Characterisation of organisations for resilient detection of threats : a cluster of multiplicity

Erwan Beurier

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Mathématiques et leurs Interactions*

Par
Erwan BEURIER

Characterisation of organisations for resilient detection of threats

A cluster of multiplicity

Thèse présentée et soutenue à IMT Atlantique, campus de Brest, le mardi 3 novembre 2020
Unité de recherche : LabSTICC, Equipe MATRIX
Thèse N° : 2020IMTA0188

Rapporteurs avant soutenance :

Mathias BÉJEAN	Maître de conférences HDR, Université Paris 12, France
Isar STUBBE	Maître de conférences HDR, Université du Littoral-Côte d'Opale, France

Composition du Jury :

Président :	Alain HILLION	Professeur émérite, IMT Atlantique, France
Examinateurs :	Arnaud DURAND	Professeur, Université Paris 7, France
	Andrée EHRESMANN	Professeure émérite, Université de Picardie Jules Verne, France
	René GUITART	Maître de conférences HDR émérite, Université Paris 7, France
Rapporteurs :	Mathias BÉJEAN	Maître de conférences HDR, Université Paris 12, France
	Isar STUBBE	Maître de conférences HDR, Université du Littoral-Côte d'Opale, France
Dir. de thèse :	Dominique PASTOR	Professeur, IMT Atlantique, France
Co-dir. de thèse :	Roger WALDECK	Maître de conférences HDR, IMT Atlantique, France

Invité(s) :

Nicolas TABAREAU	Inria Senior Researcher	INRIA, IMT Atlantique, France

Acknowledgement

I am alone in making the transformation from man into god

Nile (2012) *Supreme Humanism of Megalomania*, At the Gate of Sethu.

Bon, c'est le moment de faire preuve d'humilité et de remercier ceux qui ont contribué, directement ou non, à la réussite de cette thèse. Mais je ne vais pas m'empêcher de raconter plein de bêtises. D'ailleurs, l'ordre d'apparition des noms ne retranscrit pas l'importance respective des personnes concernées, c'est juste l'ordre dans lequel ils me viennent à l'esprit. Il faut donc en vouloir à mes connexions neuronales.

Je voudrais d'abord remercier mes deux directeurs de thèse, Dominique Pastor et Roger Waldeck, pour m'avoir aiguillé (= supporté) pendant ces trois années, pour leur disponibilité, pour m'avoir présenté au monde de la recherche sous un jour réaliste, avec ses qualités et ses travers, pour leur disponibilité, leurs conseils, leur amabilité, et pour m'avoir accepté en thèse !

Je remercie infiniment ma famille, plus ou moins proche. Ce petit monde n'a jamais vraiment douté de ma réussite, car tous savaient que j'étais assez brillant pour cette tâche, mais je sais que j'aurais pu compter sur chacun d'entre eux si jamais j'avais eu des problèmes.

Il est aussi de bon ton de remercier mes deux meilleurs amis, Axel et Melina, que je remercie infiniment certes, mais d'un infini de l'ordre du cardinal fortement inaccessible, peut-être compact ou mesurable, pour avoir supporté mes détresses et m'avoir donné les coups de pieds au fesses dont j'avais parfois besoin. Du côté de l'écoute des détresses, mais pas des coups de pieds aux fesses, je remercie aussi Viviane Le Naour.

On va aussi citer d'autres amis, sinon ils vont m'en vouloir : Alexandre M., Andrés, Julien, Sébastien, Armand, Kimberly, Aimenallah, Nolwenn, Kawther, Morgane, Robin, et ceux que j'oublie, pour leur influence positive dans ma vie.

Les précaires des département LUSSI et Optique, avec qui j'ai passé de bonnes soirées, et qui continuaient à m'accueillir alors que j'avais arrêté d'aller dans des bars avec eux (en vrai je déteste les bars et l'alcool, mais c'est pas de votre faute).

Guillaume, mon frère de thèse, pour son support DISI/LaTeX/Python indéfectible. Les deux Thierry, Jean-Marc, les deux Christophe, Martine, Frédéric, Mo-

hammed et ceux que je voyais moins souvent aux pauses chocolatées et largement trop longues, où on a parlé pendant trois ans de la fusion *exemplaire* Telecom Bretagne et Mines de Nantes, de Travel Planet / GFD, et de mes habitudes alimentaires discutables...

Mes co-bureaux: Aurélien, co-bureau à 20%, pour son humour, et Duong, co-bureau à 100%, pour sa gentillesse innée.

Si je partageais mon quotidien et mes locaux avec tous ces gens-là, ils n'ont cependant pas eu grande influence sur mes travaux de thèse, si ce n'est un soutien moral dont j'avais grandement besoin. Je voudrais donc remercier les deux mathématiciens avec qui j'ai travaillé ces trois dernières années.

David Spivak, chercheur au MIT, qui a récemment été promu au rang de "mathématicien avec une page Wikipedia", pour m'avoir accueilli pendant deux semaines au MIT, pour avoir étendu mon category-*fu* et mon topos-*fu*.

Andrée Ehresmann, chercheuse émérite de l'Université de Picardie Jules Verne, depuis longtemps "mathématicienne avec une page Wikipedia", pour tout son soutien, pour ses explications, pour avoir toléré toutes les modifications que j'apportais à ses rédactions, pour m'avoir fait découvrir une facette plus concrète des catégories, et pour avoir plus ou moins redéfini la manière dont je visualise la vie d'un point de vue biologique, systémique (lisez *Memory Evolutive Systems* (2007), par Andrée Ehresmann et Jean-Paul Vanbremeersch, et on verra si ça ne change pas votre compréhension du monde).

Je remercie le jury, René Guitart, Mathias Béjean, Isar Stubbe, Nicolas Tabareau, Alain Hillion pour leur intérêt en mon travail, ainsi que leurs conseils et remarques pour l'améliorer. Remerciement spécial à Arnaud Durand, qui a accepté de faire partie du jury quelques semaines avant la soutenance, résolvant d'un coup des problèmes administratifs infinis qui semblaient d'un cardinal inaccessible.

Je vais aussi glisser un mot pour les quelques professeurs ayant contribué à faire le mathématicien que je suis : Pierre-François "pifzeprof" Berger, professeur de maths de MP*, Jean-Louis Garcin, professeur de maths de MPSI, et (c'est assez rare qu'on y pense), à Christian Allemand, professeur de maths de collège (désolé, je ne sais plus dans quelle(s) classe(s) je vous ai eu !). Chacun de ces messieurs a alimenté ma curiosité, ne se contentant pas d'un simple "tu verras ça en Master", probablement car, en plus de l'envie d'enseigner, ils avaient eux aussi une envie constante d'apprendre.

Et bien sûr, je remercie le Erwan Beurier du passé. Franchement, t'as été génial mec. Ta thèse est une tuerie. Si si, j'te jure ! Allez, fais pas le modeste... <3

Abstract

Force m'est de constater qu'il me devient de plus en plus difficile de faire coïncider mes levers tardifs avec mes couchers prématurés par le biais de mes siestes prolongées !

Basile Landouye (1992). *Léonard, Tome 22 : Cadeau de génie*, par Turk et Bob de Groot.

Résumé. Le point de départ de cette thèse est les réseaux de capteurs, et comment les rendre résilients. Notre approche utilise le langage de la théorie des catégories.

Nous abordons en premier lieu l'usage de systèmes dynamiques, et leur composition. Il s'avère que chaque système dynamique peut être décomposé en systèmes plus simples, dits réactifs, qui pourraient être des capteurs.

Dans une deuxième partie, nous cherchons à utiliser un langage catégorique utilisé pour la description de systèmes biologiques, naturellement résilients. Les systèmes biologiques présentent une forme de redondance fonctionnelle et non-structurelle. Cette propriété s'appelle *degeneracy*, et sa traduction catégorique, le *principe de multiplicité* (PM). Le PM nous semble donc être à la base de la résilience. Cependant, le PM requiert la notion de cluster, qui est en fait le nom des flèches dans les ind-catégories. Nous étudions donc la notion de cluster, exhibant de nouvelles propriétés et définitions (utilisant les composantes connexes de comma-catégories) que nous utiliserons pour trouver une caractérisation non-catégorique du PM dans le cas simple, mais important, des préordres.

Mots-clés : résilience, théorie des catégories, cluster, multiplicité, composantes connexes

Abstract. The starting point of this thesis is sensor networks, and how to instigate resilience in them. Our approach relies on category theory.

We first tackle the use of dynamical systems and their composition. We prove that every dynamical system may be decomposed into simpler, reactive systems, that could be seen as sensors.

In a second part, we use a categorical language first meant for biological systems, that are resilient by nature. Biological systems enjoy a form of functional, non-

structural redundancy that biologists call *degeneracy*. Category theorists translate it into the *multiplicity principle* (MP). MP seems to constitute a fertile ground for resilience. However, MP relies on the notion of cluster, which are the arrows of ind-categories. We thus study that notion of a cluster, exhibit some new properties and definitions, which use the connected components of the comma-cateogry, and that we use to find a non-categorical characterisation of MP in the special, simpler, but important case of preorders.

Keywords : resilience, category theory, cluster, multiplicity, connected components

Résumé long

En fait, quand vous avez commencé votre thèse il y a trois ans, vous ne saviez pas ce qui vous attendait. Ça aurait pu être du courage. C'est dommage que dans votre cas, ce soit de l'inconscience.

Arthur (2005). Kaamelott, saison 2, épisode 68
(enfin, à peu près)

Contexte

Les systèmes biologiques sont des systèmes complexes, avec nombre de propriétés et de fonctions. Pourtant, un animal, une plante, un insecte, le blobfish et le rat-taupe nu ne sont constitués que d'éléments simples, qui opèrent de manière asynchrone, pour faire fonctionner un organisme complet, alors que chaque cellule n'a accès qu'à une information restreinte (son environnement proche).

Parmi toutes les propriétés du vivant, l'une en particulier attire notre attention, tant elle semble propre aux entités naturelles : la résilience. Il s'agit de la capacité des systèmes naturels à continuer de fonctionner alors que certaines parties du corps pourraient ne plus répondre normalement, ne plus être fiables. Le fonctionnement se fait éventuellement en mode dégradé. Le monde vivant regorge d'exemples de résilience. Lorsque l'on est malade, par exemple, d'un rhume, l'organisme continue de fonctionner malgré les symptômes (nez qui coule, fatigue, toux). La résilience permet à l'organisme de survivre en attendant une réparation.

Le système immunitaire, d'ailleurs, est un bon exemple de système résilient. Les constituants, de nouveau, sont des cellules qui communiquent entre elles via des molécules appelées interleukines. Il réagit continuellement à de l'inconnu, les agents pathogènes, et est capable de garder en mémoire les anticorps qu'il produit (c'est le principe de la vaccination).

A contrario, les systèmes créés par des humains peuvent présenter de la mémoire (ordinateurs), mais la résilience ne semble pas être une propriété atteignable.

Peut-on s'inspirer du comportement des systèmes biologiques pour concevoir des systèmes artificiels résilients ?

L'objectif de cette thèse est de proposer une approche mathématique de la

résilience, dans l'optique de présenter des fondements pour une *mathématique de la résilience*.

Notre traitement se base particulièrement sur la théorie des catégories. Cette branche des mathématiques réputée pour son caractère abstrait, présente néanmoins de plus en plus d'applications concrètes, que ce soit en biologie, en ingénierie, en informatique, en géométrie algébrique, en statistiques. De plus, la théorie des catégories semble être un langage adéquat pour modéliser et formaliser les systèmes. Cette thèse s'inscrit dans cette volonté d'application, voire d'interface entre la théorie des catégories appliquée à la biologie, les statistiques, et les systèmes complexes (le but était d'ailleurs d'aboutir à des réseaux de capteurs résilients).

Le premier chapitre est une introduction à la théorie des catégories. Nous proposons donc une introduction succincte et ciblée. On définit les notions de base : catégorie, foncteur (qui peut être défini comme un morphisme entre catégories), transformation naturelle. Les différentes propriétés de morphismes: isomorphisme, monomorphisme, épimorphisme. Le foncteur Hom-set. Les tailles de catégories. Nous introduisons ensuite les catégories monoidales et les foncteurs monoidaux. Ensuite, nous nous étendons un peu plus sur les notions de diagramme, cône, cocône, limite et colimite, catégories filtrées et cofiltrées, complètes et cocomplètes. Nous introduisons ensuite les limites et colimites standards : objets initial et terminal, produit et coproduit, égaliseur et coégaliseur, produit fibré et somme amalgamée. Ces notions seront illustrées à chaque fois par leur application dans la catégorie des ensembles, et dans une catégorie de type "préordre". Cette introduction se termine par le calcul explicite des limites et colimites dans le cas particulier de la catégorie des ensembles, en utilisant la construction des limites à partir des produits et égaliseurs (et colimites à partir des coproduits et coégaliseurs). On obtient deux formules générales qui seront utilisées plus loin dans la thèse.

Systèmes dynamiques

Nous nous sommes d'abord intéressés à l'émergence de la mémoire. La communication entre les cellules du système immunitaire, mentionnée plus haut, semble être la source de la capacité de mémoire. Notre premier résultat confirme cette hypothèse. Nous introduisons un formalisme existant, issu de la théorie des catégories, pour présenter les systèmes dynamiques, ici, une variante des automates. Un système dynamique est présenté comme un triplet constitué d'un ensemble d'états, une fonction de changement d'états en fonction des entrées et de l'état courant, et une fonction de production de sortie, en fonction de l'état courant. Ces automates dépassent le cadre de la calculabilité car l'espace des états peut être aussi grand que l'on veut, donc éventuellement infini, e.g. avec la puissance du continu.

Les automates sont placés dans un cadre, qu'on nomme "boîte". Les boîtes im-

posent les entrées et sorties des systèmes dynamiques qui sont placés dedans. Une de leurs composantes importantes, est leur capacité à se composer, en série ou en parallèle, et à s'imbriquer pour former des systèmes de plus en plus complexes. On a alors une algèbre, ou plutôt, une catégorie monoidale (pour la mise en parallèle), constituée des boîtes et des branchements qu'elles peuvent subir.

Cette catégorie vient avec un foncteur "naturel". Ce foncteur associe à une boîte, l'ensemble des systèmes dynamiques qui peuvent habiter dedans. Les opérations de parallélisation de boîtes se traduisent alors en un produit de systèmes dynamiques. En d'autres termes, ce foncteur est monoidal.

Bien sûr, toutes ces notions sont expliquées au moyen d'exemples, afin que le lecteur non familier puisse visualiser les objets manipulés.

Profitant de ce cadre, nous définissons des systèmes dynamiques sans mémoire. En effet, l'espace des états des systèmes dynamiques est vu comme la mémoire des évènements passés qui s'accumule, et l'état courant est en fait compris comme le contenu de la mémoire à un instant donné. Donc, on peut définir une classe de systèmes dynamiques dont la fonction de transition ne prend pas en compte l'état courant. C'est ce que nous appelons un système sans mémoire.

De tels systèmes deviennent alors *réactifs*, dans le sens où ils ne stockent aucune information, ils ne font que réagir à leurs entrées. Dans le cadre de cette thèse, les systèmes sans mémoire pourraient être de simples capteurs. Nous prouvons alors que de tels systèmes, avec les bons branchements, peuvent être rendus équivalents à des systèmes dynamiques généraux, avec mémoire. Nous retrouvons ici l'hypothèse que la mémoire réside dans les connexions.

Ce travail ouvre de nouveaux problèmes : en effet, si un système avec mémoire est équivalent à un système sans mémoire avec les bons branchements, le système sans mémoire n'est pas forcément "plus simple". Une telle décomposition pourrait se faire de manière plus optimale, par exemple en limitant le nombre d'états possibles, ou en limitant le nombre de branchements. Cette question n'est pas traitée dans cette thèse.

Multiplicité et clusters

Si la partie précédente s'attelait à imiter la gestion de la mémoire dans le système immunitaire, soulignant l'importance des connexions entre diverses entités, cette partie s'occupe plutôt de la résilience plus générale.

Comme mentionné plus haut, la théorie des catégories a été très tôt appliquée à de la modélisation des systèmes biologiques, notamment chez Rosen, dans *The Representation of biological systems from the standpoint of the theory of categories* (1958). Dans cette partie, nous nous inscrivons dans cette démarche. Le but est

d'extraire de cette étude une marche à suivre à appliquer à des réseaux de capteurs.

La vision catégorique de cette seconde partie est différente de celle de la première partie. En effet, au lieu de définir une catégorie de boîtes et un foncteur qui définit un ensemble de systèmes dynamiques, ici, les catégories ont une autre signification. Une catégorie représente la configuration d'un système quelconque à un instant donné (dans l'idéal, un système biologique ou complexe de quelque nature que ce soit). Un élément du système est un objet de la catégorie, une flèche entre deux objets dénote une interaction, et un diagramme représente un sous-système.

On peut dès lors tenter de définir des conditions qui mènent à la résilience. Celle explorée ici repose sur le principe biologique de *degeneracy*. Le nom est trompeur car désigne la capacité d'un système à proposer plusieurs solutions pour accomplir une même fonction. Il s'agit d'une redondance fonctionnelle, et non pas matérielle et/ou structurelle, contrairement à ce que l'on retrouve en ingénierie. Un sous-système peut arrêter de fonctionner, mais sa fonction principale sera assurée (peut-être de manière non-optimale) par un autre sous-système dont ce n'est pas la fonction principale. Le remplacement dure suffisamment longtemps pour que le système ait le temps de se réparer, afin que le premier sous-système récupère sa fonction.

On doit alors introduire de nouvelles notions catégoriques, la principale étant celle de *cluster*[1]. Si un diagramme représente un sous-système du système principal, un cluster représente les interactions entre deux sous-systèmes. Les chapitres suivants se basent partiellement sur un livre d'audience multidisciplinaire intitulé *Memory Evolutive Systems* (2007) par Andrée Ehresman et Jean-Paul Vanbremeersch. Nous entamons alors le premier chapitre avec une exploration, voire une étude, de la définition de cluster.

Nous introduisons la notion d'ind-catégorie (c'est-à-dire, une cocomplétion d'une catégorie). Mathématiquement, un cluster entre deux sous-systèmes est une variante des flèches dans la ind-catégorie du système. Par le calcul, nous obtenons une première version de définition d'un cluster : il s'agit d'un ensemble de flèches, union disjointe d'un tuple de classes d'équivalence de flèches.

Mais nous ne nous arrêtons pas là. Le but est d'extraire plusieurs définitions équivalentes de cluster. Nous trouvons deux nouvelles définitions, plus descriptives. Un cluster est alors un ensemble de flèches vérifiant certaines propriétés, au choix parmi deux listes ayant certains critères en commun.

Pour finir, étudions deux nouvelles propriétés des clusters. En effet, un cluster peut aussi être compris comme un certain type de préfaisceau, qui associe un élément

1. En français, le terme "cluster" est utilisé principalement pour de la classification de données, ou en intelligence artificielle. Ici, ce n'est pas le cas. Le mot "cluster" est le terme anglais. Le terme utilisé par Ehresmann et Vanbremeersch est plutôt "gerbe" (comme une "gerbe de flèches"), ce qui se traduit en "sheaf" en anglais, ce qui est malheureusement un terme réservé. J'ai donc choisi de garder le terme de "cluster", puisque je ne parle jamais d'intelligence artificielle dans ce manuscrit.

du domaine du premier diagramme, à un ensemble de composantes connexes. Le nombre et la qualité des composantes connexes sont intimement liés aux définitions descriptives mentionnées plus haut. Nous terminons ce chapitre avec une conjecture sur le cardinal de l'ensemble des clusters.

La notion de cluster semble être intimement liée à la structure de graphe des catégories concernées.

Pour le chapitre suivant, la définition descriptive de cluster sous forme d'ensemble de flèches semble être plus appropriée. En effet, dans le-dit chapitre, nous voyons comment construire un cluster en tant qu'ensemble de flèches. Nous introduisons la notion de protocluster (qui peut être vu comme un ensemble de flèches entre deux diagrammes, éventuellement avec quelques propriétés) ainsi que la notion de *protocluster plein* (ensemble de toutes les flèches possibles entre deux diagrammes). Un tel protocluster possède des propriétés intéressantes : en effet, un cluster est forcément inclus dedans. Nous proposons alors un théorème permettant de déterminer, à l'aide du protocluster plein, s'il y a des clusters entre deux diagrammes. On peut déjà dire que s'il manque des flèches dans le protocluster plein, il n'y aura pas de cluster. Par exemple, si un objet du premier diagramme n'a pas de flèche allant dans le deuxième diagramme, alors il ne peut pas y avoir de cluster entre le deuxième diagramme. Ceci peut se déduire en observant le protocluster plein. Enfin, nous réutilisons la notion de cluster comme foncteur vue au chapitre précédent, et déduisons une condition nécessaire et suffisante pour qu'un cluster existe, en fonction des propriétés du protocluster plein.

Cette étude concernait les clusters dans les catégories générales et sert de prérequis au chapitre suivant. En effet, l'étude sur les clusters était nécessaire afin de bien aborder le problème de la degeneracy. Son équivalent catégorique est le Principe de Multiplicité (PM). Catégoriquement, on dit que deux diagrammes vérifient le PM lorsque les deux propriétés suivantes s'appliquent :

1. Les diagrammes ont des cocones isomorphes
2. Soit il n'y a pas de clusters entre ces deux diagrammes, soit les clusters ne définissent pas d'isomorphisme entre les cocones

La deuxième condition est formalisée de la manière suivante. Un cluster définit un foncteur de composition. En prenant un cocône du second diagramme, on peut le composer avec un cluster, et cela donne un foncteur de composition. Pour qu'une catégorie vérifie le PM, il faut que ce foncteur de composition ne définisse pas un isomorphisme.

L'étude du cas général nous semblait hors de portée. Nous nous concentrons sur le cas du PM dans les préordres. Dans de telles catégories, l'isomorphisme de cocônes induit par les clusters est plus facile à étudier. On obtient alors une caractérisation du PM dans les préordres.

Deux diagrammes dans un préordre vérifient le PM lorsque l'une des propriétés suivantes est vraie :

1. Soit il n'y a pas de cluster entre les deux diagrammes, ce qui peut se vérifier en utilisant les résultats du chapitre précédent, notamment en vérifiant les propriétés du full protocluster
2. Soit en vérifiant que les cocônes des deux diagrammes sont isomorphes, mais différents

Nous utilisons ce résultat pour repérer le MP dans diverses catégories.

Le chapitre suivant, par exemple, introduit un préordre comparant les capacités de différents tests statistiques : les tests de Neyman-Pearson (NP), et les test RDT (Random Distorsion Testing), qui bien que structurellement différents, accomplissent la même fonction. Les tests NP sont optimaux en termes de probabilité de détection, mais très sensibles aux interférences. Les tests RDT, quant à eux, sont optimaux dans un sens un peu moins fort, mais très robustes vis-à-vis des interférences. Ce résultat suggère donc de combiner des tests NP et RDT dans des systèmes de capteurs, car les deux se complètent bien, au sens de la *degeneracy*.

Contributions

Nous listons ici les résultats nouveaux présentés dans cette thèse.

Le chapitre 2 est une introduction à la théorie des catégories ; aucun résultat présenté ici n'est nouveau.

Le chapitre 3 contient une longue partie consacrée à la présentation du formalisme catégorique des systèmes dynamiques. La section 3.4 contient uniquement des résultats nouveaux.

Les deux chapitres suivants sont consacrés aux clusters, ce que c'est, et comment on juge de leur présence ou absence. Le chapitre 4 contient des résultats connus (le lien avec les ind-catégories, la description de cluster comme dual des flèches de la procatégorie) et des résultats nouveaux : les deux définitions sous forme d'ensembles de flèches vérifiant certaines conditions, ainsi que les liens avec les foncteurs de composantes connexes.

