is the category of graphs and graph morphisms. For any graph G the identity graph morphism id_G is given by the set identities $id_{GV} = id_{V_G}$ and $id_{GE} = id_{E_G}$. Given three graphs G, H, I and two graph morphism $f : G \to H$ and $g : H \to I$, the composite graph morphism $g \circ f : G \to I$ is given by the composites $(g \circ f)_V = g_V \circ f_V$ and $(g \circ f)_E = g_E \circ f_E$.

Categories of presheaves are defined in the same manner.

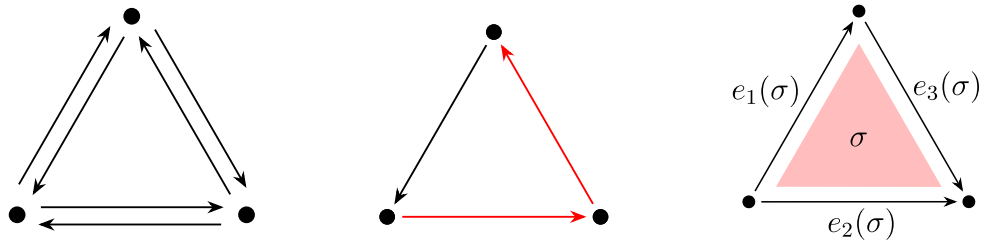
Definition 4.1.7. The category $\hat{\mathbf{C}}$ of presheaves over $\mathbf{C}$ is the category having for objects presheaves over $\mathbf{C}$ and for morphisms natural transformations between theses presheaves. For any presheaf p the identity natural transformation id_p is given by the components $id_{p_c} = id_{p(c)}$ for any $c \in \mathbf{C}$. Natural transformations are composed using vertical composition 3.1.2.

Proposition 4.1.3. *The category* $\hat{\mathbf{G}}$ *and* **Graph** *are isomorphic.*

Proof. We have already seen in proposition 4.1.1 and proposition 4.1.2, that objects and morphisms of $\hat{\mathbf{G}}$ and **Graph** are equivalent. Then one needs only to ensure that the composition and identities of $\hat{\mathbf{G}}$ and **Graph** do correspond, which is obtained directly by unfolding the definitions. □

By this fact, we consider categories of presheaves over an arbitrary category $\mathbf{C}$ as some kind of "generalized graphs". Here we give three other examples of categories of presheaves that can be seen as some kind of graphs.

Example 4.1.1. Consider the category $\mathbf{U}$ having two objects V and E and having morphisms $s, t : V \to E$ and an involution $f : E \to E$ such that $f \circ s = t$ and $f \circ t = s$ apart from identities. Apart from some technicalities on loops, the presheaf category



(a) An undirected graph (b) A bi-colored graph (c) A Δ-set with one face σ

Figure 4.3: Example objects of categories of presheaves

$\hat{\mathbf{U}}$ corresponds to the category of undirected multi-graphs (figure 4.3a). Indeed, for any presheaf $p \in \hat{\mathbf{U}}$, the function $p(f) : p(E) \to p(E)$ sends an edge $e \in p(E)$ to its reversed edge e' such that $p(s)(e') = p(s)(p(f)(e)) = p(f \circ s)(e) = p(t)(e)$ and $p(t)(e') = p(s)(e)$, the fact that $p(f)$ must be an involution pairs edges and their reversals together, guarantying that no edge is the reversal of multiple edges.

Example 4.1.2. Consider the category $\mathbf{E}_2$ having three objects V, E_1 and E_2 and having morphisms $s_1, t_1 : V \to E_1$ and $s_2, t_2 : V \to E_2$ apart from identities. The presheaf category $\hat{\mathbf{E}}_2$ corresponds to the category of graphs having edges of two colors that morphisms must respect (figure 4.3b).

Example 4.1.3. Consider the category $\mathbf{T}$ having three objects V, E and T and having morphisms $s, t : V \to E$, $e_1, e_2, e_3 : E \to T$ and $r : T \to T$ such that $e_3 \circ s = e_1 \circ t$, $e_3 \circ t = e_2 \circ t$ and $e_2 \circ s = e_1 \circ s$ apart from identities. The presheaf category $\hat{\mathbf{T}}$ corresponds to the category of two-dimensional semi-simplicial sets or Δ-sets. Objects in this category are triangular meshes with oriented edges and faces (figure 4.3c). However, this is still an adapted category to formalize triangular mesh refinement global transformation presented in the introduction.

Representable Presheaves. Any category of presheaves is known to form a topos (example A.2.1.3 in [Johnstone, 2002]), a category that shares a lot of properties with **Set**. To help us to work with these categories, we need to observe some special properties of some presheaves called *representable presheaves*.

Let us exemplify this construction on graphs first. The *single vertex graph* 1_V, the *single edge graph* 1_E and the *source* and *target* graph morphisms $m_s, m_t : 1_V \to 1_E$ (figure 4.4) are special graphs in the category **Graph** in the sense that they share some properties with the singleton sets in **Set**. In particular, they are enough to describe any object and morphisms of **Graph**, in the same manner that any set s is characterized by its elements which can be identified by the set of morphisms $e : \{\star\} \to s$ from some singleton set $\{\star\}$. In the same fashion, for any arbitrary graph G, the morphisms $v_i : 1_V \to G$ and $e_j : 1_E \to G$ from those elementary graphs can be used to probe for edges and vertices of G. Combined by the commutations $e_j \circ m_s = v_i$ or $e_j \circ m_t = v_i$, they are sufficient to describe the structure of G.

Notice the similarity between figure 4.4 and the category $\mathbf{G}$. Moreover, notice how the graphs 1_V and 1_E are related to the morphisms of $\mathbf{G}$. Particularly, for the single edge graph 1_E, the object E is the codomain of three morphisms: id_E corresponding to the sole edge, and $s, t : V \to E$ corresponding respectively to the source and target of that edge. There is no coincidence. It shows how the category $\mathbf{G}$ describes how the objects (and morphisms) of **Graph** $\cong \hat{G}$ are built from elementary

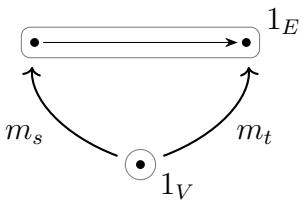


Figure 4.4: Single vertex and single edge graphs with source and edges morphisms

pieces. This also holds for any category of presheaves $\hat{\mathbf{C}}$ where any presheaf is canonically described by special objects and morphisms of $\hat{\mathbf{C}}$ corresponding to the notions of *representable* presheaves and *representable* natural transformations.

Definition 4.1.8. The *Yoneda embedding* for a category $\mathbf{C}$ is the functor $\mathbf{y} : \mathbf{C} \to \hat{\mathbf{C}}$ mapping each $c \in C$ to the hom-functor $\mathrm{Hom}_{\mathbf{C}}(-, c)$ and each morphism $f : c \to d \in C$ the natural transformation $f \circ - : \mathrm{Hom}_{\mathbf{C}}(-, c) \Rightarrow \mathrm{Hom}_{\mathbf{C}}(-, d)$. The presheaves that are isomorphic to $\mathbf{y}c$ for any $c \in \mathbf{C}$ are called *representable presheaves*. In the same manner, natural transformations isomorphic to images of morphisms of $\mathbf{C}$ through $\mathbf{y}$ are called *representable natural transformations*.

In the case of graphs, the representable presheaves are isomorphic to $\mathbf{y}V$ or $\mathbf{y}E$. Presheaves $\mathbf{y}V$ and $\mathbf{y}E$ correspond to the single vertex graph 1_V and the single edge graph 1_E respectively. Indeed, $\mathbf{y}V$ seen as a graph is given by $V_{\mathbf{y}V} = \{id_V\}$, $E_{\mathbf{y}V} = \emptyset$ and $s_{\mathbf{y}V}$ and $t_{\mathbf{y}V}$ being empty maps, and $\mathbf{y}E$ seen as a graph is given by $V_{\mathbf{y}E} = \{s, t\}$, $E_{\mathbf{y}E} = \{id_E\}$, $s_{\mathbf{y}E} = [id_E \mapsto s]$ and $t_{\mathbf{y}E} = [id_E \mapsto t]$. Apart from identities, the representable natural transformations in $\hat{\mathbf{G}}$ are isomorphic to $\mathbf{y}s : \mathbf{y}V \to \mathbf{y}E$ or $\mathbf{y}t : \mathbf{y}V \to \mathbf{y}E$, and they correspond to the source morphism and the target morphism. Indeed, $\mathbf{y}s$, seen as graph morphism is given by $\mathbf{y}s_V = [id_V \mapsto s]$ and $\mathbf{y}s_E$ being the empty mapping, and $\mathbf{y}t$ seen as a graph morphism is given by $\mathbf{y}t_V = [id_V \mapsto t]$ and $\mathbf{y}t_E$ being the empty mapping.

As previously said, graph morphisms from 1_V and from 1_E and can be used to probe for the vertices and edges of any graph. In other words given any graph G, we have $\mathrm{Hom}_{\mathbf{Graph}}(1_V, G) \cong V_G$ and $\mathrm{Hom}_{\mathbf{Graph}}(1_E, G) \cong E_G$. This is expressed on any category of presheaves $\hat{\mathbf{C}}$ by the well known Yoneda lemma.

Proposition 4.1.4 (Yoneda lemma)**.** *For any presheaf $p : \mathbf{C}^{op} \to \mathbf{Set}$ there is a canonical isomorphism*

$$\mathrm{Hom}_{\hat{\mathbf{C}}}(\mathbf{y}c, p) \cong p(c),$$

for any $c \in \mathbf{C}$.

Applied for graphs, for any presheaf p of $\hat{\mathbf{G}}$, so any graph, there is a canonical isomorphism $\mathrm{Hom}_{\hat{\mathbf{G}}}(\mathbf{y}V, p) \cong p(V)$, and a canonical isomorphism $\mathrm{Hom}_{\hat{\mathbf{G}}}(\mathbf{y}E, p) \cong p(E)$.

Remark 4.1.1. Given some $e : \mathbf{y}c \to p$ we can also denote by e the corresponding element $e \in p(c)$. Given some natural transformations $f : p \to q$ and $e : \mathbf{y}c \to p$, as e corresponds to an element e in $p(c)$, the composite $f \circ e : \mathbf{y}c \to q$ corresponds to $f_c(e) \in q(c)$ where $f_c : p(c) \to q(c)$ is the component of f on c.

Colimits in Categories of Presheaves. An important fact about the category of presheaves $\hat{\mathbf{C}}$ is that they are cocomplete, *i.e.*, any diagram in $\hat{\mathbf{C}}$ has a colimit.

Proposition 4.1.5. *For any category* $\mathbf{C}$*, the category* $\hat{\mathbf{C}}$ *is cocomplete, i.e., any (small) diagram* $F : \mathbf{I} \to \hat{\mathbf{C}}$ *has a colimit.*

Proof. See Corollary 4 of [MacLane and Moerdijk, 2012]. □

In fact any presheaf can be described by a colimit of a diagram of representable presheaves and representable natural transformations. Natural transformations between theses presheaves correspond to mediating morphisms between those colimits.

Proposition 4.1.6. *Any presheaf* p *in* $\hat{\mathbf{C}}$ *is a colimit of a diagram of representable presheaves and representable natural transformations.*

Proof. See Theorem 2.15.6 in [Borceux, 1994] (or Corollary 3 / Proposition 1 of [MacLane and Moerdijk, 2012]). □

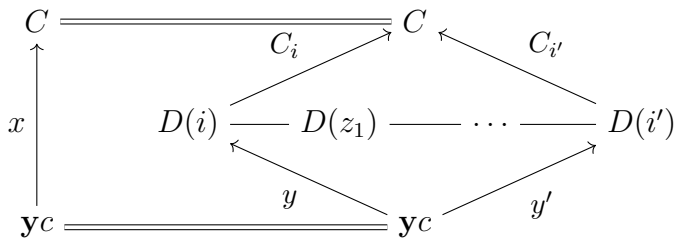
There is a way to explicitly compute colimits in $\hat{\mathbf{C}}$. This comes from the observation that colimits in a category of presheaves can be computed only using colimits in **Set**.

Proposition 4.1.7. *Colimits in* $\hat{\mathbf{C}}$ *can be computed pointwise. Given any* $F : \mathbf{I} \to \hat{\mathbf{C}}$*, and the "evaluate at* c*" functor* $E_c : \hat{\mathbf{C}} \to \mathbf{Set}$ *given by* $E_c(p) = p(c)$ *and* $E_c(\eta : p \to q) = \eta_c$*, we have* $\mathrm{Colim}(F)(c) = \mathrm{Colim}(E_c \circ F)$ *for any* $c \in \mathbf{C}$*. Moreover, for any* $i \in \mathbf{I}$ *the corresponding cocone component* $\mathrm{Colim}(E_c \circ F)_i$ *is equal to the component* $\mathrm{Colim}(F)_{ic}$*.*

Proof. The proof can be found for the case of limits in Functor categories in Proposition 2.15.1 of [Borceux, 1994] and in Theorem 1 of section V.3 of [MacLane, 2013]. The latter proof can be adapted to colimits of categories of presheaves by reversing the morphisms and acknowledging that **Set** is cocomplete (dual of Theorem 1 in section V.1 of [MacLane, 2013]). □

We now express an explicit construction for colimits in $\hat{\mathbf{C}}$. The usual description of colimits in $\hat{\mathbf{C}}$, which is based on equivalence classes of vertices and edges, can be rephrased in terms of *zigzag*. A *zigzag* z in a category $\mathbf{X}$ is given by some natural number $|z|$, a sequence $\langle z_i \rangle_{0 \leq i \leq |z|}$ of $|z|+1$ objects of $\mathbf{X}$ and a sequence $\langle \overline{z}_i \rangle_{0 \leq i \leq |z|-1}$ of morphisms in $\mathbf{X}$ of the form $z_0 \to z_1 \leftarrow z_2 \to z_3 \cdots z_{|z|}$ or $z_0 \leftarrow z_1 \to z_2 \leftarrow z_3 \cdots z_{|z|}$. Given a functor $F : \mathbf{X} \to \mathbf{Y}$ and two morphisms $g : c \to F(z_0)$ and $g' : c \to F(z_{|z|})$ in $\mathbf{Y}$, we write $F(z)$ for the zigzag defined with $|F(z)| = |z|$, $F(z)_i = F(z_i)$, and $\overline{F(z)}_i = F(\overline{z}_i)$. Given two morphisms $f_0 : c \to z_0$ and $f_{|z|} : c \to z_{|z|}$ in $\mathbf{Y}$, z is said to *link* f_0 *and* $f_{|z|}$ if there is a sequence $\langle f_i : c \to z_i \rangle_{1 \leq i \leq |z|-1}$ of morphisms such that, for any $i \in \{0, \ldots, |z|-1\}$, $f_i = \overline{z}_i \circ f_{i+1}$ or $\overline{z}_i \circ f_i = f_{i+1}$ depending on the direction of $\overline{z}_i$. We say that z *links* g *and* g' *through* F if the zig-zag $F(z)$ links g and g'.

Proposition 4.1.8. *For any diagram* $D : \mathbf{I} \to \hat{\mathbf{C}}$ *of (small) domain* $\mathbf{I}$ *with* $C = \mathrm{Colim}(D)$*, any* $c \in \mathbf{C}$*, and any* $x : \mathbf{y}c \to C$*, there exists is at least one pair of* $\langle i \in \mathbf{I}, y : \mathbf{y}c \to D(i) \rangle$ *such that* $x = C_i \circ y$*. Moreover, for any two such pairs* $\langle i, y \rangle$ *and* $\langle i', y' \rangle$ *have a zigzag* z *in* $\mathbf{I}$ *that links* y *and* y' *through* D*.*



Proof. Fix some $c \in \mathbf{C}$. Any $x : \mathbf{y}c \to C$ corresponds to an element $x \in C(c)$ by the Yoneda lemma 4.1.4, so we can apply proposition 4.1.7 that gives that $x \in \mathrm{Colim}(F)$ where $F = E_c \circ D$ where $E_c : \hat{\mathbf{C}} \to \mathbf{Set}$ is the functor given by $E_c(p) = p(c)$ and $E_c(\eta : p \to q) = \eta_c$.

By the existence Theorem of colimits (dual of theorem 1 in section V.2 of [MacLane, 2013]) and examples 2.2.4.a and 2.4.6.b of [Borceux, 1994] there is an explicit characterization of the colimit of such a **Set**-valued functor. It is a quotient set over the disjoint union $\coprod_{i \in \mathbf{I}} F(i)$:

$$\mathrm{Colim}(F) \cong \left(\coprod_{i \in \mathbf{I}} F(i) \right) /_{\sim},$$

where the equivalence relation $\sim$ is generated by

$$(y \in F(i)) \sim (y' \in F(i')) \quad \text{if} \quad \exists f : i \to i' \text{ with } F(f)(y) = y'.$$

The cocone components $\mathrm{Colim}(F)_i : F(i) \to \mathrm{Colim}(F)$ are the obvious injections: for any $x \in \mathrm{Colim}(F)$ and each pair $i \in \mathbf{I}$ and $y \in F(i)$ corresponding to x, we have $\mathrm{Colim}(F)_i(y) = x$.

Now, fix some $x \in \mathrm{Colim}(F)$. Take any such pair $\langle i, y \rangle$, note that $y \in F(i)$ means $y \in (D \circ E_c)(i)$, *i.e.*, $y \in D(i)(c)$. As $D(i)$ is a presheaf, the Yoneda lemma gives us a pair $\langle i \in I, y : \mathbf{y}c \to D(i) \rangle$. It rests to show that $C_i \circ y = x$. Proposition 4.1.7 gives $\mathrm{Colim}(F)_i = C_{ic}$ so we have the equality $C_{ic}(y) = x$, which gives the desired equality by the Yoneda correspondence (remark 4.1.1).

Moreover, given any other such pair $\langle i \in I, y : \mathbf{y}c \to D(i) \rangle$ corresponding to x, the equivalence relation $\sim$ tells us that must exists $i_0, \ldots, i_k \in \mathbf{I}$ with $i_0 = i$ and $i_k = i'$, elements $y_j \in F(i_j)$ for $0 \leq j \leq k$ with $y_0 = y$ and $y_k = y'$, and arrows $f_j : i_j \to i_{j+1}$ (or $f_j : i_{j+1} \to i_j$) such that $F(f_j)(y_j) = y_{j+1}$ (or $F(f_j)(y_{j+1}) = y_j$) for $0 \leq j < k$. As before, applying the Yoneda lemma on elements $y_j \in F(i_j)$ gives us morphisms $y_j : \mathbf{y}c \to D(i_j)$, and the equalities $F(f_j)(y_j) = y_{j+1}$ (or $F(f_j)(y_{j+1})) = y_j$) becomes $D(f_j) \circ y_{j+1} = y_j$ (or $D(f_j) \circ y_{j+1} = y_j$). Then if we close the sequence $f_0, \ldots, f_{k-1}$ using composition we get precisely a zigzag in $\mathbf{I}$ that links y and y' through D. □

Injective Presheaf Morphisms. To keep up with the intuition of the other chapters, we focus on categories of presheaves whose morphisms express an "inclusion" of presheaves. These particular natural transformations are cases of *monomorphisms*.

Monomorphisms can be though as a generalization of injective functions to arbitrary categories. In the particular case of a category of presheaves $\hat{\mathbf{C}}$, we will see that they correspond to such natural transformations $\eta : p \to q$ whose components

functions η_c are injective for any $c \in \mathbf{C}$. An injective function is a function $f : x \to y$ such that whenever some pair of elements e_1, e_2 in x are sent to the same element $f(e_1) = f(e_2)$ in y, $e_1 = e_2$. The general definition of monomorphism the follows the same pattern.

Definition 4.1.9. A morphism $f : x \to y$ in a category $\mathbf{C}$ is a *monomorphism* if for any two other morphisms $g_1, g_2 : z \to x$ we have $f \circ g_1 = f \circ g_2 \implies g_1 = g_2$.

Remark 4.1.2. Monomorphisms are closed under composition. Indeed, given two monomorphisms $f : x \to y$ and $h : y \to z$, and any two morphisms $g_1, g_2 : z \to x$ such that $h \circ f \circ g_1 = h \circ f \circ g_2$ we have that $f \circ g_1 = f \circ g_2$ as h is a monomorphism and then $g_1 = g_2$ as f is a monomorphism.

The definition of monomorphisms uses morphisms g_1, g_2 into x in place of elements e_1, e_2. But in the case of a presheaves, representable presheaves can be used to probe for the elements constituting an arbitrary presheaf, and the property of being a monomorphism can be specified in terms of elements as for injective functions.

Proposition 4.1.9. *A morphism $f : p \to q$ in some category of presheaves $\hat{\mathbf{C}}$ is a monomorphism* iff. *for any two morphisms $e_1, e_2 : \mathbf{y}c \to p$ with $c \in \mathbf{C} \implies f \circ e_1 = f \circ e_2$, then $e_1 = e_2$.*

Proof. ($\Rightarrow$) The first direction is simply a consequence of definition 4.1.9 of monomorphisms.

($\Leftarrow$) Example 4.5.17.b in [Borceux, 1994] (due to proposition 4.1.6 and definition 4.5.1 in [Borceux, 1994]) states that morphisms $f, g : p \to q$ in $\hat{\mathbf{C}}$ can be distinguished elementwise, that is:

$$\forall f, g : p \to q \in \hat{\mathbf{C}}, [\forall c \in \mathbf{C}, \forall e : \mathbf{y}c \to p, f \circ e = g \circ e] \implies f = g \tag{4.1}$$

Now suppose that for any $c \in \mathbf{C}$ and any two morphisms $e_1, e_2 : \mathbf{y}c \to p$ we have $f \circ e_1 = f \circ e_2 \implies e_1 = e_2$. So consider any $g_1, g_2 : r \to p$ such that $f \circ g_1 = f \circ g_2$, we must then show that $g_1 = g_2$. Then, for any $e : \mathbf{y}c \to r$ with $c \in \mathbf{C}$ we have $f \circ g_1 \circ e = f \circ g_2 \circ e$. As $g_1 \circ e$ and $g_2 \circ e$ are morphisms from $\mathbf{y}c$ to p, the hypothesis gives that $g_1 \circ e = g_2 \circ e$. Then $g_1 = g_2$ follows from equation (4.1). □

Combining proposition 4.1.9 and the Yoneda lemma 4.1.4 we get that monomorphisms in $\hat{\mathbf{C}}$ are precisely the morphisms whose components are injective.

Category of Presheaves and Locality. To summary, we have seen that the graphs are a special case of presheaves and that a category of presheaves $\hat{\mathbf{C}}$ can be seen representing a particular class of (generalizations of) graphs made from elementary bricks described in a presentation category $\mathbf{C}$ Moreover, focusing on monomorphisms of presheaves, *i.e.*, on inclusions between such objects, provides us with a description of a natural structure for the locality of these objects, a prerequisite of global transformations as presented in chapter 2.

In this chapter, we illustrate our proposition of algorithm in this context. In the following, we consider an arbitrary category $\mathbf{C}$, its category of presheaves $\hat{\mathbf{C}}$, and $\hat{\mathbf{C}}_{\mathcal{M}}$ which is the subcategory of $\hat{\mathbf{C}}$ consisting of all objects of $\hat{\mathbf{C}}$ and every monomorphism between them. It is indeed a category, as monomorphisms are closed under the composition (remark 4.1.2). Morphisms of $\hat{\mathbf{C}}$ and monomorphisms of $\hat{\mathbf{C}}_{\mathcal{M}}$

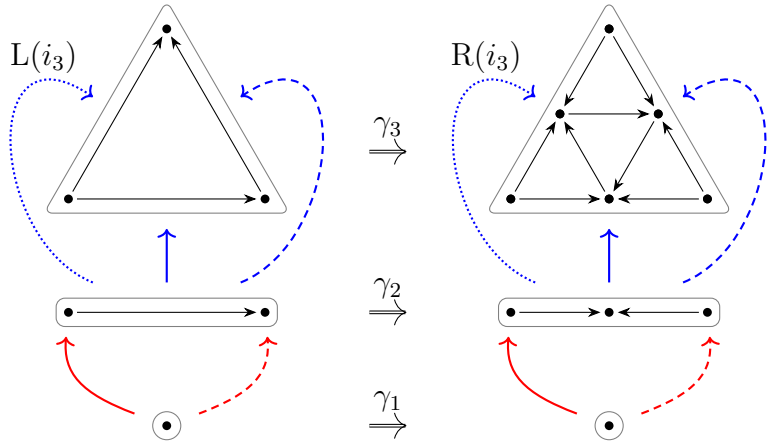


Figure 4.5: Sierpinski rule system: one rule to divide the relevant triangles, the two others and their inclusions to specify the connections in the output based on the connections in the input.

are written $p \hookrightarrow p'$. The examples are spelled out using $\mathbf{C} = \mathbf{G}$ the category that induces graphs in the last section as the category of presheaves $\hat{\mathbf{G}}$. We will make use of the following particular graphs: d_k the discrete graph with k vertices and no edges, p_k the path of length k, and c_k the cycle of length k, $k > 0$.

4.2 Global Transformations for Presheaves

In this chapter, we restrict ourselves to global transformations acting on $\hat{\mathbf{C}}_{\mathcal{M}}$. At a basic level, they are rewriting systems transforming presheaves into presheaves and sends monomorphisms to monomorphisms. As defined in chapter 2, a global transformation is locally presented by a rule system T consisting of a category $\mathbf{\Gamma}_T$ a full and faithful l.h.s. functor $\mathrm{L}_T : \mathbf{\Gamma}_T \to \hat{\mathbf{C}}_{\mathcal{M}}$ and a r.h.s. functor $\mathrm{R}_T : \mathbf{\Gamma}_T \to \hat{\mathbf{C}}_{\mathcal{M}}$.