Tout le chapitre 5 sur la construction de clusters est nouveau en termes de résultats. Enfin, le chapitre 6, sur le MP dans les préordres, est aussi nouveau.

Perspectives

En l'état, cette thèse présente des travaux très abstraits, ainsi qu'un résultat presque applicable. En effet, dans la deuxième partie de cette thèse, nous nous sommes concentrés sur l'un des outils principaux de la théorie, à savoir, les clusters. Bien que le résultat de multiplicité des tests statistiques soit assez proche de l'applicabilité, il serait souhaitable de présenter un exemple concret d'architecture de réseau de capteurs utilisant cette propriété.

Par ailleurs, ce résultat, démontré pour les tests "en bloc", devrait pouvoir être étendu aux tests séquentiels. En effet, nous émettons la conjecture que l'équivalent séquentiel des tests de Neymann-Pearson, ainsi que l'équivalent séquentiel des tests RDT, vérifient eux aussi une multiplicité. Cette question n'est pas traitée dans cette thèse.

Enfin, l'étude pourrait être poursuivie pour des systèmes plus généraux, car la résilience est une propriété souhaitable à n'importe quel système complexe. Le réseau de capteurs résilient évoqué plus haut pourrait alors faire partie du système complexe, par exemple sous la forme d'un système de surveillance interne (un système immunitaire, en somme), vérifiant à chaque instant le maintien des variables dans les seuils voulus.

De la même manière, un réseau de capteurs résilient pourrait être particulièrement utile dans toutes les applications liées à la cyber-sécurité, remplaçant alors les capteurs par des antivirus, tout en conservant la propriété de résilience.

L'ambition de cette thèse était aussi de poser les fondations d'une mathématique de la résilience, qui étudierait mathématiquement les moyens de doter les systèmes industriels de résilience.

Table of Contents

Part I

Introduction and a warm welcome to category theory

Chapter 1

Introduction

Il y a des gens qui ont pris la peine de faire une thèse, la moindre des choses, c'est de rester pour la lire !

Dame Séli (2005). Kaamelott, saison 1, épisode 2
(enfin, à peu près)

1.1 Context

Biological systems are complex systems, achieving a number of properties and functions. And yet, an animal, a plant, an insect, a blobfish or a fly amanita are made only of simpler elements, cells. They work together asynchronously in order to make a whole organism work, while each cell only has partial information at each instant (its close environment).

Among all the properties of the living, we focus on one in particular, one that seem inherent to natural entities: resilience. Resilience is the ability of natural systems to keep working at an acceptable level, even while some of its parts could be non functioning (due to an injury for example) or non reliable. The natural entity may keep working in a degraded mode. Nature is overflowed with examples of resilience. For instance, when one is sick, for example with a cold, the organism keeps functioning in spite of the symptoms (running nose, cough, fatigue, fever). Resilience lets the organism survive while it is being repaired.

The immune system is a great example of resilient system. Its constituents again are cells. Those cells communicate together via molecules, called interleukines. The immune system is constantly facing both internal and external threats, both known and unknown pathogens, and has the capability to store and remember the antibodies it produces in order to deal with them. This is how inoculation works.

On the contrary, systems created by humans may have memory (computers), discover and adapt to the world (artificial intelligence), however resilience does not seem attainable just yet. So, can we adapt the behaviour of biological systems in order to design resilient artificial systems?

The goal of this thesis is to propose a mathematical approach of resilience, with the will to develop the fundaments of the mathematics of resilience.

Our approach is based on category theory (CT). This branch of mathematics was created (or discovered) by Eilenberg and MacLane [1] between 1942-1945. Although being very abstract and potentially a part of foundations of mathematics [2], category theory may be applied to several branches of mathematics (topology, homotopy theory, algebraic geometry, logic) and also to broader, more applicative sciences: engineering [3][4], many areas of computer science [5][6][7][8], quantum physics, neuroscience [9] and biology [10][11][12]. See [13] for a longer list of applications of category theory. These applications rely on the high capability of CT for modelling and designing patterns:

> *[I intend to show that] category theory is incredibly efficient as a language for experimental design patterns, introducing formality while remaining flexible.* ([14, Chapter 1, Section 1.2])

This thesis is within the scope of applied category theory, and maybe more in the will to promote and develop and interface between biology, statistics and complex systems. Our goal is to provide a framework allowing to endow sensor networks with resilience, taking inspiration from natural resilience.

In this context, we target at a multidisciplinary audience with little to no background on category theory. Thus, the first chapter of this thesis is a concise and self-sufficient introduction to category theory. We define the most basic notions: category, functor (morphism between categories), natural transformations (morphisms between functors). The different properties of morphisms: isomorphism, monomorphisms, epimorphisms. The Hom-set functor. Sizes of categories. We then introduce the less basic notions of monoidal categories and monoidal functors. Finally, we spend some time on the notions of diagrams, cones and cocones, limits and colimits, filtered and cofiltered categories, complete and cocomplete categories. We give examples of standard limits and colimits: initial and terminal objects, product and coproduct, equaliser and coequaliser, pullback and pushout. Throughout this whole chapter, we give examples for two basic examples of categories: that of sets, and a preorder. We conclude this chapter constructing general formulas for limits and colimits in the category of sets, using the proof of a weel-known lemma, seeing limits as an equaliser between two product arrows. These formulas will be used in Chapter 4 as the starting point of the definition of clusters (see below).

1.2 Dynamical systems

We first worked on the emergence of memory. Communication is key in the immune system, as memory seems to be a consequence of the way cells communicate.

One of the strengths of the immune system is its memory. The immune system contains naive T-cells that have a whole set of specific receptors. When a given organism meets an antigen, the naive T-cells "try" to react to the antigen. Those that do react begin a self-promoted division, creating the first specific immune response. Then some of the dividing T-cells become "quiescent", i.e. waiting for the next occurrence of the antigen. The next time the organism meets the antigen, it will react much faster. This is the principle of inoculation.

The previous paradigm was that memory is maintained by T-cells created during the first contact of the organism with a pathogen. In [15], memory is shown to also depend on dividing T-cells. In fact, dividing T-cells constitute a network of communicating cells, promoting their own division. The protocol is as follows:

1. First, create the memory. The immune system of mice encounters a pathogen (here, transplants and virus antigens). Dividing T-cells are created, some become quiescent T-cells.
2. Secondly, remove the dividing T-cells. This is done by injecting a certain toxin promoting cell suicide. Note that the quiescent T-cells are still present. According to the previous paradigm, the mice should still be "vaccinated".
3. Thirdly, expose again the mice to the pathogens (transplants or virus). It is observed that the immune response is that of a naive immune system, rather than that of an experienced (vaccinated) immune system.

As the dividing T-cells act in a communicating network of similar cells, and seem to be necessary for memory to be attained, one might conclude that memory relies in the communication.

Our first result confirms this hypothesis. In Chapter 3, we introduce an existing formalism, based on category theory, to describe dynamical systems, here a variant of automata. A dynamical system is seen as a tuple consisting of a state space, a state-changing function that depends on the inputs and current state, and a function producing an output from the current state. These automata are more general than the standard automata, including Turing machines, because their state spaces may be as big as we want, including bigger than the continuum.

Automata are set into a frame, called a "box". Boxes impose input and output types of the dynamical systems that are inside. One their important properties is that they can be composed at will, in series or in parallel, allowing for a fractal-like intertwining of boxes and systems, making more and more complex systems. Thus, we obtain an algebra, or rather, a monoidal category, as introduced in Chapter 2, consisting of boxes and their wirings.

This category comes with a "natural" functor that sends a box to the set of all dynamical systems that fit inside. The parallelisation operations on boxes then becomes a "product" of dynamical systems. In other words, this functor is monoidal.

Of course, all these notions come with several examples to illustrate and explain visually what is happening at each step, for the reader non-familiar with this framework.

The state space of the dynamical system is seen as its memory, as a stacking sequence of past events. This naturally induces the definition of a memoryless system: a system whose transition function ignores the current state of the system. As a consequence, its output only relies on the input. Such a system could be called "simple-reflex", we call them "memoryless". Another word could be "reactive", in the sense that memoryless systems do not store any information, and thus, only always react to their perception of the world.

We prove that memoryless systems, with the right wirings, can be made equivalent to any general dynamical system. We thus recover the hypothesis that memory is a consequence of connections.

This work open new perspectives. If a dynamical system is equivalent to a memoryless system with the right wirings, the latter is not necessarily "simpler". Such a decomposition would be done in a more optimal way, for example with a limit of size on the state space, or a limit in the number of wirings. This question is left untreated.

1.3 Multiplicity and clusters

This part of the thesis aims at studying resilience in the most general sense.

As mentioned above, category theory was soon applied to biological systems modelling, first promoted by Robert Rosen [10] [11], pursued by Andrée Ehresmann and Jean-Paul Vanbremeersch [16] [12].

In this part, we follow this work. The goal is to study and extract a procedure to use when designing sensor networks.

The categorical vision of this section is quite different to that of the previous part. In fact, instead of defining a category of boxes and a functor that associates a set of dynamical systems to a box, here, categories bare another meaning. A category represents the configuration of system at a given time (ideally, a biological system, or any complex system of any kind). An element of the system is an object of that category, and an arrow between two objects denotes an interaction. A diagram represents a subsystem of the whole system. Colimits are used in order to introduce a hierarchy in a complex system, the hierarchy being different levels of zooms (for example, a biological system is made of organs, made of tissues, made of cells, made of proteins, made of molecules, made of atoms). Then, that system (that category) evolves through time, and each configuration is linked to another via a partial functor. It is then possible to model the evolution of a component of the

system through time. This approach is fully developed in [12], on which we base this part.

We can then define the conditions that may lead to resilience. The one explored here is based on the biological principle of *degeneracy* [17]. The term may read misleading, but it actually conveys the idea of a system presenting two or more solutions in order to carry a given function. It is a functional redundancy, rather than a structural and/or material redundancy, of the kind we often see in engineering. In other words, a subsystem could stop working, but its principal function may be fulfilled by another subsystem, whose principal function is generally different. The replacement may last until the system can recover (for example, from an injury). [17, Table 1, page 13764] gives many examples. We give here a selection:

1. Protein folds. A protein is a sequence of aminoacids that fold several times to yield a certain spacial shape, which determines part of its function. Several sequences of aminoacids may lead to similar folds, yielding similar functions.
2. Metabolism. Cells, tissues, organs may have different, parallel metabolic pathways. For instance, cells may consume ATP, sugar (glycogen) or fat as energy.
3. Body movements. The body of humans, and most vertebrates, is filled with different muscles. Subsets of muscles work together. For example, in the human arm, the biceps, brachialis and brachioradialis are all responsible for the flexion of the elbow. Athletes that tear their biceps (this may happen while deadlifting heavy weights with the alternate grip for example) may still perform flexion of the elbow thanks to the other two muscles.

Degeneracy then occurs at every level of the organism.

In order to continue the thesis, we have to introduce several more notions of category theory, the principal being that of a cluster. If a diagram represents a subsystem of the whole system (at a given instant), then a cluster represents all the interactions between two subsystems. As [12] is intended for multidisciplinary audience, the description of clusters can lead to confusion. The first chapter of this part, Chapter 4, explores the definition of a cluster.

To this end, we introduce the notion of ind-category (that is, a kind of cocompletion of a category). Mathematically speaking, a cluster between two subsystems is a variant of the arrows in the ind-category of the system. Through computation, we obtain a first version of a cluster: it is a disjoint union of equivalence classes of arrows.

This is not all. We aim at finding several definitions of clusters. We find two new definitions, more descriptive. A cluster becomes a set of arrows verifying certain properties, in a choice of two lists having common criteria.

Finally, we conclude with two new properties of clusters. In fact, a cluster may be seen as a certain presheaf, sending an object to a connected component of a certain

comma-category. The number and nature of the connected components are closely linked to the two descriptions given above.

A cluster then seems to be linked to the graph structure of the categories under study.

In Chapter 5, we continue with clusters. One of the two descriptive definitions seems more useful than the rest in this chapter. Indeed, we see how to construct a cluster as a st of arrows. We introduce the notion of protocluster (which is a generic term for a subset of arrows between two diagrams, possibly with more properties), together with that of full protocluster (the set of all arrows between two diagrams). Such a protocluster has interesting and useful properties for our purpose; the first of them being that any cluster is a specific subset of them. We prove a theorem linking the properties of the full protocluster with the existence or non-existence of clusters. For example, if some object of the domain diagram has no arrow to the other diagram, then there will be no cluster at all. Then, using the functorial properties of clusters seen in Chapter 4, we deduce a necessary and sufficent condition for a cluster to exist, based on the observation of the full protocluster.

This study in two chapters tackles the subject of clusters in general categories. It is a prerequisite for Chapter 6. The study of clusters was necessary in order to deal with the problem of degeneracy (the property of a system to have several subsystems achieving the same function). Its categorical description is called the Multiplicity Principle (MP). Categorically, we say that two diagrams verify the MP when the following two statements hold:

1. The two diagrams have isomorphic cocone categories
2. Either there is no cluster between the two diagrams, or none of them defines an isomorphism

In order to make the second condition formal, we define a functor between the two cocone categories of the two diagrams. This functor sends a cocone from a category to the composite of the cocone and the cluster. We call this functor the cluster-composition functor. The second condition imposes that such a functor never is an isomorphism.

The general case seemed out of our reach. We thus reduces the study to the case of a preorder category. In such a category, the isomorphism induced by clusters is easier to study. We thus obtain a characterisation of MP in preorders.

Two diagrams in a preorder verify the MP when one of the following properties holds:

1. Either there is no cluster between the diagrams, which translates into properties that the full protocluster may have or not
2. Either the cocone categories are isomorphic, but the cocones have different

peaks.

We then use this result in order to detect the MP in several categories.

Chapter 7 uses this result to find multiplicity between Neymann-Pearson (NP) tests and RDT tests, obtained in an article coauthored by the author of this thesis. Here, the multiplicity suggests aa functional equivalence between the two classes of statistical tests, although being designed for different statistical problems and having different properties. The NP tests are optimal among tests, however they need to be closely adapted to the signal they are searching for. RDT tests are optimal, but for a different property, and they are robust. Thus, this result suggests that a combination of RDT and NP tests should lead to a resilient detection.

1.4 Contributions

We list here all the results that are new in this thesis.

Chapter 2 is a basic introduction to category theory; no result introduced here is new.

Chapter 3 is based on an article we published [18]. The categorical framework for dynamical systems is based on a previous work from one of the coauthors [19]. Section 3.4 only contains new results.

The following chapters tackle the topic of clusters: their definitions, their existence. Chapter 4 is a compilation of existing work from renown categoricists (the link with ind-categories, the description of arrows of pro-categories). The two descriptive definitions of clusters as sets of arrows, and the link with the presheaf of connected components are new. The conjecture on the cardinal of clusters has, as far as I know, never been tackled yet.

Chapters 5 and 6 mainly consist of new results.

Chapter 7 consists of a brief introduction to statistical decision testing, and the results presented here are from our conference article [20].

Other work related to the PhD, but not relevant for this manuscript, will be mentioned in the conclusion (Chapter 8).

Chapter 2

A mere crash course on category theory

We will need to use some very simple notions of category theory, an esoteric subject noted for its difficulty and irrelevance

Moore, Seiberg (1989) [21]
J'avoue, c'est peu original, parce que tous les catégoriciens citent cette phrase !

2.1 Introduction

This thesis is intended to readers with signal processing background, as this PhD initiated in this domain. The use of category theory became necessary, first when dealing with dynamical systems (see Chapter 3) and secondly, when studying resilience in a more general context (see [12] and Chapters 4, 5 and 6).

In this chapter, we introduce all the categorical background that signal processors need to understand this thesis. Most of the notions will be illustrated with mathematical examples, using mainly background from set theory and the theory of preorders. We focus here on the bare essential, which consists in categories, functors, natural transformations, types of morphisms, limits/colimits and their usual special cases, a bit of monoidal categories. Most other basic notions will not be described in these pages (monads, adjoints, presheaves, topoi...).

2.2 Basic notions

This section will introduce some basic notions about category theory: categories, functors, opposite categories, natural transformations, monomorphisms and epimorphisms.

The following constitute the basic knowledge of a category-theorist. Renown

resources on category theory include [22], [14], [23], [24], [25] and [26] (ordered by personal, perceived difficulty). The definitions and lemmas we give here are a synthesis of such references.

Definition 2.2.1 (Category)

A *category* $\mathscr{C}$ consists of the following data:

- A collection of *objects*, denoted $\mathrm{Ob}_{\mathscr{C}}$
- A collection of *morphisms*, or *arrows*, denoted $\mathrm{Mor}_{\mathscr{C}}$
- A map $\mathrm{dom} : \mathrm{Mor}_{\mathscr{C}} \to \mathrm{Ob}_{\mathscr{C}}$; for each morphism c, $\mathrm{dom}(c)$ is called the *domain* of c
- A map $\mathrm{cod} : \mathrm{Mor}_{\mathscr{C}} \to \mathrm{Ob}_{\mathscr{C}}$; for each morphism c, $\mathrm{cod}(c)$ is called the *codomain* of c
- For each morphism $c \in \mathrm{Mor}_{\mathscr{C}}$, we write $c : C \to C'$ if $C = \mathrm{dom}(c)$ and $C' = \mathrm{cod}(c)$
- A *composition law* $\circ$ such that, for all $c : C \to C'$ and $c' : C' \to C''$, there is a chosen morphism $c' \circ c : C \to C''$
- For each object $C \in \mathrm{Ob}_{\mathscr{C}}$, there is a chosen morphism $\mathrm{id}_C : C \to C$ called *identity morphism of* C

The composition law is required to be associative: $\forall C_1, C_2, C_3, C_4 \in \mathrm{Ob}_{\mathscr{C}}$, $\forall c_1 : C_1 \to C_2$ and $c_2 : C_2 \to C_3$ and $c_3 : C_3 \to C_4$, $(c_3 \circ c_2) \circ c_1 = c_3 \circ (c_2 \circ c_1)$. Identity morphisms are required to act like identities: $\forall C, C' \in \mathrm{Ob}_{\mathscr{C}}$, $\forall c : C \to C'$, $c \circ \mathrm{id}_C = \mathrm{id}_{C'} \circ c = c$.

In the rest of the course, a category $\mathscr{C}$ will be described according to the following presentation:

Objects: An object in $\mathscr{C}$ is...

Morphisms: A morphism in $\mathscr{C}$ is...

Identities: An identity morphism is...

Composition: The composition law for morphisms is...

Usually, the description of morphisms suffices to implicitly define dom and cod, as in the following examples.

Example 2.2.2 (Category of **Sets**). One of the easiest categories is the category of sets. We define the category **Sets** as the following:

Objects: An object in **Sets** is any set X

Morphisms: A morphism in **Sets** is any function $f : X \to X'$

Identities: An identity morphism is an identity function $\mathrm{id}_X : X \to X$

Composition: The composition law for morphisms is the usual composition of functions

Example 2.2.3 (Preorder category). Another different but useful example of category is the category based on a preorder. If $(P, \leqslant)$ is a preordered set (we will refer to this as a *proset*), then we can define the following category:

Objects: The objects are the elements of the set P

Morphisms: There is an arrow $p \to q$ if and only if $p \leqslant q$

Identities: An identity morphism is an arrow $p \to p$ representing the trivial equality $p = p$

Composition: The composition law for morphisms is the transitivity of the preorder $\leqslant$: if $p_0 \to p_1$ and $p_1 \to p_2$ then the transitivity of $\leqslant$ implies that $p_0 \to p_2$

Note that here, the arrows have a very different meaning to the ones in **Sets**. Arrows are not at all similar to functions, but rather the representation of the preorder. Note that there is at most one arrow between two objects in the proset.

This example will be useful not to base our intuition only on the category of **Sets**; **Sets** is a very nice category with lots of properties and examples, however, it does not represent the "generic" category. There are categories that behave differently and we need examples of them.

Notation 2.2.4. As stated in Definition 2.2.1, a category is mainly composed of two pieces of data: objects and arrows. For the sake of readability, for a category $\mathscr{C}$, we write $C \in \mathscr{C}$ instead of $C \in \mathrm{Ob}_{\mathscr{C}}$ and $c : C \to C' \in \mathscr{C}$ instead of $c : C \to C' \in \mathrm{Mor}_{\mathscr{C}}$. In other words, a variable C or $c : C \to C'$ implicitly states its type (object or arrow). This allows for more compact notation.

Definition 2.2.5 (Hom-set)

Let $\mathscr{C}$ be a category, and let C and C' be two objects of $\mathscr{C}$. We denote by $\mathrm{Hom}_{\mathscr{C}}(C, C')$ the collection of arrows $C \to C'$ in the category $\mathscr{C}$.

Example 2.2.6 (Hom-sets in **Sets**). In the category **Sets**, X and X' are two sets, and $\mathrm{Hom}_{\mathbf{Sets}}(X, X')$ is the set of functions $f : X \to X'$.

Example 2.2.7 (Hom-sets in a proset). In a proset $(P, \leqslant)$, the hom-set is defined by: $\mathrm{Hom}_P(p, q) = \{(p, q)\} \Leftrightarrow p \leqslant q$; otherwise, $\mathrm{Hom}_P(p, q) = \varnothing$.

Let us study some properties of the arrows of a category. We start by considering isomorphisms and will then study weaker properties (the categorical equivalents of surjective and injective functions).

Definition 2.2.8 (Isomorphism)

Let $\mathscr{C}$ be a category. A morphism $c : C \to C' \in \mathscr{C}$ is an *isomorphism* when there exists $c' : C' \to C \in \mathscr{C}$ such that $c' \circ c = \mathrm{id}_C$ and $c \circ c' = \mathrm{id}_{C'}$. Such a c' is denoted c^{-1}.

Example 2.2.9 (Isomorphisms in **Sets**). An isomorphism in **Sets** is a function that is invertible. Thus, an isomorphism in **Sets** is a bijection.

Example 2.2.10 (Isomorphisms in a proset). In a proset category, there is at most one arrow $p \to q$. Thus, an arrow is an isomorphism whenever we have two arrows $p \to q \to p$.

Remark 2.2.11 (Isomorphisms in other categories). **Sets** and preorders are the canonical examples of categories. There are lots of other categories. Some of them are refered to *categories of structured sets*:

1. $\mathbf{Vect}_{\mathbb{F}}$: the category of vector spaces over a field $\mathbb{F}$, with linear mappings
2. **Groups** the category of groups, with group homomorphisms
3. **Rings** the category of rings, with ring homomorphisms
4. **Fields** the category of fields, with ring homomorphisms (this one has interesting properties)

In most *structured sets categories*, a bijective morphism is an isomorphism, just like in **Sets**. However, there exist bijective morphisms that are not isomorphisms (in **Top**, the category of topological spaces), and in more complicated categories, there exist isomorphisms that are not bijective (see the homotopy category of CW complexes). This is because bijectivity is a set-theoretic notion that does *not* make sense in terms of categories.

We have just introduced the notion of an isomorphism, and we saw that in **Sets**, they were exactly the bijections (Example 2.2.9). Thus, isomorphisms generalise the concept of bijection to other categories. Now, one could ask: how to generalise the concept of injections and surjections?