The generation of a Sierpinski gasket is an example of the kind of global transformations that we expect to be able to express in this setting. Figure 4.5 depicts the corresponding rules and rule inclusions. The rule system is composed of 3 rules transforming locally vertices (γ_1), edges (γ_2) and acyclic triangles (γ_3). These rules are related by 5 main rule inclusions: $i_1 : \gamma_1 \to \gamma_2$ (plain red), $i_2 : \gamma_1 \to \gamma_2$ (dashed red), $i_3 : \gamma_2 \to \gamma_3$ (dotted blue), $i_4 : \gamma_2 \to \gamma_3$ (plain blue), $i_5 : \gamma_2 \to \gamma_3$ (dashed blue). For instance, consider the inclusion, i_3 which expresses that the left edge of the triangle $\mathrm{L}(\gamma_3)$ is transformed into the left double-edge of $\mathrm{R}(\gamma_3)$. Formally, this is specified via the inclusion i_3 whose both components $\mathrm{L}(i_3)$ and $\mathrm{R}(i_3)$ are depicted in dotted blue arrows. The reader is invited to pay attention that even if figure 4.5 does not show them, the category $\mathbf{\Gamma}$ also contains compositions of the 5 main rule inclusions (*e.g.*, $i_3 \circ i_1$), identities and symmetries, that, as functors, L and R do respect. The three steps for applying the rule system T on some input are illustrated figure 4.6. The input is depicted at the top left and the output at the top right.

However, working with monomorphisms may lead to specific issues at the reconstruction step since colimits do not behave well in this setting. Indeed, by restricting the category to $\hat{\mathbf{C}}$, some morphisms are now missing to characterize the desired object as a colimit. For instance, consider a diagram $F : \mathbf{I} \to \hat{\mathbf{G}}_{\mathcal{M}}$, where

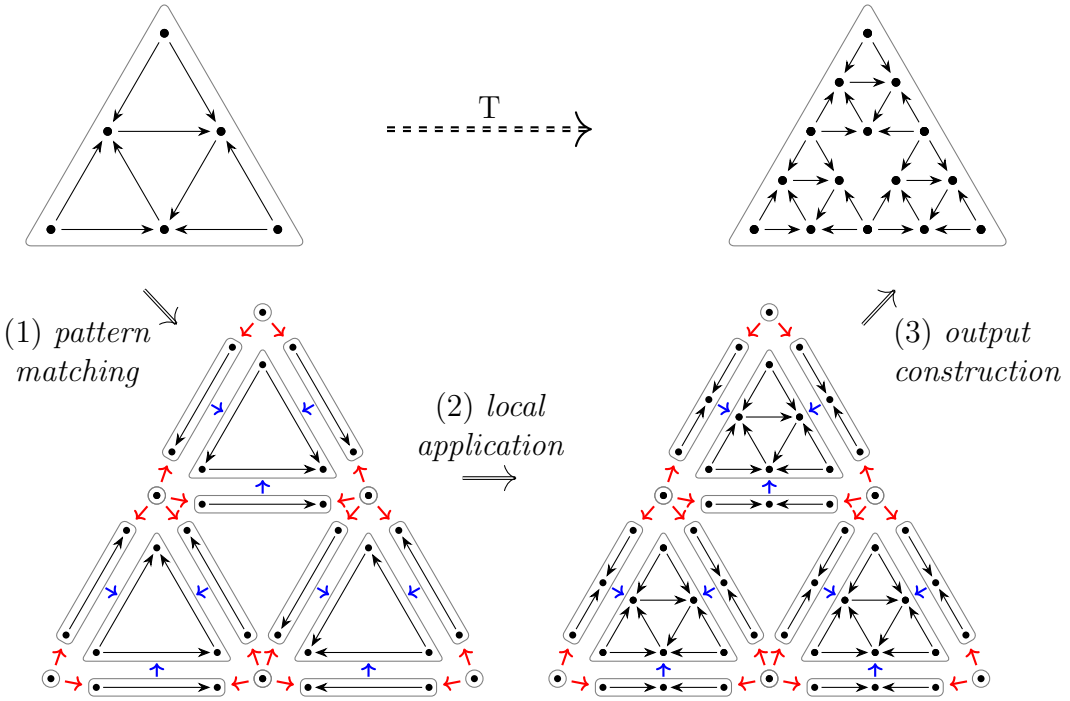


Figure 4.6: Step of computation of the Sierpinski gasket using the rules of figure 4.5.

$\mathbf{I}$ is the discrete category with only two objects 0 and 1 and their identities, and F be the constant functor mapping each object to the single vertex graph d_1, *i.e.*, $F(0) = F(1) = d_1$. One expects the desired colimit of such a diagram to be the graph d_2 with only two vertices and no edge. But consider the other cocone over F with apex d_1 and identities as components. But there is no mediating morphism $m : d_2 \to d_1$ in $\hat{\mathbf{G}}_{\mathcal{M}}$, as it would be non-injective on vertices, which shows that d_2 is not the colimit. In fact, the diagram F does not have any colimits in $\hat{\mathbf{G}}_{\mathcal{M}}$. But we can also remark that the colimit exists in $\hat{\mathbf{G}}$ and is exactly the expected one.

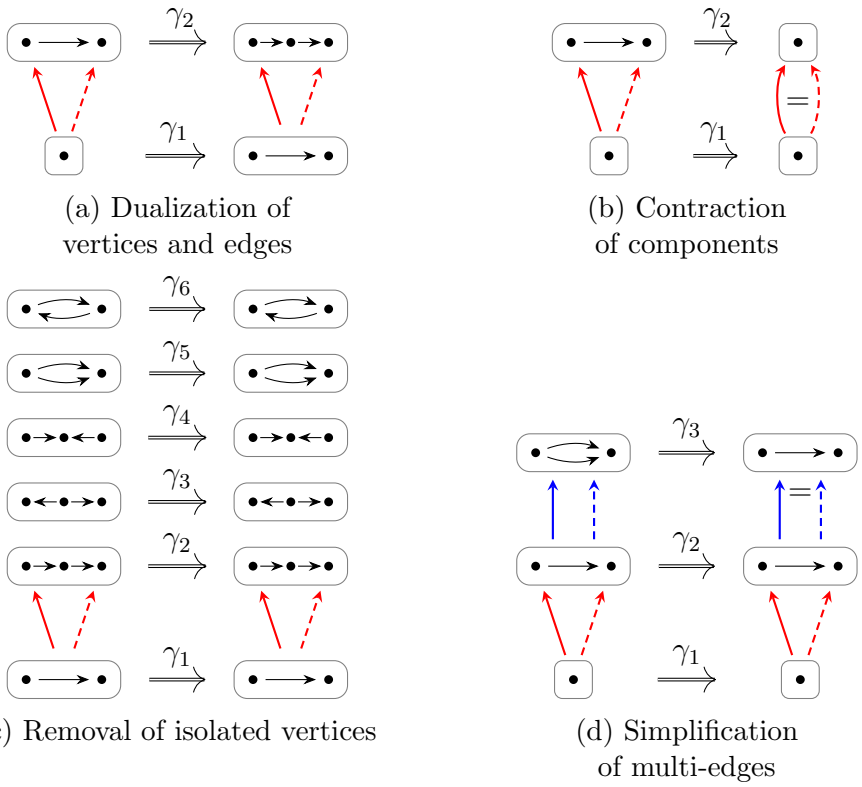
To circumvent this issue, we extend the definition of global transformations on a category of presheaves $\hat{\mathbf{C}}_{\mathcal{M}}$ to permit the colimit to be computed in the category $\hat{\mathbf{C}}$ with any type of morphisms between them. We then focus on those rule systems where the results stay inside $\hat{\mathbf{C}}_{\mathcal{M}}$, *i.e.*, such that all the mediating morphisms required by the global transformation are monomorphisms. This leads to the following definition of global transformation for $\hat{\mathbf{C}}_{\mathcal{M}}$.

Definition 4.2.1. Given a rule system T over $\hat{\mathbf{C}}_{\mathcal{M}}$, we consider the following functor $\overline{T} : \hat{\mathbf{C}}_{\mathcal{M}} \to \hat{\mathbf{C}}$:

$$\overline{T}(-) = \mathrm{Colim}(\mathrm{D}_T(-)) \qquad \text{with} \qquad \mathrm{D}_T(-) = \mathrm{U} \circ \mathrm{R}_T \circ \mathrm{Proj}[\mathrm{L}_T/-] \tag{4.2}$$

using the forgetful functor $\mathrm{U} : \hat{\mathbf{C}}_{\mathcal{M}} \to \hat{\mathbf{C}}$. A *global transformation* T is a rule system such that $\overline{T}$ factors through the forgetful functor $\mathrm{U} : \hat{\mathbf{C}}_{\mathcal{M}} \to \hat{\mathbf{C}}$. In this case, we denote $T : \hat{\mathbf{C}}_{\mathcal{M}} \to \hat{\mathbf{C}}_{\mathcal{M}}$ the functor such that $\mathrm{U} \circ T = \overline{T}$.

Remark 4.2.1. Echoing section 2.1.6, notice that $\overline{T}$ is a complete functor also acting on morphisms. Consider a monomorphism $h : p \hookrightarrow p'$. By definition of colimits, $\overline{T}(p)$ is the universal cocone with components $\overline{T}(p)_{\langle\gamma,f\rangle} : \mathrm{D}_T(p)(\langle\gamma, f\rangle) \to \overline{T}(p)$ for each instance $\langle\gamma, f\rangle \in \mathrm{L}_T/p$. We have a similar construction for $\overline{T}(p')$ which gives



(a) Dualization of vertices and edges

(b) Contraction of components

(c) Removal of isolated vertices

(d) Simplification of multi-edges

Figure 4.7: Some rule-systems. Note that all of them remove self-loops.

rise to a cocone C as the restriction of $\overline{T}(p')$ on the diagram of $\overline{T}(p)$. Formally, C is defined with apex $C = \overline{T}(p')$ and components $C_{\langle\gamma,f\rangle} = \overline{T}(p')_{\langle\gamma,h\circ f\rangle}$. The image $\overline{T}(h)$ is the mediating morphism from colimit $\overline{T}(p)$ to C.

The Sierpinski rule system of figure 4.5 is a global transformation. Its behavior is to split all edges and to add some edges for acyclic triangles. Thus, adding vertices and edges to an input only adds vertices and edges to the output. The induced functor maps monomorphisms to monomorphisms, so factors through U.

Figure 4.7 introduces four additional examples of rule systems that illustrate the various properties that we consider exhaustively. Let us see which of them are global transformations as a preparation for later considerations.

Example 4.2.1. The removal of isolated vertices (figure 4.7c) is a global transformation, since removal is definitely a functorial behavior from $\hat{\mathbf{C}}_{\mathcal{M}}$ to $\hat{\mathbf{C}}_{\mathcal{M}}$.

Example 4.2.2. Simplification of multi-edges (figure 4.7d) is also a global transformation. For any two vertices a and b with at least an edge from a to b, it merges all edges from a to b into a single edge.

Example 4.2.3. On the contrary, the dualization of vertices and edges (figure 4.7a) is not a global transformation. Indeed, consider a monomorphism $i : p_2 \hookrightarrow c_3$ from the path of length 2 p_2 to the cycle of length 3 c_3. In this case $\overline{T}(p_2) = p_3$, $\overline{T}(c_3) = c_3$, and there is no monomorphism sending the four vertices of p_3 to the three vertices of c_3.

Example 4.2.4. Contraction of components (figure 4.7b) is not a global transformation. Consider the monomorphism $i : d_2 \hookrightarrow p_1$ from the graph d_2 with only vertices to the path p_1 of length 1. In this case, $\overline{T}(d_2) = d_2$ and $\overline{T}(p_1) = d_1$ and there is no monomorphism sending the two vertices of d_2 to the single vertex of d_1.

4.3 Accretion and Incrementality

We are interested in having an online algorithm computing Equation (4.2) where the output $\overline{T}(p)$, for any p, is built while the comma category L_T/p is discovered by pattern matching. Informally, starting from a seed corresponding to the r.h.s. of some initial instance, L_T/p is visited from neighbor to neighbor, each instance providing a new piece of material to accumulate to the current intermediate result. We first give the formal features to be able to speak about intermediate results, leading to the notion of *accretive rule system*. Then we give a criterion, called *incrementality*, for a rule system to be an accretive global transformation.

In this algorithmic perspective, we restrict our discussion to finite presheaves and finite rule systems, so that L_T/p is finite as well. Moreover, we assume that L_T/p is connected; disconnected components can be processed independently. Finally, we fix a given finite rule system $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$ and a finite presheaf p.

4.3.1 Accretive Rule Systems and Global Transformations

Informally, an intermediate result consists in the application of $\overline{T}$ on an incomplete knowledge of L/p, *i.e.*, on a partial decomposition of the input. Our study of partial decompositions starts with some remarks. To begin, the comma category L/p is a preordered set.

Proposition 4.3.1. *For any category* $\mathbf{I}$*, any full embedding* $F : \mathbf{I} \to \hat{\mathbf{C}}_{\mathcal{M}}$*, and any presheaf* $p \in \hat{\mathbf{C}}_{\mathcal{M}}$*, the comma category* F/p *is* thin*, i.e., there is at most one morphism between any two objects.*

Proof. For any $i_1, i_2 \in \mathbf{I}$, consider $\langle i_1, m_1 : F(i_1) \hookrightarrow p \rangle$ and $\langle i_2, m_2 : F(i_2) \hookrightarrow p \rangle$ of F/p together with two parallel morphisms $\langle f : i_1 \to i_2, m_2 \rangle$ and $\langle g : i_1 \to i_2, m_2 \rangle$ between them. We want to prove that $f = g$. By definition of morphisms in F/p, $m_2 \circ F(f) = m_1 = m_2 \circ F(g)$. Since m_2 is a monomorphism, we get $F(f) = F(g)$. Since F is in particular faithful, we obtain the desired equality $f = g$. □

Therefore, thinking in terms of maximal, non-maximal and minimal instances, sub- and super-instances actually makes sense for any comma category L/p.

Moreover, L/p being a preorder makes the colimit $\overline{T}(p)$ of the diagram $\mathrm{D}_T(p) : \mathrm{L}/p \to \hat{\mathbf{C}}$ special. Informally, only maximal instances matter, sub-instances being used to specify how to amalgamate the r.h.s. of maximal instances. Formally, whenever we have a morphism $\langle e, f' \rangle : \langle \gamma, f' \circ \mathrm{L}(e) \rangle \to \langle \gamma', f' \rangle \in \mathrm{L}/p$, the r.h.s. of γ' contains the r.h.s. of γ through $e : \gamma \to \gamma'$. With $f = f' \circ \mathrm{L}(e)$, this means that $\langle \gamma, f \rangle$ does not contribute more data to the output. The role of the non-maximal $\langle \gamma, f \rangle$ is to specify how the r.h.s. of some γ'' should be aligned with the r.h.s. of γ' in the resulting presheaf when there is a second morphism $\langle \gamma, f \rangle \to \langle \gamma'', f'' \rangle$ to another maximal instance $\langle \gamma'', f'' \rangle$.

As an example, consider the diagram depicted on the bottom right of figure 4.6 and its colimit on the top right. All the elements of the colimit are given in the r.h.s. of the maximal instances (the 3 triangles). The minimal instances (the 6 one-vertex graphs) are used to specify how the 9 vertices of the 3 refined triangles should be merged to get the colimit.

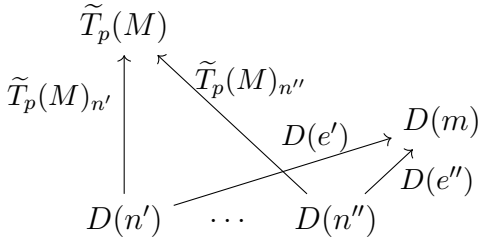
Computing $\overline{T}$ Online. Given a presheaf $p \in \hat{\mathbf{C}}$, we call a *partial decomposition of p with respect to* L a subset M of maximal instances of L/p such that the restriction $\widetilde{M}$ of L/p to M and morphisms into M remains connected. We write $\widetilde{\mathrm{L}/p}$ for the category of partial decompositions of p with set inclusions as morphisms. $\widetilde{\mathrm{L}/p}$ represents the different ways L/p can be visited from maximal instance to maximal instance by the use of non-maximal instances to guide the merge. We extend the action of $\overline{T}$ on p into a function $\widetilde{T}_p : \widetilde{\mathrm{L}/p} \to \hat{\mathbf{C}}$ as follows:

$$\widetilde{T}_p(M) = \mathrm{Colim}(\mathrm{D}_T(p) \restriction \widetilde{M}). \tag{4.3}$$

The definition $\widetilde{T}_p$ is in fact a complete functor also acting on any morphism $M \subseteq M'$ of $\widetilde{\mathrm{L}/p}$ using the exact same construction as given in remark 4.2.1 for $\overline{T}$.

In the case of figure 4.6, the maximal instances being the three triangles, the partial decompositions consist of subsets having 0, 1, 2 or 3 of these triangles. As an example, choose arbitrarily two of these triangles, say t_1 and t_2. The intermediate result $\widetilde{T}_p(\{t_1, t_2\})$ is the graph consisting of the two refinements of t_1 and t_2 glued by their common vertex.

The online computation of $\overline{T}(p)$ consists in iterating a simple step that builds $\widetilde{T}_p(M \cup \{m\})$ from $\widetilde{T}_p(M)$ as soon as a new maximal instance m has been discovered. This step is the local amalgamation of $\mathrm{D}_T(p)(m)$ with $\widetilde{T}_p(M)$ considering all the non-maximal sub-instances, say $\{n', \ldots, n''\}$, shared by the elements of M and m. Indeed, each such sub-instance n gives rise to a span $\widetilde{T}_p(M) \leftarrow \mathrm{D}_T(p)(n) \to \mathrm{D}_T(p)(m)$. Gathering all these spans leads to the *suture* diagram $S_{M,m}$ defined as follows:



(4.4)

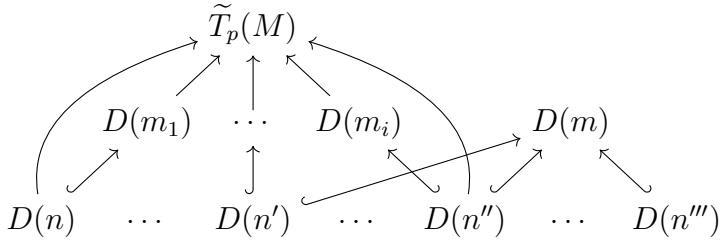
where D stands for $\mathrm{D}_T(p)$ for simplicity, the notation that we will use from now on. The colimit of a single span being called a pushout, we call the colimit of this collection of spans a *generalized pushout*. The fact that this generalized pushout $\mathrm{Colim}(S_{M,m})$ and the desired colimit $\widetilde{T}_p(M \cup \{m\})$ as given in Equation (4.3) do coincide is formalized by the following proposition.

Proposition 4.3.2. *Let $M' = M \cup \{m\} \in \widetilde{\mathrm{L}/p}$. As a cocone, $\widetilde{T}_p(M')$ has the same apex as $\mathrm{Colim}(S_{M,m})$ and has components*

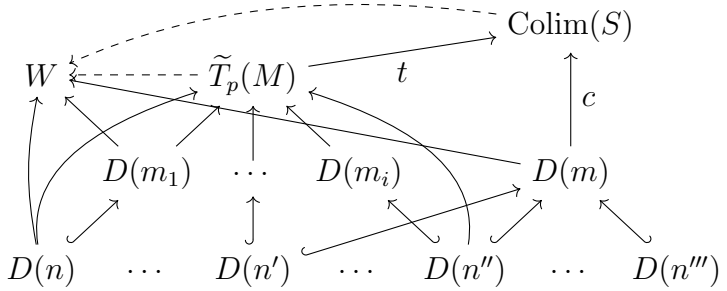
$$\widetilde{T}_p(M')_n = \begin{cases} \mathrm{Colim}(S_{M,m})_{\widetilde{T}_p(M)} \circ \widetilde{T}_p(M)_n & \text{if } n \in \widetilde{M}, \\ \mathrm{Colim}(S_{M,m})_{D(m)} \circ D(e) & \text{for any } e : n \hookrightarrow m, \end{cases}$$

where $S_{M,m}$ is diagram (4.4).

Proof. For simplicity, we write S for $S_{M,m}$ and K (resp. K') for $\widetilde{T}_p(M)$ (resp. $\widetilde{T}_p(M')$) as a cocone. Let us consider a bigger diagram containing the diagrams $D \restriction \widetilde{M}$, S and $D \restriction \widetilde{M'}$. For this, we take $D \restriction \widetilde{M'}$ and add the object $\widetilde{T}_p(M)$ and all morphisms $K(o) : D(o) \to \widetilde{T}_p(M)$ for $o \in \widetilde{M}$. By universality of $\widetilde{T}_p(M)$, there is a bijection between cocones on this big diagram and cocones on $D \restriction \widetilde{M'}$.



where $M = \{m_1, \ldots, m_i\}$. Adding to the picture $\mathrm{Colim}(S)$ with its associated components $t : \widetilde{T}_p(M) \to \mathrm{Colim}(S)$ and $c : D(m) \to \mathrm{Colim}(S)$, recall that $K'_o = t \circ K_o$ for all $o \in \widetilde{M}$, and $K'_o = c \circ D(e)$ for all $e : o \hookrightarrow m$. This is indeed a cocone on $D \restriction \widetilde{M}'$ by properties of t and c, and extending it with $K'_{\widetilde{T}_p(M)} = t$ gives us the unique corresponding cocone on the big diagram. To establish that it is universal, let us consider another arbitrary cocone W on $D \restriction \widetilde{M}'$.



By diagram chasing, one can see that this cocone can be restricted to its components on $\widetilde{M}$, *i.e.*, on $\{n, \ldots, n'', m_1, \ldots, m_i\}$. This gives us a unique extension of W on all the big diagram. By restricting again to its components on S, *i.e.*, components at $n', \ldots, n'', \widetilde{T}_p(M), m$, and by universality of $\widetilde{T}_p(M')$, we obtain a unique mediating from $\widetilde{T}_p(M')$ to W, as wanted. □

As an illustration, for computing $\widetilde{T}_p(\{t_1, t_2, t_3\})$ (which is in fact the output in figure 4.6), it is enough to amalgamate $\widetilde{T}_p(\{t_1, t_2\})$ (already considered) with the refinement $D(t_3)$ of the last triangle t_3, using as suture the two vertices of t_3 shared with t_1 and t_2 playing the role of n' and n'' in diagram (4.4).

Remark 4.3.1. To summarize, computing $\overline{T}(p)$ online is a matter of collecting the finite set of all maximal instances $\{m_1, m_2, \ldots, m_k\}$ of L/p in any order satisfying that m_{i+1} is connected to $\widetilde{M}_i$ where $M_i = \{m_1, \ldots, m_i\}$. This allows to replace the single colimit computation of the whole diagram, as in Equation (4.2), by a sequence of smaller colimit computations using the induction relation:

$$\widetilde{T}_p(M_i) = \begin{cases} D(m_1) & \text{if } i = 1 \\ \mathrm{Colim}(S_{M_{i-1},m_i}) & \text{otherwise.} \end{cases} \tag{4.5}$$

The base case is obtained from equation 4.3 for a singleton set of maximal instance, and the inductive one is established by proposition 4.3.2, S_{M_{i-1},m_i} being a generalized pushout diagram linking $\widetilde{T}_p(M_{i-1})$ and $D(m_i)$. The final value $\widetilde{T}_p(M_k)$ of this sequence is the colimit of $\widetilde{M}_k$. $\widetilde{M}_k$ is exactly the diagram D without the morphisms between the non-maximal instances. But the colimit $\widetilde{T}_p(M_k)$ of $\widetilde{M}_k$ is necessarily the same as the colimit $\overline{T}(p)$ of D by the following proposition.

Proposition 4.3.3. *The subcategory* $\widetilde{M_k}$ *of* L/p *given by all instances, but only morphisms to maximal instances is final in* L/p*, in the sense of final functor.*

Proof. We need to show that for any instance $o \in \mathrm{L}/p$, and any two morphisms $e_0 : o \hookrightarrow o_0$ and $e_1 : o \hookrightarrow o_1$, there is a zigzag z in the subcategory and a sequence of morphisms of L/p from o to each z_k that commutes with each $\overline{z}_k$. Take $i \in \{0, 1\}$. If o_i is maximal we set $\overline{z}_i = e_i$. If it is non-maximal, we set $\overline{z}_i = e'_i \circ e_i$ for any $e'_i : o_i \hookrightarrow o'_i$ to some maximal o'_i. We have just built a valid z of length 2 in the subcategory, the associated sequence of morphisms being simply $\overline{z}_0, id_o, \overline{z}_1$, which trivially commutes as wanted. □

Accretive Rule Systems. We are interested in those rule systems where the intermediate results stay inside $\hat{\mathbf{C}}_{\mathcal{M}}$, *i.e.*, such that $\widetilde{T}_p(M \subseteq M')$ are monomorphisms for any p and any $M \subseteq M' \in \widetilde{\mathrm{L}/p}$. This leads to the following definition of accretive rule systems.

Definition 4.3.1. An *accretive rule system* T is a rule system such that for any $p \in \hat{\mathbf{C}}$, $\widetilde{T}_p$ factors through the forgetful functor $\mathrm{U} : \hat{\mathbf{C}}_{\mathcal{M}} \to \hat{\mathbf{C}}$.

Example 4.3.1. The rule system of figure 4.7b is accretive. Focusing on connected L/p, its l.h.s. implies that p is a connected graph. Any $M \in \widetilde{\mathrm{L}/p}$ corresponds to a connected sub-graph of p and is sent to the single vertex graph if it is not empty, or to the empty graph otherwise. So for any relation $M \subseteq M'$, $\widetilde{T}_p(M \subseteq M')$ is the empty morphism or the identity morphism, and both are monomorphisms.

Example 4.3.2. The rule system of figure 4.7d is also accretive. Again, p is a connected graph and any $M \in \widetilde{\mathrm{L}/p}$ corresponds to a connected sub-graph of p. Here M is sent to the same graph with parallel edges simplified into single edges. So for any relation $M \subseteq M'$, M' is sent either to the same thing as M when M' only adds more parallel edges, or to a strictly greater graph otherwise. In both case $\widetilde{T}_p(M \subseteq M')$ is a monomorphism.

Example 4.3.3. On the contrary, the rule system of figure 4.7a is not accretive. Consider the cycle c_3 of length 3, and the associated L/c_3. The latter contains 3 instances of the rule γ_1 and 3 maximal instances of the rule γ_2. Consider the relation $M \subseteq M'$ where M' contains all three maximal instances and M only two of them. We have $\widetilde{T}_p(M) = p_3$ and $\widetilde{T}_p(M') = c_3$, but there are no monomorphisms between these two graphs.

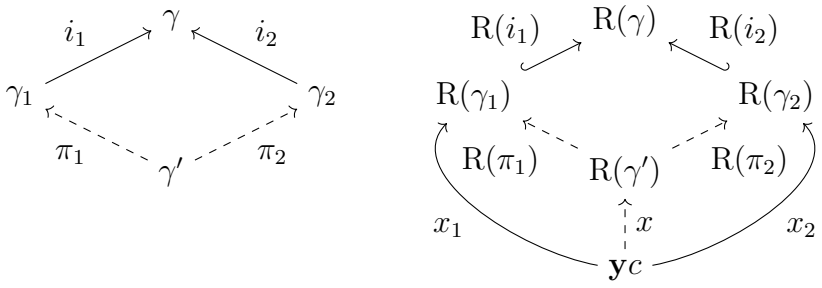
Example 4.3.4. By the exact same reasoning, the rule system of figure 4.7c is also not accretive.

4.3.2 Incremental Rule Systems

We are interested in giving sufficient conditions for rule systems to be global transformations. These conditions also imply accretiveness.

Our strategy consists in preventing any super-rule to merge by itself the r.h.s. of its sub-rules. In other words, the rule only adds new elements to the r.h.s. of its sub-rules in an *incremental* way. A positive expression of this constraint is as follows: if the r.h.s. of two rules overlap in the r.h.s. of a common super-rule, this overlap must have been required by some common sub-rules.

Definition 4.3.2. Given a rule system $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$, we say that a rule $\gamma \in \mathbf{\Gamma}$ is incremental if for any two sub-rules $\gamma_1 \xrightarrow{i_1} \gamma \xleftarrow{i_2} \gamma_2$ in $\mathbf{\Gamma}$, any representable presheaf $\mathbf{y}c$, and any $\mathrm{R}(\gamma_1) \xleftarrow{x_1} \mathbf{y}c \xrightarrow{x_2} \mathrm{R}(\gamma_2)$ such that $\mathrm{R}(i_1) \circ x_1 = \mathrm{R}(i_2) \circ x_2$, there are some $\gamma_1 \xleftarrow{\pi_1} \gamma' \xrightarrow{\pi_2} \gamma_2$ and $x : \mathbf{y}c \to \mathrm{R}(\gamma')$ such that the following diagrams commute.

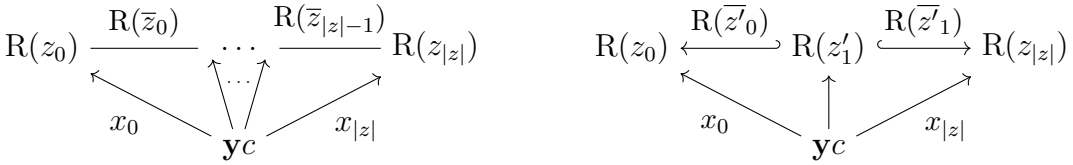


A rule system T is said *incremental* if every $\gamma \in \mathbf{\Gamma}$ is incremental.

The Sierpinski gasket rule system (figure 4.5) is incremental. The only non-trivial case is when the sub-rules γ_1 and γ_2 of definition 4.3.2 are set to the edge rule of figure 4.5 and γ to be the triangle rule, such that the r.h.s. of γ_1 and γ_2 overlap on a common vertex in $\mathrm{R}(\gamma)$ (morphisms x_1 and x_2 of definition 4.3.2). This vertex is nothing but the image of the vertex of $\mathrm{L}(\gamma)$ common to the inclusions of $\mathrm{L}(\gamma_1)$ and $\mathrm{L}(\gamma_2)$ in $\mathrm{L}(\gamma)$. This invites us to set γ' to the vertex rule of figure 4.5 and complete the requirements of definition 4.3.2 to get incrementality.

The main constraint enforced by the incrementality criterion is that any merge is always required by sub-rules, as stated by the following lemma.

Lemma 4.3.4. *Given an incremental rule system $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$, a zigzag z in $\mathbf{\Gamma}$, a representable presheaf $\mathbf{y}c$ and two morphisms $x_0 : \mathbf{y}c \to \mathrm{R}(z_0)$, $x_{|z|} : \mathbf{y}c \to \mathrm{R}(z_{|z|})$ such that z links x_0 and $x_{|z|}$, i.e., there is a cone $\langle x_i : \mathbf{y}c \to \mathrm{R}(z_i) \rangle_{0 \leq i \leq |z|}$ that commutes with $\mathrm{R}(z)$. Then there is a zigzag z' in $\mathbf{\Gamma}$ of the form $z_0 = z'_0 \leftarrow z'_1 \rightarrow z'_2 = z_{|z|}$ that also links x_0 and $x_{|z|}$.*



Proof. This is proved by induction on the length of z. We show the base case with $|z| = 0$ by taking $z' = \langle id_{z_0}, id_{z_0} \rangle$. For the induction case, we assume that the proposition is true for any zigzag of size k and take z of size $k+1$. Then we have two cases that depends on the direction of the first morphism of z:

- If $\overline{z}_0 : z_1 \to z_0$ we can apply the induction hypothesis on the zigzag $y = \langle \overline{z}_1, \ldots \overline{z}_{k+1} \rangle$ to get a zigzag y' of the form $z_1 = y'_0 \leftarrow y'_1 \rightarrow y'_2 = z_{k+1}$ that links x_1 and x_{k+1}. Then we take $z' = \langle \overline{z}_0 \circ \overline{y'}_0, \overline{y'}_1 \rangle$ to conclude this case.

- If $\overline{z}_0 : z_0 \to z_1$ we first apply the induction hypothesis on $y = \langle \overline{z}_1, \ldots \overline{z}_{k+1} \rangle$ to get a zigzag y' of the form $z_1 = y'_0 \leftarrow y'_1 \rightarrow y'_2 = z_{k+1}$ that links x_1 and x_{k+1}. Let $x' : \mathbf{y}c \to \mathrm{R}(y'_1)$ be the induced linking morphism such that $\mathrm{R}(\overline{z}_0) \circ x_0 = \mathrm{R}(\overline{y'}_0) \circ x'$. Applying definition 4.3.2 on this square, we get a

quadruplet $\langle \gamma' \in \mathbf{\Gamma}, \pi_1 : \gamma' \to z_0, \pi_2 : \gamma' \to y'_1, x : \mathbf{y}c \to \mathrm{R}(\gamma') \rangle$ such that $\mathrm{R}(\overline{z}_0) \circ \mathrm{R}(\pi_1) = \mathrm{R}(\overline{y'}_0) \circ \mathrm{R}(\pi_2)$, $x_0 = \mathrm{R}(\pi_1) \circ x$, and $x' = \mathrm{R}(\pi_2) \circ x$. Here the wanted z' is $\langle \pi_1, \overline{y'}_1 \circ \pi_2 \rangle$.

□

Consider any monomorphism $h : p \hookrightarrow p'$ of presheaves such that some merge is required by the computation of $\overline{T}(p')$ between some elements of r.h.s. instances also involved by $\overline{T}(p)$. lemma 4.3.4 ensures that it is required by a sub-rule which must also be instantiated by $\overline{T}(p)$ so that the merge is also required by the computation of $\overline{T}(p)$. In other words, $\overline{T}(h)$ is a monomorphism as established by the following theorem.

Theorem 4.3.5. *Any incremental rule system is a global transformation.*

Proof. Consider the setting of remark 4.2.1. We need to show that $\overline{T}(f)$ is a monomorphism for $f : p \hookrightarrow p'$. For any two morphisms $x_1, x_2 : \mathbf{y}c \to \overline{T}(p)$ for any $c \in \mathbf{C}$ such that $\overline{T}(f) \circ x_1 = \overline{T}(f) \circ x_2$. We are left to show that $x_1 = x_2$.

Take $k \in \{1, 2\}$. By proposition 4.1.8, x_k factors through some cocone component of $\overline{T}(p)$. Say $x_k = \overline{T}(f)_{\langle \gamma_k, m_k \rangle} \circ y_k$ where $\langle \gamma_k, m_k \rangle$ is an object of L/p. So $\overline{T}(f) \circ x_k = \overline{T}(f) \circ \overline{T}(p)_{\langle \gamma_k, m_k \rangle} \circ y_k = \overline{T}(p')_{\langle \gamma_k, f \circ m_k \rangle} \circ y_k$. Since $\overline{T}(f) \circ x_1 = \overline{T}(f) \circ x_2$ then $\overline{T}(p')_{\langle \gamma_1, f \circ m_1 \rangle} \circ y_1 = \overline{T}(p')_{\langle \gamma_2, f \circ m_2 \rangle} \circ y_2$. Using proposition 4.1.8 on the latter equality into the colimit $\overline{T}(p')$, there is a zigzag z in L/p' from $\langle \gamma_1, f \circ m_1 \rangle$ to $\langle \gamma_2, f \circ m_2 \rangle$ that links y_1 and y_2 through $\mathrm{D}(p')$. The zigzag $\mathrm{Proj}[\mathrm{L}/p'](z)$ in $\mathbf{\Gamma}$ obviously links y_1 and y_2 through $\mathrm{U} \circ \mathrm{R}$. Since the rule system T is incremental, we apply lemma 4.3.4 to obtain a zigzag z' in $\mathbf{\Gamma}$ of the form $\gamma_1 = z'_0 \leftarrow z'_1 \to z'_2 = \gamma_2$ that links y_1 and y_2 through $\mathrm{U} \circ \mathrm{R}$. Let us define the zigzag z'' in L/p to be $\langle \gamma_1, m_1 \rangle \hookleftarrow \langle z'_1, h \rangle \hookrightarrow \langle \gamma_2, m_2 \rangle$ where $h = m_1 \circ \mathrm{L}(\overline{z'}_1) = m_2 \circ \mathrm{L}(\overline{z'}_2)$ and $\overline{z''}_j = \langle \overline{z'}_k, m_k \rangle$ for $k \in \{1, 2\}$. Clearly, $\mathrm{Proj}[\mathrm{L}/p](z'') = z'$, so z'' links y_1 and y_2 through $\mathrm{D}(p)$. By commutation properties of cocones over $\mathrm{D}(p)$, we have that $\overline{T}(p)_{\langle \gamma_1, m_1 \rangle} \circ y_1 = \overline{T}(p)_{\langle \gamma_2, m_2 \rangle} \circ y_2$, which implies $x_1 = x_2$ as wanted. □

The previous remark also applies for intermediate results leading to the following theorem concerning accretiveness.

Theorem 4.3.6. *Any incremental rule system is accretive.*

Proof. The proof is similar to the proof of theorem 4.3.5. □

However, the converses of these theorems do not hold so incrementality is sufficient but not necessary as illustrated by the following examples.

Example 4.3.5. The rule system of figure 4.7c is a global transformation as explained in example 4.2.1, but not incremental. Consider $e_1 : \gamma_1 \to \gamma_2$ be the plain arrow into γ_2 and $e_2 : \gamma_1 \to \gamma_2$ the dashed arrow into γ_2. The cospan $\gamma_1 \hookrightarrow \gamma_2 \hookleftarrow \gamma_1$ is such that $h_1 : d_1 \hookrightarrow \mathrm{R}(\gamma_1)$ and $h_2 : d_1 \hookrightarrow \mathrm{R}(\gamma_1)$ such that $\mathrm{R}(e_1) \circ h_1 = \mathrm{R}(e_2) \circ h_2$ but there is no rule γ' to ensure the incrementality condition.

Example 4.3.6. Similarly, the rule system of figure 4.7d is a global transformation (example 4.2.2) but is not incremental. Consider $e_1 : \gamma_2 \to \gamma_3$ be the plain arrow into γ_3 and $e_2 : \gamma_2 \to \gamma_3$ the dashed arrow into γ_3. The cospan $\gamma_2 \hookrightarrow \gamma_3 \hookleftarrow \gamma_2$ is such that $h_1 : l_1 \hookrightarrow \mathrm{R}(\gamma_2)$ and $h_2 : l_1 \hookrightarrow \mathrm{R}(\gamma_2)$ such that $\mathrm{R}(e_1) \circ h_1 = \mathrm{R}(e_2) \circ h_2$ but there is no rule γ' to ensure the incrementality condition.

Example 4.3.7. The rule system of figure 4.7b is accretive (example 4.3.1) but non-incremental. Consider $e_1 : \gamma_1 \to \gamma_2$ the plain arrow into γ_3 and $e_2 : \gamma_1 \to \gamma_2$ the dashed arrow into γ_3. Observe that for the cospan $\gamma_1 \hookrightarrow \gamma_2 \hookleftarrow \gamma_1$ we have $h_1 : d_1 \hookrightarrow \mathrm{R}(\gamma_2)$ and $h_2 : d_1 \hookrightarrow \mathrm{R}(\gamma_2)$ such that $\mathrm{R}(e_1) \circ h_1 = \mathrm{R}(e_2) \circ h_2$ but there is no rule γ' to ensure the incremental condition.

Example 4.3.8. Similarly, the rule system of figure 4.7d is accretive (example 4.3.2) but non-incremental.

Summarizing the properties collected with the four examples of figure 4.7 and with the one of figure 4.5 in a table, we can see that being a global transformation and being accretive are orthogonal properties, but incrementality forces the two.

	non-incr.		incr.	
	non-G.T.	G.T.	non-G.T.	G.T.
non-accr.	ex. f. 4.7a	ex. f. 4.7c	None, t. 4.3.5/4.3.6	None, t. 4.3.6
accr.	ex. f. 4.7b	ex. f. 4.7d	None, t. 4.3.5	Sierpenski

4.4 Computing Accretive Global Transformations

This section is devoted to the description of an effective implementation of the on-line procedure considered in section 4.3.1. In this context, we focus on incremental global transformations. We first explain how the categorical concepts of section 4.3 are represented computationally (section 4.4.1). This is followed by a detailed presentation of the algorithm (section 4.4.2).

4.4.1 Categorical Constructions Computationally

Up to now, we exposed everything formally using categorical concepts: the category of presheaves $\hat{\mathbf{C}}_{\mathcal{M}}$, finite incremental rule systems $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$, and the comma category L/p for some finite presheaf p. We now describe their computational counterparts. First, let us introduce some notations used in the algorithm:

- X^* stands for the set of finite words on the alphabet X; the empty word is denoted by ε and the concatenation by $u \cdot v$ for any two words $u, v \in X^*$.
- $\coprod_{a \in A} B(a)$ is the set of pairs (a, b) where $a \in A$ and $b \in B(a)$.
- $\prod_{a \in A} B(a)$ is the set of functions $f : A \to \bigcup_{a \in A} B(a)$ such that for any $a \in A$, $f(a) \in B(a)$. Such functions are also manipulated as sets of pairs. Those pairs are written $a \mapsto f(a)$.

The Category of Presheaves with Monomorphisms. The category $\hat{\mathbf{C}}_{\mathcal{M}}$ is the formal abstraction for a library providing a data structure suitably captured by presheaves (like sets, graphs, Petri nets, etc.) and how an instance of that data structure (presheaves) is part of another one (monomorphisms). Two functions - $- \circ -$ and $- = -$ need to be provided to compute composition and equality test of sub-parts. The library also needs to come with a pattern matching procedure taking as input two finite presheaves p and p' and returning the set $\mathrm{Hom}_{\hat{\mathbf{C}}_{\mathcal{M}}}(p, p')$ of occurrences of p in p'. Finally, the library is assumed to provide a particular construction operation called `generalizedPushout`(p_1, p_2, S) computing the generalized pushout,

i.e., the colimit of the collection of spans S, each span being represented as a triplet $(p \in \hat{\mathbf{C}}_{\mathcal{M}}, f_1 : p \hookrightarrow p_1, f_2 : p \hookrightarrow p_2)$. The resulting colimit is returned as a triplet $(c \in \hat{\mathbf{C}}_{\mathcal{M}}, g_1 : p_1 \hookrightarrow c, g_2 : p_2 \hookrightarrow c)$ where c is the apex and g_1, g_2 the corresponding component morphisms.

Finite Incremental Rule System. A finite rule system is described as a finite graph whose vertices are rules $l \Rightarrow r$ as pairs of presheaves and edges are pairs of monomorphisms $\langle i_l : l_1 \hookrightarrow l_2, i_r : r_1 \hookrightarrow r_2 \rangle$. Functors L and R return the first and second components of these pairs respectively. At the semantic level, $\mathbf{\Gamma}$ is the category generated from this graph. Finally, incrementality as presented in definition 4.3.2 is clearly decidable on finite rule systems, giving rise to an accretive global transformation by theorems 4.3.5 and 4.3.6.

The Category of Instances. By proposition 4.3.1, L/p is a preordered set, but in our implementation, any time an instance is matched, all of its isomorphic instances are taken care of at the same time. This corresponds informally to taking the poset of equivalence classes of the preordered set. Also, by proposition 4.3.3, morphisms between non-maximal instances can be ignored. All in all, L/p is adequately thought of as an abstract undirected bipartite graph that we call *the network*.

Finally, the L/p is never entirely represented in memory (neither is the cocone associated to the resulting colimit). A first instance is constructed, and the others are built from neighbor to neighbor through the operation $\coprod_n \mathrm{Hom}_{\mathrm{L}/p}(n, m)$ and $\coprod_m \mathrm{Hom}_{\mathrm{L}/p}(n, m)$. The former lists the sub-instances of m and the latter lists the super-instances of n. For the "incoming neighbors", that are the sub-instances $\coprod_n \mathrm{Hom}_{\mathrm{L}/p}(n, \langle \gamma', f' \rangle)$, they are specified as

$$\{(\langle \gamma, f' \circ \mathrm{L}(e) \rangle, \langle e, f' \rangle) \mid e : \gamma \to \gamma'\}.$$

This corresponds simply to the composition $- \circ -$ in $\hat{\mathbf{C}}_{\mathcal{M}}$ discussed earlier. On the contrary, "outgoing neighbors" or super-instances $\coprod_m \mathrm{Hom}_{\mathrm{L}/p}(\langle \gamma', f' \rangle, m)$ correspond to extensions and are obtained by pattern matching.

$$\{\langle e', f'' \rangle : \langle \gamma', f' \rangle \to \langle \gamma'', f'' \rangle \mid \\ e' : \gamma' \to \gamma'', f'' \in \mathrm{Hom}_{\hat{\mathbf{C}}_{\mathcal{M}}}(\mathrm{L}(\gamma''), p) \text{ s.t. } f' = f'' \circ \mathrm{L}(e')\}.$$

Notice that these two specifications are obtained by simply unfolding the definition of the morphisms of the comma category. Also, the set of incoming morphisms $e : \gamma \to \gamma'$ and outgoing morphisms $e' : \gamma' \to \gamma''$ in $\mathbf{\Gamma}$ are directly available in the graph representation of the rule system T as said earlier. These operations are used to implement a breadth-first algorithm, earlier instances being dropped away as soon as their maximal super-instances have been found.

4.4.2 The Global Transformation Algorithm

Algorithm 1 gives a complete description of a procedure to compute $T(p)$ online. The algorithm manages four variables P, N, E and C. The variable P contains intermediate results and finally the output presheaf. The part of the network that is kept in memory is represented by variables N and E: N is a queue containing, in order of discovery, the non-maximal instances that might still have a role to play.

Algorithm 1:

Input: T : rule system on $\hat{\mathbf{C}}_{\mathcal{M}}$
Input: $p : \hat{\mathbf{C}}_{\mathcal{M}}$
Variable: $P : \hat{\mathbf{C}}_{\mathcal{M}}$
Variable: $N : (\mathrm{L}/p)^*$
Variable: $E \subseteq \coprod_{n \in N} \coprod_m \mathrm{Hom}_{\mathrm{L}/p}(n, m)$
Variable: $C : \prod_{n \in N} \mathrm{Hom}_{\hat{\mathbf{C}}_{\mathcal{M}}}(D(n), P)$

1 **let** $n = \texttt{findAnyMinimal}(T, g)$, *i.e.*, any minimal element in L/p
2 **let** $E = \emptyset$, $C = \{n \mapsto id_{D(n)}\}$, $N = n$, $P = D(n)$
3 **while** $N \neq \varepsilon$ **do**
4 **let** $n = \texttt{head}(N)$, *i.e.*, the first instance in the queue without modifying N
5 **let** $M' = \coprod_m \mathrm{Hom}_{\mathrm{L}/p}(n, m)$
6 **for** $(m, e) \in M'$ *s.t.* $(n, m, e) \notin E$ *and* m *is maximal* **do**
7 **let** $E' = \coprod_{n' \neq m} \mathrm{Hom}_{\mathrm{L}/p}(n', m)$
8 **let** $S = \{(n', C(n'), D(e')) \mid (n', e') \in E', n' \in N\}$
9 **let** $(P', t, c) = \texttt{generalizedPushout}(P, D(m), S)$
10 $E := E \cup \{(n', m, e') \mid (n', e') \in E'\}$
11 $C := C \cup \{n' \mapsto c \circ D(e') \mid (n', e') \in E', n' \notin N\}$
12 $N := N \cdot \langle n' \mid (n', e') \in E', n' \notin N \rangle$
13 $P := P'$
14 $E := \{(n', m, e') \in E \mid n' \neq n\}$
15 $C := \{n' \mapsto C(n') \in C \mid n' \neq n\}$
16 $N := \texttt{tail}(N)$, *i.e.*, removes the first instance n from the queue
17 **return** P

E associates each instance in N to the set of its maximal super-instances that have already been processed. For simplicity, E is not represented as a function from N to sets, but as a relation. The r.h.s. $D(n)$ of each instance $n \in N$ is already in the current result P through the morphism kept as $C(n)$.

Figure 4.8 illustrates the first steps of algorithm 1 representing maximal instances as black dots, and non-maximal instances as white squares. The initialization step is to find a first instance (line 1). For that, we try each minimal pattern, and start with the first founded minimal instance, say n_1. At this point, the first intermediate result P_0 is simply the r.h.s. $D(n_1)$; we memorize the (identity) relationship between $D(n_1)$ and P_0, call it $C(n_1) : D(n_1) \to P_0$, and enqueue n_1 (line 2). Enqueued non-maximals are treated one after the other (lines 3, 4, 16). For each, we consider all maximal super-instances of n_1 (lines 5, 6). In figure 4.8, we assume three such super-instances m_1, m_2 and m_3. They are processed one after the other (line 6).

The first iteration processes m_1 by taking all its sub-instances $n_1, \ldots, n_4$ (line 7). The suture S is computed (line 8) by considering all already computed non-maximals (*i.e.*, in N) among these sub-instances. Here, only n_1 is already known and serves to define a one-span suture with morphisms $C(n_1) : D(n_1) \to P_0$ and $D(e') : D(n_1) \to D(m_1)$, where e' is the morphism from n_1 to m_1. The generalized pushout of P_0 and the r.h.s. $D(m_1)$ is therefore computed and gives the new intermediate result P_1 (lines 9, 13). Since P_1 includes the r.h.s. of all discovered non-maximals

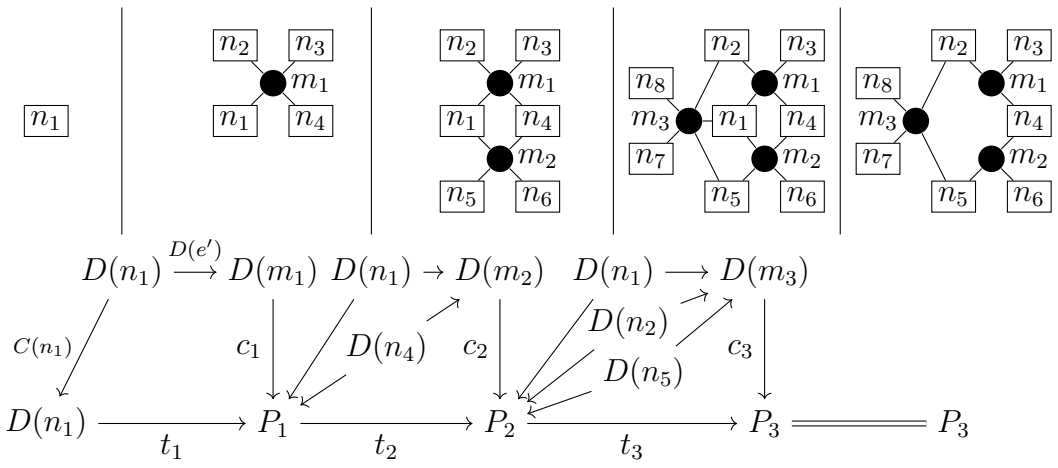


Figure 4.8: Evolution of the data during the four firsts steps of the algorithm. From left to right: we start with a non-maximal instance, process its associated maximal instances successively, and finally drop the non-maximal. At each stage, the output is updated by generalized pushout.

$N = \{n_1, \ldots, n_4\}$, we memorize as $C(n) : D(n) \to P_1$ for $n \in N$ the locations of these r.h.s. in P_1 (line 11). The newly discovered non-maximal instances n_2, n_3 and n_4 are enqueued (line 12).

The second iteration processes m_2 similarly and all its sub-instances n_1, n_4, n_5, and n_6 are computed. This time, n_1 and n_4 are used for computing the new intermediate result P_2 by generalized pushout using the two spans $\langle n_1, C(n_1) : D(n_1) \to P_1, D(j) : D(n_1) \to D(m_2)\rangle$ and $\langle n_4, C(n_4) : D(n_4) \to P_1, D(k) : D(n_4) \to D(m_2)\rangle$ as suture. The set N of discovered non-maximals is updated by adding n_5 and n_6 as well as the locations C of their r.h.s. in P_2.

The processing of m_3 is similar and shows no novelty. At this point, non-maximal n_1 does not have any further role to play: the r.h.s. of all its associated maximals are already amalgamated to the current intermediate result. n_1 is dropped together with all data associated to it (lines 14–16), as shown in the last step of Figure4.8. Non-maximal instances being processed in the order of first discovery, the next one is n_2 in the example.