Definition 2.2.12 (Epimorphisms and monomorphisms)

Let $\mathscr{C}$ be a category and let $c : C \to C'$ be an arrow in $\mathscr{C}$.
The arrow c is a *monomorphism*, or is *monic*, if, for all $f, g : A \to C$, $c \circ f = c \circ g \Rightarrow f = g$:

$$A \underset{g}{\overset{f}{\rightrightarrows}} C \xrightarrow{\ c\ } C'$$

The arrow c is an *epimorphism*, or is *epic*, if, for all $f, g : C' \to B$, $f \circ c = g \circ c \Rightarrow$

$f = g$:

$$C \xrightarrow{c} C' \underset{g}{\overset{f}{\rightrightarrows}} B$$

Example 2.2.13 (Epis and monos in **Sets**). In **Sets**, suppose $f : X \to X'$ is monic. Let $x, y \in X$ such that $f(x) = f(y)$. Let f_x and f_y be the functions:

$$f_x : \begin{cases} 1 & \longrightarrow & X \\ i & \longmapsto & x \end{cases} \qquad \text{and} \qquad f_y : \begin{cases} 1 & \longrightarrow & X \\ i & \longmapsto & y \end{cases}$$

As f is monic, we have $f \circ f_x = f \circ f_y \Rightarrow f_x = f_y \Rightarrow x = y$. In other words, f is injective.

Conversely, if f is injective, then for all $g, h : Y \to X$, if $f \circ g = f \circ h$, then for all $y \in Y$, $f \circ g(y) = f \circ h(y)$ which by injectivity means $g(y) = h(y)$ and then $g = h$. In other words, f is monic.

Now, if $f : X \to X'$ is epic, let $\chi_{f(X)} : X' \to 2$ be the characteristic function of $f(X)$ (the image of f), and let $\mathrm{cst}_1 : x \mapsto 1$ be the constant function. We have $\chi_{f(X)} \circ f = \mathrm{cst}_1 \circ f$, which by epicity gives $\chi_{f(X)} = \mathrm{cst}_1$, and thus $X' = f(X)$, from which we deduce the surjectivity.

If $f : X \to X'$ is surjective, let $g, h : X' \to Y$ such that $g \circ f = h \circ f$. For all $x' \in X'$, there exists an x such that $x' = f(x)$ and $g(x') = g \circ f(x) = h \circ f(x) = h(x')$, which gives $g = h$, and f is epic.

In summary, in **Sets**, monomorphisms are exactly injective functions, and epimorphisms are exactly surjective functions.

Example 2.2.14 (Epis and monos in a proset). In a proset category $(P, \leqslant)$, every arrow is monic and epic. This is due to the unicity of the arrow between two objects. Note that, here, the arrows that are both monic and epic, are not necessarily isomorphisms.

Remark 2.2.15 (Epis and monos in other categories). In most "structured sets" categories, for example, in **Monoids**, in **Groups**, in $\mathbf{Vect}_{\mathbb{F}}$, the monomorphisms are exactly the injective morphisms. However, the epimorphisms are not exactly the surjective morphisms. For more information, see [22, Section 2.1, pp30-31].

Proposition 2.2.16

Let $c : C \to C'$ and $c' : C' \to C$ such that $c' \circ c = \mathrm{id}_C$. Then c is monic while c' is epic.

Proof. Let $a, a' : C' \to A$ such that $c \circ a = c \circ a'$, then $c' \circ c \circ a = c' \circ c \circ a' \Rightarrow a = a'$, so

c is monic.

Let $b, b' : C \to B$ such that $b \circ c' = b' \circ c'$, then $b \circ c' \circ c = b' \circ c' \circ c \Rightarrow b = b'$, so c' is epic. □

From Remark 2.2.13, we deduce that a function in **Sets** is an isomorphism if and only if it is both monic and epic. However, the "if and only if" does not hold for most categories (see Example 2.2.14 or [22, Section 2.1.1, pp32-33] for an example). What does hold is the following:

Corollary 2.2.17

If $c : C \to C'$ is an isomorphism, then c is both a monomorphism and an epimorphism.

A final important notion regarding arrows is the following:

Definition 2.2.18 (Factor through)

Let $f : B \to C'$ be an arrow in a category $\mathscr{C}$. Assume that there is C, $b : B \to C$ and $c : C \to C'$ such that $f = c \circ b$. In this context, we say that f *factors through* C, b *and* c.

This definition is very practical. For example, if we have $f = c \circ b$ but we do not care what $b : B \to C$ and $c : C \to C'$ actually are, we simply say that f factors through C.

We now go back to studying a bit more about categories. We consider here the size of categories, which might be a concern of a reader with set-theoretic background.

Nothing in Definition 2.2.1 implies that $\mathrm{Ob}_{\mathscr{C}}$ or $\mathrm{Mor}_{\mathscr{C}}$ should be sets (nor should be $\mathrm{Hom}_{\mathscr{C}}(C, C')$). In fact, $\mathrm{Ob}_{\mathbf{Sets}}$ is not a set. In that sense, categories can be as *big* as possible. However, in the scope of this course, we will only use *somewhat small* categories, in the following sense.

Definition 2.2.19 (Small, locally small and large categories)

A category $\mathscr{C}$ is *small* if both $\mathrm{Ob}_{\mathscr{C}}$ and $\mathrm{Mor}_{\mathscr{C}}$ are sets; otherwise, it is *large*. A category $\mathscr{C}$ is *locally small* if, for all objects $C, C' \in \mathscr{C}$, the Hom-set $\mathrm{Hom}_{\mathscr{C}}(C, C')$ is a set.

Example 2.2.20. **Sets** is large but locally small.

Example 2.2.21. If $(P, \leqslant)$ is a proset, then it is a small (thus locally small) category.

Example 2.2.22. The following example is inspired from set-theory. If V_α is the α-th set of the von Neumann hierarchy [27, Definition 2.1, p. 95], and if λ is a limit ordinal, then we define the category V_λ by:

Objects: An object in V_λ is any set $X \in V_\lambda$

Morphisms: A morphism in V_λ is any function $f : X \to X'$ for $X, X' \in V_\lambda$

Identities: An identity morphism is an identity function $\mathrm{id}_X : X \to X'$

Composition: The composition law for morphisms is the usual composition of functions

We can see V_λ as a truncated **Sets** category. The category V_λ is a small category.

Example 2.2.23. For an example of a large, non-locally small category, see [28].

Remark 2.2.24. Small categories are locally small (because "*sets contain sets*").

In this course, we will consider locally-small categories, for a reason explained later. For now, we continue with a few more basic notions.

We also define mappings somewhat similar to functions, or homomorphisms, between categories.

Definition 2.2.25 (Functor)

Let $\mathscr{C}$ and $\mathscr{X}$ be categories.
A *functor* $F : \mathscr{C} \to \mathscr{X}$ is a mapping from $\mathscr{C}$ to $\mathscr{X}$ such that:

- $\forall C \in \mathrm{Ob}_{\mathscr{C}},\ F(C) \in \mathrm{Ob}_{\mathscr{X}}$
- $\forall c : C \to C' \in \mathrm{Mor}_{\mathscr{C}},\ F(c) : F(C) \to F(C') \in \mathrm{Mor}_{\mathscr{X}}$
- $\forall C \in \mathrm{Ob}_{\mathscr{C}},\ F(\mathrm{id}_C) = \mathrm{id}_{F(C)}$
- $\forall c : C \to C', c' : C' \to C'' \in \mathrm{Mor}_{\mathscr{C}},\ F(c' \circ c) = F(c') \circ F(c)$

In other words, a functor $F : \mathscr{C} \to \mathscr{X}$ sends the objects (resp. morphisms) in $\mathscr{C}$ to objects (resp. morphisms) in $\mathscr{X}$, preserving domains and codomains of morphisms, as well as identities and composition.

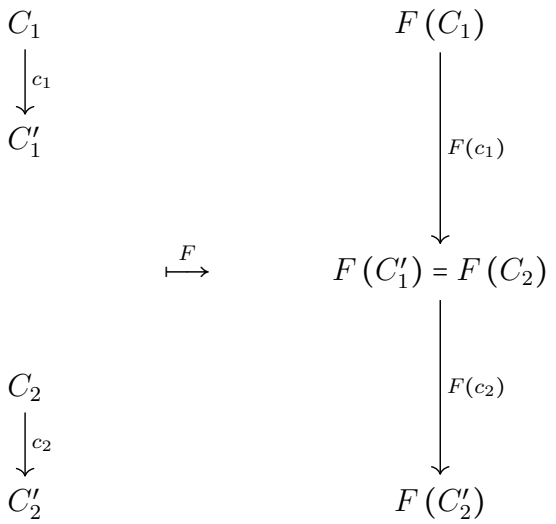
Example 2.2.26 (Functors between prosets). If $(P_1, \leqslant_1)$ and $(P_2, \leqslant_2)$ are prosets, then a functor between those two categories is a monotone function such that $p \leqslant_1 q \Rightarrow F(p) \leqslant_2 F(q)$.

Example 2.2.27 (Forgetful functors). Every category of structured sets $\mathscr{C}$, for example $\mathscr{C} = \mathbf{Vect}_{\mathbb{F}}$ or $\mathscr{C} = \mathbf{Fields}$, comes with a functor $F : \mathscr{C} \to \mathbf{Sets}$ that "takes away the structure". For example, if $\mathscr{C} = \mathbf{Vect}_{\mathbb{F}}$, then it sends a vector space to its underlying set. Such a functor generally has interesting properties as well (i.e. it may

have a left adjoint, however, despite being a captivating topic, this is far beyond the scope of this thesis).

One can interpret a functor $\mathscr{C} \to \mathscr{X}$ as a way to have the *picture- perhaps distorted* of the category $\mathscr{C}$ into the category $\mathscr{X}$ ([22, section 1.4, page 9]). It is the idea behind diagrams as we will see in Section 2.4.

Remark 2.2.28. It is important to note here that the image of a category by a functor is not necessarily a category. Consider the following functor:



In the domain category, there is no composite $c_2 \circ c_1$ because the domain of c_2 is not the codomain of c_1. However, in the image of the functor, we have an arrow $F(c_2)$ whose domain coincides with the codomain of $F(c_1)$. If it were a category, it would need a composite arrow $F(?) = F(c_2) \circ F(c_1)$, which doesn't exist in the first category.

Of course, we can complete the image of a functor and make it a category.

In the special case of functors $\mathscr{C} \to \mathbf{Sets}$, we can define the pretty straightforward notion of subfunctor:

Definition 2.2.29 (Subfunctor)

Let $F : \mathscr{C} \to \mathbf{Sets}$ be a functor.
A subfunctor of F is a functor $G : \mathscr{C} \to \mathbf{Sets}$ such that:

1. for all C, $G(C) \subset F(C)$
2. for all $c : C \to C'$, $G(c) = F(c)|_{G(C)}$ (restriction of $F(c)$ to $G(C)$)

Sometimes, we come across some functors that behave strangely. Namely, some-

times a functor $F : \mathscr{C} \to \mathscr{X}$ may send $c : C \to C'$ to $F(c) : F(C') \to F(C)$ (note the inversion). We will give an example of such a functor. What is happening, is that F is actually not a functor $\mathscr{C} \to \mathscr{X}$ but somehow defined on a similar, but "reversed" category of $\mathscr{C}$.

Definition 2.2.30 (Opposite category)

Let $\mathscr{C}$ be any category. We call *opposite, or dual category of* $\mathscr{C}$, denoted by $\mathscr{C}^{\mathrm{op}}$, the following category:

Objects: An object in $\mathscr{C}^{\mathrm{op}}$ is an object in $\mathscr{C}$

Morphisms: An arrow $c : C' \to C$ in $\mathscr{C}^{\mathrm{op}}$ is an arrow $c : C \to C'$ in $\mathscr{C}$

Identities: An identity in $\mathscr{C}^{\mathrm{op}}$ is an identity in $\mathscr{C}$

Composition: The composition law in $\mathscr{C}^{\mathrm{op}}$ is the same as in $\mathscr{C}$

Basically, the opposite category $\mathscr{C}^{\mathrm{op}}$ is the same category as $\mathscr{C}$, with inverted arrows.

If a functor $F : \mathscr{C} \to \mathscr{X}$ sends $c : C \to C'$ to $F(c) : F(C') \to F(C)$, then F is not actually defined on $\mathscr{C}$ but rather on $\mathscr{C}^{\mathrm{op}}$: $F : \mathscr{C}^{\mathrm{op}} \to \mathscr{X}$. However, it is often simpler to consider only functors on $\mathscr{C}$, hence the following notions:

Definition 2.2.31 (Covariant and contravariant functor)

A functor $F : \mathscr{C} \to \mathscr{X}$ is called *covariant* if it sends $c : C \to C'$ to $F(c) : F(C) \to F(C')$ (for all $c \in \mathscr{C}$).
A functor $G : \mathscr{C} \to \mathscr{X}$ is called *contravariant* if it sends $c : C \to C'$ to $G(c) : G(C') \to G(C)$, or equivalently, if $G : \mathscr{C}^{\mathrm{op}} \to \mathscr{X}$ is a covariant functor.

Two examples of such functors are the following:

Definition 2.2.32 (Covariant Hom-set functor)

Let $\mathscr{C}$ be a (locally small) category, and let $C \in \mathscr{C}$ be an object.
The mapping

$$\mathrm{Hom}_{\mathscr{C}}(C, -) : \begin{cases} \mathscr{C} & \longrightarrow & \mathbf{Sets} \\ C' & \longmapsto & \mathrm{Hom}_{\mathscr{C}}(C, C') \end{cases}$$

defines the *covariant Hom-set functor*. It sends an object $C'_0 \in \mathscr{C}$ to the set $\mathrm{Hom}_{\mathscr{C}}(C, C'_0)$ of arrows from C to C'_0, and an arrow $c'_0 : C'_0 \to C''_0$ to the arrow $\mathrm{Hom}_{\mathscr{C}}(C, c'_0) : c \mapsto c'_0 \circ c$ in **Sets**.

Definition 2.2.33 (Contravariant Hom-set functor)

Let $\mathscr{C}$ be a (locally small) category, and let $C' \in \mathscr{C}$ be an object.
The mapping

$$\mathrm{Hom}_{\mathscr{C}}(-, C') : \begin{cases} \mathscr{C}^{\mathrm{op}} & \longrightarrow & \mathbf{Sets} \\ C & \longmapsto & \mathrm{Hom}_{\mathscr{C}}(C, C') \end{cases}$$

defines the *contravariant Hom-set functor*. It sends an object $C_0 \in \mathscr{C}^{\mathrm{op}}$ to the set $\mathrm{Hom}_{\mathscr{C}}(C_0, C')$ of arrows from C_0 to C', and an arrow $c_0 : C_0 \to C_0'$ to the arrow $\mathrm{Hom}_{\mathscr{C}}(c_0, C') : c \mapsto c \circ c_0$ in **Sets**.

Remark 2.2.34. Their names are not stolen: $C' \to \mathrm{Hom}_{\mathscr{C}}(C, C')$ is a covariant functor and $C \to \mathrm{Hom}_{\mathscr{C}}(C, C')$ is a contravariant functor.

Note that both Hom-set functors assume $\mathscr{C}$ to be locally small. As stated a few paragraphs before, all the categories we will encounter in this course are locally small, unless stated otherwise, because we will often need this functor to be defined.

Also note that along this course, we will encounter lots of examples of contravariant functors. This notion may look confusing. With a bit of practice, it is no more a problem.

We continue and end this section with a final basic notion of category theory, namely, natural transformations, which are a kind of mappings between functors.

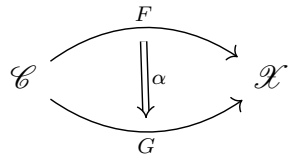
Definition 2.2.35 (Natural transformation)

Let $\mathscr{C}$ and $\mathscr{X}$ be two categories, and let $F, G : \mathscr{C} \to \mathscr{X}$ be functors. A *natural transformation* $\alpha : F \to G$ consists of a collection of morphisms in $\mathscr{X}$ $(\alpha_C : F(C) \to G(C))_{C \in \mathrm{Ob}_{\mathscr{C}}}$ such that, for all $C, C' \in \mathscr{C}$, and for all $c : C \to C'$, the following square commutes:

$$\begin{array}{ccc} C & & F(C) \xrightarrow{\alpha_C} G(C) \\ {\scriptstyle c}\downarrow & \rightsquigarrow & {\scriptstyle F(c)}\downarrow \quad \checkmark \quad \downarrow{\scriptstyle G(c)} \\ C' & & F(C') \xrightarrow[\alpha_{C'}]{} G(C') \end{array} \tag{2.1}$$

For each object $C \in \mathscr{C}$, the morphism α_C is called the C-component of α.

The natural transformation $\alpha : F \to G$ can be written in the following diagram:



We denote by $\mathrm{Nat}\,(F, G)$ the collection of all natural transformations between F and G.

Depending on the context, and for the sake of readability, the C-component of a natural transformation α can be written α_C as above (C as an index) or $\alpha(C)$ (C as a parameter).

Natural transformations can be seen as a way to extract the parameters C, C' and c from $F(C)$, $F(C')$ and $F(c)$ and input them into G, while preserving arrows. It's a variable substitution.

Remark 2.2.36. Consider two functors $F, G : \mathscr{C} \to \mathscr{X}$, and their respective (categorified) images $\mathrm{Im}\,(F)$ and $\mathrm{Im}\,(G)$. A natural transformation $\alpha : F \to G$ may be seen as a functor $\hat{\alpha} : \mathrm{Im}\,(F) \to \mathrm{Im}\,(G)$ such that:

1. for all object $C \in \mathscr{C}$, $\hat{\alpha}(F(C)) = G(C)$ ($\hat{\alpha}$ preserves the objects)
2. for all arrow $c : C \to C' \in \mathscr{C}$, $\hat{\alpha}(F(c)) = G(c)$ with $F(c) : F(C) \to F(C')$ and $G(c) : G(C) \to G(C')$ ($\hat{\alpha}$ preserves the arrows)
3. $\hat{\alpha}$ makes the natural transformation diagram (Diagram 2.1) commute

Note that this view of natural transformations is not standard, but it might help the reader to grasp this notion.

Before introducing the notion of natural isomorphism, we need to make something clear on the nature of natural transformations.

Definition 2.2.37 (Composition of natural transformations)

Let $\mathscr{C}$, $\mathscr{X}$ be categories, and let F, G and H be functors $\mathscr{C} \to \mathscr{X}$.
If $\alpha : F \to G$ is the natural transformation $\alpha = \left(F(C) \xrightarrow{\alpha_C} G(C)\right)_{C \in \mathscr{C}}$ and $\eta : G \to H$ is the natural transformation $\eta = \left(G(C) \xrightarrow{\eta_C} H(C)\right)_{C \in \mathscr{C}}$ then the composition of α by η is $\eta \circ \alpha : F \to H$, defined by $\eta \circ \alpha = \left(F(C) \xrightarrow{\eta_C \circ \alpha_C} H(C)\right)_{C \in \mathscr{C}}$.

Definition 2.2.38 (Functor category)

Let $\mathscr{C}$ and $\mathscr{X}$ be two categories. The *functor category*, denoted by $\mathscr{X}^{\mathscr{C}}$, or by $\mathbf{Func}(\mathscr{C}, \mathscr{X})$, is the following category:

Objects: The objects are the functors $F : \mathscr{C} \to \mathscr{X}$

Morphisms: A morphism between two functors F and G is a natural transformation $\alpha : F \to G = \left(F(C) \xrightarrow{\alpha_C} G(C)\right)_{C \in \mathscr{C}}$

Identities: An identity on a functor F is the identity natural transformation $\mathrm{id}_F = \left(F(C) \xrightarrow{\mathrm{id}_{F(C)}} F(C)\right)_{C \in \mathscr{C}}$

Composition: The composition law in $\mathbf{Func}(\mathscr{C}, \mathscr{X})$ is defined in Definition 2.2.37.

Natural transformations are morphisms between functors. Besides, if F and G are two functors $\mathscr{C} \to \mathscr{X}$, then the notation $\mathrm{Nat}(F, G)$ actually stands for:

$$\mathrm{Nat}(F, G) = \mathrm{Hom}_{\mathbf{Func}(\mathscr{C}, \mathscr{X})}(F, G)$$

however $\mathrm{Nat}(F, G)$ is usually more convenient.

Using Definition 2.2.38 (functor category), and Definition 2.2.8 (isomorphism), we deduce the definition of a natural isomorphism:

Definition 2.2.39 (Natural isomorphism)

Let $F, G : \mathscr{C} \to \mathscr{X}$ be functors. A *natural isomorphism* $\alpha : F \to G$ is a natural transformation that is an isomorphism in the functor category $\mathbf{Func}(\mathscr{C}, \mathscr{X})$.

It is easy to see that:

Lemma 2.2.40

A natural transformation $\alpha : F \to G$ is a natural isomorphism whenever the C-components $\alpha_C : F(C) \to G(C)$ are isomorphisms.

This lemma gives a useful description of what a natural isomorphism is. It makes it easier to look for an inverse. We will use this lemma in the following section.

This lemma does not exactly hold for monic or epic natural transformations. In fact, we have only one implication.

Proposition 2.2.41

Let $\mathscr{C}$ and $\mathscr{X}$ any categories. Let $F, G : \mathscr{C} \to \mathscr{X}$ be two functors and let $\alpha : F \to G$ be a natural transformation between those two functors.
If for all $C \in \mathscr{C}$, $\alpha_C : F(C) \to G(C)$ is monic (resp. epic), then so is $\alpha : F \to G$.

Proof. Suppose that each C-component is monic. The proof is similar when we are considering epic components.

Consider $\beta, \beta' : H \to F$ such that $\alpha \circ \beta = \alpha \circ \beta'$.

$$H \underset{\beta'}{\overset{\beta}{\rightrightarrows}} F \xrightarrow{\alpha} G \qquad \Leftrightarrow \qquad H(C) \underset{\beta'_C}{\overset{\beta_C}{\rightrightarrows}} F(C) \xrightarrow{\alpha_C} G(C)$$

In terms of components, this means that for all $C \in \mathscr{C}$, we have $\alpha_C \circ \beta_C = \alpha_C \circ \beta'_C$. As every component is monic, this gives $\beta_C = \beta'_C$, and then $\beta = \beta'$. Thus, α is monic. □

Surprisingly, the converse does not hold in general. In fact, it needs some more properties about the codomain category, but this is far beyond the scope of this crash course.

We have now introduced the basic notions of category theory: categories, hom-sets, isomorphisms, monomorphisms, epimorphisms, opposite categories, (covariant or contravariant) functors, natural transformations. We can now move on to the next section, in which we introduce the very first important result about category theory.

2.3 Monoidal categories

In Chapter 3, we work with a certain type of categories that we introduce here. This is a short section because we do not need much more than the following basic notions.

In terms of monoidal categories, the references are [25] and [29].

Definition 2.3.1 (Monoidal category)

A *monoidal category* is a 6-tuple $(\mathscr{C}, \otimes, I, a, r, l)$, consisting of a category $\mathscr{C}$, a functor $\otimes : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$, an object $I \in \mathscr{C}$, and three natural isomorphisms a, r, and l of the following form

1. $a(A,B,C) : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$
2. $r(A) : A \otimes I \to A$
3. $l(A) : I \otimes A \to A$

such that, for all objects A, B, C and D of $\mathscr{C}$, the following diagrams commute:

$$\begin{array}{ccc} A \otimes (I \otimes B) & \xrightarrow{a(A,I,B)} & (A \otimes I) \otimes B \\ & {\scriptstyle A\otimes l(B)} \searrow \quad \swarrow {\scriptstyle r(A)\otimes B} & \\ & A \otimes B & \end{array} \tag{2.2}$$

$$\begin{array}{ccc} & A \otimes (B \otimes (C \otimes D)) & \\ {\scriptstyle A\otimes a(B,C,D)} \swarrow & & \searrow {\scriptstyle a(A,B,C\otimes D)} \\ A \otimes ((B \otimes C) \otimes D) & & (A \otimes B) \otimes (C \otimes D) \\ {\scriptstyle a(A,B\otimes C,D)} \downarrow & & \downarrow {\scriptstyle a(A\otimes B,C,D)} \\ (A \otimes (B \otimes C)) \otimes D & \xrightarrow[a(A,B,C)\otimes D]{} & ((A \otimes B) \otimes C) \otimes D \end{array} \tag{2.3}$$

We also say that $(\otimes, I, a, r, l)$ forms a *monoidal structure* on $\mathscr{C}$.

Roughly, a monoidal category is a category with an operation $\otimes$ which can be seen as associative (Diagram 2.3), and a distinguished element I that behaves like a unit for the operation (Diagram 2.2).

Example 2.3.2. The category $(\mathbf{Sets}, \times, 1, a, r, l)$ is monoidal for the usual product of sets, where a, r, l are the obvious isomorphisms and 1 is the singleton $1 = \{0\}$.

Remark 2.3.3. Readers not familiar with category theory may wonder what functors were used to define the natural transformations a, r and l.

The natural transformation a is defined in item 1 by its (A,B,C)-component. Define the two functors:

$$F : \left\{ \begin{array}{ccc} \mathscr{C} \times \mathscr{C} \times \mathscr{C} & \longrightarrow & \mathscr{C} \\ (A,B,C) & \longmapsto & A \otimes (B \otimes C) \end{array} \right. \qquad G : \left\{ \begin{array}{ccc} \mathscr{C} \times \mathscr{C} \times \mathscr{C} & \longrightarrow & \mathscr{C} \\ (A,B,C) & \longmapsto & (A \otimes B) \otimes C \end{array} \right.$$

then a is defined as the natural transformation $a : F \to G$. The same holds for r and l.

Define:

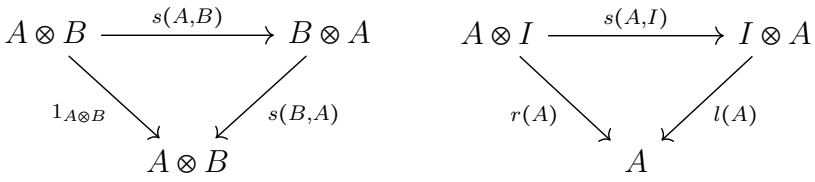
$$R : \begin{cases} \mathscr{C} & \longrightarrow & \mathscr{C} \\ A & \longmapsto & A \otimes I \end{cases} \qquad L : \begin{cases} \mathscr{C} & \longrightarrow & \mathscr{C} \\ A & \longmapsto & I \otimes A \end{cases} \qquad \mathrm{id}_{\mathscr{C}} : \begin{cases} \mathscr{C} & \longrightarrow & \mathscr{C} \\ A & \longmapsto & A \end{cases}$$

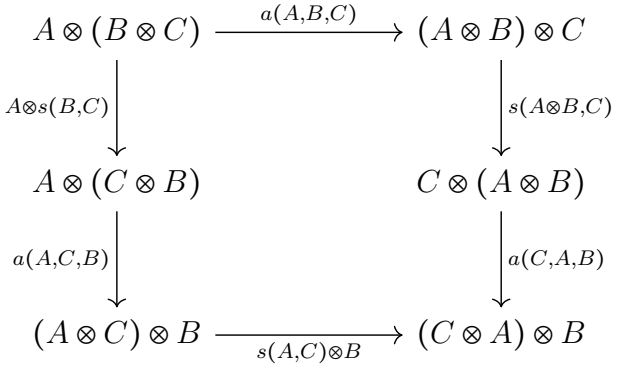
then r is the natural transformation $r : R \to \mathrm{id}_{\mathscr{C}}$ and l is the natural transformation $l : L \to \mathrm{id}_{\mathscr{C}}$.

Definition 2.3.4 (Symmetric monoidal category)

A *symmetric monoidal category* is a 7-tuple $(\mathscr{C}, \otimes, I, a, r, l, s)$ such that:

1. $(\mathscr{C}, \otimes, I, a, r, l)$ is a monoidal category
2. s is a natural isomorphism $s(A, B) : A \otimes B \to B \otimes A$ such that, for all A, B and C of $\mathscr{C}$, the following diagrams commute:





(2.4)

A symmetric monoidal category has an associative and commutative law, with a unit object. Again, **Sets** is a symmetric monoidal category.

Definition 2.3.5 (Lax monoidal functor)

Let $(\mathscr{C}, \boxplus, I, a_{\mathscr{C}}, r_{\mathscr{C}}, l_{\mathscr{C}})$ and $(\mathscr{X}, \otimes, J, a_{\mathscr{X}}, r_{\mathscr{X}}, l_{\mathscr{X}})$ be monoidal categories. A monoidal functor between $\mathscr{C}$ and $\mathscr{X}$ is a 3-tuple (F, σ, σ') where:

— $F : \mathscr{C} \to \mathscr{X}$ is a functor

— σ is a natural transformation

$$\sigma = \left(\sigma_{A,B} : F(A) \otimes F(B) \to F\left(A \boxplus B\right)\right)_{A,B \in \mathscr{C}}$$

between two $\mathscr{C} \times \mathscr{C} \to \mathscr{X}$ functors

— σ' is a morphism $\sigma' : J \to F(I)$

such that, for all $A, B, C \in \mathscr{C}$, the following three diagrams commute:

$$\begin{array}{ccc}
(F(A) \otimes F(B)) \otimes F(C) & \xrightarrow{a_{\mathscr{X}}(F(A),F(B),F(C))} & F(A) \otimes (F(B) \otimes F(C)) \\
\downarrow{\scriptstyle \sigma_{A,B} \otimes 1_{F(C)}} & & \downarrow{\scriptstyle 1_{F(A)} \otimes \sigma_{B,C}} \\
F\left(A \boxplus B\right) \otimes F(C) & & F(A) \otimes F\left(B \boxplus C\right) \\
\downarrow{\scriptstyle \sigma_{A \boxplus B, C}} & & \downarrow{\scriptstyle \sigma_{A, B \boxplus C}} \\
F\left(\left(A \boxplus B\right) \boxplus C\right) & \xrightarrow[F(a_{\mathscr{C}}(A,B,C))]{} & F\left(A \boxplus \left(B \boxplus C\right)\right)
\end{array}$$

$$\begin{array}{ccc}
F(A) \otimes F(I) & \xleftarrow{1_{F(A)} \otimes \sigma'} & F(A) \otimes J \\
\downarrow{\scriptstyle \sigma_{A,I}} & & \downarrow{\scriptstyle r_{\mathscr{X}}(F(A))} \\
F\left(A \boxplus I\right) & \xrightarrow[F(r_{\mathscr{C}}(A))]{} & F(A)
\end{array}
\qquad
\begin{array}{ccc}
F(I) \otimes F(A) & \xleftarrow{\sigma' \otimes 1_{F(A)}} & J \otimes F(A) \\
\downarrow{\scriptstyle \sigma_{I,A}} & & \downarrow{\scriptstyle l_{\mathscr{X}}(F(A))} \\
F\left(I \boxplus A\right) & \xrightarrow[F(l_{\mathscr{C}}(A))]{} & F(A)
\end{array}$$

The pair (σ, σ') is called the *coherence maps of* F. We sometimes refer to σ as the *first coherence map of* F.

Remark 2.3.6. Just like we call $\mathbb{R}$ a field without clarifying its two laws and its two units, we often write $(\mathscr{C}, \otimes, I)$, omitting the natural isomorphisms a, r and l, because they are only a matter of "bookkeeping". We may even write $\mathscr{C}$ for a monoidal category when the context makes it clear what the unit and monoidal product are.

2.4 Your only colimit is yourself

2.4.1 Basics

In category theory, there are a few constructions that are refered to as "universal constructions". These are [30]:

1. Representable functor
2. Adjunctions
3. Limits and colimits
4. End and coends
5. Kan extensions

These structures are universal in the sense that, whenever another object has the same properties than them, then that objects is linked (in a way or another) to the corresponding structure. An intriguing property is that there is a way to see each of them as a special case of the others.

All five structures were given for the sake of completeness. However, in the context of this thesis, we will only introduce limits and colimits. The interested reader may refer to [24], [22], [25] or any other great introductory read on category theory for the remaining universal structures.

Roughly, a functor $P : \mathscr{P} \to \mathscr{C}$ (with $\mathscr{P}$ small) will be seen as a (small) subcategory of $\mathscr{C}$. If $\mathscr{C}$ has a rich enough structure, there may exist objects that "react", or "behave", particularly well to these subcategories. If $\mathscr{C}$ has an even richer structure, then one object among these will "represent" the other objects. Such functors will be called *diagrams*, the objects will be *cones* to that diagram, and the representative will be the *limit* of the diagram.

Let us give the formal definitions for all these notions.

Definition 2.4.1 (Diagram)

Let $\mathscr{C}$ and $\mathscr{P}$ be categories. A *diagram in* $\mathscr{C}$ *of shape* $\mathscr{P}$ is a functor $\mathscr{P} \to \mathscr{C}$. The category $\mathscr{P}$ is called the *index category*. If $\mathscr{P}$ is finite, then the diagram is said *finite*.

In the rest of this thesis, we only consider *small diagrams*, that is, diagrams whose index categories are small (Definition 2.2.19). This is usual in category theory and justifies the use of a different name: a functor comes from any category and goes to any category, while diagrams are functors from a small category.

In Chapters 4, 5 and 6, diagrams will be denoted by the letters P and Q, following the convention from [12], where diagrams are called *patterns*.

Among all possible diagrams, we need to discriminate the following one:

Definition 2.4.2 (Diagonal functor)

Let $\mathscr{C}$ and $\mathscr{P}$ be categories.
The *diagonal functor* Δ is the functor $\Delta : \mathscr{C} \to \mathscr{C}^{\mathscr{P}}$ defined as:

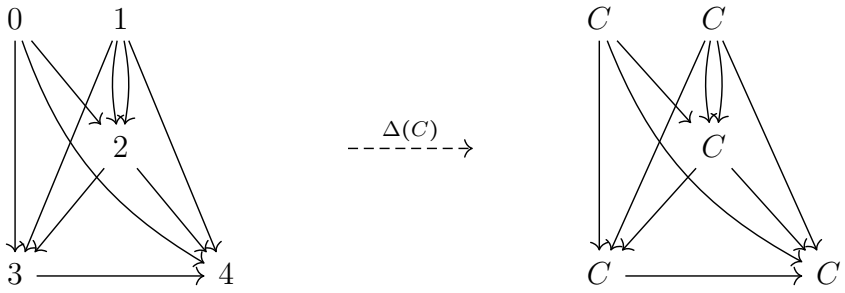
1. For all object $C \in \mathscr{C}$, $\Delta(C)$ is the diagram:

$$\Delta(C) : \begin{cases} \mathscr{P} & \longrightarrow & \mathscr{C} \\ p & \longmapsto & C \\ p \to p' & \longmapsto & \mathrm{id}_C \end{cases}$$

2. For all arrow $c : C \to C' \in \mathscr{C}$, $\Delta(c) : \Delta(C) \to \Delta(C')$ is the natural transformation $\Delta(c) = \left(C \xrightarrow{c} C'\right)_{p \in \mathscr{P}}$ (each component $\Delta(c)_p$ is a copy of c).

In summary, the functor $\Delta(C)$ "collapses" the category $\mathscr{P}$ into one element C.

One can also see $\Delta(C)$ as a sequence of copies of C, indexed by the objects of $\mathscr{P}$. Here, the arrows of $\mathscr{P}$ do not matter, as they always become id_C. A complementary view of the action of $\Delta(C)$ is the following diagram:



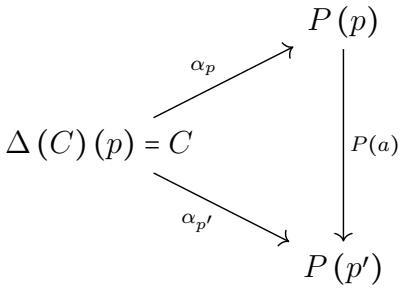
where all the arrows in the right diagram are identity arrows.

Formally, there is exactly one diagonal functor per pair $(\mathscr{P}, \mathscr{C})$, so these categories should appear in the notation. It is not rare to see notation like $\Delta_{\mathscr{C}}^{\mathscr{P}}$. In most cases, $\mathscr{C}$ is fixed, and $\mathscr{P}$ does not really matter, as the image of Δ has only one object and one arrow. Unlike what we said under the definition of functor, Δ does not exactly give a picture of $\mathscr{P}$ inside $\mathscr{C}$. This allows for a flexible notation and the following formal definition:

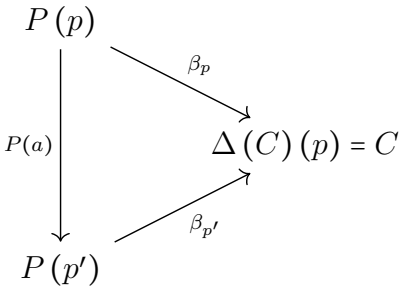
Definition 2.4.3 (Cone, cocone)

Let $P : \mathscr{P} \to \mathscr{C}$ be a diagram to $\mathscr{C}$, and let $C \in \mathscr{C}$.
A *cone from C to P* is an arrow $\Delta(C) \to P$ in the functor category $\mathbf{Func}(\mathscr{P}, \mathscr{C})$.
A *cocone from P to C* is an arrow $P \to \Delta(C)$ in the functor category $\mathbf{Func}(\mathscr{P}, \mathscr{C})$.

In other words, a cone from C to P is a natural transformation $\alpha : \Delta(C) \to P$ such that, for all $a : p \to p' \in \mathscr{P}$, the following triangle commutes:



Similarly, a cocone from P to C is a natural transformation $\beta : P \to \Delta(C)$ such that, for all $a : p \to p' \in \mathscr{P}$, the following triangle commutes:



We cannot help ourselves making categories out of nothing:

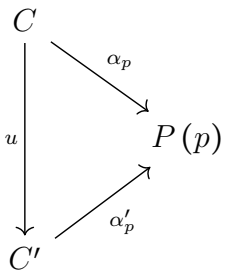
Definition 2.4.4 (Category of cones, category of cocones)

Let $P : \mathscr{P} \to \mathscr{C}$.
The *category of cones to P*, denoted $\mathbf{Cones}(P)$, is the following category:

Objects: Objects are cones $\alpha : \Delta(C) \to P$

Morphisms: An arrow $u : \alpha \to \alpha'$, where $\alpha : \Delta(C) \to P$ and $\alpha' : \Delta(C') \to P$ are cones, is an arrow $u : C \to C' \in \mathscr{C}$ such that, for all $p \in \mathscr{P}$, we have

$\alpha_p = \alpha'_p \circ u$, just as in the following diagram:



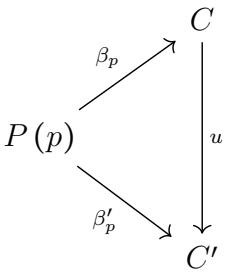
Identities: The identity of $\alpha : \Delta(C) \to P$ is the identity of C in $\mathscr{C}$

Composition: The composition of arrows is that of $\mathscr{C}$

Dually, the *category of cocones from* P, denoted $\mathbf{Cocones}(P)$, is the following category:

Objects: Objects are cocones $\beta : P \to \Delta(C)$

Morphisms: An arrow $u : \beta \to \beta'$, where $\beta : P \to \Delta(C)$ and $\beta' : P \to \Delta(C')$ are cocones, is an arrow $u : C \to C' \in \mathscr{C}$ such that, for all $p \in \mathscr{P}$, we have $\beta'_p = \beta_p \circ u$, just as in the following diagram:



Identities: The identity of $\beta : P \to \Delta(C)$ is the identity of C in $\mathscr{C}$

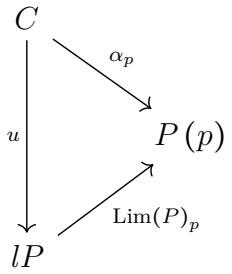
Composition: The composition of arrows is that of $\mathscr{C}$

It is important to distinguish between a cone $\alpha : \Delta(C) \to P$ and its "peak" C. In fact, in the category of cones, there may be several cones $\Delta(C) \to P$, so that C may appear as the peak of different cones. However, most of the time, the confusion causes no harm.

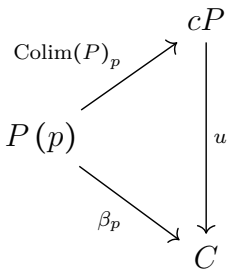
Finally, limits and colimits are specific cones and cocones:

Definition 2.4.5 (Limit, colimit)

Let $P : \mathscr{P} \to \mathscr{C}$ be a diagram to $\mathscr{C}$.
The *limit of* P, if it exists, is the cone $\mathrm{Lim}\,(P) : \Delta\,(lP) \to P$ with the property that, for all cone $\alpha : \Delta\,(C) \to P$, there exists a unique arrow $u : C \to lP$ such that for all $p \in \mathscr{P}$, we have $\alpha_p = \mathrm{Lim}\,(P)_p \circ u$:



Dually, the *colimit of* P, if it exists, is the cocone $\mathrm{Colim}\,(P) : P \to \Delta\,(cP)$ with the property that, for all cocone $\beta : P \to \Delta\,(C)$, there exists a unique arrow $u : cP \to C$ such that for all $p \in \mathscr{P}$, we have $\beta_p = u \circ \mathrm{Colim}\,(P)_p$:



In other words:

Fact 2.4.6

A limit is a cone $\mathrm{Lim}\,(P)$ such that, for each $\alpha \in \mathbf{Cones}\,(P)$, there is a unique arrow $u : \alpha \to \mathrm{Lim}\,(P) \in \mathbf{Cones}\,(P)$.
Dually, a colimit is a cocone $\mathrm{Colim}\,(P)$ such that, for each $\alpha \in \mathbf{Cocones}\,(P)$, there is a unique arrow $u : \mathrm{Colim}\,(P) \to \alpha \in \mathbf{Cocones}\,(P)$.

Keep this in mind for Section 2.4.2.

The unique arrow is often called the universal arrow [1], and the existence of such arrow is often referred to as the Universal Mapping Property, or UMP for short.

1. Note: this is an extremely ambiguous name for such an arrow, because the expression "universal arrow" does have a specific meaning (see [26, chapter III, section 1, page 62]). However, I use this footnote in order to insist on the fact that what we call *universal arrow* in this thesis is not

Just like cones and cocones, a limit/colimit is said finite, or small, if the corresponding diagram is finite, or small, respectively.

Note that limits and colimits, just like cones and cocones, do not always exist in every category.

Definition 2.4.7 (Cofiltered category, filtered category)

A category is called *cofiltered* when every finite diagram has at least one cone. Dually, a category is called *filtered* when every finite diagram has at least one cocone.

Note the inversion: a category is *filtered* when it has cocones, and cofiltered when it has *cones*.

Proposition 2.4.8

Let $\mathscr{P}$ be a category. The following two statements are equivalent:

1. $\mathscr{P}$ is non-empty, for all $p, p' \in \mathscr{P}$ there exist $a : p \to p_0$ and $a' : p' \to p_0$ targeting the same p_0, and for all $a_1, a_2 : p \to p'$, there exists an arrow $a_0 : p' \to p'_0$ such that $a_0 \circ a_1 = a_0 \circ a_2$
2. Every finite diagram $D : \mathscr{X} \to \mathscr{P}$ admits a cocone in $\mathscr{P}$

Proof. (2) ⇒ (1) is obvious, as (1) describes special cases of diagrams (the first one is the two-object diagram serving as base for coproducts, and the second one is the two-arrow diagram serving as base for coequalisers).

The proof of (1) ⇒ (2) is a simpler version of that of Lemma 2.4.40 (if a category has products and equalisers, then it has limits), because we are considering cocones, instead of limits, but the idea remains the same.

□

Also note that the notion refers only to finite diagrams. However, we introduce it just for Chapter 4, in which we do not need any "small" filtered-ness.

A similar notion exists for limits and colimits:

Definition 2.4.9 (Complete category, cocomplete category)

A category is called *complete* (respectively, *finitely complete*) when every small diagram (resp. finite diagram) has a limit. We also say that the category has

the one described in [26]!

small (resp. finite) limits.
Dually, a category is called *cocomplete* (respectively, *finitely cocomplete*) when every small diagram (resp. finite diagram) has a colimit. We also say that the category has *small (resp. finite) colimits.*

It is easy to see that:

Fact 2.4.10

Let $\mathscr{C}$ be a category.

1. $\mathscr{C}$ is complete $\Rightarrow$ $\mathscr{C}$ is cofiltered
2. $\mathscr{C}$ is cocomplete $\Rightarrow$ $\mathscr{C}$ is filtered

Example 2.4.11. The usual categories of mathematical structures are complete and cocomplete: **Sets** the category of sets and functions, **Groups** the category of groups and group morphisms, $\mathbf{Vect}_{\mathbb{F}}$ the category of vector spaces and linear maps, **Rings** the category of rings and ring morphisms, are complete. However, **Fields** the category of fields and field morphisms is neither filtered nor cofiltered, let alone complete nor cocomplete. This is because the category of fields is "divided", and not connected; for example there is no morphism between fields with characteristic 7 and fields with characteristic 11.

As we will see in Chapter 4, given a category $\mathscr{C}$, it is possible to "plunge" it into a bigger category that will contain all the limits or all the colimits of $\mathscr{C}$, and these are called the completion, or cocompletion of $\mathscr{C}$. We see a partial cocompletion in Section 4.3 (meaning: a cocompletion only for a certain kind of diagrams). The completion is beyond the scope of this thesis, but is mentioned in Sections 4.3.1 and 4.5. References for further reading are given in these sections.

Another important result is the following:

Proposition 2.4.12

Let $P : \mathscr{P} \to \mathscr{C}$ be a diagram.
If P has a limit, then it is unique up to isomorphism.
Dually, if P has a colimit, then it is unique up to isomorphism.

Proof. We consider the case of limits; the case with colimits derives dually.

Let $\mathrm{Lim}\,(P)$ and λ be two limits of P. Both are cones to P. By Fact 2.4.6, as $\mathrm{Lim}\,(P)$ is a limit, there is only one arrow $u : \lambda \to \mathrm{Lim}\,(P)$ in $\mathbf{Cones}\,(P)$. Similarly, as λ is a limit, there is only one arrow $v : \mathrm{Lim}\,(P) \to \lambda$ in $\mathbf{Cones}\,(P)$.

Also, there is only one arrow $\mathrm{Lim}\,(P) \to \mathrm{Lim}\,(P)$ and one arrow $\lambda \to \lambda$: the identity. We thus deduce $u \circ v = \mathrm{id}_{\mathrm{Lim}(P)}$ and $v \circ u = \mathrm{id}_{\lambda}$, which means that u and v are inverses of each other. □

In category theory, we distinguish four limits and their dual colimits. These are the basic bricks for building any limits and colimits. We dedicate each of them a subsection.

2.4.2 Initial object, terminal object

Consider the simplest category $\mathscr{P} = 0$, that is, the empty category, with no object nor arrow.

The empty diagram $P : 0 \to \mathscr{C}$ is a functor that takes nothing and turns it into nothing. A cone $\alpha : \Delta\,(C) \to P$ is the empty natural transformation $\alpha = (\alpha_p : C \to P(p))_{p \in \mathscr{P}} = \varnothing$. There is exactly one cone (because there is exactly one empty natural transformation) per object $C \in \mathscr{C}$, and every arrow of $\mathscr{C}$ is an arrow between cones. The category of cones to P is thus isomorphic to $\mathscr{C}$.