During these processings, other non-maximal instances see some of their associated maximals being processed. We have to keep track of this to avoid double processing of maximals which would cause infinite loops (condition at line 6). This is the role of E to maintain this information. Clearly E contains only the useful part of the network: edges from maximals to their sub-instances are registered when discovered (line 10) but cleared up as soon as a non-maximal is dropped (line 14). Considering that non-maximal instances are treated in order of appearance, the algorithm will process the maximals at distance 1 from n_1 first, then those at distance 2, and so on, until the complete connected component of the network is processed. In memory, there are never stored more than four "radius" of instances d, $d+1$, $d+2$ and $d+3$ from n_1.

Theorem 4.4.1. *Algorithm 1 is correct, i.e., the final value of P is $\overline{T}(p)$.*

Proof. We ignore the case when L/p is composed of a single instance, since the algorithm behaves trivially in that case.

Ignoring lines 8, 9, 11, 13, and 15, variables P and C, and looking only at non-maximal instances (variable n), the algorithm behaves like a usual breadth-first search. Indeed, the search begins by enqueueing a first non-maximal instance at line 2. Each iteration of the while loop (line 3) processes the next non-maximal instance n in the queue (line 4), lists all its "neighbors via a maximal instance" (lines 5–7) and enqueues those that have not yet been visited (lines 10, 12) before popping n out of the queue (line 16). Variables E and N serve as the set of visited non-maximal instances. The reason line 14 can remove all occurrences of n in the set E without creating an infinite loop is that E memorizes the maximal instances m' from which each enqueued non-maximal instance n' has been reached (line 10). The constraint $(n, m, e) \notin E$ of the for loop (line 6) prevents this path to be taken in the other direction.

Since all non-maximal instances are assigned to n at line 4, and each maximal instance is a super-instance of some non-maximal instance, we have that m goes through all maximal instances as well (line 6). Let us call m_1, ..., m_k, the successive values taken by m and define the sequence of set of maximal instances $M_i = \{m_1, \ldots, m_i\}$ for $i \in \{1, \ldots, k\}$. The breath-first traversal ensures that each newly considered m_{i+1} is connected to some maximal instance in M_i by some non-maximal sub-instances. Let us show now that the successive values taken by P at line 13, numbered P_1, P_2, ..., P_k, are such that $P_i = \widetilde{T}_p(M_i)$. Using remark 4.2.1, it is enough to show that $P_1 = D(m_1)$ and $P_{i+1} = \mathrm{Colim}(S_{M_i, m_{i+1}})$.

For P_1, consider the first steps of the algorithm before the first execution of line 13. Call n_1 the value of n at line 1 and note that P is assigned to $D(n_1)$ at line 2 and $C(n_1)$ to $id_{D(n_1)} : D(n_1) \to D(n_1)$. At lines 4, 5 and 6, n is assigned to n_1, m to m_1 and e to the corresponding morphism from n_1 to m_1. Lines 7 and 8 lead S to be $\{(n_1, id_{D(n_1)} : D(n_1) \to D(n_1), D(e) : D(n_1) \to D(m_1))\}$. So the first execution of line 9 computes this simple pushout and sets (P', t, c) to $(D(m_1), D(e), id_{D(m_1)})$, so $P_1 = D(m_1) = \widetilde{T}_p(M_1)$ at line 13.

To establish that $P_{i+1} = \mathrm{Colim}(S_{M_i, m_{i+1}})$ in the $(i+1)$-th execution of line 13, we need to show that the parameters $(P, D(m), S)$ provided in $(i+1)$-th execution of line 9 correspond to the diagram $S_{M_i, m_{i+1}}$ given in (4.4). Firstly, by induction hypothesis, we have that $P = P_i = \widetilde{T}_p(M_i)$ and $m = m_{i+1}$. The collection of spans S computed at line 8 is correct because E' is the set of sub-instances of m (line 7), and N contains all sub-instances of the maximal instances in M_i that could have a morphism to m_{i+1}. Indeed, a non-maximal instance is discarded from N, E and C (lines 14–16) only after that all of its maximal super-instances have been processed (for loop at lines 5 to 13). Line 11 and 15 ensure that C always contain the correct morphism $D(n) \to P$ for all non-maximal instances n contained in N.

Finally, line 11 modifies C without updating the cocone compounds already stored in C, resulting in mixing morphisms with codomain P and P'. It is correct considering the following fact. For accretive global transformations, t (line 9) is always a monomorphism and can be designed for t to be a trivial inclusion. In that case, any morphism to P is also a morphism to P', the latter materially including P. In other words, everything is implemented to ensure that the modification on intermediate results are realized *in place*. □

4.5 Final Discussion

This chapter presented an online algorithm for computing the application of global transformations. To this end, we focused on global transformations of presheaves that preserve injective presheaves morphisms. We also focused on a particular class of global transformations such that, when the pattern matching is done in an online fashion, the local results can be aggregated by only adding elements to a growing global result. This led to the local characterization of such global transformations using the incremental criterion on rule systems.

The incremental criterion can be studied for itself. An alternative equivalent expression of definition 4.3.2 is stated as follows: given a super-rule, its r.h.s. contains the r.h.s. of its sub-rules as if they were considered independently. Intuitively, this prevents from non-local behavior like collapsing non-empty graphs to a single vertex since the empty graph remains empty for example as in figure 4.7b. From that point of view, incremental global transformations follow the research direction of causal graph dynamics [Arrighi et al., 2018]. In this work any produced element in the output is attached to an element of the input graph and a particular attention is put on preventing two rule instances to produce a common element.

At the algorithmic level, there remain many interesting considerations that need to be settled. One of them is that the way this algorithm goes from maximal instances to maximal instances using common sub-instances reminds of the strategy of the famous knuth-morris-pratt algorithm [Knuth et al., 1977]: in both cases, the content of one match is used to guide following subsequent pattern matching. This link is reinforced by the work of [Srinivas, 1993] that extend the knuth-morris-pratt algorithm to sheaves. In algorithm 1, we used pattern matching as a black-box, but opening it should permit to mix the outer maximal-to-maximal strategy with the knuth-morris-pratt considerations inside the pattern matching algorithm of [Srinivas, 1993]. Another important aspect is the complexity of this online approach and its natural extensions. Indeed, we described how a full input is decomposed in an online fashion, and the parts also treated online. The full picture includes the input itself being received by part, or even treated in a distributed way. Each of these versions deserve a careful study of their online complexity, *i.e.*, the complexity of the computation happening between each outputted data. We are also interested in the detailed study of the problem consisting of deciding, given a rule system, if it is a global transformation or not. Incremental rule systems form a particularly easy subclass for this problem, but we are talking here about the complete class of all rule systems.

Note that this work was originally restricted to global transformations of graphs, but the extension to any category of presheaves appears to be straightforward. Indeed, we recall that the classical category of graphs is equivalent to the category of presheaves from some simple finite category. We also gave some examples of other categories of presheaves describing other generalized graphs. In particular, this gave a characterization to the category of meshes, informally described in the introduction and in chapter 2, as the category of semi-simplicial sets. Categories of presheaves are a very convenient setting for global transformations, as we recall that they have similar properties than the category of sets. We recall a notable such property: these categories are cocomplete and the colimit can be computed element-wise using zig-zags between elements in the diagram. It is

also known that any category of presheaves form a Grothendieck topos (example A.2.1.3 in [Johnstone, 2002]). It is then natural to expect the extensions to other well-known classes of categories, such as arbitrary toposes or even adhesive categories [Lack and Sobociński, 2005, Ehrig et al., 2010] used in the field of graph grammar, any topos being adhesive [Lack and Sobociński, 2006].

Chapter 5

Non-Deterministic Global Transformations

Global Transformations are spacialized dynamical systems for which the synchronous application of the local rules at any place is decided deterministically. Still, it is necessary to talk about non-deterministic, stochastic or quantum systems for many practical and theoretical purpose, and notable such extensions of cellular automata and Lindenmayer systems were thoroughly studied in the literature.

A usual approach for modeling non-determinism from a deterministic system consists of generalizing the deterministic object into a non-deterministic one and showing that the original deterministic case is a degenerate case of the new non-deterministic setting. We rather seek to demonstrate that non-determinism is already encompassed as a particular case of the current definition of global transformations. This is in complete analogy with dynamical systems. Indeed, the general definition of dynamical systems is in terms of sets of states and evolution functions. Non-deterministic dynamical systems are *particular* dynamical systems where the sets happen to be powersets of an underlying set of real states.

It is not obvious at first that is possible, because global transformations are mainly about the notion of locality, but non-deterministic, probabilistic and quantum systems all exhibit the somewhat non-local features of correlations. The notion of correlations in non-deterministic global transformations are worth to study and we also seek to characterize rule systems that lead to these non-local effects.

In this chapter, we come back on the Lindenmayer systems of chapter 2, not as a pedagogical exercise but to explore non-determinism in global transformations in the simplest possible concrete setting. Formal definitions of the involved objects are introduced in section 5.1. This will allow us to show that a too naive approach of this quest, based on powersets, does not work as explored in section 5.2. In section 5.3, we circumvent the obstruction in the most natural way and go to an intuitive solution that does not look as a dynamical system though, because the transformation does not go from a set to itself but from a set to a bigger set. Section 5.4 comes back to the relation between dynamical system and non-deterministic system in terms of monad and Kleisli category. This well-known categorical point of view on dynamical systems leads us to consider a 2-monad and its Kleisli 2-category, linking together the previous attempts with this setting as a coherent whole, and providing a positive answer to our quest. Sections 5.5 studies some examples of non-deterministic global graph transformations and informally observe that correlations tends to appear when

a non-deterministic choice for a local rule cannot be decided independently of the choices of its neighboring rules. We finally express formally in section 5.6 that correlation-free global transformations are precisely those systems such that each non-deterministic choice can be decided independently. We finally observe that this property is connected to the notion of being a sheaf for some topology.

5.1 Preliminaries

The following notations are used along this chapter. Formal language operations are written as follows. For a given alphabet Σ, Σ^* is the set of finite words on Σ, and ε is the empty word. The length of a word $u \in \Sigma^*$ is written $|u|$ and its ith letter, for $0 \leq i < |u|$, is written u_i. The concatenation of two words $u, v \in \Sigma^*$ is written $u \cdot v$. Concatenation of sets $U, V \subseteq \Sigma^*$ is written $U \cdot V$ and is defined by $\{u \cdot v \mid u \in U, v \in V\}$. For a given set U, the powerset of U is written $\mathrm{P}(U)$. The cartesian product of a family of sets $\{U_i\}_{i \in I}$ is written $\prod_{i \in I} U_i$, and the projection on component i for an element x of that product is written $x(i)$.

Restrictions of categories and functors are needed in this chapter. The restriction of a category $\mathbf{C}$ to the full subcategory with objects $S \subset \mathbf{C}$ is written $\mathbf{C} \restriction S$. The restriction of a functor F to a subcategory $\mathbf{C}$ of its domain is written $F \restriction \mathbf{C}$ as well.

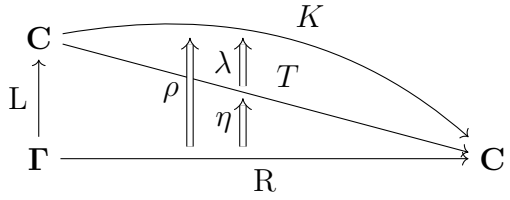
5.1.1 Global Transformations

Multiple definitions of global transformations have been considered so far. The initial and simplest definition of chapter 2 has been related to the notion of left Kan extension in chapter 3 and has been extended to export reconstruction in a suitable category in chapter 4.

In this chapter, we restrict ourselves to the definition of chapter 2 together with its Kan extension counterpart. Thus, we consider rule systems on a category $\mathbf{C}$ of the form $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$ with L a full embedding functor. The associated global transformation, when it is well-defined, is given by the functor:

$$T(-) = \mathrm{Colim}(\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/-]). \tag{5.1}$$

Alternatively, the global transformation T is a (pointwise) left Kan extensions of R along L, *i.e.*, a pair $\langle T : \mathbf{C} \to \mathbf{C}, \eta : \mathrm{R} \Longrightarrow T \circ \mathrm{L} \rangle$ such that any other pair $\langle K, \rho : \mathrm{R} \Longrightarrow K \circ \mathrm{L} \rangle$ factorizes through η by a unique $\lambda : T \Longrightarrow K$ as in the following diagram. The natural transformation η is the identity, *i.e.*, $T \circ \mathrm{L} = \mathrm{R}$.



5.1.2 Non-Deterministic Lindenmayer Systems

In this chapter we focus on *non-deterministic* Lindenmayer systems without context, where a word u on an alphabet Σ is synchronously rewritten by mapping each

individual letter to a set of words. Definition 2.2.1 of deterministic L-systems is then extended as follows.

Definition 5.1.1. A *non-deterministic Lindenmayer system* on an alphabet Σ is given by a function $\delta : \Sigma \to \mathrm{P}(\Sigma^*)$ and produces the function on words $\Delta : \Sigma^* \to \mathrm{P}(\Sigma^*)$:

$$\Delta(u) = \{v_0 \cdot \ldots \cdot v_{|u|-1} \mid (v_0, \ldots, v_{|u|-1}) \in \delta(u_0) \times \ldots \times \delta(u_{|u|-1})\} \tag{5.2}$$

and the dynamical system $\overline{\Delta} : \mathrm{P}(\Sigma^*) \to \mathrm{P}(\Sigma^*)$:

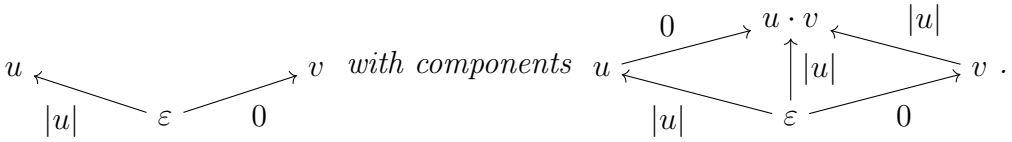
$$\overline{\Delta}(U) = \bigcup_{u \in U} \Delta(u). \tag{5.3}$$

Notice that the deterministic case is retrieved when function δ maps each letter to a singleton.

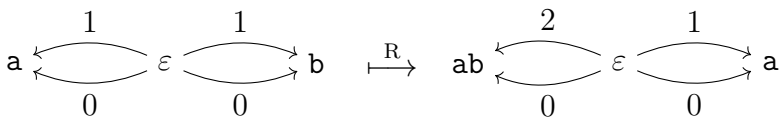
Example 5.1.1. Consider the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ with function δ defined by $\delta(\mathtt{a}) = \{\mathtt{a}, \mathtt{ab}\}$ and $\delta(\mathtt{b}) = \{\varepsilon, \mathtt{b}\}$. In this system, each $\mathtt{a}$ may potentially produce a new $\mathtt{b}$ on its right, and each $\mathtt{b}$ remains or vanishes. The behavior on the word $\mathtt{ab}$ is given by $\Delta(\mathtt{ab}) = \{\mathtt{a} \cdot \varepsilon, \mathtt{ab} \cdot \varepsilon, \mathtt{a} \cdot \mathtt{b}, \mathtt{ab} \cdot \mathtt{b}\} = \{\mathtt{a}, \mathtt{ab}, \mathtt{abb}\}$. Notice that $\mathtt{ab}$ is produced in two different ways.

Given some alphabet Σ, we consider the category $\mathbf{W}_{\Sigma^\star}$ of chapter 2 (definition 2.2.2) of finite words on Σ which plays a crucial role in this study. Let us fix some alphabet Σ so we drop the subscript $\Sigma^\star$. Recall that category $\mathbf{W}$ records the many places words appear in each other, which is the relevant notion of locality for L-systems, and that the concatenations involved in equation (5.2) correspond to colimits in $\mathbf{W}$:

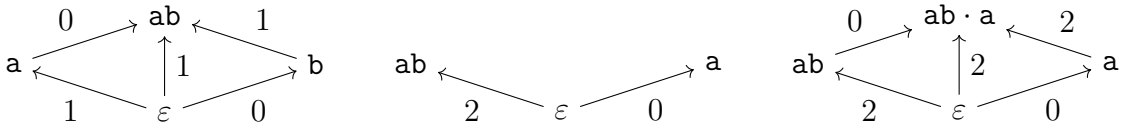
Theorem 5.1.1 (from chapter 2)**.** *For any two words u, v, the word $u \cdot v$ is the colimit in $\mathbf{W}$ of the diagram:*



Example 5.1.2. Let us illustrate the construction of chapter 2 with a simple example. Consider the deterministic Lindenmayer system defined on $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ by $\delta(\mathtt{a}) = \{\mathtt{ab}\}$ and $\delta(b) = \{\mathtt{a}\}$. The associated global transformation $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$ is completely determined by the following diagrams presenting respectively the category of rules $\mathbf{\Gamma}$ as a full subcategory of $\mathbf{W}$ with inclusion functor L, and the image of $\mathbf{\Gamma}$ by R:



The computation of $T(ab)$ as defined by Equation (5.1) is pictured by the following diagrams:

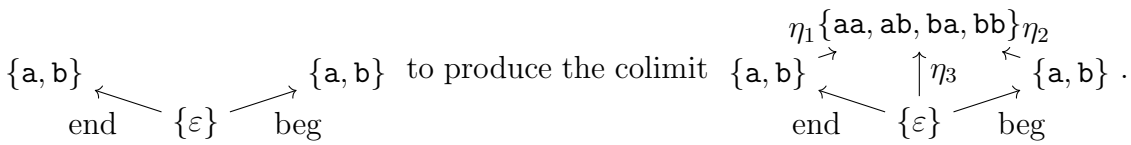


On the left, the diagram illustrates the information provided by the comma category L/ab. It corresponds to the pattern matching of the rules l.h.s. in the input word ab. In the middle, the diagram R ∘ Proj[L/ab] is represented. On the right, the application of theorem 5.1.1 for computing the colimit requested by $T(\mathtt{ab})$ constructs the expected result aba.

5.2 The Challenge of Powersets

Let us recall the goal and make it more precise in light of the formal definitions. We already know from chapter 2 that deterministic Lindenmayer systems are global transformations, and now we want to establish that non-deterministic Lindenmayer systems are also global transformations, without any extension of the framework. This means that we want to provide a rule system based on δ such that $\overline{\Delta}$ (definition 5.1.1) is obtained by the colimit formula of equation (5.1). This implies in particular that we need to design the appropriate category, say **PW**, with a calibrated notion of morphisms to capture what we can informally call *non-deterministic locality*. The first idea that comes to mind is to take $\mathrm{P}(\Sigma^*)$ as set of objects for **PW** so that an object represents a set of possibilities. It remains to define morphisms of **PW**. However, it will cause difficulties, as we are now going to see.

For concreteness, let us take the simple example of $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $\delta(\mathtt{a}) = \{\mathtt{a}, \mathtt{b}\}$ and $\delta(\mathtt{b}) = \{\varepsilon\}$. First, notice that $\Delta(\varepsilon) = \{\varepsilon\}$. Second, it produces on the input aa the behavior $\Delta(\mathtt{aa}) = \{\mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$ corresponding to the four possibilities combining the possible evolutions of each a. Using the lessons learned in the deterministic case as illustrated in example 5.1.2, we make the educated guess that the involved diagram should be of the form:



The diagram contains $\{\varepsilon\}$ for the matched intercalary ε between the two a in aa, and two times $\{\mathtt{a}, \mathtt{b}\}$ for each occurrence of a. The morphisms in the diagram indicate that the empty words at the end of the words in the left object need to correspond to the empty words at the beginning of the words in the right object, as in theorem 5.1.1. The expected colimit results in the concatenation $\{\mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$. Based on these assumptions, we know that:

- the morphism "end" of **PW** should be based on morphisms $1 : \varepsilon \to \mathtt{a}$ and $1 : \varepsilon \to \mathtt{b}$ of **W**,
- the morphism "beg" of **PW** should be based on morphisms $0 : \varepsilon \to \mathtt{a}$ and $0 : \varepsilon \to \mathtt{b}$ of **W**,
- the morphism η_1 should be based on $0 : \mathtt{a} \to \mathtt{aa}$, $0 : \mathtt{a} \to \mathtt{ab}$, $0 : \mathtt{b} \to \mathtt{ba}$, and $0 : \mathtt{b} \to \mathtt{bb}$,

- the morphism η_2 should be based on $1 : \mathtt{a} \to \mathtt{aa}$, $1 : \mathtt{b} \to \mathtt{ab}$, $1 : \mathtt{a} \to \mathtt{ba}$, and $1 : \mathtt{b} \to \mathtt{bb}$,
- the morphism η_3 should be based on $1 : \varepsilon \to \mathtt{aa}$, $1 : \varepsilon \to \mathtt{ab}$, $1 : \varepsilon \to \mathtt{ba}$, and $1 : \varepsilon \to \mathtt{bb}$.

A natural choice for designing the morphisms of **PW** is then to gather of these morphisms of **W** into sets of morphisms, leading to the following definition.

Definition 5.2.1. Let **PW** be the category having $\mathrm{P}(\Sigma^*)$ as set of objects, and

$$\mathrm{Hom}_{\mathbf{PW}}(U, V) := \mathrm{P}(\{\, p : u \to v \in \mathbf{W} \mid u \in U, v \in V \,\})$$

as set of morphisms from any $U \in \mathrm{P}(\Sigma^*)$ to any $V \in \mathrm{P}(\Sigma^*)$. As usual, we write $P : U \to V$ for $P \in \mathrm{Hom}_{\mathbf{PW}}(U, V)$. The composite $Q \circ P : U \to W$ of any two morphisms $P : U \to V$ and $Q : V \to W$ is given by $\{p + q : u \to w \mid p : u \to v \in P, q : v \to w \in Q\}$, $\{0 : u \to u \mid u \in U\}$ being the identity morphism of any $U \in \Sigma^*$.

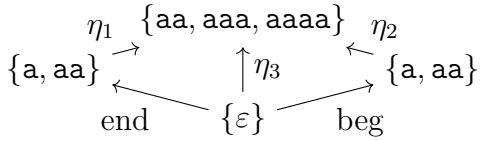
Notice that for any $P \in \mathrm{Hom}_{\mathbf{PW}}(U, V)$, P does not contain integers but morphisms of **W** with their domain and codomain. To avoid any confusion, we always write $p : u \to v \in P$ for elements of P.

Example 5.2.1. With this category, the previous example works as expected if we take our previous list of constraints to define end, beg, η_1, η_2, and η_3 as sets of morphisms: $\mathrm{end} = \{1 : \varepsilon \to \mathtt{a}, 1 : \varepsilon \to \mathtt{b}\}$, $\mathrm{beg} = \{0 : \varepsilon \to \mathtt{a}, 0 : \varepsilon \to \mathtt{b}\}$, $\eta_1 = \{0 : \mathtt{a} \to \mathtt{aa}, 0 : \mathtt{a} \to \mathtt{ab}, 0 : \mathtt{b} \to \mathtt{ba}, 0 : \mathtt{b} \to \mathtt{bb}\}$, and so on so forth. To see this, consider another cocone to some object $U \in \mathbf{PW}$ with components $\rho_1 : \{\mathtt{a}, \mathtt{b}\} \to U$, $\rho_2 : \{\mathtt{a}, \mathtt{b}\} \to U$, $\rho_3 : \{\varepsilon\} \to U$. We aim at showing that there is a unique morphism $\mu : \{\mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\} \to U$ such that $\rho_i = \mu \circ \eta_i$ for $i \in \{1, 2, 3\}$.

First, notice that ρ_1, ρ_2, and ρ_3 are bijectively related. Indeed, by definition of the composition in **PW** and by commutativity of the cocone, each $f_1 : l_1 \to u \in \rho_1$ compose with the unique appropriate morphism $1 : \varepsilon \to l_1$ of "end" to give a bijectively corresponding morphism in $f_1 + 1 : \varepsilon \to u \in \rho_3$. Surjectivity holds by the definition of the composition. Injectivity stands on the fact that an occurrence of $\mathtt{a}$ and an occurrence of $\mathtt{b}$ in a given word (here u) cannot be at the same position. Therefore, given $f_1 + 1 : \varepsilon \to u \in \rho_3$, there is a unique letter l_1 such that $f_1 : l_1 \to u \in \rho_1$. Similarly, there is also such a bijection mapping each $f_2 : l_2 \to u \in \rho_2$ to $f_2 : \varepsilon \to u \in \rho_3$. Consequently, the mediating μ has to contain exactly morphisms with domain the concatenation of source letters l_i of two bijectively related morphisms $f_1 : l_1 \to u \in \rho_1$ and $f_2 : l_2 \to u \in \rho_2$, and with codomain u. Formally μ has to be $\{f_1 : l_1 \cdot l_2 \to u \mid f_1 : l_1 \to u \in \rho_1$ and corresponding $f_1 + 1 : l_2 \to u \in \rho_2\}$. The fact that this mediating commutes appropriately comes directly from the fact that $\mu \circ \eta_1 = \{f_1 : l_1 \to u \mid 0 : l_1 \to l_1 \cdot l_2 \in \eta_1, f_1 : l_1 \cdot l_2 \to u \in \mu\}$ which turns to be ρ_1. Commutations for ρ_2 and ρ_3 hold as well. Finally, μ is unique by construction.

While the previous example works, the proof uses explicitly properties specific to the example. In the general case, the construction fails as shown by the following example.

Example 5.2.2. We consider a different non-deterministic Lindenmayer system where $\Sigma = \{\mathtt{a}\}$ and $\delta(\mathtt{a}) = \{\mathtt{a}, \mathtt{aa}\}$. This produces the behavior $\Delta(\mathtt{aa}) = \{\mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}\}$. Notice in particular that the concatenations $\mathtt{a} \cdot \mathtt{aa}$ and $\mathtt{aa} \cdot \mathtt{a}$ give the same word $\mathtt{aaa}$ in $\Delta(\mathtt{aa})$. Following the same construction as presented in example 5.2.1, we consider the following colimit:



where

- $\eta_1 = \{0 : \mathtt{a} \to \mathtt{aa}, 0 : \mathtt{a} \to \mathtt{aaa}, 0 : \mathtt{aa} \to \mathtt{aaa}, 0 : \mathtt{aa} \to \mathtt{aaaa}\}$,
- $\eta_2 = \{1 : \mathtt{a} \to \mathtt{aa}, 1 : \mathtt{aa} \to \mathtt{aaa}, 2 : \mathtt{a} \to \mathtt{aaa}, 2 : \mathtt{aa} \to \mathtt{aaaa}\}$, and
- $\eta_3 = \{1 : \varepsilon \to \mathtt{aa}, 1 : \varepsilon \to \mathtt{aaa}, 2 : \varepsilon \to \mathtt{aaa}, 2 : \varepsilon \to \mathtt{aaaa}\}$.