If the empty diagram has a limit $\mathrm{Lim}\,(P)$ in $\mathscr{C}$, then it has exactly one arrow from any cone to P to that limit. Isomorphically, there is exactly one arrow $C \to \mathrm{Lim}\,(P)$ for each $C \in \mathscr{C}$. This limit has a name:

Definition 2.4.13 (Initial object, terminal object)

An object T is called *terminal* in $\mathscr{C}$ when, for all $C \in \mathscr{C}$, there is exactly one arrow $t_C : C \to T$.
Dually, an object I is called *initial* in $\mathscr{C}$ when, for all $C \in \mathscr{C}$, there is exactly one arrow $i_C : I \to C$.

The following statement will be useful in Section 6.2:

Proposition 2.4.14

Let $P : \mathscr{P} \to \mathscr{C}$.
$\mathrm{Lim}\,(P)$ is the terminal element of $\mathbf{Cones}\,(P)$. Dually, $\mathrm{Colim}\,(P)$ is the initial element of $\mathbf{Cocones}\,(P)$.

Proof. Combine Definition 2.4.13 with Fact 2.4.6. □

Example 2.4.15. In **Sets**, any singleton $\{a\}$ is a terminal object, because there is only one function $A \to \{a\}$ for every set A (the constant function $x \mapsto a$). Besides,

the empty set $\varnothing$ is the unique initial object; for set-theoretic reasons, there is only one function $\varnothing \to A$ (the empty function).

Example 2.4.16. If $(P, \leqslant)$ is a preorder, then the initial object is the minimal object $\min(P)$ (if it exists) and the terminal object is the maximum $\max(P)$ (if it exists).

2.4.3 Products and coproducts

We need the following definition:

Definition 2.4.17 (Discrete category)

A category $\mathscr{C}$ is called *discrete* when, for all $c : C \to C' \in \mathscr{C}$, we have $C = C'$ and $c = \mathrm{id}_C$.

That is, the only morphisms in a discrete category are the identities.

Let $\mathscr{P}$ be a discrete category. A diagram $P : \mathscr{P} \to \mathscr{C}$ does not have any arrow of the form $P(a) : P(p) \to P(p')$ other than the identity. Thus, a cone from C to P is any set of arrows of the form $C \to P(p)$. The limit $\mathrm{Lim}(P)$ of this diagram is a cone such that each arrow $C \to P(p)$ factors through the peak of $\mathrm{Lim}(P)$.

It may be tricky to see what limit this actually is, so we spoil the suspense and reveal its name:

Definition 2.4.18 (Product, coproduct)

The limit of a diagram whose domain is discrete is called a *product*. Dually, the colimit of a diagram whose domain is discrete is called a *coproduct*.

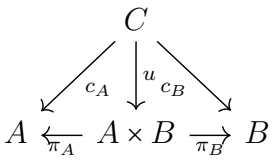
We give below the equivalent, but more frequent definition of a product and coproduct.

Fact 2.4.19

Let $(C_p)_{p \in \mathscr{P}}$ be an indexed set of objects of $\mathscr{C}$.
The product of $(C_p)_{p \in \mathscr{P}}$ consists of an object of $\mathscr{C}$ denoted $\prod_{p \in \mathscr{P}} C_p$ and a set of arrows $\pi = \left(\pi_{p_0} : \prod_{p \in \mathscr{P}} C_p \to C_{p_0} \right)_{p_0 \in \mathscr{P}}$ such that, for all object $C \in \mathscr{C}$ and all set of arrows $(c_p : C \to C_p)_{p \in \mathscr{P}}$, there is a unique arrow $u : C \to \prod_{p \in \mathscr{P}} C_p$ such that for all $p \in \mathscr{P}$, $\pi_p \circ u = c_p$. The arrows π_p are often called the *projection*

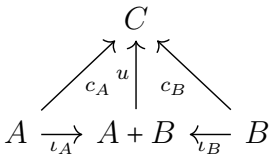
maps.
The coproduct of $(C_p)_{p\in\mathscr{P}}$ consists of an object of $\mathscr{C}$ denoted $\sum_{p\in\mathscr{P}} C_p$ and a set of arrows $\iota = \left(\iota_p : C_p \to \sum_{p\in\mathscr{P}} C_p\right)_{p\in\mathscr{P}}$ such that, for all object $C \in \mathscr{C}$ and all set of arrows $(c_p : C_p \to C)_{p\in\mathscr{P}}$, there is a unique arrow $u : \sum_{p\in\mathscr{P}} C_p \to C$ such that for all $p \in \mathscr{P}$, $u \circ \iota_p = c_p$. The arrows ι_p are often called the *inclusion maps.*

With only two objects A and B, the product $A \times B$ can be described with the following diagram:



For all pair of arrows $(c_A : C \to A, c_B : C \to B)$, there is a unique arrow $u : C \to A \times B$ such that each triangle commutes.

Dually, the coproduct $A + B$ is summarised in the following diagram:



For all pair of arrows $(c_A : A \to C, c_B : B \to C)$, there is a unique arrow $u : A+B \to C$ such that each triangle commutes.

Example 2.4.20. In **Sets**, the product is the usual Cartesian product, and the coproduct is their disjoint union. As we need arrows in order to make a cone (or cocone), we need to specify what those are. The arrows of the product cone are the projections, and the arrows of the coproduct cocones are the inclusion maps. (So these have a meaningful name, fortunately).

Example 2.4.21. In a preorder, the product (if it exists) is the greatest lower bound and the coproduct (if it exists) is the least upper bound.

We complete the section with a couple definitions.

Definition 2.4.22 (Small and finite product)

We say a category $\mathscr{C}$ has *finite products* when, for all finite set of objects $(C_i)_{i\in n}$ of $\mathscr{C}$, the product, denoted $\prod_{i\in n} C_i$, exists.

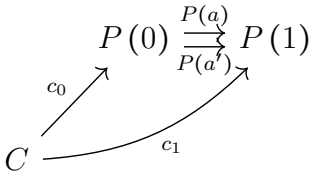
We say a category $\mathscr{C}$ has *small products* when, for all set of objects $(C_i)_{i\in I}$ of $\mathscr{C}$, the product, denoted $\prod_{i\in I} C_i$, exists.

2.4.4 Equalisers and coequalisers

Consider the following category $\mathscr{P}$:

$$0 \underset{a'}{\overset{a}{\rightrightarrows}} 1$$

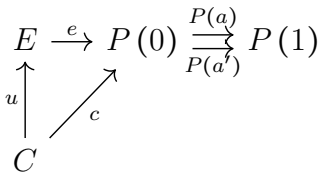
Let $P : \mathscr{P} \to \mathscr{C}$ be a diagram. A cone to P is a pair of arrows (c_0, c_1) such that the following diagram commutes:



In summary, we have:

$$c_1 = P(a) \circ c_0 = P(a') \circ c_0$$

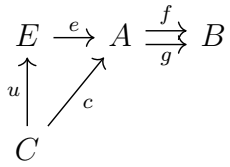
So, a cone to P is characterised, here, by the arrow $c_0 : C \to P(0)$. The limit of P consists then of an object E and an arrow $e : E \to P(0)$ such that $P(a) \circ e = P(a') \circ e$ and every arrow $c : C \to P(0)$ that verifies $P(a) \circ c = P(a') \circ c$ factors uniquely through e:



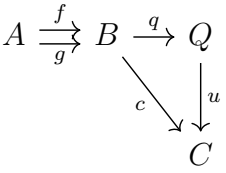
Such a limit is called an equaliser:

Definition 2.4.23 (Equaliser, coequaliser)

Let $f, g : A \to B$ be two arrows of $\mathscr{C}$.
The *equaliser of f and g* is a pair $(E, e : E \to A)$ with the property that $f \circ e = g \circ e$, and for all pair $(C, c : C \to A)$ such that $f \circ c = g \circ c$, there is a unique arrow $u : C \to E$ such that $e \circ u = c$:



Dually, *coequaliser of f and g* is a pair $(Q, q : B \to Q)$ with the property that $q \circ f = q \circ g$, and for all pair $(C, c : B \to C)$ such that $c \circ f = c \circ g$, there is a unique arrow $u : Q \to B$ such that $u \circ q = c$:



Example 2.4.24. In **Sets**, given two functions $f, g : A \to B$, their equaliser is (E, e) where $E = \{x \in A \mid f(x) = g(x)\} \subset A$ and $e : E \to A$ is the canonic inclusion.

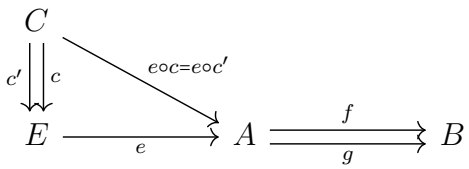
Example 2.4.25. In a preorder category $(P, \leqslant)$, there is at most one arrow $p \to q$. Thus, the equaliser of $f, g : p \to q$, with $f = g$ is their domain together with its identity (p, id_p).

Equalisers and coequalisers intervene when considering isomorphisms.

Proposition 2.4.26

Let (E, e) be the equaliser of $f, g : A \to B$.
Then e is monic.

Proof. Let $c, c' : C \to E$ such that $e \circ c = e \circ c'$.



By definition of an equaliser, we have $f \circ e \circ c = g \circ e \circ c$, so there exists a unique

$u : C \to E$ such that $e \circ u = e \circ c = e \circ c'$. By unicity of u, we have $u = c = c'$, hence e is monic. □

Proposition 2.4.27

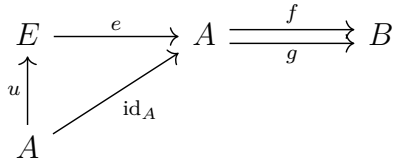
Let (E, e) be the equaliser of $f, g : A \to B$.
If e is an epimorphism then e is an isomorphism.

Proof. Suppose e is epic. As an equaliser, we have the following diagram:

$$E \xrightarrow{e} A \underset{g}{\overset{f}{\rightrightarrows}} B$$

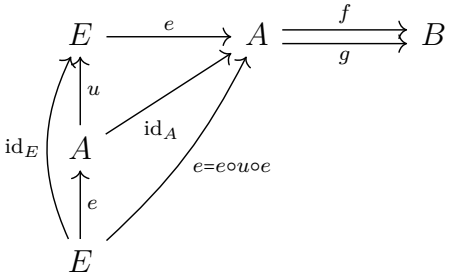
and as an epimorphism, we deduce that $f \circ e = g \circ e \Rightarrow f = g$.

Thus, the identity $\mathrm{id}_A : A \to A$ verifies $f \circ \mathrm{id}_A = g \circ \mathrm{id}_A$. Consequently, there exists a unique $u : A \to E$ such that the following diagram commutes:



from which we deduce $e \circ u = \mathrm{id}_A$.

The same occurs with $e : E \to A$:



We know that $e = e \circ \mathrm{id}_E = e \circ (u \circ e) = (e \circ u) \circ e = \mathrm{id}_A \circ e$. As an equaliser, e is monic, so $u \circ e = \mathrm{id}_E$; e is an isomorphism and $e^{-1} = u$. □

We deduce from this proposition what a *monic epimorphism/epic monomorphism* (or simply *monic/epic*) lacks to be an isomorphism:

Corollary 2.4.28

Let $c : C \to C'$ be any arrow.
The arrow c is an isomorphism $\Leftrightarrow$ c is an epic equaliser.

These results dualise:

Proposition 2.4.29

Let (Q, q) be the coequaliser of $f, g : A \to B$.

1. q is epic
2. if q is monic then q is an isomorphism
3. an arrow c is an isomorphism iff it is a monic coequaliser

Example 2.4.30. In **Sets**, take $f, g : A \to B$. Let R be the relation such that $\forall a \in A, (f(a), g(a)) \in R$, and let $\bar{R}$ be the smallest equivalence relation containing R. Consider $(B/R, b)$, where B/R is the quotient of B by the equivalence relation R, and b is the function that sends an element of B to its equivalence class. Then, $(B/R, b)$ is the coequaliser of f and g.

For more details, see [25, Section 9.4.1, pp 278-279].

Example 2.4.31. Just as in Remark 2.4.25, as there is only one arrow between any two objects, the coequaliser of $f, g : p \to q$ is their codomain: (q, id_q).

2.4.5 Pullbacks and pushouts

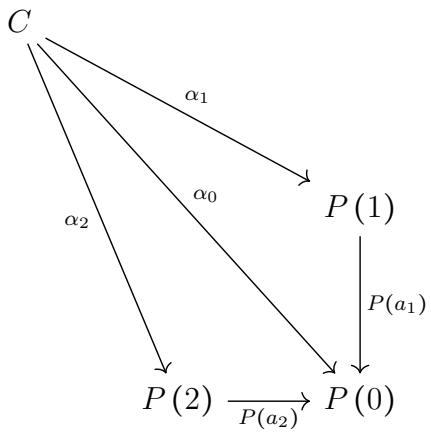
Consider the following category $\mathscr{P}$:

$$\begin{array}{ccc} & & 1 \\ & & \downarrow{\scriptstyle a_1} \\ 2 & \xrightarrow[a_2]{} & 0 \end{array}$$

Let $P : \mathscr{P} \to \mathscr{C}$ be a diagram [2] to $\mathscr{C}$.

A cone to P is a set of three arrows $(\alpha_p : C \to P(p))_{p \in \mathscr{P}}$ such that:

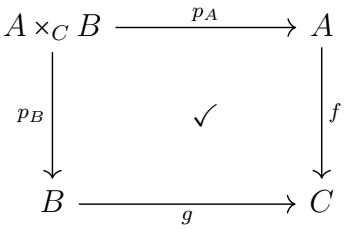
$$P(a_1) \circ \alpha_1 = \alpha_0 = P(a_2) \circ \alpha_2$$



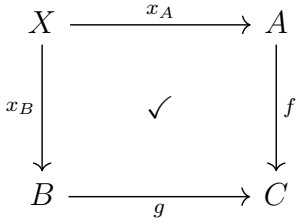
So, a cone to P is characterised by the two arrows α_1 and α_2. The limit of such a diagram is called a pullback:

Definition 2.4.32 (Pullback)

Let $\mathscr{C}$ be a category. Let $f : A \to C$ and $g : B \to C$ be arrows with same codomain.
The *pullback of f and g* is a 3-tuple $(A \times_C B, p_A, p_B)$ such that the following diagram commutes:



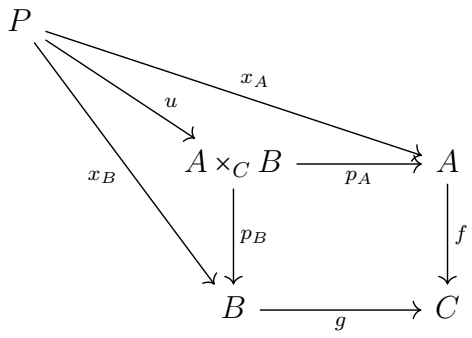
and such that, for all (X, x_A, x_B) such that the following diagram commutes:



there is a unique arrow $u : X \to A \times_C B$ such that $x_A = p_A \circ u$ and $x_B = p_B \circ u$,

2. A diagram from such a category is often called a cospan, but this is out of the scope of this thesis.

that is, such that the triangles and squares commute:



Example 2.4.33 (Pullbacks in **Sets**). In **Sets**, let $f : A \to C$ and $g : B \to C$ be two functions. Their pullback $(A \times_C B, p_A, p_B)$ is:

$$\begin{aligned} A \times_C B &= \{z \in \mathcal{P}(\mathcal{P}(A \cup B)) \mid f \circ p_A(z) = g \circ p_B(z)\} \\ &\cong \{(x, y) \in A \times B \mid f(x) = g(y)\} \end{aligned}$$

with projections $p_A : A \times_C B \to A$ and $p_B : A \times_C B \to B$.

Note that there is the idea of "equalising" two functions. As we will see in a following proposition, there is a link between equalisers and pullbacks, and the explicit construction is based on this idea.

Consider the special case where f and g are inclusion mappings (that is: functions of the form $f : \begin{cases} A & \longrightarrow & C \\ x & \longmapsto & x \end{cases}$ for $A \subset C$ and $g : \begin{cases} B & \longrightarrow & C \\ x & \longmapsto & x \end{cases}$ for $B \subset C$). The pullback of f and g is then:

$$\begin{aligned} A \times_C B &= \{(a, b) \in A \times B \mid a = b\} \\ &= \{(a, a) \in A \times B\} \\ &\cong \{a \in A \mid a \in B\} \\ &= A \cap B \end{aligned}$$

The intersection of sets consists in a pullback of inclusion mappings in **Sets**.

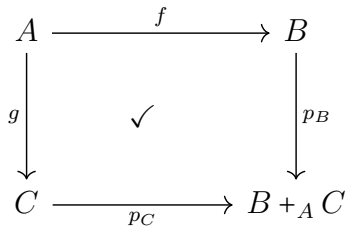
It is interesting to notice that the pullback is a subset of the product.

Example 2.4.34 (Pullbacks in a preorder). In a preorder category $(P, \leqslant)$, as there is at most one arrow between two objects, we don't need to check that any diagram commutes. In fact, the pullback is exactly the same as a product; that is, a pullback

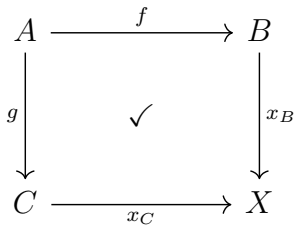
between $p \to q$ and $p' \to q$ is $p \times_q p' = p \times p' = \inf(p, p')$.

Definition 2.4.35 (Pushout)

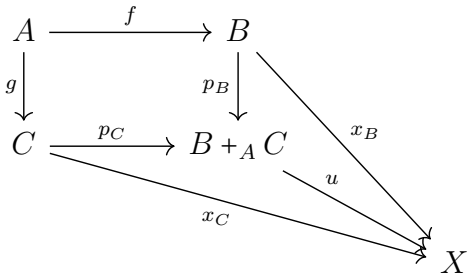
Let $\mathscr{C}$ be a category. Let $f : A \to B$ and $g : A \to C$ be arrows with same domain.
The *pushout of f and g* is a 3-tuple $(B +_A C, p_B, p_C)$ such that the following diagram commutes:



and such that, for all (X, x_A, x_B) such that the following diagram commutes:



there is a unique arrow $u : B +_A C \to X$ such that $x_B = p_B \circ u$ and $x_C = p_C \circ u$, that is, such that the triangles and squares commute:



The arrows $p_B : B \to B +_A C$ and $p_C : C \to B +_A C$ are often called the inclusion mappings, just like in the coproduct.

Example 2.4.36 (Pushout in **Sets**). In **Sets**, consider the functions $f : A \to B$ and $g : A \to C$. Then their pushout $B +_A C$ is identified with a subset of $B + C$; in fact, it is:

$$B +_A C = (B + C) / \equiv$$

where $\equiv$ is the smallest equivalence relation on $B + C$ such that for all $a \in A$, $f(a) \equiv g(a)$.

Another interesting special case is the following. In Example 2.4.33, we defined the intersection $A \cap B$ of two sets A and B. This intersection comes with trivial inclusion mappings $i_A : \begin{cases} A \cap B & \longrightarrow & A \\ x & \longmapsto & x \end{cases}$ and $i_B : \begin{cases} A \cap B & \longrightarrow & B \\ x & \longmapsto & x \end{cases}$, so we can compute its pushout.

$$\begin{array}{ccc} A \cap B & \xrightarrow{i_A} & A \\ {\scriptstyle i_B}\downarrow & \swarrow & \downarrow{\scriptstyle p_A} \\ B & \xrightarrow[p_B]{} & A +_{A \cap B} B \end{array}$$

We have $A +_{A \cap B} B = (A + B) / \equiv$ where $\equiv$ is the smallest equivalence relation such that for all $a \in A$, $i_A(a) \equiv i_B(a)$. In our case, $i_A(a) = i_B(a) = a$, so $\equiv$ is simply the equality $=$. This means that, in the coproduct, which is a disjoint union in **Sets**, the pushout doesn't contain duplicates of the same element a if a is in both A and B. Thus, the pushout $A +_{A \cap B} B$ is simply the union $A \cup B$.

Example 2.4.37 (Pushout in a preorder). Just as pointed in Example 2.4.34 about pullbacks, in a preorder, the pushout is exactly the same as a coproduct.

2.4.6 Links between limits and their special cases

All four of them are crucial in category theory, and in the study of limits and colimits, because they are the basic bricks with which we build limits and colimits.

Theorem 2.4.38

Let $\mathscr{C}$ be any category. The following propositions are equivalent:

1. $\mathscr{C}$ has finite products and equalisers
2. $\mathscr{C}$ has pullbacks and a terminal object
3. $\mathscr{C}$ has finite limits

Of course, the dual theorem is also true:

Theorem 2.4.39

Let $\mathscr{C}$ be any category. The following propositions are equivalent:

1. $\mathscr{C}$ has finite coproducts and coequalisers
2. $\mathscr{C}$ has pushouts and an initial object
3. $\mathscr{C}$ has finite colimits

The proof of this theorem is a long, long run, and most lemmas it requires will not be used afterwards. Here, we give the proof of only one of them, because it will be used to build limits and colimits in **Sets** in the next section and use that expression in Chapter 4.

Lemma 2.4.40

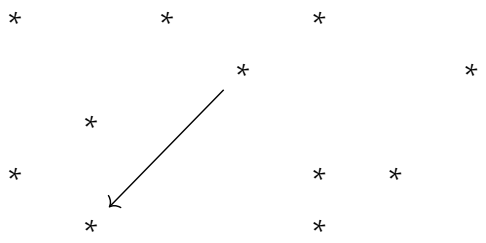
Let $\mathscr{C}$ be any category.
If $\mathscr{C}$ has small products and equalisers, then $\mathscr{C}$ has small limits.

This lemma is about *small* products/limits, but the same lemma with *small* replaced by *finite* also holds and may be used as a lemma to Theorem 2.4.38. However we need this lemma for *small* limits.

Proof. (The proof written here is a resolution of [25, Exercise 3, Section 2.13, Chapter 9])

We will start the proof with one special case of index category. We then give a hint for a second special case. Those two proofs are given just as an exemplification of the general case that we detail right after.

Suppose $\mathscr{P}$ is any small category with only one non-identity arrow $a : p \to p'$. It will then look like this category:



Now let $P : \mathscr{P} \to \mathscr{C}$ be any diagram.

As $\mathscr{C}$ has small products, the product $\prod_{p \in \mathscr{P}} P(p)$ with arrows $\pi_p : \prod_{p \in \mathscr{P}} P(p) \to P(p)$

exists. As $\mathscr{C}$ has equalisers, consider the equaliser (E, e) of $P(a) \circ \pi_p$ and $\pi_{p'}$.

$$E \xrightarrow{\quad e \quad} \prod_{p \in \mathscr{P}} P(p) \underset{\pi_{p'}}{\overset{P(a)\circ\pi_p}{\rightrightarrows}} P(p')$$

Define $\varepsilon = (e_p = \pi_p \circ e)_{p \in \mathscr{P}}$. By definition of (E, e), we have:

$$P(a) \circ \pi_p \circ e = \pi_{p'} \circ e$$

which proves that ε is a natural transformation $\Delta(\mathrm{Lim}(P)) \to P$ (there is only one arrow to check).

We now prove that (E, ε) is the limit of the diagram $P : \mathscr{P} \to \mathscr{C}$. Let $\alpha : \Delta(C) \to P$ be a cone to P; we have $P(a) \circ \alpha_p = \alpha_{p'}$.