To see that this cocone is *not* a colimit, consider the cocone to $\{\mathtt{aaa}\}$ having components $\rho_1 = \{0 : \mathtt{a} \to \mathtt{aaa}\}$, $\rho_2 = \{1 : \mathtt{aa} \to \mathtt{aaa}\}$, and $\rho_3 = \{1 : \varepsilon \to \mathtt{aaa}\}$. The only possible mediating is $\mu = \{0 : \mathtt{aaa} \to \mathtt{aaa}\}$ but it fails to respect the required commutation property. Indeed, $\mu \circ \eta_1 = \{0 : \mathtt{a} \to \mathtt{aaa}, 0 : \mathtt{aa} \to \mathtt{aaa}\}$ which definitively differs from ρ_1.

So the first and simplest intuitive idea does not work, and we do not have designed the appropriate category. In particular, the construction fails since it is not able to distinguish the different ways for generating a given output (case `aaa` in example 5.2.2). We then deduce that an appropriate category, if it exists, needs to keep track of this information. Moreover, notice that in equation (5.1), the morphisms of the category are not only used for constructing the result as a colimit, but also to decompose an input U into a comma category L/U and produce the diagram by the formula $\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/U]$. So the previous discussion has only scratched one side of the problem.

5.3 From Sets to Indexed Families

There are plenty of other possible definitions for designing morphisms of **PW** and the research space to get the right one is pretty large. However, having considered several attempts of definitions, we come to the following working hypothesis.

Conjecture 5.3.1. There is no category with set of objects $\mathrm{P}(\Sigma^*)$ and an appropriate choice of morphisms $\mathrm{beg}_U, \mathrm{end}_U : \{\varepsilon\} \to U$ producing concatenation as a colimit.

Irrespectively of the validity of that conjecture, we choose to circumvent directly the problem that occurred in the previous example and to jump to other aspects of the program. As previously evoked, the obstruction was that $\mathtt{a} \cdot \mathtt{aa}$ and $\mathtt{aa} \cdot \mathtt{a}$ merged into a single word `aaa`, so that the mediating morphism could not specify whether it needed this word as a result of the first concatenation or of the second one. To keep track of that information, we simply allow for a word to appear many times, and we take families of words instead of sets of words. Taking this path of least resistance, we obtain the following category where a morphism from a source family of words to a target family of words, picks a unique word from the source for each word in the target. This additional "unique source word" property is respected by all the morphisms considered up to now. Taking the view that a morphism $p : u \to v$ in **W** selects a subword of v, this constraint says that a morphism to a family of words similarly selects a subword for each member of that family.

Definition 5.3.1. For any $\mathbf{C} \in \mathbf{Cat}$, $\mathbb{O}\mathbf{C} \in \mathbf{Cat}$ has pairs $(I : \mathrm{Set}, U \in \mathbf{C}^I)$ as objects and

$$\mathbb{O}\mathbf{C}((I,U),(J,V)) := \{\, (P : J \to I, P' : \prod_{j \in J} \mathrm{Hom}_{\mathbf{C}}(U_{P(j)}, V_j)) \,\}$$

as set of morphisms from any $(I,U) \in \mathbb{O}\mathbf{C}$ to any $(J,V) \in \mathbb{O}\mathbf{C}$. As usual, we write $(P,P') : (I,U) \to (J,V)$ for $(P,P') \in \mathrm{Hom}_{\mathbb{O}\mathbf{C}}((I,U),(J,V))$. The composite $(Q,Q') \circ (P,P') : (I,U) \to (K,W)$ of any two morphisms $(P,P') : (I,U) \to (J,V)$ and $(Q,Q') : (J,V) \to (K,W)$ is given by the pair $(R : K \to I, R' : \prod_{k \in K} \mathbf{C}(U_{R(k)}, W_k))$ where :

$$R(k) = P(Q(k)) \text{ and } R'(k) = Q'(k) \circ P'(Q(k)) : U_{P(Q(k))} \to V_{Q(k)} \to W_k \in \mathbf{C}.$$

The identity morphism of any (I,U) is (P,P') where $P(i) = i$ and $P'(i) = \mathrm{id}_{P(i)} : P(i) \to P(i)$.

One may have recognized in this definition the construction of non-determinism of [Lehmann, 1976]. It happens to be the free cartesian completion of $\mathbf{C}$, the dual of $\mathrm{Fam}(\mathbf{C})$ construction for free coproduct completion [Hu and Tholen, 1995], i.e. $\mathbb{O}\mathbf{C} = \mathrm{Fam}(\mathbf{C}^{\mathrm{op}})^{\mathrm{op}}$.

It is not hard to see that each object of this category has many isomorphic objects of bijective index set, so that the particular index set used is irrelevant. The real information contained in an equivalence class of isomorphic objects is the number of times each word occurs. Here, we allow this cardinality to be arbitrary. The issue of cardinality is a detail at this point, and we do not bother commenting on this issue before the conclusion. In the meantime, one can freely add the word *finite* anywhere one feels it is needed.

At this time, some notations are required to handle families and some relevant elements of $\mathbb{O}\mathbf{W}$. Given an arbitrary set U, we write $\overline{U}$ for the corresponding family containing each element of U exactly once and given by the pair (U, id_U). Also, for any $(I,U) \in \mathbb{O}\mathbf{W}$, we consider the appropriate morphisms $\mathrm{beg}_{(I,U)}, \mathrm{end}_{(I,U)} : \overline{\{\varepsilon\}} \to (I,U)$ identifying the occurrences of the empty words respectively at the beginning and at the end of the words of the family (I,U), and which are given by $\mathrm{beg}_{(I,U)} = ([i \mapsto \varepsilon], [i \mapsto (0 : \varepsilon \to U_i)])$ and $\mathrm{end}_{(I,U)} = ([i \mapsto \varepsilon], [i \mapsto (|U_i| : \varepsilon \to U_i)])$. We make use here of the notation $[x \mapsto f(x)]$ to specify succinctly an anonymous function; domains and codomains can always be retrieved from the context.

With the category $\mathbb{O}\mathbf{W}$ and these beginning and ending morphisms, we obtain concatenation as wanted and in a very similar way to concatenation in $\mathbf{W}$ with theorem 5.1.1.

Proposition 5.3.1. *For any two families $(I,U),(J,V) \in \mathbb{O}\mathbf{W}$, a colimit of the diagram*

$$(I,U) \xleftarrow{\mathrm{end}_{(I,U)}} \overline{\{\varepsilon\}} \xrightarrow{\mathrm{beg}_{(I,U)}} (J,V)$$

is given by the cocone

$$\begin{array}{ccccc} & & (I \times J, (i,j) \mapsto U_i \cdot V_j) & & \\ & \nearrow^{\eta_1} & \uparrow{\scriptstyle \eta_3} & \nwarrow^{\eta_2} & \\ (I,U) & \xleftarrow{\mathrm{end}_{(I,U)}} & \overline{\{\varepsilon\}} & \xrightarrow{\mathrm{beg}_{(I,U)}} & (J,V) \end{array}$$

where

- $\eta_1 = ([(i,j) \mapsto i], [(i,j) \mapsto (0 : U_i \to U_i \cdot V_j)])$,
- $\eta_2 = ([(i,j) \mapsto j], [(i,j) \mapsto (|U_i| : V_j \to U_i \cdot V_j)])$, *and*
- $\eta_3 = ([(i,j) \mapsto \varepsilon], [(i,j) \mapsto (|U_i| : \varepsilon \to U_i \cdot V_j)])$.

Proof. Indeed, consider another cocone to some (K, W) with components, $(P, P') : (I, U) \to (K, W)$, $(Q, Q') : (J, V) \to (K, W)$, and $(R, R') : \overline{\{\varepsilon\}} \to (K, W)$. The mediating morphism is given by $([k \mapsto (P(k), Q(k))], [k \mapsto P'(k) : U_{P(k)} \cdot V_{Q(k)} \to W_k])$. Notice that this mediating follows the exact same definition as the one exhibited in example 5.2.1. □

It is now time to summarize what we have just achieved. We took a diagram shape that allowed to obtain *deterministic* Lindenmayer systems as a global transformation by encoding concatenation as colimit. Then, we changed the objects and morphisms in this diagram to obtain what we can informally call *non-deterministic concatenation*. But the diagram shape itself arises from the *deterministic* decomposition of a single input word (cf. chapter 2). In other words, the pattern matching is not considered in $\mathbb{O}\mathbf{W}$ but in $\mathbf{W}$. So for now, we only have the following.

Definition 5.3.2. For any non-deterministic Lindenmayer system $(\Sigma, \delta : \Sigma \to \mathrm{P}(\Sigma^*))$, we write $\mathbf{D} : \mathbf{W} \to \mathbb{O}\mathbf{W}$ for the functor mapping words $u \in \mathbf{W}$ to $\mathbf{D}(u) = (I, V)$ where

$$I := \delta(u_0) \times \ldots \times \delta(u_{|u|-1}) \text{ and } V_{(v_0,\ldots,v_{|u|-1})} := v_0 \cdot \ldots \cdot v_{|u|-1}$$

and morphisms $p : u' \to u$ to $\mathbf{D}(p) = (P, P')$ where

$$P((v_0, \ldots, v_{|u|-1})) := (v_p, \ldots, v_{p+|u'|-1}) \text{ and}$$

$$P'((v_0, \ldots, v_{|u|-1})) := |v_0| + \ldots + |v_{p-1}| : v_p \cdot \ldots \cdot v_{p+|u'|-1} \to v_0 \cdot \ldots \cdot v_{|u|-1}.$$

The functor $\mathbf{D}$ is a categorical counterpart of $\Delta : \Sigma^* \to \mathrm{P}(\Sigma^*)$ of definition 5.1.1 but with families making sure that we keep the multiple instances of each word. Indeed, for $\mathbf{D}(u) = (I, U)$, each index $(v_0, \ldots, v_{|u|-1}) \in I$ corresponds to a choice of a word v_i among the possibilities provided by $\delta(u_i)$, for each letter u_i of u. For such a choice, the associated resulting word $V_{(v_0,\ldots,v_{|u|-1})}$ is simply given by the concatenation of the v_i. The definition of $\mathbf{D}(p)$ expresses the monotony of Δ. The monotony can be illustrated as follows. Consider $u = \alpha_1 \cdot u' \cdot \alpha_2$ with $|\alpha_1| = p$. Taking $v' \in \Delta(u')$, $\gamma_1 \in \Delta(\alpha_1)$, and $\gamma_2 \in \Delta(\alpha_2)$, we have $v = \gamma_1 \cdot v' \cdot \gamma_2 \in \Delta(u)$. So we have a morphism $|\gamma_1| : v' \to v$. As a family of morphisms, $\mathbf{D}(p)$ gathers all of these morphisms. In the formula of definition 5.3.2, we have $\alpha_1 = u_0 \ldots u_{p-1}$, $u' = u_p \ldots u_{p+|u'|-1}$, $\gamma_1 = v_0 \cdot \ldots \cdot v_{p-1}$, and $v' = v_p \cdot \ldots \cdot v_{p+|u'|-1}$.

Exactly as Δ is generated from its sole behavior on letters given by δ as stated by definition 5.1.1, we will see that the functor $\mathbf{D}$ is generated from its restriction to the letters and ε. We start by defining the categorical counterpart $\mathbf{d}$ of δ.

Definition 5.3.3. For any non-deterministic Lindenmayer system $(\Sigma, \delta : \Sigma \to \mathrm{P}\Sigma^*)$, we write $\mathbf{d} : \mathbf{W} \restriction \Sigma \cup \{\varepsilon\} \to \mathbb{O}\mathbf{W}$ for the functor from the full subcategory $\mathbf{W} \restriction \Sigma \cup \{\varepsilon\}$ of $\mathbf{W}$ to $\mathbb{O}\mathbf{W}$ defined as $\mathbf{d} = \mathbf{D} \restriction (\mathbf{W} \restriction \Sigma \cup \{\varepsilon\})$.

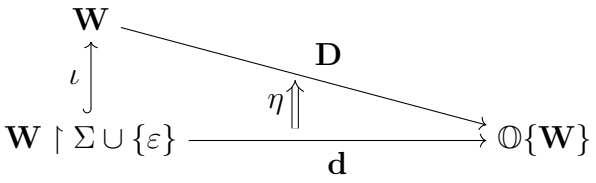
The functor $\mathbf{d}$ is entirely characterized in terms of morphisms beg and end.

Lemma 5.3.2. *For any $a \in \Sigma$, we have $\mathbf{d}(0 : \varepsilon \to a) = \mathrm{beg}_{\mathbf{d}(a)}$ and $\mathbf{d}(1 : \varepsilon \to a) = \mathrm{end}_{\mathbf{d}(a)}$.*

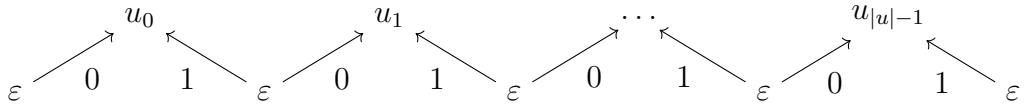
Proof. Consider $0 : \varepsilon \to a$. By defs. 5.3.3 and 5.3.2, $\mathbf{d}(\varepsilon) = \mathbf{D}(\varepsilon) = (\{\varepsilon\}, [\varepsilon \mapsto \varepsilon]) = \overline{\{\varepsilon\}}$, and $\mathbf{d}(a) = \mathbf{D}(a) = (\delta(a), [i \mapsto i])$. By the same definitions, $\mathbf{d}(0 : \varepsilon \to a) = \mathbf{D}(0 : \varepsilon \to a) = (P, P')$ with $P(i) = \varepsilon$ and $P'(i) = |\varepsilon| : \varepsilon \to i$ where i ranges over $\delta(a)$. Clearly, $\mathbf{d}(0 : \varepsilon \to a) = \mathrm{beg}_{(\delta(a),[i\mapsto i])} = \mathrm{beg}_{\mathbf{d}(a)}$ as expected. We get $\mathbf{d}(1 : \varepsilon \to a) = \mathrm{end}_{\mathbf{d}(a)}$ similarly. □

We can now establish that $\mathbf{D}$ is obtained as an extension of $\mathbf{d}$ thereby providing a categorical counterpart of definition 5.1.1.

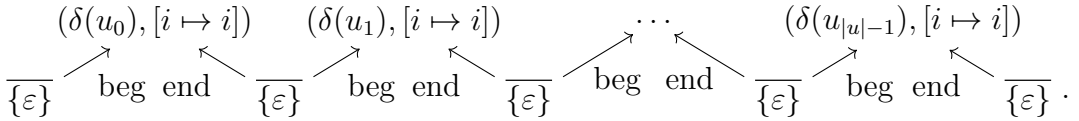
Proposition 5.3.3. $\mathbf{D}$ *is a pointwise left Kan extension of* $\mathbf{d}$ *along the inclusion functor $\iota : \mathbf{W} \restriction \Sigma \cup \{\varepsilon\} \to \mathbf{W}$ as in the following diagram where η is the identity.*



Proof. Using the explicit definition of pointwise left Kan extensions in terms of colimit, we are left to show that $\mathbf{D}(-) = \mathrm{Colim}(\mathbf{d} \circ \mathrm{Proj}[\iota/-])$. In chapter 2, we already proved that the diagram $\mathrm{Proj}[\iota/u]$ has the following zigzag shape (theorem 2.2.2):



Using lemma 5.3.2, the diagram $\mathbf{d} \circ \mathrm{Proj}[\iota/u]$ is:



Using iteratively proposition 5.3.1 on this finite sequence, the colimit of this diagram is clearly the non-deterministic concatenation of the $(\delta(u_k), [i \mapsto i])$, $0 \leq k < |u|$, which is also the definition of $\mathbf{D}(u)$ as given in definition 5.3.2. To prove that η is the identity, it is enough to consider the particular case of the previous reasoning with $|u| \leq 1$ that shows that $(\mathbf{D} \circ \iota)(u) = \mathbf{d}(u)$.

Given some $p : u' \to u$, for proving that $\mathbf{D}(p) = \mathrm{Colim}(\mathbf{d} \circ \mathrm{Proj}[\iota/p])$ we remark that $\mathbf{D}(p)$ has to be a mediating morphism. By uniqueness of the mediating morphism, it remains to show that $\mathbf{D}(p)$ obeys to the requested commutations of mediating morphisms, which is straightforward. □

So far, for a non-deterministic Lindenmayer system (Σ, δ), we have $\mathbf{d}$ as a categorical counterpart of δ, which gives rise by left Kan extension to $\mathbf{D}$, the categorical counterpart of Δ. However, we still do not have a dynamical system, since the domain and codomain of $\mathbf{D} : \mathbf{W} \to \mathbb{O}\mathbf{W}$ are not strictly the same. In other words, we now want a left Kan extension counterpart of $\overline{\Delta} : \mathrm{P}(\Sigma^*) \to \mathrm{P}(\Sigma^*)$ of definition 5.1.1, say $\overline{\mathbf{D}} : \mathbb{O}\mathbf{W} \to \mathbb{O}\mathbf{W}$. Clearly, we already know the expected definition of $\overline{\mathbf{D}}$ since we want to apply independently $\mathbf{D}$ on each element of a family (I, U) and to flatten the results altogether.

Definition 5.3.4. Let $\overline{\mathbf{D}} : \mathbb{O}\mathbf{W} \to \mathbb{O}\mathbf{W}$ be the functor defined as

$$\overline{\mathbf{D}}((I,U)) = \left(\bigcup_{i \in I} (\{i\} \times J_i) \,,\, [(i,j) \mapsto (V_i)_j] \right) \text{ where } (J_i, V_i) = \mathbf{D}(U_i),$$

and $\overline{\mathbf{D}}((P,P') : (I',U') \to (I,U)) = (Q,Q')$ such that, for each $(i,j) \in \bigcup_{i \in I} (\{i\} \times J_i)$:

$$Q((i,j)) = (P(i), R_i(j)) \text{ and } Q'((i,j)) = R'_i(j) \text{ where } (R_i, R'_i) = \mathbf{D}(P'(i)).$$

Obtaining $\overline{\mathbf{D}}$ as a Kan extension consists in embedding $\mathbf{W}$ into $\mathbb{O}\mathbf{W}$, then extending along this embedding. The notation $\overline{U}$ that we have introduced earlier can be turned into a *singleton functor* for defining this embedding.

Definition 5.3.5. The singleton functor $\mathrm{sing}_{\mathbf{W}} : \mathbf{W} \to \mathbb{O}\mathbf{W}$ is defined as

$$\mathrm{sing}_{\mathbf{W}}(u) = \overline{\{u\}} \text{ and } \mathrm{sing}_{\mathbf{W}}(p : u \to v) = ([v \mapsto u], [v \mapsto p : u \to v]).$$

Unfortunately, the program stops here since $\overline{\mathbf{D}}$ fails to be the extension of $\mathbf{d}$ along $\mathrm{sing}_{\mathbf{W}} \circ \iota : \mathbf{W} \restriction \Sigma \cup \{\varepsilon\} \to \mathbf{W} \hookrightarrow \mathbb{O}\mathbf{W}$. In fact, the morphisms of $\mathbb{O}\mathbf{W}$ are not well-suited for decomposing a family of words in the appropriate way for the expected concatenation. For instance, consider the family in inputs $\overline{\{\mathtt{aa}, \mathtt{bb}\}}$. The comma category $\mathrm{sing}_{\mathbf{W}} \circ \iota / \overline{\{\mathtt{aa}, \mathtt{bb}\}}$ fails to identify the occurrences of $\mathtt{a}$ in this family. Indeed, a morphism from $\overline{\{\mathtt{a}\}}$ to $\overline{\{\mathtt{aa}, \mathtt{bb}\}}$ requires to identify an occurrence of $\mathtt{a}$ in $\mathtt{bb}$ but there is none. So there is no morphism between those two families and the diagram $\mathbf{d} \circ \mathrm{Proj}[\mathrm{sing}_{\mathbf{W}} \circ \iota / \overline{\{\mathtt{a}, \mathtt{b}\}}]$ of equation (5.1) contains only ε's and does not exhibit the expected zigzag shape.

5.4 The Kleisli 2-Category of the 2-Monad of Families

As explained in the introduction of the chapter, we propose to solve the last issue by placing oneself in a 2-categorical context. Since this solution can seem more elaborated than necessary, let us make precise why this transition to 2-categories is conceptually natural with respect to our goal.

Let us develop the relation between dynamical systems and their non-deterministic counterparts. At a general level of description, dynamical systems can be defined once we have a collection of objects to model the states, and a way to specify endo-functions on these objects to model the dynamics. For instance, in the category of sets and functions, the states are modeled as a set and the dynamics as a function; the usual case of (deterministic) dynamical systems is captured. But in the category of topological spaces and continuous functions, states are modeled as a topological space and the dynamics by a continuous function, allowing to handle the so-called topological dynamical systems. Similarly, in the category of sets and relations, states are modeled as a set and the dynamics by a relation. This last case is particularly interesting for our concern since it is the place to deal with non-deterministic dynamical systems. Formally this latter category is equivalently described as the *Kleisli category* of the so-called *powerset monad*. This is based on the fact that $R \subseteq X \times Y$ is equivalently a function $f : X \to \mathrm{P}(Y)$, that singletons

allow any set X to be seen as included in $\mathrm{P}(X)$, and that unions allow any sets of sets in $\mathrm{P}(\mathrm{P}(X))$ to be simplified in a simple set of $\mathrm{P}(X)$. The two lessons we learn here are that (1) dynamical systems are parametrized by the nature of the objects and the morphisms they rely on, and that (2) the parametrization for the non-determinitic counterpart is based on the powerset monad.

We now proceed to apply the same scheme for the global transformation. The difference with dynamical systems is that global transformations are not defined with two layers (an object for the states and a morphism for the dynamics) but with three layers: categories, functors and natural transformations as it can be seen in the pointwise left Kan extension diagram of section 5.1.1. So they are parametrized by a 2-category. For instance, the global transformations considered so far are parametrized by **Cat**, the 2-category of categories. Following the second lesson on non-deterministic dynamical systems, for the particular case of non-deterministic global transformations, we propose this 2-category parameter to be set to the Kleisli 2-category induced by the 2-monad of families.

We already have all the ingredients of a 2-monad on **Cat** as we now proceed to show. Firstly, the construction that we applied on the category $\mathbf{W}$ to obtain $\mathbb{O}\mathbf{W}$ can actually be applied to any category, and be extended to act on functors and natural transformations, and yields a 2-functor $\mathbb{O} : \mathbf{Cat} \to \mathbf{Cat}$ for which we already have the behavior on objects by definition 5.3.1.

Definition 5.4.1. For any functor $F : \mathbf{C} \to \mathbf{C}'$, the functor $\mathbb{O}F : \mathbb{O}\mathbf{C} \to \mathbb{O}\mathbf{C}'$ is defined as $\mathbb{O}F((I, U)) = (I, F \circ U)$, and $\mathbb{O}F((P, P') : (I, U) \to (J, V)) = (P, F \circ P')$.

Definition 5.4.2. For any natural transformation $\alpha : F \Longrightarrow G : \mathbf{C} \to \mathbf{C}'$, the natural transformation $\mathbb{O}\alpha : \mathbb{O}F \Longrightarrow \mathbb{O}G : \mathbb{O}\mathbf{C} \to \mathbb{O}\mathbf{C}'$ has components $(\mathbb{O}\alpha)_{(I,U)} = ([i \mapsto i], [i \mapsto \alpha_{U_i}]) : (I, F \circ U) \to (I, G \circ U)$.

To make the 2-functor $\mathbb{O}$ into a 2-monad, we need an additional structure composed of two operations. The first operation lifts an object of a category $\mathbf{C}$ to a singleton family of $\mathbb{O}\mathbf{C}$, like $\mathrm{sing}_{\mathbf{W}}$ for $\mathbf{W}$.

Definition 5.4.3. For any category $\mathbf{C}$, the singleton functor $\mathrm{sing}_{\mathbf{C}} : \mathbf{C} \to \mathbb{O}\mathbf{C}$ is defined as

$$\mathrm{sing}_{\mathbf{C}}(x) = \overline{\{x\}} \text{ and } \mathrm{sing}_{\mathbf{C}}(f : x \to y) = ([y \to x], [y \mapsto (f : x \to y)]).$$

The second operation flattens a family of families of objects into a simple family of objects.

Definition 5.4.4. For any category $\mathbf{C}$, let $\mu_{\mathbf{C}} : \mathbb{O}\mathbb{O}\mathbf{C} \to \mathbb{O}\mathbf{C}$ be the functor defined as $\mu_{\mathbf{C}}((I, U)) = (I', U')$ where,

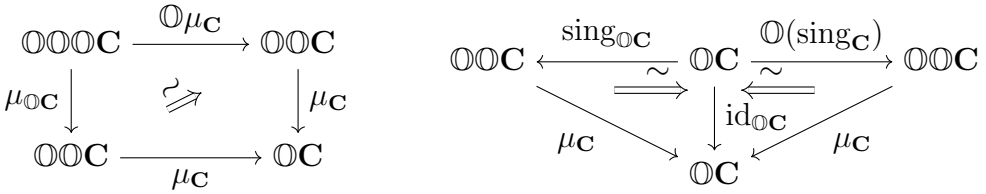
$$\mu_{\mathbf{C}}((I, U)) = \left(\bigcup_{i \in I} (\{i\} \times J_i), [(i, j) \mapsto (V_i)_j] \right) \text{ with } U_i = (J_i, V_i),$$

and $\mu_{\mathbf{C}}((P, P') : (I', U') \to (I, U)) = (Q, Q')$ such that, for each $(i, j) \in \bigcup_{i \in I} (\{i\} \times J_i)$, we have:

$$Q((i, j)) = (P(i), R_i(j)) \text{ and } Q'((i, j)) = R'_i(j) \text{ with } P'(i) = (R_i, R'_i).$$

Notice that this exact construction has already been encountered in definition 5.3.4 of $\overline{\mathbf{D}}$, whose role of flattening has also been underlined above. In particular, the function $\overline{\mathbf{D}}$ is in fact obtained as $\mu_{\mathbf{W}} \circ \mathbb{O}\mathbf{D}$ from definition 5.3.2. The two operations form indeed a 2-monad.

Proposition 5.4.1. *Operations* $\mathrm{sing}_{\_}$ *and* $\mu_{\_}$ *make* $\mathbb{O} : \mathbf{Cat} \to \mathbf{Cat}$ *into a 2-monad, i.e., all instances of the following diagrams weakly commute.*



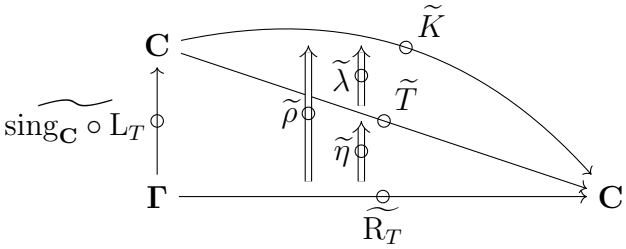
Proof. For the first square, and given an object $(I, U) \in \mathbb{O}\mathbb{O}\mathbb{O}\mathbf{C}$, the top-right path leads to index set of the form $(i, (j, k))$ for the object in $\mathbb{O}\mathbf{C}$ while the left-bottom path leads to the form $((i, j), k)$, hence the weak commutation. For the triangles on the right, an object $(I, U) \in \mathbb{O}\mathbf{C}$ sees each index $i \in I$ transformed into $((I, U), i) \in \{(I, U)\} \times I$ by the left path and into $(i, U_i) \in \{(i, U_i) \mid i \in I\}$ by the right path. □

In order to ease the reading of elements of the Kleisli weak 2-category, let us introduce some notations. We write $\widetilde{F} : \mathbf{C} \nrightarrow \mathbf{D}$ to represent a functor $F : \mathbf{C} \to \mathbb{O}\mathbf{D}$ of the Kleisli weak 2-category. A 2-arrow $\widetilde{\eta} : \widetilde{F} \Rightarrow \widetilde{G}$ stands simply for a natural transformation $\eta : F \Longrightarrow G$. The composition of morphisms in the Kleisli weak 2-category, written $\widetilde{G} \circ \widetilde{F} : \mathbf{C} \nrightarrow \mathbf{D} \nrightarrow \mathbf{E}$, is the functor $\mu_{\mathbf{E}} \circ \mathbb{O}G \circ F : \mathbf{C} \to \mathbb{O}\mathbf{D} \to \mathbb{O}\mathbb{O}\mathbf{E} \to \mathbb{O}\mathbf{E}$.

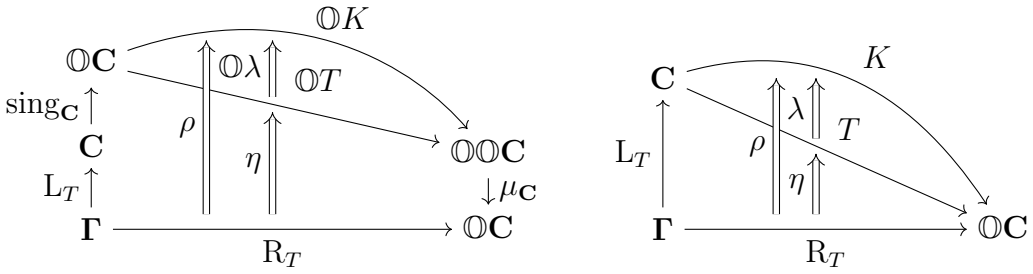
We finally reach our initial goal, as we are now able to show that the diagram of proposition 5.3.3 is in fact a summary of a global transformation in the Kleisli weak 2-category induced by the 2-monad $\mathbb{O}$. More accurately, considering the global transformation diagram of the rule system $\langle \mathbf{W} \upharpoonright \Sigma \cup \{\varepsilon\}, \widetilde{\mathrm{sing}_{\mathbf{W}} \circ \iota}, \widetilde{\mathbf{d}} \rangle$ is completely equivalent to considering the diagram of proposition 5.3.3, achieving the fact that the initial non-determinitic Lindenmayer system is indeed a global transformation. Moreover, this works for any non-deterministic rule system $\langle \mathbf{\Gamma}, \widetilde{\mathrm{sing}_{\mathbf{C}} \circ \mathrm{L}}, \widetilde{\mathrm{R}} \rangle$ on any category $\mathbf{C}$. Notice the particular form of the l.h.s. functor defined using, $\mathrm{L} : \mathbf{\Gamma} \to \mathbf{C}$ which is still required to be a full embedding.

Theorem 5.4.2. *Let* $\widetilde{T} = \langle \mathbf{\Gamma}_T, \widetilde{\mathrm{sing}_{\mathbf{C}} \circ \mathrm{L}_T}, \widetilde{\mathrm{R}_T} \rangle$ *be a rule system in the Kleisli weak 2-category induced by the 2-monad* $\mathbb{O}$*.* $\widetilde{T}$ *is a global transformation* iff T *is the left Kan extension of* R_T *along* L_T *in the 2-category* **Cat**.

Proof. The rule system being a global transformation, we have a pair $\langle \widetilde{T}, \widetilde{\eta} : \widetilde{\mathrm{R}_T} \Longrightarrow \widetilde{T} \circ \widetilde{\mathrm{sing}_{\mathbf{C}} \circ \mathrm{L}_T} \rangle$ which is a left Kan extension in the Kleisli 2-category and takes the following diagrammatic form for any other pair $\langle \widetilde{K}, \widetilde{\rho} : \widetilde{\mathrm{R}_T} \Longrightarrow \widetilde{K} \circ \widetilde{\mathrm{sing}_{\mathbf{C}} \circ \mathrm{L}_T} \rangle$:



This diagram corresponds by definition to the diagram in **Cat** on the left below. By naturality of sing and proposition 5.4.1, it (weakly) simplifies into the expected diagram, on the right below.



□

5.5 Non-Deterministic Global Transformations of Graphs

We have shown that working in another 2-category allows us to consider non-deterministic rule systems while keeping the usual presentation of global transformations as left Kan extensions. In this section, we develop a more involved study case. The chosen example illustrates the genericity of the proposition with the application of the 2-monad $\mathbb{O}$ on a category of graphs. However, since it is not be useful to be as formal as in the previous sections, the presentation is only described graphically. We end with a discussion about the interactions between non-determinism and locality.

5.5.1 Random Generation of a River

An example of local non-deterministic process can be found in [Peyrière, 1981] and [Prusinkiewicz and Hammel, 1993] for computing a squig curve. It is presented as a random rewriting process that aims at generating the shape of a river by subdividing a triangular mesh. The coding of the river in the mesh is done as follows. The edges of the triangles are labelled by a binary information specifying if the river goes through the edge or not. We will speak about *neutral* edges and *river* edges. A triangle with three neutral edges, say a *neutral* triangle, is not traversed by the river. Conversely, if a triangle has two river edges, it is a *river* triangle, the river going from one of the river edge to the other (the direction of the stream is not considered). We suppose that the river does not have any branching or merging. Consequently, the system is restricted to triangular meshes with those two types of triangles. In other words, triangles with one or three river edges are forbidden.

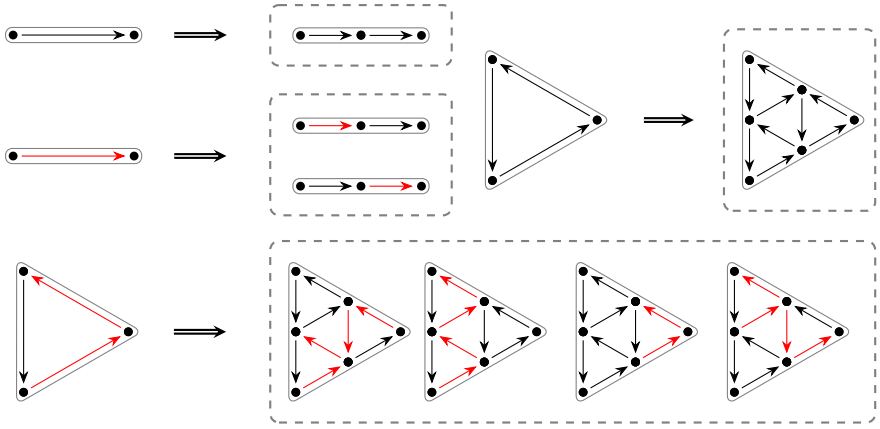


Figure 5.1: Rules for normal edges/triangles and river edges/triangles.

The rewriting process works as the refinement of triangular meshes given in the introduction. Non-deterministic features stand in the managing of the labels. Starting from a river triangle, the triangle is refined. The river yard is refined as well by deciding randomly by which part of the refined boundary the river goes. The four possibilities are illustrated at the bottom row of figure 5.1 (river edges are pictured in red).

In this section, we revisit this example in the context of global transformations. This computation can be modeled using the category $\hat{\mathbf{E}}_2$ of graphs with edges having two colors (see example 4.1.2 for the formal definition). The rule system is depicted in figure 5.1. It extends the rule system given in the introduction for the triangular mesh refinement with the labelling management. Neutral edges evolve to double edges composed of two neutral edges, and neutral triangles to subdivided triangles composed of four neutral triangles. For river edges there are two possible outcomes consisting of double edges composed of one normal edge and one river edge. Finally, a river triangle gives four possible outcomes induced by the results of its edges.

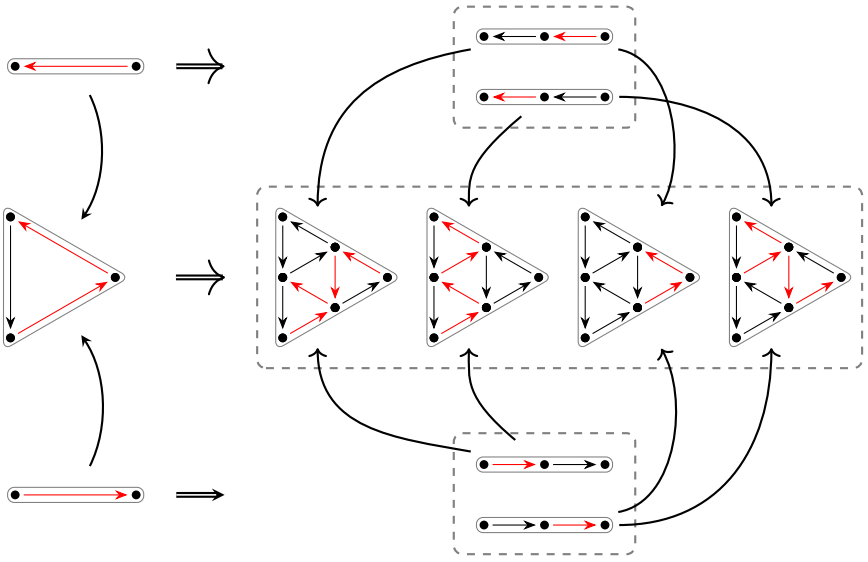


Figure 5.2: Evolution of the two river edge inclusions in a triangle.

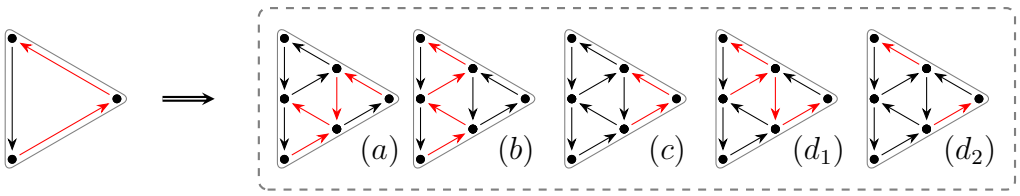


Figure 5.3: Alternative rule for river triangles: a river can flow underground.

The relationship between the possible outcomes of river edges inside a river triangle is described by the rule inclusions depicted in figure 5.2: for each possible outcome of the river triangle, an inclusion from the corresponding outcome of the river edge is specified.

The global coherence of this rule system comes from the fact the outcomes of a pair of adjacent triangles are constrained by the outcomes of their common edge. Figure 5.4 describes the rewriting of a river traversing such a pair of triangles. There are eight possible results corresponding to the combinations of the r.h.s. of the river triangle rule applied twice, that are coherent with the r.h.s. of the rule applied on their common edge. They correspond to the connected components composed of a morphism in the r.h.s. of f and a morphism in the r.h.s. of g. For instance, (a) is given by the pair $\langle f_a, g_b \rangle$, (c) by $\langle f_c, g_b \rangle$ and (f) by $\langle f_c, g_c \rangle$.

5.5.2 Non-determinism and Locality

One can observe that the rule system in figures 5.1/5.2 have two particular properties. First, the outcomes of a river triangle are entirely determined by the outcomes of its edges, *i.e.*, there is exactly one possible outcome triangle for every combination of outcomes of its edges. Non-deterministic global transformations does not always have this property: a rule can have multiple outcomes for a combination of outcomes of its sub-rules. For instance, consider the update of the rule for river triangles pictured in figure 5.3. The four first cases (a), (b), (c) and (d_1) are as before. However, for the exact same choice of edge refinement of (d_1), the alternative case (d_2) is now also proposed. One could interpret this new result as the river flowing underground. The rule inclusions are updated accordingly in the obvious way[1]. Since the outcomes (d_1) and (d_2) correspond to the same combination of outcomes for their edges, fixing the outcomes for the edges does not fully determine the outcomes for the triangles. This shows that the formalism allows to add non-determinism in over-rules, without constraining sub-rules.

The second property of the rule system in figures 5.1/5.2 is that for each combination of outcomes of the edges in a triangle, there is at least one corresponding outcome for the triangle. So the sub-rules can be decided independently, there will always be a possible outcome of the over-rule to conciliate these decisions. However, rule systems without this property can be designed and can exhibit some non-local effects called *correlations*. For instance, suppose that we now update the rule for river triangles by not considering some outcomes anymore, like in figure 5.5. For this new version of the rule system, river edges in a river triangle must have the same outcome: either it is the first edge of the two resulting double edges that is colored

[1]Since the result (d_2) contains triangles with only one river edge, a full specification should also consider a rule for this new kind of triangle.

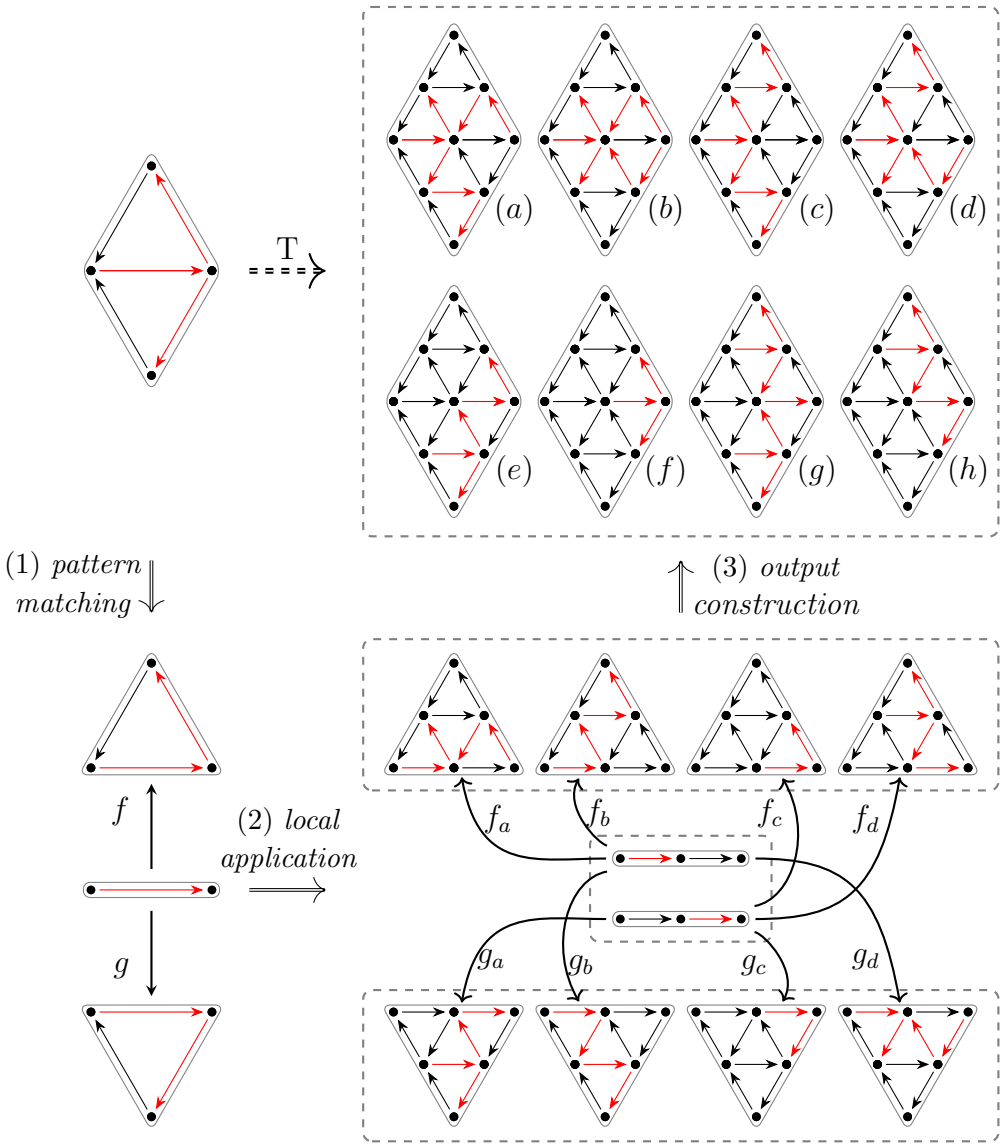


Figure 5.4: Application of the global transformation over the top left graph.

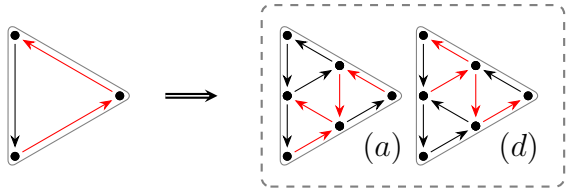


Figure 5.5: Alternative rule for river triangles exhibiting correlations.

(case (a)), or it is the second edge (case (d)). The two other possibilities are dropped away. Consequently, the results of edges cannot be considered independently. These dependencies can propagate through the entire graph and lead to non-local effects, such as in figure 5.6. In this example, a river flows through a chain of three triangles. As the outcome for river edges are dependent, there are only two possible outcomes for the full graph. These outcomes correspond either to take (a) as the outcomes of all river edges or (d). This example can be generalized with a river involving an arbitrary number of triangles showing that the consequence cannot be bounded in space. This time, we have shown that the formalism allows non-determinism in over-rules to constrain sub-rules, loosing the locality of the behavior in some way.

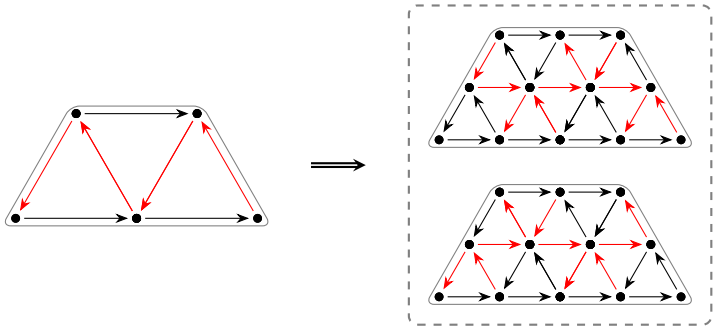


Figure 5.6: Correlations leading to a non-local effect.

5.6 Correlations

This section investigates the notion of correlation as introduced above to characterize *correlation-free* systems that do not exhibit non-local effects. Intuitively, a non-deterministic global transformation T is correlation-free if, for any object c and any combination of possible outcomes in its local results, there must be some corresponding global result in $T(c)$. This notion of dependencies between the local results relies not only on the non-determinism of the rules as we have seen in the previous example, but also on the properties of the category $\mathbf{C}$. Indeed, the possible outputs for an arbitrary input are computed by colimit in $\mathbf{C}$ and $\mathbf{C}$ may not propose such a result for some local choice. To avoid the latter difficulty coming from the category and to focus on rules only, we restrict ourselves to a simpler setting where $\mathbf{C}$ happens to be cocomplete.

We first study the shortcomings induced by considering a cocomplete category. We then characterize formally the notion of correlation in this setting, giving a local criterion on the rule system to ensure the absence of correlation. We finally relate this formal result to sheaf theory.

5.6.1 Colimits of Families in a Cocomplete Category

For a given input c, each object of the family $T(c)$ corresponds to a choice of an outcome for each matching in c. Such a choice can specified by giving the index of the chosen outcome for each local application of the rules. We call such a specification a *compatible family of indices* and it can be defined for any diagram in $\mathbb{O}\mathbf{C}$. For conciseness, for a given pair $C = (I, U)$, we denote I by $\pi_1(C)$ and U by $\pi_2(C)$.

Definition 5.6.1. For a diagram $D : \mathbf{I} \to \mathbb{O}\mathbf{C}$, a *compatible family of indices* of D is a family $\{e_i \in \pi_1(D(i))\}_{i \in \mathrm{Ob}_\mathbf{I}}$ such that for any $f : i \to j \in \mathbf{I}$, $\pi_1(D(f))(e_j) = e_i$.

For a cocone C over D, an *amalgamation* of such a compatible family is an element $e \in \pi_1(C)$ such that $\pi_1(C_i)(e) = e_i$ for any $i \in \mathrm{Ob}_\mathbf{I}$.

When $\mathbf{C}$ happens to be cocomplete, we can give a precise definition of correlations only in terms of the index sets involved in the rules. Indeed, in that case, the index set of the colimit of a diagram $D : \mathbf{I} \to \mathbb{O}\mathbf{C}$ corresponds to the compatible families of indices of the diagram, regardless of the content of the families: cocompleteness ensures that for each possible compatible family, a unique amalgamation, so output, exists. Moreover, such an output is the colimit in $\mathbf{C}$ of the outcomes specified by the family.

Definition 5.6.2. Let $D : \mathbf{I} \to \mathbb{O}\mathbf{C}$ be a diagram and $\mathcal{F} = \{e_i \in \pi_1(D(i))\}_{i \in \mathrm{Ob}_\mathbf{I}}$ a compatible family of indices of D. The *restriction of D to $\mathcal{F}$* is the diagram $D \restriction \mathcal{F} : \mathbf{I} \to \mathbf{C}$ such that for any $i \in \mathrm{Obj}_\mathbf{I}$, we have $(D \restriction \mathcal{F})(i) = \pi_2(D(i))(e_i)$, and for any $f : i \to j \in \mathbf{I}$, we have $(D \restriction \mathcal{F})(f) = \pi_2(D(f))(e_j) : \pi_2(D(i))(e_i) \to \pi_2(D(j))(e_j)$.

Proposition 5.6.1. *Consider a cocomplete category $\mathbf{C}$ and a diagram $D : \mathbf{I} \to \mathbb{O}\mathbf{C}$ such that $C = \mathrm{Colim}(D)$ exists in $\mathbb{O}\mathbf{C}$. Every compatible family of indices $\mathcal{F}$ of D has a unique amalgamation $e \in \pi_1(C)$. Moreover, $\pi_2(C)(e) \cong \mathrm{Colim}(D \restriction \mathcal{F})$.*

Proof. Let us fix a compatible family $\mathcal{F} = \{e_i \in \pi_1(D(i))\}_{i \in \mathrm{Ob}_\mathbf{I}}$. Consider $X = \mathrm{Colim}(D \restriction \mathcal{F})$ which exists since $\mathbf{C}$ is cocomplete. Now take the object $E = \mathrm{sing}_\mathbf{C}(X) = (\{\star\}, [\star \mapsto X]) \in \mathbb{O}\mathbf{C}$, together with the family $E_i : D(i) \to E = ([\star \mapsto e_i], [\star \mapsto X_i : \pi_2(D(i))(e_i) \to X])$ for each $i \in \mathrm{Ob}_\mathbf{I}$.

We first show that E is a cocone over D. For doing so, we need to check that for any $f : i \to j \in \mathbf{I}$, we have $E_j \circ D(f) = E_i$. First observe that $\pi_1(D(f)) \circ \pi_1(E_j) = \pi_1(E_i)$ since $\pi_1(E_i) = [\star \mapsto e_i]$, $\pi_1(E_j) = [\star \mapsto e_j]$ and $\pi_1(D(f))(e_j) = e_i$. Secondly, observe that $\pi_2(E_j) \circ \pi_2(D(f)) = [\star \mapsto \pi_2(E_j)(\star) \circ \pi_2(D(f))(\pi_1(E_j)(\star))]$. Since $\pi_1(E_j)(\star) = e_j$ and $\pi_2(E_j)(\star) = X_j$, we get $(\pi_2(E_j) \circ \pi_2(D(f)))(\star) = X_j \circ \pi_2(D(f))(e_j) = X_j \circ (D \restriction \mathcal{F})(f) = X_i$ by definition of $D \restriction \mathcal{F}$ and since X is a cocone. Since $\pi_1(E_j) = [\star \mapsto X_i]$, we get that $E_j \circ D(f) = E_i$ as required and E forms a cocone over D.

By universality of C, there exists a unique mediating morphism $m : C \to E$ such that $m \circ C_i = E_i$ for any $i \in \mathrm{Ob}_\mathbf{I}$. In particular, $\pi_1(m) : \pi_1(E) \to \pi_1(C)$ precisely gives an element $e = \pi_1(m)(\star)$ in $\pi_1(C)$. It is an amalgamation of $\mathcal{F}$ since $\pi_1(C_i)(e) = \pi_1(C_i)(\pi_1(m)(\star)) = (\pi_1(C_i) \circ \pi_1(m))(\star) = \pi_1(E_i)(\star) = e_i$ as required. For uniqueness, consider any other amalgamation e'. We are able to define $m' : C \to E$ with $\pi_1(()m')(\star) = e'$ giving $m' \circ C_i = E_i$. But by uniqueness of the mediating morphism, $m' = m$ and $e' = e$.