Consider the function $\Pi\alpha : C \to \prod_{p \in \mathscr{P}} P(p)$ such that $\forall p \in \mathscr{P},\ \pi_p \circ \Pi\alpha = \alpha_p$. We have:

$$\begin{aligned} P(a) \circ \alpha_p &= \alpha_{p'} \\ P(a) \circ \pi_p \circ \Pi\alpha &= \pi_p \circ \Pi\alpha \end{aligned}$$

As (E, e) is an equaliser of $P(a) \circ \pi_p$ and π_p, there exists a unique $u : C \to E$ such that $e \circ u = \Pi\alpha$, from which we infer, for all $p \in \mathscr{P}$:

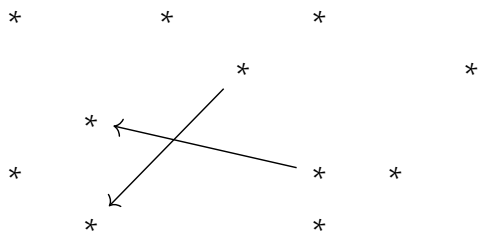
$$\begin{aligned} e \circ u &= \Pi\alpha \\ \pi_p \circ e \circ u &= \pi_p \circ \Pi\alpha \\ e_p \circ u &= \alpha_p \\ \Rightarrow\ \varepsilon \circ \Delta(u) &= \alpha \end{aligned}$$

So (E, ε) is the limit of P.

In short, we have built a limit as the equaliser of (two arrows from) a product.

Now suppose $\mathscr{P}$ is any small category with only two non-identity arrow $a_0 : p_0 \to$

p'_0 and $a_1 : p_1 \to p'_1$. It will then look like this category:



Note that no assumption is made about a_0 and a_1 being distinct; we only suppose that $p'_0 \neq p_1$ and $p_0 \neq p'_1$; otherwise they would compose and give birth to a third arrow.

For a diagram $P : \mathscr{P} \to \mathscr{C}$, we also build the product $\prod_{p \in \mathscr{P}} P(p)$ with its projections $\pi_p : \prod_{p \in \mathscr{P}} P(p) \to P(p)$.

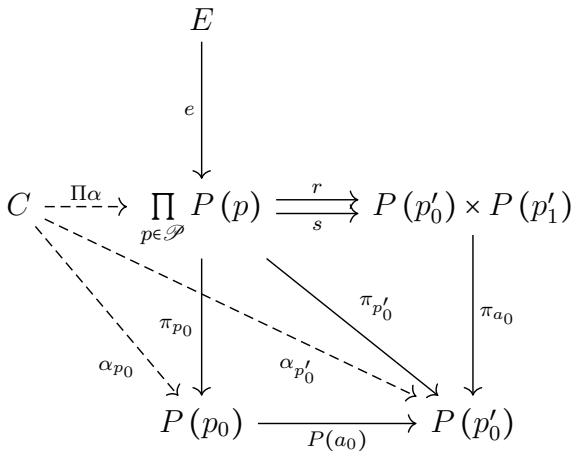
We also define the following arrows:

$$
\begin{aligned}
r_0 &= P(a_0) \circ \pi_{p_0} \\
r_1 &= P(a_1) \circ \pi_{p_1} \\
s_0 &= \pi_{p'_0} \\
s_1 &= \pi_{p'_1} \\
r &= (r_0, r_1) \\
s &= (s_0, s_1)
\end{aligned}
$$

$$
\begin{array}{ccc}
E & & \\
\downarrow{\scriptstyle e} & & \\
\prod_{p \in \mathscr{P}} P(p) & \underset{s}{\overset{r}{\rightrightarrows}} & P(p'_0) \times P(p'_1) \\
\downarrow{\scriptstyle \pi_{p_0}} & \searrow^{\pi_{p'_0}} & \downarrow{\scriptstyle \pi_{a_0}} \\
P(p_0) & \xrightarrow[P(a_0)]{} & P(p'_0)
\end{array}
$$

As $\mathscr{C}$ has equalisers, consider the equaliser (E, e) of $r : \prod_{p \in \mathscr{P}} P(p) \to P(p'_0) \times P(p'_1)$ and $s : \prod_{p \in \mathscr{P}} P(p) \to P(p'_0) \times P(p'_1)$. The proof is very similar to the previous one. The

reader may follow the proof while referring the following diagram:



If $\alpha : \Delta(C) \to P$ is a cone to P, then we define $\Pi\alpha$ to be the concatenation of the components of α: $\forall p \in \mathscr{P}, \alpha_p = \pi_p \circ \Pi\alpha$. We check that $s \circ \Pi\alpha = r \circ \Pi\alpha$ using the fact that α is a natural transformation. As (E, e) is an equaliser, there exists a unique $u : C \to E$ such that $e \circ u = \Pi\alpha$, and we conclude that $\varepsilon \circ \Delta(u) = \Pi\alpha$, with $\varepsilon = (e_p = \pi_p \circ e)_{p \in \mathscr{P}}$ (which is a natural transformation $\Delta(E) \to P$). Finally, (E, ε) is the limit of P.

Again, the limit of P is built as the equaliser of (two arrows from) a product.

As the final case, let $\mathscr{P}$ be any small category. Again, $\mathscr{C}$ has small products, so we define $\prod_{p \in \mathscr{P}} P(p)$ and its projections π_p. As the set of arrows in $\mathscr{P}$ is also small, we can consider all arrows $a : p \to p' \in \mathscr{P}$ and define the product $\prod_{a : p \to p' \in \mathscr{P}} P(p')$, that is, the product of all codomains of all arrows in $\mathscr{P}$. For $a_0 : p_0 \to p'_0$, the projection of index a_0 will be denoted $\pi_{a_0} : \prod_{a : p \to p' \in \mathscr{P}} P(p') \to P(p'_0)$.

We now define:

$$r, s : \prod_{p \in \mathscr{P}} P(p) \to \prod_{a : p \to p' \in \mathscr{P}} P(p')$$

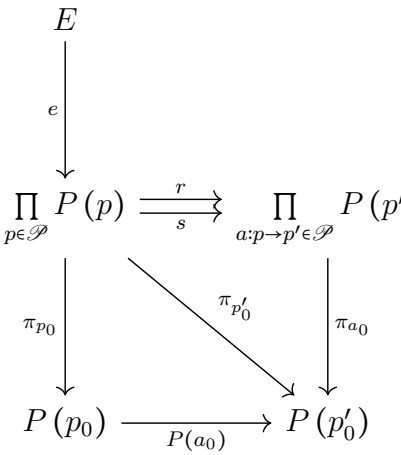
such that, for all $a : p \to p' \in \mathscr{P}$, we have:

$$\pi_a \circ r = P(a) \circ \pi_p = P(a) \circ \pi_{\operatorname{dom} a}$$
$$\pi_a \circ s = \pi_{p'} = \pi_{\operatorname{cod} a}$$

So roughly, r is the concatenation of all the $P(a)$'s, and s is simply a concate-

nation of identities:

$$r \underset{\text{roughly}}{=} \left(P(a) \circ \pi_p\right)_{a:p\to p'\in\mathscr{P}}$$
$$s \underset{\text{roughly}}{=} \left(\mathrm{id}_{p'} \circ \pi_{p'}\right)_{a:p\to p'\in\mathscr{P}}$$



(2.5)

These two arrows r and s both exist due to the UMP of $\prod_{a:p\to p'\in\mathscr{P}} P(p')$: r is the unique arrow of the UMP due to the cone $\left(P(a_0)\circ\pi_{p_0} : \prod_{p\in\mathscr{P}} P(p) \to P(p'_0)\right)$, and s is the unique arrow of the UMP due to the cone $\left(\pi_{p'_0} : \prod_{p\in\mathscr{P}} P(p) \to P(p'_0)\right)$.

Let (E, e) be an equaliser of r and s. the rest of the proof is very similar to the previous two ones, and we deduce that (E, e) is the limit of P. □

Lemma 2.4.41

Let $\mathscr{C}$ be any category.
If $\mathscr{C}$ has small coproducts and coequalisers, then $\mathscr{C}$ has small colimits.

2.4.7 Explicit computation in Sets

The reason why we proved Lemma 2.4.40 while introducing Theorems 2.4.38 and 2.4.39 without proof, is that we wanted to make explicit the construction of limits and colimits in **Sets**. The proof of Lemma 2.4.40 makes it possible to decompose a limit into products and equalisers. Besides, in **Sets**, we know what products and equalisers are! (cf. Examples 2.4.20 and 2.4.24)

In this Section, we use the proof of Lemma 2.4.40 in order to explicitly write what limits and colimits (by duality) are in **Sets**.

Let $P : \mathscr{P} \to \mathbf{Sets}$. The product of sets is the usual Cartesian product and we consider the following products $\prod_{p\in\mathscr{P}} P(p)$ and $\prod_{a:p\to p'\in\mathscr{P}} P(p')$. Here, the product

$\prod_{p \in \mathscr{P}} P(p)$ is the product of all the $P(p)$'s for each object $p \in \mathscr{P}$. The other product $\prod_{a:p \to p' \in \mathscr{P}} P(p')$ has indices in all the arrows $a : p \to p'$ of $\mathscr{P}$ (as in Notation 2.2.4) and makes the product of all $P(p') = P(\mathrm{cod}(a))$.

For $p_0 \in \mathscr{P}$, the projection of index p_0 will be denoted by $\pi_{p_0} : \prod_{p \in \mathscr{P}} P(p) \to P(p_0)$.

For $a_0 : p_0 \to p_0'$, the projection of index a_0 will be denoted by $\pi_{a_0} : \prod_{a:p \to p' \in \mathscr{P}} P(p') \to P(p_0')$.

We can now define:

$$r, s : \prod_{p \in \mathscr{P}} P(p) \to \prod_{a:p \to p' \in \mathscr{P}} P(p')$$

such that, for all $a : p \to p' \in \mathscr{P}$, we have:

$$\pi_a \circ r = P(a) \circ \pi_p = P(a) \circ \pi_{\mathrm{dom}\, a}$$
$$\pi_a \circ s = \pi_{p'} = \pi_{\mathrm{cod}\, a}$$

Just as in the proof of Lemma 2.4.40, the arrows r and s exist both as the UMP of $\prod_{a:p \to p' \in \mathscr{P}} P(p')$:

1. r is the UMP arrow to the cone $\left(P(a_0) \circ \pi_{p_0} : \prod_{p \in \mathscr{P}} P(p) \to P(p_0') \right)$
2. s is the UMP arrow to the cone $\left(\pi_{p_0'} : \prod_{p \in \mathscr{P}} P(p) \to P(p_0') \right)$

The limit of P is the equaliser of r and s (just as in the proof of Lemma 2.4.40). According to Example 2.4.24, it is:

$$\begin{aligned}
\mathrm{Lim}\,(P) &= \left\{ \bar{x} \in \prod_{p \in \mathscr{P}} P(p) \;\middle|\; r\,(\bar{x}) = s\,(\bar{x}) \right\} \\
&= \left\{ \bar{x} \in \prod_{p \in \mathscr{P}} P(p) \;\middle|\; \forall a : p \to p' \in \mathscr{P}, \pi_a \circ r\,(\bar{x}) = \pi_a \circ s\,(\bar{x}) \right\} \\
&= \left\{ \bar{x} \in \prod_{p \in \mathscr{P}} P(p) \;\middle|\; \forall a : p \to p' \in \mathscr{P}, P(a) \circ \pi_p\,(\bar{x}) = \pi_{p'}\,(\bar{x}) \right\} \\
&= \left\{ \bar{x} \in \prod_{p \in \mathscr{P}} P(p) \;\middle|\; \forall a : p \to p' \in \mathscr{P}, P(a)(x_p) = x_{p'} \right\}
\end{aligned}$$

Dually, we construct a colimit as the coequaliser of a coproduct.

Explicitly, in **Sets**, the coproduct is the disjoint union. We consider the sets $\sum_{p \in \mathscr{P}} P(p)$ and $\sum_{a : p \to p' \in \mathscr{P}} P(p')$.

For $p_0 \in \mathscr{P}$, the inclusion of index p_0 will be denoted $\iota_{p_0} : P(p_0) \to \sum_{p \in \mathscr{P}} P(p)$.

For $a_0 : p_0 \to p_0'$, the inclusion of index a_0 will be denoted $\iota_{a_0} : P(p_0') \to \sum_{a : p \to p' \in \mathscr{P}} P(p')$.

We now define:

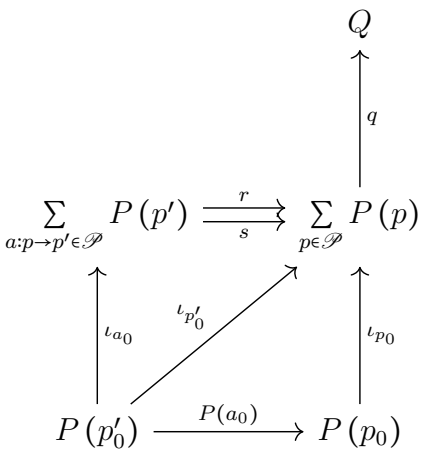
$$r, s : \sum_{a : p \to p' \in \mathscr{P}} P(p') \to \sum_{p \in \mathscr{P}} P(p)$$

such that, for all $a_0 : p_0 \to p_0' \in \mathscr{P}$, we have:

$$\begin{aligned}
r \circ \iota_{a_0} &= \iota_{p_0} \circ P(a_0) = \iota_{\mathrm{dom}\, a_0} \circ P(a_0) \\
s \circ \iota_{a_0} &= \iota_{p_0'} = \iota_{\mathrm{cod}\, a_0}
\end{aligned}$$

We obtain the following diagram, which is basically Diagram 2.5 with inverted

arrows:



The colimit of P is then the coequaliser of r and s. According to Example 2.4.30, we need to define the equivalence relation $\sim$ on $\sum_{p\in\mathscr{P}} P(p)$ generated by: for all $x \in \sum_{a:p\to p'\in\mathscr{P}} P(p')$, $r(x) \sim s(x)$.

Then x must be in one of the $P(p'_0)$ (because x is in the disjoint union of them). So we have:

$$\begin{aligned} x &= \iota_{a_0}(x) \\ s(x) &= s \circ \iota_{a_0}(x) = \iota_{p'_0}(x) = x \\ r(x) &= r \circ \iota_{a_0}(x) = \iota_{p_0} \circ P(a_0)(x) = P(a_0)(x) \end{aligned}$$

Basically, the equivalence relation is generated by: for all $a_0 : p'_0 \to p_0$, for all $x \in P(p'_0)$, $P(a_0)(x) \sim x$, and the colimit of P is then the quotient set $\sum_{p\in\mathscr{P}} P(p) \,/\sim$.

Theorem 2.4.42 (Limits and colimits in Sets)

Let $P : \mathscr{P} \to \mathbf{Sets}$ be a diagram.
Its limit exists and is written:

$$\mathrm{Lim}\,(P) = \left\{ \bar{x} \in \prod_{p\in\mathscr{P}} P(p) \;\middle|\; \forall a : p \to p' \in \mathscr{P}, P(a)(x_p) = x_{p'} \right\}$$

Its colimit exists and is written:

$$\mathrm{Colim}\,(P) = \sum_{p \in \mathscr{P}} P\,(p)\,/\sim$$

where $\sim$ is the equivalence relation generated by: for all $a : p \to p'$, for all $x \in P(p)$, $P(a)(x) \sim x$.

Part II

Dynamical systems

Chapter 3

Dynamical systems

[...] In mathematics you don't understand things.
You just get used to them.

John Von Neumann.

3.1 Introduction

The following chapter is a detailed version of our published work [18].

Automata represent systems that receive inputs, alter their internal states, and produce outputs. The state set of the automaton is to be interpreted as the set of all potential memories, or storable experiences. In automata theory, the state set is typically finite. In this case, one can view this memory capacity as limited. On the contrary, when the memory of the automaton is not assumed to be limited (human brain), or its capacity can always be extended (RAM-machines or Turing machines as models of computers in computation theory), the automaton should have an infinite state set.

In the theory of dynamical systems, we use a generalisation of automata in which the size of the state space is not restricted to finiteness, or even to countability. Dynamical systems with the behaviour of an automaton, that is, taking inputs in discrete time, are called discrete systems. The state space of such a system acts as a sort of memory of the inputs. Each input influences the current state of the automaton, and the current state is the result of the system's own form—how it deals with inputs—together with the system's history.

One can imagine a dynamical system whose state space is that of all possible input-histories; a new input simply appends to the existing history to become a new history. On the other hand, one can imagine the "opposite" kind of system: one that completely forgets the previous inputs. These systems are referred to as "simple-reflex" in [31, p. 49], reactive, or memoryless in this chapter. The transitions of these automata depend only on the input, as no experience is stored. The system decides according to the current perception of the world, rather than current perception

together with past perception. In fact, these memoryless systems could act by making a single distinction in the input—a yes/no Boolean response—and nothing more; we call these Boolean reactive systems.

In this chapter, we will study the links between systems that have memory and those that do not. More precisely, we prove that systems with memory can be simulated by wiring together systems without memory. Our result provides a theoretical framework that supports artificial neural network approaches. Memory is carried by connections, and not only by individuals, within a compositional hierarchy of parts. In particular, feedback generates memory. This result is already known by electronicians and computer scientists (transistors), but this chapter formally proves and generalises this intuitive result to any kind of automaton in discrete time, which takes any kind of inputs and returns any kind of outputs. As a special case, when the automata are boolean (which can compare to transistors), we generate the class of finite automata (which can compare to computers).

This chapter lies between two fields of mathematics: category theory and dynamical systems. In Section 2, we introduce all the background related to category theory and its use in the study of discrete systems. Readers already familiar with the study of boxes and discrete systems from a category-theoretic point of view (as in [19]) can skip Section 3.2. Here again, we only need the basic understanding of $\mathscr{C}$-typed-finite sets, $\mathscr{C}$-boxes, wiring diagrams and discrete systems inside a $\mathscr{C}$-box.

In Section 3.3, we introduce discrete systems and a specific mapping that will serve our purposes (Section 3.3.1). We then introduce two equivalence relations between discrete systems. Both are bigger than the usual bisimulation used in automata theory (in the sense of set inclusion). One corresponds to an external point of view; two systems are equivalent if they transform input streams into output streams in the same way (Section 3.3.2). The other relation corresponds to an internal point of view: two systems are equivalent if they have "the same structure" (in a sense that is defined in Section 3.3.3). We prove that these are just two perspectives on the same relation.

This equivalence relations play a crucial role in the two results we show in Section 3.4. First, we show that any discrete system is equivalent to some wiring-together of memoryless systems (Section 3.4.2). Second, we show that any discrete system with a finite state set is equivalent to a combination of finitely many Boolean reactive systems (Section 3.4.3).

3.2 Background on boxes and wiring diagrams

We will now apply the categorical framework to build discrete systems. Our approach is different from the one in [32]. The dynamical systems presented here are defined as a generalisation of automata whose input and output spaces are

predetermined. We will define a category of lists, a category of boxes, and diverse operations on them.

In this section, $\mathscr{C}$ will be any category with finite products (typically **Sets**). Most of the following notions were already defined in [19]; we only recall them without proving their properties. We also give examples in order to help for comprehension.

3.2.1 The category of typed finite sets

Before defining proper boxes, we need to define the notion of input and output ports. These will eventually be the sides of our boxes.

Definition 3.2.1 (Category of $\mathscr{C}$-typed finite sets [19])

The category $\mathbf{TFS}_{\mathscr{C}}$ *of $\mathscr{C}$-typed finite sets* is defined as follows:

Objects: An object is any pair (P, τ) such that P is a finite set and $\tau : P \to \mathrm{Ob}_{\mathscr{C}}$ is a function

Morphisms: A morphism from (P, τ) to (P', τ') is a function $\gamma : P \to P'$ such that $\tau = \tau' \circ \gamma$

Identities: The identity morphism on (P, τ) is the identity function of the set P

Composition: The composition of morphisms is the usual composition of functions

An object in $\mathbf{TFS}_{\mathscr{C}}$ is called a *$\mathscr{C}$-typed finite set*; a morphism in $\mathbf{TFS}_{\mathscr{C}}$ is called a *$\mathscr{C}$-typed function*.

We can rewrite a $\mathscr{C}$-typed finite set (P, τ) as the finite sequence

$$\langle \tau(p_0), \ldots, \tau(p_{n-1}) \rangle$$

where $P = \{p_0, \ldots, p_{n-1}\}$. A $\mathscr{C}$-typed finite set is simply a list of objects in $\mathscr{C}$, indexed by a finite set P. If $\mathscr{C} = \mathbf{Sets}$, a **Sets**-typed finite set is a list of sets.

A $\mathscr{C}$-typed function $\gamma : (P, \tau) \to (P', \tau')$ can be then seen as a means to obtain the former list $\langle \tau(p_0), \ldots, \tau(p_{n-1}) \rangle$ from the latter list $\langle \tau'(p_0), \ldots, \tau'(p_{n-1}) \rangle$, by reordering, duplicating or even ignoring its elements. As $\tau = \tau' \circ \gamma$, the list $\langle \tau(p_0), \ldots, \tau(p_{n-1}) \rangle$ can be rewritten $\langle \tau'(\gamma(p_0)), \ldots, \tau'(\gamma(p_{n-1})) \rangle$. Beware of the inversion: γ goes from (P, τ) to (P', τ') and we see it as a transformation of the list (P', τ') into the list (P, τ).

Example 3.2.2. Let A, B and C be objects of $\mathscr{C}$ and consider the following three $\mathscr{C}$-typed finite sets:

— $(2, \tau_2)$ such that $\tau_2(0) = A$ and $\tau_2(1) = B$; thus $(2, \tau_2)$ is the list $\langle A, B\rangle$
— $(3, \tau_3)$ such that $\tau_3(0) = B$, $\tau_3(1) = C$ and $\tau_3(2) = A$; thus $(3, \tau_3)$ is the list $\langle B, C, A\rangle$
— $(4, \tau_4)$ such that $\tau_4(0) = \tau_4(1) = \tau_4(2) = A$ and $\tau_4(3) = B$; thus $(4, \tau_4)$ is the list $\langle A, A, A, B\rangle$

(Remember our set-theoretic notation: $2 = \{0, 1\}$, $3 = \{0, 1, 2\}$ and $4 = \{0, 1, 2, 3\}$.)

The list $\langle A, B\rangle$ can be obtained from the list $\langle B, C, A\rangle$ by taking its third and first elements in this order. A $\mathscr{C}$-typed function from $(2, \tau_2)$ to $(3, \tau_3)$ could be $\gamma : 2 \to 3$ such that $\gamma(0) = 2$ and $\gamma(1) = 0$.

Similarly, the morphisms $\gamma' : 2 \to 4$ that convert the list $\langle A, A, A, B\rangle$ to $\langle A, B\rangle$ are such that $\gamma'(0) = 0$ and $\gamma'(1) = 3$, or $\gamma'(0) = 1$ and $\gamma'(1) = 3$, or $\gamma'(0) = 2$ and $\gamma'(1) = 3$.

We let the reader find the morphism $(4, \tau_4) \to (3, \tau_3)$ that transforms the list $\langle B, C, A\rangle$ into the list $\langle A, A, A, B\rangle$, and the (unique) morphism $(4, \tau_4) \to (2, \tau_2)$ that transforms the list $\langle A, B\rangle$ into the list $\langle A, A, A, B\rangle$.

What about the morphisms from $(3, \tau_3)$ to $(2, \tau_2)$? The list $\langle A, B\rangle$ does not contain the object C. There is simply no morphism $(3, \tau_3) \to (2, \tau_2)$. The same argument applies to morphisms from $(3, \tau_3)$ to $(4, \tau_4)$.

Definition 3.2.3 (Sum of typed finite sets [19])

Let $(P_0, \tau_0), (P_1, \tau_1) \in \mathbf{TFS}_{\mathscr{C}}$ be two $\mathscr{C}$-typed finite sets.
We define their *sum* by $(P_0, \tau_0) + (P_1, \tau_1) = (P_0 + P_1, \tau_0 + \tau_1)$ as the usual disjoint union of sets $P_0 + P_1$ and $\tau_0 + \tau_1$ as τ_i on P_i for $i \in 2$.