Finally, to show that $\pi_2(C)(e) \cong X$, it is enough to observe that m provides us with a morphism $\pi_2(m)(\star) : \pi_2(C)(e) \to X$. Moreover, for any $X_i : \pi_1(D(i))(e_i) \to$

X, we have $\pi_2(m)(\star) \circ \pi_2(C_i)(e) = \pi_2(m)(\star) \circ \pi_2(C_i)(\pi_2(E_i)(\star)) = \pi_2(m \circ C_i)(\star) = \pi_2(E_i)(\star) = X_i$. But, universal cocones only factor through isomorphic universal conones, so $\pi_2(C)(e) \cong X$ which concludes the proof. □

This proposition does not hold in general when $\mathbf{C}$ is an arbitrary category. In some cases, there can be more than one element e for a given compatible family of indices, leading to non-determinism emerging from the category $\mathbf{C}$.

Example 5.6.1. Consider $\mathbf{C}$ with four objects a, b, u_1, u_2 and four morphisms $a_1 : a \to u_1, b_1 : b \to u_1, a_2 : a \to u_2, b_2 : b \to u_2$ apart from identities. Notice that the discrete diagram in $\mathbf{C}$ consisting of only a and b have two cones u_1 and u_2 but none of them is a colimit as there is no morphism between them. So $\mathbf{C}$ is not cocomplete. Consider the equivalent diagram in $\mathbb{O}\mathbf{C}$, *i.e.*, with only $(\{\star\}, [\star \mapsto a])$ and $(\{\star\}, [\star \mapsto b])$ as objects. The diagram can be interpreted as specifying the merge of a and b with no alternative outcome. A colimit of this diagram is $C = ([\{\star_1, \star_2\}, [\star_1 \mapsto u_1, \star_2 \mapsto u_2])$, and with components $C_a = ([\star_1 \mapsto \star, \star_2 \mapsto \star], [\star_1 \mapsto a_1, \star_2 \mapsto a_2])$ and $C_b = ([\star_1 \mapsto \star, \star_2 \mapsto \star], [\star_1 \mapsto b_1, \star_2 \mapsto b_2])$. This colimit proposes to use non-deterministically either u_1 or u_2 as the result of the merge. The indices $\star_1$ and $\star_2$, corresponding to these two possibilities, are both the amalgamation of the same and unique compatible family, in contradiction with proposition 5.6.1.

Conversely, there can be no such amalgamation e for a given compatible family of indices. This leads to correlations that do not come from the index sets but from the category $\mathbf{C}$ as well.

Example 5.6.2. Consider $\mathbf{C}$ with six objects a, u_1, b, c, u_2, d and four morphisms $a_1 : a \to u_1, b_1 : b \to u_1, c_2 : c \to u_2, d_2 : d \to u_2$ apart from identities. Notice that the discrete diagram in $\mathbf{C}$ consisting of only a and d does not have any colimit, so $\mathbf{C}$ is not cocomplete. The same observation can be made for b and c. Consider the diagram in $\mathbb{O}\mathbf{C}$, *i.e.*, with only $(\{\star_1, \star_2\}, [\star_1 \mapsto a, \star_2 \mapsto c])$ and $(\{\star_1, \star_2\}, [\star_1 \mapsto b, \star_2 \mapsto d])$ as objects. The diagram can be interpreted as specifying the merge of two objects, but this time the first object is to be chosen among a and c, and the second object among b and d. A colimit of this diagram is $C = ([\{\star_1, \star_2\}, [\star_1 \mapsto u_1, \star_2 \mapsto u_2])$. This colimit proposes to use non-deterministically either u_1 or u_2 as the result of the merge, but illustrate that if a (resp. c) is chosen for the first object, b (resp. d) has to be chosen for the second. The compatible families consisting in choosing a and d, or c and b, cannot be amalgamated, in contradiction with proposition 5.6.1.

Proposition 5.6.1, combined by the fact that any element $e \in \pi_1(C)$ is by definition the amalgamation of a corresponding family of indices through the cocone components, means that $\pi_1(C)$ is isomorphic to the set of compatible families of indices. So $\mathbf{C}$ does not constrain the computation of the index set $\pi_1(C)$. Notice that this formulation in terms of a family of indices characterizes $\pi_1(C)$ as a limit in **Set** (see theorem 1 of section V.1 of [MacLane, 2013]). This is not surprising, as limits are equivalent to colimits in the opposite category. So, if one considers the functor $\Pi_1 : \mathbb{O}\mathbf{C} \to \mathbf{Set}^{op}$ as the obvious projection sending families to their index sets, $\Pi_1 \circ D : \mathbf{I} \to \mathbf{Set}^{op}$ is the index part of the diagram $D : \mathbf{I} \to \mathbb{O}\mathbf{C}$, and we have:

$$\pi_1(C) \cong \mathrm{Lim}((\Pi_1 \circ D)^{op}) \cong \mathrm{Colim}(\Pi_1 \circ D).$$

In the following, we denote by D^1 the presheaf $(\Pi_1 \circ D)^{op} : \mathbf{I}^{op} \to \mathbf{Set}$. Particularly, we will make use of T^1 and R^1, the respective index parts of the global functor associated with a global transformation T, and of its r.h.s. functor R.

To summarize, when $\mathbf{C}$ is cocomplete, the study of the correlations can be safely restricted to the behavior of the index part only and its limit.

5.6.2 Correlations in Terms of Index Sets

To keep up with the intuitions of the other chapters, we consider a non-deterministic global transformation $T = \langle \mathbf{\Gamma}, \mathrm{L} : \mathbf{\Gamma} \to \mathbf{C}_{\mathcal{M}}, \mathrm{R} : \mathbf{\Gamma} \to \mathbb{O}\mathbf{C} \rangle$, where the pattern matching is done in the category of monomorphisms $\mathbf{C}_{\mathcal{M}}$. Also, for reasons we will make explicit later, we suppose that for any $c \in \mathbf{C}$ we suppose that $\mathrm{Ob}_{\mathrm{L}/c}$ is countable.

We now formalize the idea of a global transformation to be correlation-free. Since we want the non-determinism to not induce non-local effects, the local outcomes of the rules instances should be somehow independent. A local choice is expressed as a compatible family of indices for some subset of the instances. To describe the expected independence, we thus consider all the families of indices (*i.e.*, all choices) for all subsets of the instances (*i.e.*, for all local views), and we ask for each of them that there exists some global outcome compatible with it. In other words, any local choice cannot be forbidden by the global behavior.

We first define formally the notion of local choice. The choice expressed by the family of indices must respect the morphisms of L/c. Indeed, for any rule inclusion, an instance of the over-rule contains an instance of the sub-rule, so that, when a choice is made for the over-instance, it induces a choice for the sub-instance. This leads to the following definition.

Definition 5.6.3. Consider some global transformation $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$, some input object $c \in \mathbf{C}$, and a downward closed full subcategory $\mathbf{I} \subset \mathrm{L}/c$, *i.e.*, such that for any morphism of instances $\iota : \langle \gamma_1, f_1 \rangle \to \langle \gamma_2, f_2 \rangle \in \mathrm{L}/c$ with $\langle \gamma_2, f_2 \rangle \in \mathbf{I}$, $\iota \in \mathbf{I}$.

A *compatible family of indices of $T(c)$ restricted to I* is compatible family of indices of the functor $\mathrm{R} \circ \mathrm{Proj}[\mathbf{I}]$, *i.e.*, a collection $\{e_i \in \mathrm{R}^1(\gamma_i)\}_{\langle \gamma_i, f_i \rangle \in \mathbf{I}}$ such that for any instance inclusion $\iota : \langle \gamma_i, f_i \rangle \to \langle \gamma_j, f_j \rangle \in \mathbf{I}$, $e_i = \mathrm{R}^1(\iota)(e_j)$.

Correlation-free global transformations can now be formally characterized in terms of amalgamations.

Definition 5.6.4. Given a global transformation $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$ and an input object $c \in \mathbf{C}$, we say that T *is correlation-free at c* if for any downward closed full subcategory $\mathbf{I} \subset \mathrm{L}/c$ and any compatible family $\{e_i \in \mathrm{R}^1(\gamma_i)\}_{\langle \gamma_i, f_i \rangle \in \mathbf{I}}$ of $T(c)$ restricted to $\mathbf{I}$, there exists an amalgamation $e \in T^1(c)$, *i.e.*, such that $e_i = T^1(f_i)(e)$ for any $\langle \gamma_i, f_i \rangle \in \mathbf{I}$.

T is said *correlation-free* if for any $c \in \mathbf{C}$, T is correlation free at c.

Notice that in definition 5.6.4, we do not require that there exists some *unique* amalgamation $e \in T^1(c)$. Indeed, as $T^1(\mathrm{L}(\gamma)) \cong \mathrm{R}^1(\gamma)$ since L is full faithfull (as recalled in the background section 5.1.1), this property also applies on rules. If this e were unique, the set $T^1(\mathrm{L}(\gamma))$ for any rule γ would be a singleton. Therefore, any rule would have a unique possible result, and the computation would be deterministic. This point is discussed in more details in the following section.

5.6.3 Local Criterion

Given a global transformation T, exhibiting a criterion on the sole rule system for ensuring that T is correlation-free is meaningful. In this section, we give such a local criterion. We restrict ourselves to the practical case where $\mathbf{C}$ is a cocomplete category (or at least contains all the required colimits) and has only monomorphisms[2]. Thanks to this last point, the comma category L/c is thin for any input object c. Moreover, we suppose that L/c is countable so that rules instances can be iterated.

To get such a criterion, we simply restrict the property of a global transformation to be correlation-free to its rules. For doing so, we look at the r.h.s. functor R as a global transformation itself. The action of a global transformation on its rules is given by the functor:

$$\mathrm{R} \cong T \circ \mathrm{L} \cong \mathrm{Colim}(\mathrm{R} \circ \mathrm{Proj}[\mathrm{L}/\mathrm{L}(-)]).$$

Since L is fully faithful, working with $\mathrm{L}/\mathrm{L}(-)$ is totally equivalent to working with $\mathbf{\Gamma}/-$. Indeed, for any $\gamma \in \mathbf{\Gamma}$, there is an obvious isomorphism of category $\overline{\mathrm{L}}_\gamma : \mathbf{\Gamma}/\gamma \to \mathrm{L}/\mathrm{L}(\gamma)$, where $\mathbf{\Gamma}/\gamma$ denotes the comma category $id_{\mathbf{\Gamma}}/\gamma$ with $id_{\mathbf{\Gamma}} : \mathbf{\Gamma} \to \mathbf{\Gamma}$ the identity functor of $\mathbf{\Gamma}$. Thus, we can express R without involving L:

$$\mathrm{R} \cong \mathrm{Colim}(\mathrm{R} \circ \mathrm{Proj}[\mathbf{\Gamma}/-]).$$

So R can be seen as a global transformation $\mathrm{R} = \langle \mathbf{\Gamma}, id_{\mathbf{\Gamma}}, \mathrm{R} \rangle$.

The local criterion for a global transformation to be correlation-free can now be easily stated as follows.

Theorem 5.6.2. *A global transformation $T = \langle \mathbf{\Gamma}, \mathrm{L}, \mathrm{R} \rangle$ is correlation-free* iff *the global transformation $\mathrm{R} = \langle \mathbf{\Gamma}, id_{\mathbf{\Gamma}}, \mathrm{R} \rangle$ is correlation-free.*

Proof. ($\Rightarrow$) For the first direction, suppose that T is correlation-free. Using definition 5.6.4, showing that R is correlation-free consists in proving that for any input object $\gamma \in \mathbf{\Gamma}$, any downward closed full subcategory $\mathbf{I} \subset \mathbf{\Gamma}/\gamma$, any compatible family $\varepsilon = \{e_i \in \mathrm{R}^1(\gamma_i)\}_{\langle \gamma_i, \iota_i \rangle \in \mathbf{I}}$ has an amalgamation in $\mathrm{R}^1(\gamma)$. Using the category isomorphism $\overline{\mathrm{L}}_\gamma$, we have $\varepsilon = \{e_i \in \mathrm{R}^1(\gamma_i)\}_{\langle \gamma_i, \mathrm{L}(\iota_i) \rangle \in \overline{\mathrm{L}}_\gamma(\mathbf{I})}$ where $\overline{\mathrm{L}}_\gamma(\mathbf{I})$ is the image by $\overline{\mathrm{L}}_\gamma$ of $\mathbf{I}$ and is a downward closed full subcategory of $\mathrm{L}/\mathrm{L}(\gamma)$. So ε is a compatible family of $T(\mathrm{L}(\gamma))$ restricted to $\overline{\mathrm{L}}_\gamma(\mathbf{I})$, compatibility coming from: for any $\iota : \langle \gamma_i, \mathrm{L}(\iota_i) \rangle \to \langle \gamma_j, \mathrm{L}(\iota_j) \rangle \in \overline{\mathrm{L}}_\gamma(\mathbf{I})$, $T^1(\mathrm{L}(\iota))(e_j) = \mathrm{R}^1(\iota)(e_j) = e_i$ since $\iota : \langle \gamma_i, \iota_i \rangle \to \langle \gamma_j, \iota_j \rangle \in \mathbf{I}$ and ε is compatible for $\mathrm{R}^1(\gamma_i)$ restricted to $\mathbf{I}$. As a compatible family of $T(\mathrm{L}(\gamma))$ restricted to $\overline{\mathrm{L}}_\gamma(\mathbf{I})$ with T correlation-free, we get an amalgamation $e \in T^1(\mathrm{L}(\gamma))$ such that $e_i = T^1(\mathrm{L}(\iota_i))(e)$ for $\langle \gamma_i, \mathrm{L}(\iota_i) \rangle \in \overline{\mathrm{L}}_\gamma(\mathbf{I})$. Since $T^1 \circ \mathrm{L} = \mathrm{R}^1$, this gives $e \in \mathrm{R}^1(\gamma)$ such that $\mathrm{R}^1(\iota_i)(e) = e_i$ for any $\langle \gamma_i, \iota_i \rangle \in I$. So ε has an amalgamation in $\mathrm{R}^1(\gamma)$ and R is correlation-free.

($\Leftarrow$) For the other direction, suppose that R is correlation-free. Showing that T is correlation-free consists in proving that for any input object $c \in \mathbf{C}$, any downward closed full subcategory $\mathbf{I} \subset \mathrm{L}/c$, any compatible family $\varepsilon = \{e_i \in \mathrm{R}^1(\gamma_i)\}_{\langle \gamma_i, f_i \rangle \in \mathbf{I}}$ has an amalgamation in $T^1(c)$. For doing so, we will extend the local choice specified by ε to a global choice for L/c and use proposition 5.6.1 to get an amalgamation.

[2] A mechanism similar to the factorisation by U of chapter 4, can be used to process the pattern mathing in a category of monomphisms and the reconstruction in a cocomplete category. But for the sake of simplicity, we avoid such a complication.

We proceed by induction, using that L/c is countable, to decide at each step for an instance of L/c not in $\mathbf{I}$ one of the proposed outcomes. We suppose that we iterate over the rule instances $\langle \gamma^k, f^k : \gamma^k \to c \rangle \in \mathrm{L}/c$ with for $k > 1$.

For the base case, we consider $\mathbf{I}^0 = \mathbf{I}$ and $\varepsilon^0 = \varepsilon$. And we have that ε^0 is a compatible family of $T(c)$ restricted to $\mathbf{I}^0$ where $\mathbf{I}^0$ contains all $\langle \gamma^K, f^K : \gamma^K \to c \rangle$ for $K \leq 0$.

For the induction, take $k > 0$ with the associated $\mathbf{I}^k$ and ε^k where is a compatible family of $T(c)$ restricted to $\mathbf{I}^k$ where $\mathbf{I}^k$ contains all $\langle \gamma^K, f^K : \gamma^K \to c \rangle$ for $K \leq k$. Our goal is to define $\mathbf{I}^{k+1}$ by extending $\mathbf{I}^k$ with $\langle \gamma^{k+1}, f^{k+1} \rangle$, and to define ε^{k+1} as an compatible family restricted to $\mathbf{I}^{k+1}$ such that $\varepsilon_i^{k+1} = \varepsilon_i^k$ for all $\langle \gamma_i, f_i \rangle \in I^k$.

If $\langle \gamma^{k+1}, f^{k+1} \rangle \in \mathbf{I}^k$, the case is trivial and we set $\mathbf{I}^{k+1} = \mathbf{I}^k$ and $\varepsilon^{k+1} = \varepsilon^k$. We consider now that $\langle \gamma^{k+1}, f^{k+1} \rangle \notin \mathbf{I}^k$. We set $\mathbf{I}^{k+1}$ as the downward closed full subcategory of L/c containing all the objects of $\mathbf{I}_k$ and $\langle \gamma^{k+1}, f^{k+1} \rangle$. We need to decide $e^{k+1} \in \mathrm{R}^1(\gamma^{k+1})$ for defining ε^{k+1}. Let $\{\iota_i : \gamma_i \to \gamma^{k+1}\}_{\langle \gamma_i, \iota_i \rangle \in \mathbf{J}}$ where $\mathbf{J}$ is the downward closed full subcategory of $\mathbf{\Gamma}/\gamma^{k+1}$ such that $\langle \gamma_i, \iota_i \rangle \in \mathbf{J}$ if $\langle \gamma_i, f^{k+1} \circ \mathrm{L}(\iota_i) \rangle \in \mathbf{I}$. First, we show that $\delta = \{\varepsilon_i^k\}_{\langle \gamma_i, \iota_i \rangle \in \mathbf{J}}$ is a compatible family of elements of $\mathrm{R}(\gamma^{k+1})$ restricted to $\mathbf{J}$. Let us consider $\iota : \langle \gamma_i, \iota_i \rangle \to \langle \gamma_j, \iota_j \rangle \in \mathbf{J}$. We have $\mathrm{R}^1(\iota)(\varepsilon_j^k) = T^1(\mathrm{L}(\iota))(\varepsilon_j^k)$. But ε^k is a compatible family for $T(c)$ restricted to $\mathbf{I}^k$ and $\mathrm{L}(\iota) \in \mathbf{I}^k$. So $T^1(\mathrm{L}(\iota))(\varepsilon_j^k) = \varepsilon_i^k$ so that $\varepsilon_i^k = \mathrm{R}^1(\iota)(\varepsilon_j^k)$ as expected, and δ is a compatible family of $\mathrm{R}(\gamma^{k+1})$ restricted to $\mathbf{J}$. Since R, as a global transformation, is correlation-free, there is an amalgamation in $\mathrm{R}^1(\gamma^{k+1})$ that we choose for e^{k+1}. It remains to define ε^{k+1}. Take $\langle \gamma_i, f_i \rangle \in \mathbf{I}^{k+1}$: either $\langle \gamma_i, f_i \rangle \in \mathbf{I}^k$, or $f_i = f^{k+1} \circ \mathrm{L}(\iota_i)$ for some $\iota_i : \gamma_i \to \gamma^{k+1}$. For the former, we set $\varepsilon_i^{k+1} = \varepsilon_i^k$, for the latter $\varepsilon_i^{k+1} = \mathrm{R}^1(\iota_i)(e^{k+1})$. Notice first that the latter case is well defined since L/c is thin and the decomposition $f_i = f^{k+1} \circ \mathrm{L}(\iota_i)$ is unique. Notice then that if $\langle \gamma_i, f_i \rangle$ satisfies both cases, then $\langle \gamma_i, \iota_i \rangle$ is in $\mathbf{J}$ and both definitions agree since e^{k+1} is the amalgamation of δ and, as such, $\varepsilon_i^k = \mathrm{R}^1(\iota_i)(e^{k+1})$. Finally, we check that ε^{k+1} is a compatible family for $T(c)$ restricted to $\mathbf{I}^{k+1}$. Consider $\iota : \langle \gamma_i, f_i \rangle \to \langle \gamma_j, f_j \rangle \in \mathbf{I}^{k+1}$, we need to check that $\mathrm{R}^1(\iota)(\varepsilon_j^{k+1}) = \varepsilon_i^{k+1}$. If $\iota \in \mathbf{I}^k$, $\mathrm{R}^1(\iota)(\varepsilon_j^{k+1}) = \mathrm{R}^1(\iota)(\varepsilon_j^k) = \varepsilon_i^k = \varepsilon_i^{k+1}$, since ε^k is compatible. If $\iota \notin \mathbf{I}^k$, we have $f^{k+1} \circ L(\iota_j) \circ L(\iota) = f_j \circ L(\iota) = f_i = f^{k+1} \circ L(\iota_i)$. Since f^{k+1} is a monomorphism and L is fully faithful, we have $\iota \circ \iota_j = \iota_i$. So we have $\mathrm{R}^1(\iota)(\varepsilon_j^{k+1}) = \mathrm{R}^1(\iota)(\mathrm{R}^1(\iota_j)(e^{k+1})) = \mathrm{R}^1(\iota_j \circ \iota)(e^{k+1}) = \mathrm{R}^1(\iota_i)(e^{k+1}) = \varepsilon_i^{k+1}$ as expected, and ε^{k+1} is a compatible family which completes the induction case.

The previous induction associates for each instance $\langle \gamma^k, f^k \rangle$ of L/c, an index $e^k \in \mathrm{R}^1(\gamma^k)$ and this association consists in a compatible family of $T(c)$. Then by proposition 5.6.1, there is some unique element $e \in T^1(c)$ and for any $\langle \gamma^k, f^k \rangle \in \mathrm{L}/c$, $T^1(f)(e) = e^k$. So the global transformation T is correlation-free. □

5.6.4 Connection with Sheaf Theory

One can notice that the definition 5.6.4 of correlation-free global transformations is very similar to the definition of a presheaf to be a sheaf. Sheaves are a way to attach local data to a topological space or, more generally, any category. A presheaf $F : \mathbf{C}^{op} \to \mathbf{Set}$ attaches a set of data $F(c)$ on each object $c \in \mathbf{C}$. The presheaf structure allows to restrict the global data to more local data, *i.e.*, given some $e \in F(c)$, and any $f : c' \to c \in \mathbf{C}$, a corresponding element of $e' \in F(c')$ is obtained by applying $F(f) : F(c) \to F(c')$ on e: $e' = F(f)(e)$. A sheaf is a presheaf with the added structure that permits to go the other way around by "gluing" together local

data to obtain a more global data. More precisely, given some $c \in \mathbf{C}$, some collection of morphisms $\{f_i : c_i \to c\}_{i \in I}$ that is supposed to be sufficient to "cover" c, and a collection of compatible elements $\{e_i \in F(c_i)\}_{i \in I}$, the sheaf is able to "glue" them together into a unique element $e \in F(c)$ corresponding to each e_i by $F(f_i)(e) = e_i$.

The connection with our setting is pretty clear: the attached local data is the elements of the index sets, representing non-deterministic choices, and the correlation-freeness states that local and compatible non-deterministic choices can be glued together to a global choice. However, there is a main difference with the usual concepts of sheaf theory and our notion of correlation-freeness. Indeed, we dropped the uniqueness constraint, as we want to describe that local choices taken in some of the rule instances can be extended to choices in the other rule instances, leading to choices in the global result. There might be multiple such extensions, and in fact there should be, otherwise a choice for a rule instance would induce a choice for the global result, which would not contribute to the computation. To this need, we use the finest coverage on $\mathbf{\Gamma}$, where a rule is covered by any abritrary subset of its subrules. This coverage is usually not considered as there is no interesting sheaf for this coverage.

Indeed, as we mentioned earlier, the uniqueness constraint induces determinism. Notice that any rule γ is covered by the empty family of sub-rules, and that there is only one possible (degenerate) family of compatible elements of this family: the empty family of elements. As a consequence, the uniqueness constraint gives us a unique element $e \in \mathrm{R}^1(\gamma)$ corresponding to this collection. But this implies that $\mathrm{R}^1(\gamma)$ is the singleton $\{e\}$ as if there were an other element $e' \in \mathrm{R}^1(\gamma)$, it would also trivially correspond this empty family of elements. Therefore, the only sheaf which respects this coverage is the constant presheaf sending each object to the singleton set, which corresponds to a deterministic global transformation.

So we considered presheaves that satisfy the gluing condition of sheaves without the uniqueness constraint. To our knownledge, while this consept exists, it only rarely occurs in the literature in terms of coseparated (or conjunctive, or epipresheaf), while its dual notion of separated (or monopresheaf) is more documented (see C2.2 in [Johnstone, 2002]).

As mentioned in section 5.6.1, we restricted our study of correlation to the case where $\mathbf{C}$ is cocomplete. Indeed, when it is not the case, the index sets do not hold enough information to study correlations. As mentioned in example 5.6.2, the category $\mathbf{C}$ can also constrain on the non-deterministic choices, as some choices can correspond to diagrams that have no colimit in $\mathbf{C}$, leading to correlations that cannot be characterized in terms of the index sets. Also, in example 5.6.1, we have seen that in some case, a completely deterministic diagram, where each index set is a singleton, can lead to a non-deterministic computation: the corresponding colimit in $\mathbf{C}$ does not exist, since the multiple candidate cocones cannot be compared, so there is one result for each of theses cocones. This shows how, in the general case, the interplay between the indices and the values of the families, leads to a wide range of effects. These effects deserve their proper study, which is left for future work.

5.7 Final Discussion

This journey started with the general goal of going toward non-deterministic, probabilistic and quantum global transformations. Starting with the first concrete step of capturing non-deterministic Lindenmayer systems as global transformations, we guessed some constructions based on the deterministic case. The result of these guesses did not have the precise form of a global transformation in the 2-category of categories, functors and natural transformations. But we showed they correspond in fact to a global transformation in another (weak) 2-category. The latter is induced by Kleisli's construction on a particular (weak) 2-monad that we made explicit. This is to be expected, since the same architecture happens for non-deterministic dynamical systems. Indeed, they are also dynamical systems in another category, with the latter being induced by a monad. All in all, we ended with a general solution for mixing non-determinism with locality, as described in the global transformation framework.

Along the way, we mentioned a technicality about the size of the families considered, which is related to the size issues for the "2-category of categories". For most practical purpose, it is possible to restrict oneself to finite families. In this case, a small category $\mathbf{C}$ leads to a small category $\mathbb{O}\mathbf{C}$. In this case, the 2-functor $\mathbb{O}$ is indeed an endomorphism of the 2-category of *small* categories. Dropping the finiteness constraint, though, one then considers a 2-functor $\mathbb{O}\mathbf{C}$ from *small* categories to *large* categories. This is however perfectly fine, since this describes a so-called *relative pseudomonad* with an associate Kleisli's construction, as defined in [Fiore et al., 2018].

While giving concrete examples of non-deterministic global graph transformation, we discussed the non-local effects of spacial correlations in some rule systems. First, we gave an informal description of correlations in terms of dependencies between the elements in the local results. In some extreme cases, these dependencies can propagate through neighboring local results, resulting in dependencies between arbitrary distant local results. For functors $F : \mathbf{I} \to \mathbb{O}\mathbf{C}$ where $\mathbf{C}$ is cocomplete, this description was formally expressed on the presheaf part $(\Pi_1 \circ F)^{op} : \mathbf{I}^{op} \to \mathbf{Set}$ such functors, obtained by only considering the index sets of families. The formal definition of correlations was shown to be very close to the condition for the presheaf to be a sheaf, only the uniqueness condition of gluings being missing. As discussed in the chapter, for rule systems considered in this work, if we restore the uniqueness of gluings, making the presheaf to be a sheaf, we get a deterministic system. This notion was shown to correspond to the notion of an epipresheaf, occuring rarely in the litterature. The in-depth study of the general case of non-deterministic global transformations and of the connections sheaf theory is left for future work.

The formulation of "non-deterministic functors" in terms of presheaves was taken in [Fernandez et al., 2021], which directly defines the 2-category described here in terms of 2-monad. In particular, the *open* functors and *open* natural transformations are introduced using presheaves and proved to form a weak 2-category. More precisely, an open functor F from a category $\mathbf{C}$ to a category D is the data of a presheaf on $\mathbf{C}$ together with a functor from the category of elements of that presheaf to D. The formal definition in [Fernandez et al., 2021] can be made easier to manipulate by the use of discrete fibrations instead of categories of elements through the so-called Grothendieck construction, and doing so presents this bicategory as a particular bicategory of spans. Moreover, this bicategory is a sub-bicategory of the bicategory of profunctors (*a.k.a.* distributors). Notice that the many presentations

of this bicategory are strongly related to the many possible presentations one can have of the notion of "relation": powerset monad (as in this paper), spans, and characteristic functions of the relation.

Chapter 6

Conclusion and Perspectives

This study of global transformations gave rise to new knowledge about practical aspects of spatialized synchronous dynamical systems and categorical foundations of this framework. Indeed, it is a general setting to describe such dynamical systems with dynamic space, and gives general methods to describe some global behavior from a local presentation. This work allows not only to formalize the rewriting of many data-structure in a unified context, but also to study the relationship between the local and global behaviors. Whereas our framework is based on highly abstract algebra, this work is mainly motivated by practical purposes. The original formulation of global transformations in terms of pattern matching, local application and reconstruction is meant to be implemented, which gave rise to the presheaf rewriting algorithm. In some sense, the incremental and correlation-free criterion guarantees that the computation is local; the former ensures that two distinct elements are not merged by the computation, and the latter that compatible local choices are still compatible in the global setting. Therefore, these two criteria allow more simple implementations of the computation step. The expression of global transformations in terms of Kan extensions allow to relate the local presentation and its extensions. This gives a mathematical tool to expose the properties of the extended behavior, and to find an adequate local presentation of a global behavior. The language of categories is not commonly used for our practical considerations, and this work gave many examples and intuitions about highly abstract concepts in the field of computer science. Let us now review the multiple aspects of this study.

Category Theory to Describe Spatialized Computations. In global transformations, the space is described through a category where the objects can be viewed as configurations and sub-configurations, *i.e.*, spaces and sub-parts of the spaces, and the morphisms describe the inclusions between these spaces. In contrast with approaches like in causal graph dynamics, where some addressing system is encoded within the graph structure, global transformations make use of morphisms of a category to relate objects to one another. We claim that our approach is generic enough to capture any full-synchronous spatialized dynamical system, and this work has shown computations over multiple structures, such as words, cellular automata configurations or generalized graphs. The handling of other well-known structures such as trees are ongoing work and will lead to subsequent publications. Such genericity comes from the high abstract nature of category theory, as it permits to use general operations defined in any categorical context. In this work, we

mainly focused on categories of monomorphisms, as they encode a strong intuition about the notion of a subpart of an object. However, it is worth to notice that most of this work safely extends to general categories, as comma categories, colimits and Kan extensions are defined in any categorical context. The claim of generality is reinforced by the consideration of non-determinism. Indeed, the expression of global transformations in terms of (pointwise) Left Kan extensions was shown to hold in other 2-categorical contexts such as the Kleisly 2-category of the 2-monad of families, which allows composition of non-deterministic functors. This gives another parameter to the global transformation framework.

Different points of view over spatialized data have been taken through this work. For cellular automata, the usual objects of study are global and local configurations. In this setting, sub-configurations and inclusions are only introduced as a mean to define the computation. Whereas in L-systems or global graph transformations, no such distinction can be made, as the objects of study are simply all words or graphs and the inclusions naturally emerge from these objects. However, there is a main difference that sets apart the category of graphs from the categories of words or configurations. The category of graphs, was shown to be equivalent to a category of presheaves, *i.e.*, the free cocompletion of a category describing only nodes and edges. Conversely, the category of partial configurations is not cocomplete, since any partial configuration can be described as a colimit of proper partial configurations, but the converse does not hold. As categories of presheaves can be seen as generalized graphs, the free cocompletion of the category of partial configurations contains objects specifying arbitrary assemblies of partial configurations able to exhibit branches and cycles.

If one wants to consider structures more constrained than these arbitrary assemblies, but one still desires to describe them as the extension of some category of simple objects, it would be interesting to specify how these simple objects can be extended or not. This added information consists of a *site*, *i.e.*, a categorical generalization of topological spaces designed to define sheaves. A site consists of a category, with some added information called coverage, that sheaves on this site must respect. The category of sheaves over this site is the Grothendieck topos describing only the desired presheaves that turned to be sheaves since they respect the coverage. Extending our framework in this way would lead to a finer description of the spaces and dynamics of our interest, and is part of future work.

Relating the Local and the Global Behaviors. The global rewriting step of global transformations consists of the application of every rewriting rule in an object. This is successfully formalized using the general formula involving the comma category, *i.e.*, the pattern matching, and the colimit operation, *i.e.*, the gluing of local results. In this work, we have shown how this formula is a simple case of the more abstract concept of (pointwise) left Kan extensions. More generally, Kan extensions can be used to describe the relationships between the local rules and the global behavior of the computations. In our setting, left Kan extensions describe how the global behavior is an optimal extension of the rules, whereas right Kan extensions describe how a global behavior can be specified in the right-hand side of the rules as an optimal description. The property of being pointwise was shown to be related to algorithmic considerations, since it ensures that the rewriting of an object only involves its sub-objects. We also questioned the property of rule systems where

L is fully faithful, and the relaxing of this constraint should be part of subsequent work.

The tools that we developed to capture existing systems are also able to describe original systems. For cellular automata, replacing the fully shifted transition function by any function from shifted local configurations to shifted one cell-configurations permits to describe non-uniform cellular automata. Moreover, it is feasible to design global transformations on the category of words that are not Lindenmayer systems. For instance, consider some global transformation that reverses words, that consequently does not behave like a Lindenmayer system. In this perspective, we aim at guarantying global properties of an extended system at the local level. In particular, for graph rewriting, we focused on the property of the global functor to preserve monomorphisms. This was guaranteed locally using the incrementality criterion, and the same scheme was applied to characterize correlation-free non-deterministic global transformations. Through theses two cases, we sketched a general method to ensure some global property at the local level, which shows how the global transformation framework is an appropriate context to these needs.

Generic Computation Algorithm. The application of the method to concrete situations is a central motivation for this work. Therefore, its computational aspects were studied, and we gave a global graph transformation computation algorithm. The motivation was to specify an algorithm completely generic over the structure. As the two main steps of computations of gathering all matches of left-hand sides in an object, and gluing all local results together, are complex structure dependent operations, we broke them into simpler building blocks. The algorithm recomposes these building blocks to compute a sequence of local rewritings in place of the synchronous global rewriting step. This algorithm is generic enough to be adapted to other data structures, and it was tested upon words, graphs, meshes and cellular automata configurations. The only missing step toward its general formulation is a way to guarantee that the colimit is defined for any input object, otherwise the algorithm would fail on some inputs. For this purpose, this algorithm was specified for categories of presheaves, where every colimit is defined, and should be extended to other categorical context such as adhesive categories or categories of sheaves.

This algorithm needs also further studying. On the one hand, its complexity must be discussed, but as its behavior depends not only on the rule system but also on the objects being rewritten, it must be considered in terms of parameterized complexity. Also, as it is designed to work an online algorithm, it will also be useful to consider its complexity in the context of online complexity. The comparison of its complexity with the complexity of other rewriting algorithms will be necessary to understand its performances. Moreover, albeit being a sequentialization of a massively parallel process, other execution schemes should be considered. Indeed, this procedure can be adapted to fit with a given parallel architecture by executing it in parallel at multiple places in a given object. Then, a synchronization barrier is needed when two processes overlap to maintain the synchronicity of the global rewriting step.

Similarly, asynchronous update schemes can also be considered. Notice that, whereas global transformations were designed to capture deterministic and synchronous systems, the last chapter blurs this setting. Indeed, asynchronous systems can be described by non-determinism, by including the input inside the families of

results in the output, implementing the idea that, at a given time, an object can evolve or not. Whereas the decision of choosing local results inside the families is done synchronously at some step of time, this encodes the fact that some inputs remain unchanged while some are rewritten, which is a first step toward the formulation of asynchronous global transformations.

Definition of Global Transformations. Through this work, we considered multiple definitions of global transformations. The initial definition considers rules whose left-hand sides and right-hand side are taken in the same category, and the left-hand side functor is fully faithful, so the rules can be seen as a partial endofunctor over this category. The obtained global functor is an endofunctor over this category, obtained by applying in a pointwise manner the colimit formula of global transformations. This definition is the simplest formulation of global transformations and allows to describe dynamical systems over some category in an operational fashion: it uses the three computation steps that are pattern-matching, local application and reconstruction.

This description of the global behavior was shown to correspond to the concept of pointwise left Kan extension of the right-hand side functor along the left-hand side functor. This led to multiple possible generalization of global transformations. On one hand, we considered the left-hand side functor not to be fully faithful, which induces finer semantics on rule systems. Moreover, we discussed how given a non fully faithful rule system, we can retrieve an equivalent classical global transformation rule system. On the other hand, we considered the case where the global functor is a non-pointwise Kan extension. Such a global functor cannot be computed only from its value over each object, and some non-local extra information may be needed. Consequently, our algorithm cannot be used for such extensions.

In cases when the reconstruction could not be done in the desired category, the category of graphs with monomorphisms, we used another reconstruction operations: the colimit in another category, the cocomplete category of graphs. We then focused on rule systems such that the global functor factors through the inclusion from the first category to the second. This pattern can be generalized to any two categories, where the domain of the global transformation can be related to the second through some functor. Even defining global transformations that are not endofunctor, hence not dynamical systems, can be useful to describe, for example, computations from graphs to trees. At the end, we have seen that even the 2-categorical context is a parameter of this formalism, as Kan extensions can be expressed in any 2-category. In particular, the 2-category of Kleisly of the two monad of families allowed to compose functors that are not endofunctors in the usual sense.

To summary, the most generic idea of a global transformation that we considered, is that it is some 2-categorical extension of an arbitrary pair of functors, both into categories considered representing spatialized structure and their subparts, and we seek properties to guarantee the extension process at the global level.

Bibliography

[Arnoux et al., 2004] Arnoux, P., Berthé, V., and Siegel, A. (2004). Two-dimensional iterated morphisms and discrete planes. *Theoretical Computer Science*, 319(1-3):145–176.

[Arrighi and Dowek, 2012] Arrighi, P. and Dowek, G. (2012). Causal graph dynamics. In *International Colloquium on Automata, Languages, and Programming*, pages 54–66. Springer.

[Arrighi et al., 2018] Arrighi, P., Martiel, S., and Nesme, V. (2018). Cellular automata over generalized cayley graphs. *Math. Struct. in Comp. Sc.*, 18:340–383.

[Belitsky and Ferrari, 1995] Belitsky, V. and Ferrari, P. A. (1995). Ballistic annihilation and deterministic surface growth. *Journal of statistical physics*, 80(3):517–543.

[Berlekamp et al., 2004] Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2004). *Winning ways for your mathematical plays, volume 4*. AK Peters/CRC Press.

[Boehm et al., 1987] Boehm, P., Fonio, H.-R., and Habel, A. (1987). Amalgamation of graph transformations: a synchronization mechanism. *Journal of Computer and System Sciences*, 34(2-3):377–408.

[Borceux, 1994] Borceux, F. (1994). *Handbook of categorical algebra: volume 1, Basic category theory*, volume 1. Cambridge University Press.

[Boy de la Tour and Echahed, 2018] Boy de la Tour, T. and Echahed, R. (2018). A set-theoretic framework for parallel graph rewriting. *CoRR*, abs/1808.03161.

[Boy de la Tour and Echahed, 2020] Boy de la Tour, T. and Echahed, R. (2020). Parallel coherent graph transformations. In *International Workshop on Algebraic Development Techniques*, pages 75–97. Springer.

[Burks and Von Neumann, 1966] Burks, A. W. and Von Neumann, J. (1966). Theory of self-reproducing automata. *Urbana: University of Illinois Press*.

[Cattaneo et al., 2000] Cattaneo, G., Finelli, M., and Margara, L. (2000). Investigating topological chaos by elementary cellular automata dynamics. *Theoretical computer science*, 244(1-2):219–241.

[Ceccherini-Silberstein and Coornaert, 2010] Ceccherini-Silberstein, T. and Coornaert, M. (2010). *Cellular automata and groups*. Springer Science & Business Media.

[Cook et al., 2004] Cook, M. et al. (2004). Universality in elementary cellular automata. *Complex systems*, 15(1):1–40.

[Cornforth et al., 2002] Cornforth, D., Green, D. G., Newth, D., and Kirley, M. (2002). Do artificial ants march in step? ordered asynchronous processes and modularity in biological systems. *Artificial Life VIII, MIT Press*, pages 28–32.

[Corradini et al., 2006] Corradini, A., Heindel, T., Hermann, F., and König, B. (2006). Sesqui-pushout rewriting. In *International Conference on Graph Transformation*, pages 30–45. Springer.

[Corradini et al., 1997] Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., and Löwe, M. (1997). Algebraic approaches to graph transformation–part i: Basic concepts and double pushout approach. In *Handbook Of Graph Grammars And Computing By Graph Transformation: Volume 1: Foundations*, pages 163–245. World Scientific.

[Danos and Laneve, 2004] Danos, V. and Laneve, C. (2004). Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110.

[Derbel et al., 2008] Derbel, B., Mosbah, M., and Gruner, S. (2008). Mobile agents implementing local computations in graphs. In *International Conference on Graph Transformation*, pages 99–114. Springer.

[Dobrushin et al., 1978] Dobrushin, R. L., Kryukov, V., and Toom, A. L. (1978). *Locally interacting systems and their application in biology*. Springer.

[Echahed and Maignan, 2017] Echahed, R. and Maignan, A. (2017). Parallel graph rewriting with overlapping rules. *arXiv preprint arXiv:1701.06790*.

[Ehrig, 1978] Ehrig, H. (1978). Introduction to the algebraic theory of graph grammars (a survey). In *International Workshop on Graph Grammars and Their Application to Computer Science*, pages 1–69. Springer.

[Ehrig et al., 2006] Ehrig, H., Ehrig, K., Golas, U., and Taentzer, G. (2006). *Fundamentals of Algebraic Graph Transformation*, volume XIV. Springer.

[Ehrig et al., 2010] Ehrig, H., Golas, U., Hermann, F., et al. (2010). Categorical frameworks for graph transformation and hlr systems based on the dpo approach. *Bulletin of the EATCS*, (102):111–121.

[Ehrig et al., 1997] Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., and Corradini, A. (1997). Algebraic approaches to graph transformation–part ii: Single pushout approach and comparison with double pushout approach. In *Handbook Of Graph Grammars And Computing By Graph Transformation: Volume 1: Foundations*, pages 247–312. World Scientific.

[Ehrig et al., 1973] Ehrig, H., Pfender, M., and Schneider, H. J. (1973). Graph-grammars: An algebraic approach. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 167–180. IEEE.

[Ehrig and Rosen, 1980] Ehrig, H. and Rosen, B. K. (1980). Parallelism and concurrency of graph manipulations. *Theoretical Computer Science*, 11(3):247–275.

[Eilenberg and MacLane, 1945] Eilenberg, S. and MacLane, S. (1945). General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294.

[Fernandez et al., 2021] Fernandez, A., Maignan, L., and Spicher, A. (2021). The bicategory of open functors.

[Fiore et al., 2018] Fiore, M., Gambino, N., Hyland, M., and Winskel, G. (2018). Relative pseudomonads, kleisli bicategories, and substitution monoidal structures. *Selecta Mathematica*, 24(3):2791–2830.

[Fogg et al., 2002] Fogg, N. P., Berthé, V., Ferenczi, S., Mauduit, C., and Siegel, A. (2002). *Substitutions in dynamics, arithmetics and combinatorics*. Springer.

[Grzegorz et al., 1999] Grzegorz, R., Hartmut, E., and Hans-jorg, K. (1999). *Handbook Of Graph Grammars And Computing By Graph Transformations, Vol 3: Concurrency, Parallelism, And Distribution*. World Scientific.

[Hasslacher and Meyer, 1998] Hasslacher, B. and Meyer, D. A. (1998). Modeling dynamical geometry with lattice-gas automata. *International Journal of Modern Physics C*, 9(08):1597–1605.

[Hedlund, 1969] Hedlund, G. A. (1969). Endomorphisms and automorphisms of the shift dynamical system. *Mathematical systems theory*, 3(4):320–375.

[Herman and Rozenberg, 1975] Herman, G. T. and Rozenberg, G. (1975). *Developmental systems and languages*. Elsevier Science Inc.

[Herrmann and Margenstern, 2003] Herrmann, F. and Margenstern, M. (2003). A universal cellular automaton in the hyperbolic plane. *Theoretical Computer Science*, 296(2):327–364.

[Hu and Tholen, 1995] Hu, H. and Tholen, W. (1995). Limits in free coproduct completions. *Journal of Pure and Applied Algebra*, 105(3):277–291.

[Johnstone, 2002] Johnstone, P. T. (2002). *Sketches of an Elephant: A Topos Theory Compendium: Volume 2*, volume 2. Oxford University Press.

[Kan, 1958] Kan, D. M. (1958). Adjoint functors. *Transactions of the American Mathematical Society*, 87(2):294–329.

[Klales et al., 2010] Klales, A., Cianci, D., Needell, Z., Meyer, D. A., and Love, P. J. (2010). Lattice gas simulations of dynamical geometry in two dimensions. *Physical Review E*, 82(4):046705.

[Knuth et al., 1977] Knuth, D. E., Morris, Jr, J. H., and Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350.

[Kreowski and Kuske, 2006] Kreowski, H.-J. and Kuske, S. (2006). Autonomous units and their semantics-the parallel case. In *International Workshop on Algebraic Development Techniques*, pages 56–73. Springer.

[Krug and Spohn, 1988] Krug, J. and Spohn, H. (1988). Universality classes for deterministic surface growth. *Physical Review A*, 38(8):4271.

[Kurth et al., 2004] Kurth, W., Kniemeyer, O., and Buck-Sorlin, G. (2004). Relational growth grammars–a graph rewriting approach to dynamical systems with a dynamical structure. In *International Workshop on Unconventional Programming Paradigms*, pages 56–72. Springer.

[Lack and Sobociński, 2005] Lack, S. and Sobociński, P. (2005). Adhesive and quasiadhesive categories. *RAIRO-Theoretical Informatics and Applications*, 39(3):511–545.

[Lack and Sobociński, 2006] Lack, S. and Sobociński, P. (2006). Toposes are adhesive. In *International Conference on Graph Transformation*, pages 184–198. Springer.

[Lehmann, 1976] Lehmann, D. (1976). *Categories for fixpoint semantics*. PhD thesis, University of Warwick.

[Lindenmayer, 1968] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299.

[Löwe et al., 1993] Löwe, M., Korff, M., and Wagner, A. (1993). An algebraic framework for the transformation of attributed graphs. In *Term graph rewriting: theory and practice*, pages 185–199.

[MacLane, 2013] MacLane, S. (2013). *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York.

[MacLane and Moerdijk, 2012] MacLane, S. and Moerdijk, I. (2012). *Sheaves in geometry and logic: A first introduction to topos theory*. Springer Science & Business Media.

[Maignan and Spicher, 2015] Maignan, L. and Spicher, A. (2015). Global graph transformations. In Plump, D., editor, *Proceedings of the 6th International Workshop on Graph Computation Models co-located with the 8th International Conference on Graph Transformation (ICGT 2015) part of the Software Technologies: Applications and Foundations (STAF 2015) federation of conferences, L'Aquila, Italy, July 20, 2015*, volume 1403 of *CEUR Workshop Proceedings*, pages 34–49. CEUR-WS.org.

[Mammen et al., 2010] Mammen, S. v., Phillips, D., Davison, T., and Jacob, C. (2010). A graph-based developmental swarm representation and algorithm. In *International Conference on Swarm Intelligence*, pages 1–12. Springer.

[Morse, 1921] Morse, H. M. (1921). Recurrent geodesics on a surface of negative curvature. *Transactions of the American Mathematical Society*, 22(1):84–100.

[Papazian and Rémila, 2002] Papazian, C. and Rémila, E. (2002). Hyperbolic recognition by graph automata. In *International Colloquium on Automata, Languages, and Programming*, pages 330–342. Springer.

[Păun, 2000] Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143.

[Peyrière, 1981] Peyrière, J. (1981). Processus de naissance avec interaction des voisins, évolution de graphes. In *Annales de l'institut Fourier*, volume 31, pages 187–218.

[Prusinkiewicz and Hammel, 1993] Prusinkiewicz, P. and Hammel, M. (1993). A fractal model of mountains and rivers. In *Graphics Interface*, volume 93, pages 174–180. Canadian Information Processing Society.

[Prusinkiewicz and Lindenmayer, 2012] Prusinkiewicz, P. and Lindenmayer, A. (2012). *The algorithmic beauty of plants*. Springer Science & Business Media.

[Queffélec, 1987] Queffélec, M. (1987). Substitution dynamical-systems-spectral-analysis. *Lecture notes in mathematics*, 1294:1–238.

[Queffélec, 2010] Queffélec, M. (2010). *Substitution dynamical systems-spectral analysis*, volume 1294. Springer.

[Róka, 1999] Róka, Z. (1999). Simulations between cellular automata on cayley graphs. *Theoretical Computer Science*, 225(1-2):81–111.

[Rozenberg, 1997] Rozenberg, G. (1997). *Handbook of graph grammars and computing by graph transformation*, volume 1. World scientific.

[Rozenberg and Salomaa, 1980] Rozenberg, G. and Salomaa, A. (1980). *The mathematical theory of L systems*, volume 90. Academic press.

[Salomaa, 1987] Salomaa, A. (1987). *Formal languages*. Academic Press Professional, Inc.

[Salzberg et al., 2004] Salzberg, C., Sayama, H., and Ikegami, T. (2004). A tangled hierarchy of graph-constructing graphs. In *Proc. Ninth International Conference on the Simulation and Synthesis of Living Systems (Artificial Life IX)*, pages 495–500.

[Sayama, 2007] Sayama, H. (2007). Generative network automata: A generalized framework for modeling complex dynamical systems with autonomously varying topologies. In *2007 IEEE Symposium on Artificial Life*, pages 214–221. IEEE.

[Schröder et al., 1998] Schröder, O. P., Schröder, P., Zorin, D., Derose, T., Forsey, D. R., W, V., Kobbelt, L., Lounsbery, M., Peters, J., Zorin, D., Schröder, P., Zorin, D., Peters, J., and Zorin, D. (1998). Siggraph 98 course notes subdivision for modeling and animation.

[Senechal, 1996] Senechal, M. (1996). *Quasicrystals and geometry*. CUP Archive.

[Smith et al., 2003] Smith, C., Prusinkiewicz, P., and Samavati, F. (2003). Local specification of surface subdivision algorithms. In *International Workshop on Applications of Graph Transformations with Industrial Relevance*, pages 313–327. Springer.

[Smith et al., 2011] Smith, D. M., Onnela, J.-P., Lee, C. F., Fricker, M. D., and Johnson, N. F. (2011). Network automata: Coupling structure and function in dynamic networks. *Advances in Complex Systems*, 14(03):317–339.

[Smith III, 1971] Smith III, A. R. (1971). Simple computation-universal cellular spaces. *Journal of the ACM (JACM)*, 18(3):339–353.

[Spicher and Giavitto, 2017] Spicher, A. and Giavitto, J.-L. (2017). Interaction-based programming in mgs. In *Advances in Unconventional Computing*, pages 305–342. Springer.

[Spicher et al., 2010] Spicher, A., Michel, O., and Giavitto, J.-L. (2010). Declarative mesh subdivision using topological rewriting in mgs. In *International conference on graph transformation*, pages 298–313. Springer.

[Srinivas, 1993] Srinivas, Y. V. (1993). A sheaf-theoretic approach to pattern matching and related problems. *Theoretical computer science*, 112(1):53–97.

[Tadaki and Kikuchi, 1994] Tadaki, S.-i. and Kikuchi, M. (1994). Jam phases in a two-dimensional cellular-automaton model of traffic flow. *Physical Review E*, 50(6):4564.

[Thue, 1906] Thue, A. (1906). Uber unendliche zeichenreihen. *Norske Vid Selsk. Skr. I Mat-Nat Kl.(Christiana)*, 7:1–22.

[Tomita et al., 2009] Tomita, K., Kurokawa, H., and Murata, S. (2009). Graph-rewriting automata as a natural extension of cellular automata. In *Adaptive Networks*, pages 291–309. Springer.

[Tullio Ceccherini-Silberstein, 2010] Tullio Ceccherini-Silberstein, M. C. a. (2010). *Cellular automata and groups*. Springer Monographs in Mathematics. Springer-Verlag Berlin Heidelberg, 1 edition.

[Wolfram, 1983] Wolfram, S. (1983). Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601.

Index