Definition 3.2.4 (Sum of typed functions [19])

Let $\gamma_i : (P_i, \tau_i) \to (P_i', \tau_i')$ $(i \in 2)$ be two $\mathscr{C}$-typed functions.
We define their *sum* as the $\mathscr{C}$-typed function $\gamma_0 + \gamma_1 : (P_0, \tau_0) + (P_1, \tau_1) \to (P_0', \tau_0') + (P_1', \tau_1')$ such that $\forall x \in P_0 + P_1$, $(\gamma_0 + \gamma_1)(x) = \gamma_i(x)$ if $x \in P_i$ $(i \in 2)$.

We can view the sum $(P, \tau) + (P', \tau')$ as the usual concatenation of the two lists

$$\langle \tau(p_0), \ldots, \tau(p_n)\rangle \text{ and } \langle \tau'(p_0'), \ldots, \tau'(p_{n'}')\rangle$$

that is, the list

$$\langle \tau(p_0), \ldots, \tau(p_n), \tau'(p_0'), \ldots, \tau'(p_{n'}')\rangle$$

and the sum of $\mathscr{C}$-typed functions as a action on each part of the concatenated list.

Proposition 3.2.5

The category $\mathbf{TFS}_{\mathscr{C}}$ has the following properties:

- The sum of $\mathscr{C}$-typed finite sets is a coproduct.
- There is only one $\mathscr{C}$-typed finite set (P, τ) where $P = \varnothing$. We denote it by 0.
- $\mathbf{TFS}_{\mathscr{C}}$ has a symmetric monoidal structure for the sum $+$, with 0 as the unit.

Proof. See [19]. □

3.2.2 Dependent products

In this subsection, we define the dependent product functor. If a $\mathscr{C}$-typed finite set can be viewed as a list of objects of $\mathscr{C}$, then the dependent product of this list is simply the product of its elements.

Definition 3.2.6 (Dependent product [19])

We define the *dependent product* as the functor $\widehat{-} : \mathbf{TFS}_{\mathscr{C}}^{\mathrm{op}} \to \mathscr{C}$ such that:

Action on objects: $\widehat{(P,\tau)} = \prod_{p \in P} \tau(p)$

Action on morphisms: If $\gamma : (P, \tau) \to (P', \tau')$, then $\widehat{\gamma} : \widehat{(P', \tau')} \to \widehat{(P, \tau)}$ is defined as the function $\widehat{\gamma} : \prod_{p' \in P'} \tau'(p') \to \prod_{p \in P} \tau(p)$ such that $\forall \left(a_{p'}\right)_{p' \in P'} \in \widehat{(P', \tau')}$, $\widehat{\gamma}\left(\left(a_{p'}\right)_{p' \in P'}\right) = \left(a_{\gamma(p)}\right)_{p \in P}$.

The interpretation of the dependent product is actually quite straightforward: the dependent product of a $\mathscr{C}$-typed finite set, viewed as a list, is the product of the elements of the list in the same order as they appear in the list.

We remind that $\mathscr{C}$ has finite products; as a consequence, the dependent product always exists.

Example 3.2.7. Consider the same A, B and C objects of $\mathscr{C}$ and $\mathscr{C}$-typed finite sets as in Example 3.2.2:

- $(2, \tau_2) = \langle A, B \rangle$
- $(3, \tau_3) = \langle B, C, A \rangle$
- $(4, \tau_4) = \langle A, A, A, B \rangle$

The dependent products of these $\mathscr{C}$-typed finite sets are:

— $\widehat{(2,\tau_2)} = A \times B$
— $\widehat{(3,\tau_3)} = B \times C \times A$
— $\widehat{(4,\tau_4)} = A \times A \times A \times B$

In order to see what the dependent product does to morphisms, take a morphism $\gamma : 2 \to 4$ that converts the list $\langle A, A, A, B\rangle$ to $\langle A, B\rangle$, for example the morphism defined by $\gamma(0) = 0$ and $\gamma(1) = 3$. Its dependent product $\widehat{\gamma}$ will be the function $\widehat{(4,\tau_4)} \to \widehat{(2,\tau_2)} = A \times A \times A \times B \to A \times B$ such that $\widehat{\gamma}(x_0, x_1, x_2, x_3) = \big(x_{\gamma(0)}, x_{\gamma(1)}\big) = (x_0, x_3)$.

We let the reader find the other dependent products as an exercise.

The dependent product is thus a functor that packages the usual operations of diagonal $A \to A \times A$, projection $A \times B \to A$, and swapping $A \times B \to B \times A$.

Proposition 3.2.8

There is a natural isomorphism $\widehat{(P_0,\tau_0) + (P_1,\tau_1)} \cong \widehat{(P_0,\tau_0)} \times \widehat{(P_1,\tau_1)}$; in other words, the dependent product functor sends coproducts in $\mathbf{TFS}_{\mathscr{C}}$ to products in $\mathscr{C}$.

Proof. See [19]. □

This property is also quite intuitive: if one views the coproduct in $\mathbf{TFS}_{\mathscr{C}}$ as the concatenation of lists, and the dependent product as the product of the elements of the list, then the dependent product of the concatenation of two lists is the product of the dependent products of each lists.

3.2.3 The category of boxes and wiring diagrams

The category $\mathbf{TFS}_{\mathscr{C}}$ is not the main purpose of this chapter; however its properties will be useful for the rest of it.

In the following, by abuse of notation, we will write $X \in \mathbf{TFS}_{\mathscr{C}}$ for (X, τ), and $\widehat{X}$ for $\widehat{(X,\tau)}$.

Definition 3.2.9 ($\mathscr{C}$-box [19])

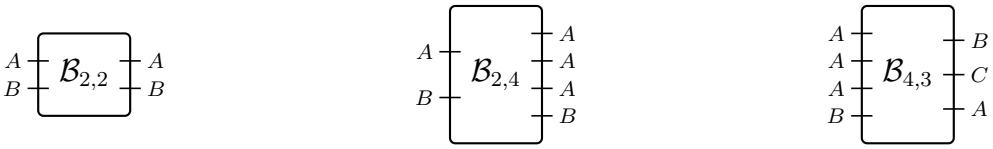
We call *$\mathscr{C}$-box* any pair $X = (X^{\text{in}}, X^{\text{out}}) \in \mathbf{TFS}_{\mathscr{C}} \times \mathbf{TFS}_{\mathscr{C}}$.

A $\mathscr{C}$-box is a pair of $\mathscr{C}$-typed finite sets $(X^{\text{in}}, X^{\text{out}})$, where X^{in} represent the list of inputs ports, and X^{out} represent the list of outputs ports.

Example 3.2.10. From the typed finite sets in Example 3.2.2, we can build the following $\mathscr{C}$-boxes:

— $\mathcal{B}_{2,2} = ((2,\tau_2),(2,\tau_2)) = (\langle A,B\rangle, \langle A,B\rangle)$
— $\mathcal{B}_{2,4} = ((2,\tau_2),(4,\tau_4)) = (\langle A,B\rangle, \langle A,A,A,B\rangle)$
— $\mathcal{B}_{4,3} = ((4,\tau_4),(3,\tau_3)) = (\langle A,A,A,B\rangle, \langle B,C,A\rangle)$

These $\mathscr{C}$-boxes are represented here:



In the rest of the chapter, the ports will no longer be labelled, for the sake of readability.

Definition 3.2.11 (Wiring diagram [19])

Let $X = (X^{\text{in}}, X^{\text{out}})$ and $Y = (Y^{\text{in}}, Y^{\text{out}})$ be $\mathscr{C}$-boxes.
A *wiring diagam* $\varphi : X \to Y$ is a pair of $\mathscr{C}$-typed functions $(\varphi^{\text{in}}, \varphi^{\text{out}})$ such that:

— $\varphi^{\text{in}} : X^{\text{in}} \to Y^{\text{in}} + X^{\text{out}}$
— $\varphi^{\text{out}} : Y^{\text{out}} \to X^{\text{out}}$

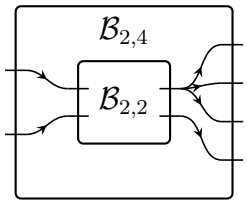
The $\mathscr{C}$-typed function φ^{in} tells what feeds the input ports of the box X: each input port of X is either connected to an input port of Y or to an output port of X (in case of feedback); the $\mathscr{C}$-typed function φ^{out} tells what feeds the output ports of Y: each output port of Y is connected to some output port of X.

Example 3.2.12. Given $\mathcal{B}_{2,2}$ and $\mathcal{B}_{2,4}$ defined in Example 3.2.10, the wiring diagrams $\varphi : \mathcal{B}_{2,2} \to \mathcal{B}_{2,4}$ will have the following form:

— $\varphi^{\text{in}} : (2,\tau_2) \to (2,\tau_2) + (2,\tau_2) = \langle A,B\rangle \to \langle A,B,A,B\rangle$
— $\varphi^{\text{out}} : (4,\tau_4) \to (2,\tau_2) = \langle A,A,A,B\rangle \to \langle A,B\rangle$

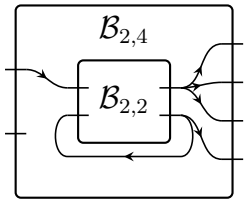
We can build specific wiring diagrams such as:

$\varphi_0^{\text{in}}(0) = 0$
$\varphi_0^{\text{in}}(1) = 1$



$\varphi_0^{\text{out}}(0) = 0$
$\varphi_0^{\text{out}}(1) = 0$
$\varphi_0^{\text{out}}(2) = 0$
$\varphi_0^{\text{out}}(3) = 1$

$\varphi_1^{\text{in}}(0) = 0$
$\varphi_1^{\text{in}}(1) = 3$



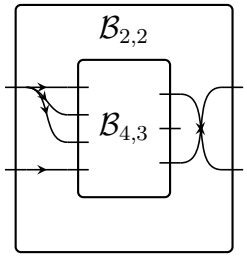
$\varphi_1^{\text{out}}(0) = 0$
$\varphi_1^{\text{out}}(1) = 0$
$\varphi_1^{\text{out}}(2) = 0$
$\varphi_1^{\text{out}}(3) = 1$

Let us consider a wiring diagram $\psi : \mathcal{B}_{4,3} \to \mathcal{B}_{2,2}$ (defined in Example 3.2.10). It will look like:

— $\psi^{\text{in}} : (4, \tau_4) \to (2, \tau_2) + (3, \tau_3) = \langle A, A, A, B \rangle \to \langle A, B, B, C, A \rangle$
— $\psi^{\text{out}} : (2, \tau_2) \to (3, \tau_3) = \langle A, B \rangle \to \langle B, C, A \rangle$

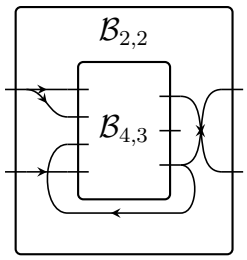
We can build specific wiring diagrams:

$\psi_0^{\text{in}}(0) = 0$
$\psi_0^{\text{in}}(1) = 0$
$\psi_0^{\text{in}}(2) = 0$
$\psi_0^{\text{in}}(3) = 1$



$\psi_0^{\text{out}}(0) = 2$
$\psi_0^{\text{out}}(1) = 0$

$\psi_1^{\text{in}}(0) = 0$
$\psi_1^{\text{in}}(1) = 0$
$\psi_1^{\text{in}}(2) = 4$
$\psi_1^{\text{in}}(3) = 1$



$\psi_1^{\text{out}}(0) = 2$
$\psi_1^{\text{out}}(1) = 0$

What about the reverse wiring diagram $\rho : \mathcal{B}_{2,2} \to \mathcal{B}_{4,3}$? It will look like:

— $\rho^{\text{in}} : (2, \tau_2) \to (4, \tau_4) + (2, \tau_2) = \langle A, B \rangle \to \langle A, A, A, B, A, B \rangle$

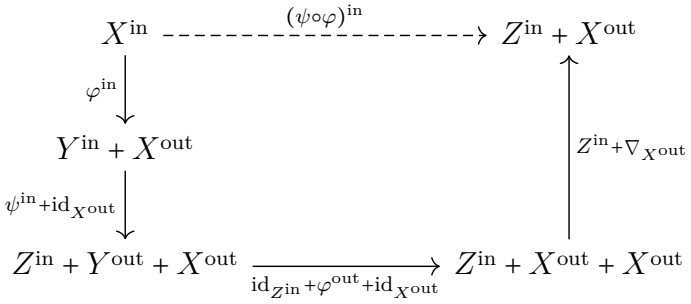
— $\rho^{\text{out}} : (3, \tau_3) \to (2, \tau_2) = \langle B, C, A\rangle \to \langle A, B\rangle$

There is no problem with the first $\mathscr{C}$-typed function, but we already know that there is no $\mathscr{C}$-typed function $(3, \tau_3) \to (2, \tau_2)$ (cf. Example 3.2.2).

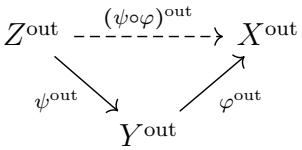
We can now compose the wiring diagrams:

Definition 3.2.13 (Composition of wiring diagrams [19])

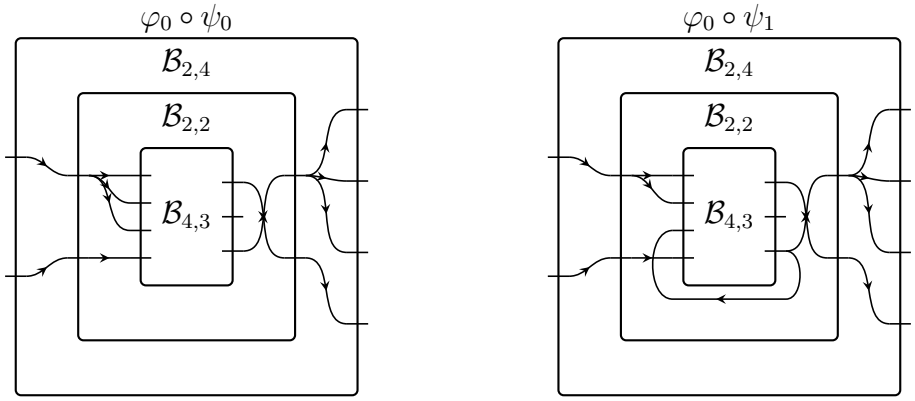
Let $\varphi : X \to Y$ and $\psi : Y \to Z$ be two wiring diagrams. We define their *composition*, denoted $\psi \circ \phi$, as the pair $((\psi \circ \phi)^{\text{in}}, (\psi \circ \phi)^{\text{out}})$, where $(\psi \circ \phi)^{\text{in}}$ is defined such that the following diagram commutes:

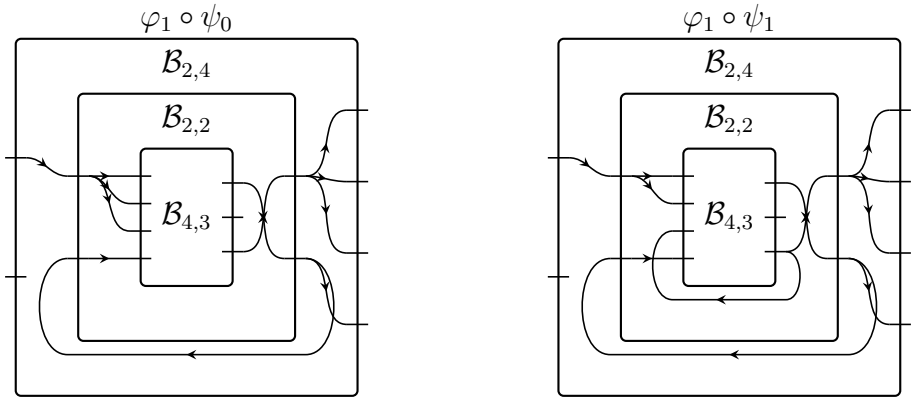


and $(\psi \circ \phi)^{\text{out}}$ is defined such that the following diagram commutes:



Example 3.2.14. We can compose $\psi_i : \mathcal{B}_{4,3} \to \mathcal{B}_{2,2}$ with $\varphi_j : \mathcal{B}_{2,2} \to \mathcal{B}_{2,4}$ $(i, j \in 2)$ (defined in Example 3.2.12).





Definition 3.2.15 (Category of $\mathscr{C}$-boxes and wiring diagrams [19])

The *category $\mathscr{W}_{\mathscr{C}}$ of $\mathscr{C}$-boxes and wiring diagrams* is defined as follows:

Objects: An object in $\mathscr{W}_{\mathscr{C}}$ is a $\mathscr{C}$-box

Morphisms: A morphism between two $\mathscr{C}$-boxes X and Y is a wiring diagram $\phi : X \to Y$

Identities: An identity morphism on X is the identity wiring diagram

Composition: The composition of wiring diagrams is the composition defined in definition 3.2.13

3.2.4 Monoidal structure of the category of boxes

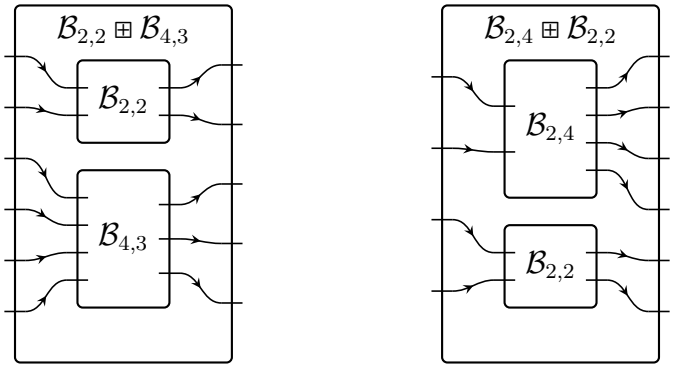
The category $\mathscr{W}_{\mathscr{C}}$ has a monoidal structure for the parallel composition of boxes, that corresponds to the intuitive idea of parallelising boxes.

Definition 3.2.16 (Parallel composition of boxes [19])

Let $X = (X^{\text{in}}, X^{\text{out}})$ and $Y = (Y^{\text{in}}, Y^{\text{out}})$ be two $\mathscr{C}$-boxes.
The *parallel composition*, or *sum*, of X and Y, denoted $X \boxplus Y$, is the box $X \boxplus Y = (X^{\text{in}} + Y^{\text{in}}, X^{\text{out}} + Y^{\text{out}})$, where $+$ is the sum of $\mathscr{C}$-typed finite sets (cf. Definition 3.2.3).

The parallel composition of two $\mathscr{C}$-boxes summarises to the concatenation of both input ports, and both output ports.

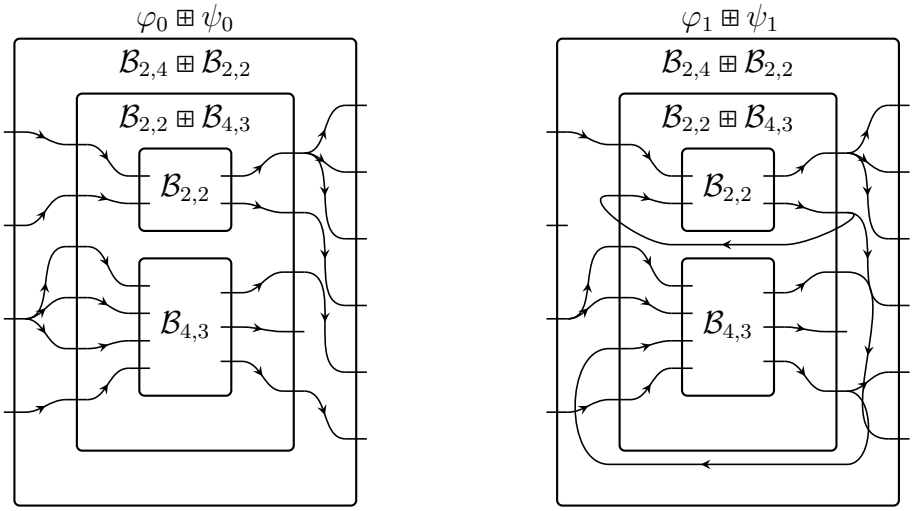
Example 3.2.17. Any two $\mathscr{C}$-boxes can be put in parallel. For example:



Definition 3.2.18 (Parallel composition of wiring diagrams [19])

Let $\varphi : X \to Y = (\varphi^{\text{in}}, \varphi^{\text{out}})$ and $\psi : Y \to Z = (\psi^{\text{in}}, \psi^{\text{out}})$ be two wiring diagrams.
The *parallel composition*, or *sum*, of φ and ψ, denoted $\varphi \boxplus \psi$, is the wiring diagram $\varphi \boxplus \psi = (\varphi^{\text{in}} + \psi^{\text{in}}, \varphi^{\text{out}} + \psi^{\text{out}})$, where $+$ is the sum of $\mathscr{C}$-typed functions (cf. Definition 3.2.4).

Example 3.2.19. Using the notations of Example 3.2.12, we can build $\varphi_i \boxplus \psi_i : \mathcal{B}_{2,2} \boxplus \mathcal{B}_{4,3} \to \mathcal{B}_{2,4} \boxplus \mathcal{B}_{2,2}$ $(i \in 2)$:



Proposition 3.2.20

The category $\mathscr{W}_{\mathscr{C}}$ has the following properties:

— The *closed box* $\square$, defined as $\square = (0, 0)$, where 0 is $\mathscr{C}$-typed finite set

$(\varnothing, \varnothing \to \mathscr{C})$ defined in 3.2.5, is the unit for the sum of boxes $\boxplus$.

— $\mathscr{W}_{\mathscr{C}}$ has a symmetric monoidal structure for the sum of boxes $\boxplus$, with $\square$ as the unit.

Proof. See [19]. □

3.2.5 Dependent product of boxes

The aim of this section is to extend the notion of dependent product (Definition 3.2.6) to $\mathscr{C}$-boxes and wiring diagrams.

Definition 3.2.21 (Dependent product of a $\mathscr{C}$-box [19])

The *dependent product* $\widehat{X}$ of the $\mathscr{C}$-box $X = (X^{\text{in}}, X^{\text{out}})$ is the pair $\widehat{X} = \left(\widehat{X^{\text{in}}}, \widehat{X^{\text{out}}}\right)$.

Remark 3.2.22. The dependent product of $X_0 \boxplus X_1$ is simply the $\mathscr{C}$-box:

$$\widehat{X_0 \boxplus X_1} = \left(\widehat{X_0^{\text{in}}} \times \widehat{X_1^{\text{in}}}, \widehat{X_0^{\text{out}}} \times \widehat{X_1^{\text{out}}}\right)$$

Definition 3.2.23 (Dependent product of wiring diagrams [19])

The *dependent product* $\widehat{X}$ of the wiring diagram $\varphi : X \to Y$ is the pair $\widehat{\varphi} = \left(\widehat{\varphi^{\text{in}}}, \widehat{\varphi^{\text{out}}}\right)$.

Remark 3.2.24. The dependent product $\varphi_0 \boxplus \varphi_1$ is $\widehat{\varphi_0 \boxplus \varphi_1} = \left(\widehat{\varphi_0^{\text{in}}} \times \widehat{\varphi_1^{\text{in}}}, \widehat{\varphi_0^{\text{out}}} \times \widehat{\varphi_1^{\text{out}}}\right)$.

Proposition 3.2.25

Let $\varphi : X \to Y$ and $\psi : Y \to Z$. The dependent product of $\psi \circ \varphi$ is the pair $\widehat{\psi \circ \varphi} = \left(\widehat{(\psi \circ \varphi)^{\text{in}}}, \widehat{(\psi \circ \varphi)^{\text{out}}}\right)$ where:

— $\widehat{(\psi \circ \varphi)^{\text{in}}}(x, z) = \widehat{\varphi^{\text{in}}}\left(\widehat{\psi^{\text{in}}}\left(z, \widehat{\varphi^{\text{out}}}(x)\right), x\right)$

— $\widehat{(\psi \circ \varphi)^{\text{out}}}(x) = \widehat{\psi^{\text{out}}}\left(\widehat{\varphi^{\text{out}}}(x)\right)$

Proof. See [19]. □

Remark 3.2.26. The dependent product of $\mathscr{C}$-boxes and wiring diagrams could be described in terms of monoidal functors; however the codomain of this functor is not $\mathscr{C} \times \mathscr{C}$ as expected, but a category that has the same objects (pairs of objects (A, B) of $\mathscr{C}$) but whose morphisms are pairs of morphisms $\left(\widehat{f^{\text{in}}}, \widehat{f^{\text{out}}}\right) : (A_0, B_0) \to (A_1, B_1)$ such that $\widehat{f^{\text{in}}}$ is the morphism $\widehat{f^{\text{in}}} : A_1 \times B_0 \to A_0$ in $\mathscr{C}$ and $\widehat{f^{\text{out}}}$ is the morphism $\widehat{f^{\text{out}}} : B_0 \to B_1$ in $\mathscr{C}$. The composition law is the one given in Proposition 3.2.25.

Until now, we have only defined a category of $\mathscr{C}$-boxes, with interesting properties. These $\mathscr{C}$-boxes are exactly as their name suggests: empty boxes. The extension of the dependent product to $\mathscr{C}$-boxes is a necessary step in order to define the "inhabitants" of $\mathscr{C}$-boxes.

3.3 Discrete systems and their equivalences

In this section and for the rest of the chapter, we will consider the special case where $\mathscr{C} = \mathbf{Sets}$. Thus, in general, we will simply call "boxes" what we introduced as "**Sets**-boxes". We denote the symmetric monoidal category of boxes as $\mathscr{W}_{\mathbf{Sets}}$.

3.3.1 Definition and basic properties

The notions introduced in this section come from [19]. The properties stated here are proven in the same article.

Definition 3.3.1 (Discrete systems [19])

Let $X = (X^{\text{in}}, X^{\text{out}}) \in \mathscr{W}_{\mathbf{Sets}}$ be a box.
A *discrete system for the box* X, or *discrete system* for short, is a 4-tuple

$$F = \left(S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}\right)$$

where:

- $S_F \in \mathbf{Sets}$ is the *state set* of F
- $f^{\text{rdt}} : S_F \to \widehat{X^{\text{out}}}$ is its *readout function*
- $f^{\text{upd}} : \widehat{X^{\text{in}}} \times S_F \to S_F$ is its *update function*
- $s_0 \in S_F$ is its *initial state*

We denote by $\mathrm{DS}(X)$ the set of all discrete systems for the box X.

Remark 3.3.2. In Proposition 3.2.20, we defined the closed box $\square = (0, 0)$, where 0 denotes $(\varnothing, \tau : \varnothing \to \mathbf{Sets})$. Its dependent product is $\widehat{\square} = \left(\prod_{p \in \varnothing} \tau(p), \prod_{p \in \varnothing} \tau(p)\right) \cong$

$(1', 1')$, where $1'$ is any typed finite set of the form $(1, \tau : 1 \to \mathbf{Sets})$. As a consequence, we have:

$$\mathrm{DS}(\square) \cong \left\{ \left(S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}\right) \;\middle|\; \begin{array}{l} S_F \in \mathbf{Sets}, f^{\mathrm{rdt}} : S_F \to 1, f^{\mathrm{upd}} : 1 \times S_F \to S_F, \\ s_{F,0} \in S_F \end{array} \right\} \tag{3.1}$$

In other words, an inhabitant of a closed box is a dynamical system with no inputs and no outputs, just a set S and a function $S \to S$.

Remark 3.3.3. From a set-theoretic point of view, $\mathrm{DS}(X)$ is too big to be a set. A potential solution is to define the $\mathrm{DS}(X)$ within a set big enough for our purposes; for example, the set V_{ω_1} from the von Neumann hierarchy of sets, which contains the usual sets, vector spaces, measurable spaces, Hausdorff spaces, fields, etc. used in mathematics ($V_{\omega \times 2}$ suffices [27, Lemma 2.9]).

In the following, we will continue to write $\mathrm{DS}(X)$ (and similarly for mappings) with the state set in $S_F \in \mathbf{Sets}$ for the sake of understandability, but in case set-theoretic problems emerge, we should not write $S_F \in \mathbf{Sets}$ but $S_F \in V_{\omega_1}$.

Note that discrete systems can be viewed as a generalisation of automata. They have no final states, the transition function is always a function, i.e. all discrete systems are deterministic, the input alphabet can be infinite, and the transition function is always defined on every input and every state. Discrete systems are not automata that recognize a language, but rather, automata that take any input stream and return an output stream based on the states it transitioned to; that is, discrete systems are a generalisation of transducers as defined in [33]. Alternatively, discrete systems exactly correspond to the *sequential automata* in [34].

Example 3.3.4. For this example, we generalise the notation seen in Definition 3.2.1 to the set $\mathbf{TFS}_{\mathbb{R}}$ of $\mathbb{R}$-typed finite sets, seen as lists of real numbers, that is, finite sequences of real numbers:

$$\mathbf{TFS}_{\mathbb{R}} = \{(n, \tau) \mid n \in \mathbb{N}, \tau : n \to \mathbb{R}\}$$

Here, this is a set, not a category; besides, we use $n \in \mathbb{N}$ instead of $P \in \mathbf{FinSets}$ so that we define a set. As a discrete category (having only identity morphisms), it is equivalent to $\mathbf{TFS}_{\mathbb{R}}$ obtained by also considering $\mathbb{R}$ as a discrete category.

We also generalise the sum of finite sequences, seen as the concatenation.

Consider the box $X_0 = (\langle \mathbb{R} \rangle, \langle \mathbb{R} \rangle) \in \mathscr{W}_{\mathbf{Sets}}$. We define the following discrete system $F_0 \in \mathrm{DS}(X_0)$:

— $S_{F_0} = \mathbf{TFS}_{\mathbb{R}}$

— $f_0^{\mathrm{upd}} : \left\{ \begin{array}{ccc} \mathbb{R} \times \mathbf{TFS}_{\mathbb{R}} & \to & \mathbf{TFS}_{\mathbb{R}} \\ (a, \langle a_0, \ldots, a_{n-1} \rangle) & \mapsto & \langle a_0, \ldots, a_{n-1}, a \rangle \end{array} \right.$

— $f_0^{\mathrm{rdt}} : \begin{cases} \mathbf{TFS}_{\mathbb{R}} & \to & \mathbb{R} \\ \langle a_0, \ldots, a_{n-1} \rangle & \mapsto & \max\left(\langle a_0, \ldots, a_{n-1} \rangle\right) \end{cases}$

— $s_{F_0,0} = 0$ (where 0 is the empty list as defined in Proposition 3.2.5)

In this example, the state set $S_{F_0} = \mathbf{TFS}_{\mathbb{R}}$ is uncountably infinite and clearly works as a memory. This discrete system F_0 takes a real number as an input, appends it to its memory, and computes the maximum value of the stored list.

A more complicated discrete system could return several results; for example, in the box $X_1 = (\langle \mathbb{R} \rangle, \langle \mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R} \rangle) \in \mathscr{W}_{\mathbf{Sets}}$, we can define $F_1 \in \mathrm{DS}(X_1)$ such that:

— $S_{F_1} = S_{F_0} = \mathbf{TFS}_{\mathbb{R}}$, $f_1^{\mathrm{upd}} = f_0^{\mathrm{upd}}$ and $s_{F_1,0} = 0$ (same state set, update function and initial state as F_0)

— $f_1^{\mathrm{rdt}} : \begin{cases} \mathbf{TFS}_{\mathbb{R}} & \to & \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \\ \langle a_0, \ldots, a_{n-1} \rangle & \mapsto & \left(\max\left(\langle a_0, \ldots, a_{n-1} \rangle\right), \min\left(\langle a_0, \ldots, a_{n-1} \rangle\right), \right. \\ & & \left. \mathrm{mean}\left(\langle a_0, \ldots, a_{n-1} \rangle\right), \mathrm{var}\left(\langle a_0, \ldots, a_{n-1} \rangle\right)\right) \end{cases}$

Or we could add a switch so that we can decide what output we want; in the box $X_2 = (\langle \mathbb{R}, 4 \rangle, \langle \mathbb{R} \rangle)$, we define F_2 as:

— $S_{F_2} = \mathbf{TFS}_{\mathbb{R}} \times 4$ (here we see 4 as its set-theoretic counterpart: $4 = \{0, 1, 2, 3\}$)

— $f_2^{\mathrm{upd}} : \begin{cases} \mathbb{R} \times 4 \times \mathbf{TFS}_{\mathbb{R}} \times 4 & \to & \mathbf{TFS}_{\mathbb{R}} \times 4 \\ (a, b, \langle a_0, \ldots, a_{n-1} \rangle, c) & \mapsto & (\langle a_0, \ldots, a_{n-1}, a \rangle, b) \end{cases}$

— $f_2^{\mathrm{rdt}} : \begin{cases} \mathbf{TFS}_{\mathbb{R}} \times 4 & \to & \mathbb{R} \\ (\langle a_0, \ldots, a_{n-1} \rangle, b) & \mapsto & \begin{cases} \max\left(\langle a_0, \ldots, a_{n-1} \rangle\right) & \text{if } b = 0 \\ \min\left(\langle a_0, \ldots, a_{n-1} \rangle\right) & \text{if } b = 1 \\ \mathrm{mean}\left(\langle a_0, \ldots, a_{n-1} \rangle\right) & \text{if } b = 2 \\ \mathrm{var}\left(\langle a_0, \ldots, a_{n-1} \rangle\right) & \text{otherwise} \end{cases} \end{cases}$

— $s_{F_2,0} = (0, 0)$

We could even add a reset button; in the box $X_3 = (\langle \mathbb{R}, 4, 2 \rangle, \langle \mathbb{R} \rangle)$, we define F_3 as:

— $S_{F_3} = S_{F_2} = \mathbf{TFS}_{\mathbb{R}} \times 4$, $f_3^{\mathrm{rdt}} = f_2^{\mathrm{rdt}}$ and $s_{F_3,0} = s_{F_2,0} = (0, 0)$ (same state set, same readout function and same initial state as F_2)

— $f_3^{\mathrm{upd}} : \begin{cases} \mathbb{R} \times 4 \times 2 \times \mathbf{TFS}_{\mathbb{R}} \times 4 & \to & \mathbf{TFS}_{\mathbb{R}} \times 4 \\ (a, b, r, \langle a_0, \ldots, a_{n-1} \rangle, c) & \mapsto & \begin{cases} s_{F_3,0} & \text{if } r = 0 \\ (\langle a_0, \ldots, a_{n-1}, a \rangle, b) & \text{otherwise} \end{cases} \end{cases}$

We previously viewed general boxes (objects in $\mathscr{W}_{\mathbf{Sets}}$) as empty frames. Discrete systems are the objects that "live" inside. One can draw a parallel with programming: a $\mathscr{C}$-box is the signature of the function, that is, its accepted types of inputs and outputs, and the discrete system is the actual code of the function.

In the rest of this chapter, a discrete system $F = (S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0})$ will often be represented as the following two-arrow graph:

$$F: \qquad \widehat{X^{\text{in}}} \times S_F \xrightarrow{f^{\text{upd}}} S_F \xrightarrow{f^{\text{rdt}}} \widehat{X^{\text{out}}}$$

The first function describes how a state and an input are transformed into a new state; the second describes how the state is output, or "read out". In general, the initial state $s_{F,0} \in S_F$ will not be represented in these diagrams, though it is implicitly there.

Discrete systems are part of the more general class of *dynamical systems*. We can define other types of dynamical systems depending on the category $\mathscr{C}$ that we are interested in. If $\mathscr{C}$ is the category **Euc** of Euclidean spaces, then we will refer to *continuous systems*. For more examples, see [19].

Definition 3.3.5 (DS-application of a wiring diagram [19])

Let $\varphi : X \to Y$ be a wiring diagram. Let $F = (S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}) \in \text{DS}(X)$.
The *DS-application of φ to F*, denoted $\text{DS}(\varphi)(F)$, is the discrete system:

$$\text{DS}(\varphi)(F) = \left(S_G, g^{\text{rdt}}, g^{\text{upd}}, s_{G,0}\right) \in \text{DS}(Y)$$

such that:

- $S_G = S_F$
- $g^{\text{rdt}} : s \mapsto \widehat{\varphi^{\text{out}}}\left(f^{\text{rdt}}(s)\right)$
- $g^{\text{upd}} : (y, s) \mapsto f^{\text{upd}}\left(\widehat{\varphi^{\text{in}}}\left(y, f^{\text{rdt}}(s)\right), s\right)$
- $s_{G,0} = s_{F,0}$

We can view $\text{DS}(\varphi)(F)$ as the discrete system we obtain from F by implementing the wiring diagram φ.

Definition 3.3.6 (Parallel composition of discrete systems [19])

Let X_0, X_1 be boxes and let $F_i = \left(S_{F_i}, f_i^{\text{rdt}}, f_i^{\text{upd}}, s_{F_i,0}\right) \in \text{DS}(X_i)$ $(i \in 2)$ be discrete systems.
The *parallel composition of F_0 and F_1*, denoted $F_0 \boxtimes F_1$, is the discrete system

$$\left(S_G, g^{\text{rdt}}, g^{\text{upd}}, s_{G,0}\right) \in \text{DS}(X_0 \boxplus X_1)$$

such that:

- $S_G = S_{F_0} \times S_{F_1}$

— $s_{G,0} = (s_{F_0,0}, s_{F_1,0})$

— $g^{\mathrm{rdt}} = f_0^{\mathrm{rdt}} \times f_1^{\mathrm{rdt}} : S_{F_0} \times S_{F_1} \to \widehat{X_0^{\mathrm{out}}} \times \widehat{X_1^{\mathrm{out}}}$

— $g^{\mathrm{upd}} : \widehat{X_0^{\mathrm{in}}} \times \widehat{X_1^{\mathrm{in}}} \times S_{F_0} \times S_{F_1} \to S_{F_0} \times S_{F_1}$ makes the following diagram commute:

$$\begin{array}{ccc} \widehat{X_0^{\mathrm{in}}} \times \widehat{X_1^{\mathrm{in}}} \times S_{F_0} \times S_{F_1} & \xrightarrow{g^{\mathrm{upd}}} & S_{F_0} \times S_{F_1} \\ \updownarrow \cong & & \updownarrow = \\ \widehat{X_0^{\mathrm{in}}} \times S_{F_0} \times \widehat{X_1^{\mathrm{in}}} \times S_{F_1} & \xrightarrow{f_0^{\mathrm{upd}} \times f_1^{\mathrm{upd}}} & S_{F_0} \times S_{F_1} \end{array}$$

We also define the parallel composition of $\mathrm{DS}(X_0)$ and $\mathrm{DS}(X_1)$, denoted $\mathrm{DS}(X_0) \boxtimes \mathrm{DS}(X_1)$, by:

$$\mathrm{DS}(X_0) \boxtimes \mathrm{DS}(X_1) = \{F_0 \boxtimes F_1 \mid F_0 \in \mathrm{DS}(X_0), F_1 \in \mathrm{DS}(X_1)\}.$$

Proposition 3.3.7

Parallel composition $(F_0, F_1) \mapsto F_0 \boxtimes F_1$ provides a natural map $\mathrm{DS}(X_0) \times \mathrm{DS}(X_1) \to \mathrm{DS}(X_0 \boxplus X_1)$.

Proof. See [19]. □

Theorem 3.3.8

$DS: \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ as defined is a lax monoidal functor.

Proof. See [19] □

3.3.2 An external equivalence relation on dynamical systems

Via the monoidal functor DS, a box contains a specified sort of discrete system (depending on the ports of the box). For an exterior spectator, the content of the box does not matter; what matters is the way it transforms input streams to output streams. Thus, even if two boxes contain different discrete systems, for example one with an infinite state set, and the other with a finite state set, as long as they both give the same output in response to the same input, then they are viewed as "equivalent" from an external point of view.

The following definitions formalise this idea.

Definition 3.3.9 (Input and output streams)

Let $F = (S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}) \in \text{DS}(X)$.
An *input stream (for X)* is a finite sequence $\overline{x^{\text{in}}} = (x_i^{\text{in}})_{i \in n} \in \left(\widehat{X^{\text{in}}}\right)^n$, where $n \in \mathbb{N}$.
The *output stream produced by F when given $\overline{x^{in}}$*, denoted $F\left(\overline{x^{\text{in}}}\right)$, is the stream $\overline{x^{\text{out}}}$ defined by the following recursive system:

$$\begin{cases} s_0 &= s_{F,0} \\ s_{i+1} &= f^{\text{upd}}\left(x_i^{\text{in}}, s_i\right) \\ x_i^{\text{out}} &= f^{\text{rdt}}\left(s_{i+1}\right) \end{cases}$$

We refer to the state s that F reaches after having processed the input stream $\overline{x^{\text{in}}}$ as *resulting state of F*, and denote it $F_{\text{res}}(\overline{x^{\text{in}}})$. Formally, if $\overline{x^{\text{in}}} = (x_i^{\text{in}})_{i \in n}$, then according to the previous recursive system, the resulting state of F is $F_{\text{res}}(\overline{x^{\text{in}}}) = s_n$.

Remark 3.3.10. According to the notation proposed in the Notation section, $\overline{x^{\text{in}}} = (x_i^{\text{in}})_{i \in n} \in \left(\widehat{X^{\text{in}}}\right)^n$ will be written $\overline{x^{\text{in}}} \in \widehat{X^{\text{in}}}$.

Remark 3.3.11. Definition 3.3.9 is a continuation of the definitions of *run maps* and *behaviours* in [34], which are functions that assign respectively the resulting state and the last output of the automaton given an input stream. The results we obtain with our notations are similar to those in [34].

Definition 3.3.12 (Equivalence as stream transducers)

Let $F = (S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0})$ and $G = (S_G, g^{\text{rdt}}, g^{\text{upd}}, s_{G,0})$ be two discrete systems.
We say that *F and G are equivalent as stream transducers*, and we write $F \equiv G$, when, $\forall \overline{x^{\text{in}}} \in \widehat{X^{\text{in}}}$, $F\left(\overline{x^{\text{in}}}\right) = G\left(\overline{x^{\text{in}}}\right)$.

It is easy to see that:

Proposition 3.3.13

The relation $\equiv$ is an equivalence relation on the set $\text{DS}(X)$, for any box X.

3.3.3 An internal equivalence relation on dynamical systems

The relation $\equiv$ defined above does not give any information on the links between two discrete systems that are equivalent as stream transducers. In this subsection, we define another equivalence relation that provides an internal point of view. We then prove that the two equivalence relations are the same.

In the following, $X = (X^{\text{in}}, X^{\text{out}})$ is any box.

Definition 3.3.14 (Simulation relation)

Suppose given $F = (S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0})$ and $G = (S_G, g^{\text{rdt}}, g^{\text{upd}}, s_{G,0})$ in $\mathrm{DS}(X)$. We say that F *simulates* G, and we write $F \vdash G$, if there exists $\alpha : S_F \to S_G$ such that $s_{G,0} = \alpha(s_{F,0})$ and such that $\alpha \circ f^{\text{upd}} = g^{\text{upd}} \circ \left(\mathrm{id}_{\widehat{X^{\text{in}}}}, \alpha\right)$, and $f^{\text{rdt}} = g^{\text{rdt}} \circ \alpha$; that is, preserving the initial state and making the following two diagrams commute:

$$
\begin{array}{lccccc}
F: & \widehat{X^{\text{in}}} \times S_F & \xrightarrow{f^{\text{upd}}} & S_F & \xrightarrow{f^{\text{rdt}}} & \widehat{X^{\text{out}}} \\
 & \Big\downarrow{\scriptstyle \left(\mathrm{id}_{\widehat{X^{\text{in}}}}, \alpha\right)} & & \Big\downarrow{\scriptstyle \alpha} & & \Big\updownarrow{\scriptstyle =} \\
G: & \widehat{X^{\text{in}}} \times S_G & \xrightarrow[g^{\text{upd}}]{} & S_G & \xrightarrow[g^{\text{rdt}}]{} & \widehat{X^{\text{out}}}
\end{array}
\tag{3.2}
$$

We refer to α as a *simulation function*: it witnesses the simulation $F \vdash G$.

A priori, the simulation relation does not relate the output of the two discrete systems F and G (though this does follow; see Lemma 3.3.19); it only declares a correspondence between both their state sets and update and readout functions. Both discrete systems can work in parallel; their state sets need not be the same, nor even of the same cardinality, but they somehow *coordinate* via the map α. The function α draws the parallel between the internal machinery of F and that of G.

For the rest of the chapter, we will be more interested in the simulation relation $F \vdash G$ than any particular simulation function witnessing it: any one will do.

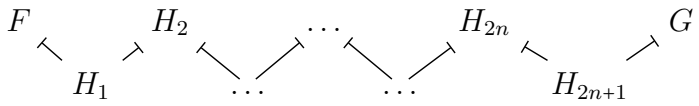
Remark 3.3.15. Definition 3.3.14 refers to the existence of morphisms between two automata as described in the automata theory litterature [34]. The existence of such morphisms suffices for our purposes. We are a bit more restrictive here, as the outputs need to be the same in both automata, while in the usual definition of morphisms, automata can have different output alphabets, as long as there is a function to convert one output into the other.

The simulation relation is not necessarily an equivalence relation and is not

enough for our purpose, but we can use it to generate the equivalence relation we actually need.

Definition 3.3.16 (Internal equivalence relation on DS (X))

Let $F, G \in \mathrm{DS}(X)$.
We say that F *and* G *are simulation-equivalent*, and we write $F \sim G$, if there exists a finite sequence $(H_i)_{i \in N} \in \mathrm{DS}(X)$ such that:



It is not hard to check that:

Theorem 3.3.17

The equivalence relation $\sim$ is the equivalence relation generated by $\vdash$, that is, $\sim$ is the smallest equivalence relation R such that $\vdash \subseteq R$.

Finally, we need to show that the equivalence relation $\sim$ actually groups discrete systems that have the same behaviour as a stream transducer, in the sense of Definition 3.3.12; that is, the external and the internal equivalence relation are the same.

Lemma 3.3.18

Let $F, G \in \mathrm{DS}(X)$. If $F \equiv G$ then $\exists H \in \mathrm{DS}(X)$ such that $H \vdash F$ and $H \vdash G$.

Proof. Let $F = (S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0})$ and $G = (S_G, g^{\mathrm{rdt}}, g^{\mathrm{upd}}, s_{G,0})$.

Take $H = (S_H, h^{\mathrm{rdt}}, h^{\mathrm{upd}}, s_{H,0})$ such that:

- $S_H = \left\{ (s, s') \in S_F \times S_G \mid \exists \overline{x^{\mathrm{in}}} \text{ such that } F_{\mathrm{res}}(\overline{x^{\mathrm{in}}}) = s \text{ and } G_{\mathrm{res}}(\overline{x^{\mathrm{in}}}) = s' \right\}$
- $h^{\mathrm{upd}}(x, (s, s')) = \left(f^{\mathrm{upd}}(x, s), g^{\mathrm{upd}}(x, s')\right)$
- $h^{\mathrm{rdt}}(s, s') = f^{\mathrm{rdt}}(s)$
- $s_{H,0} = (s_{F,0}, s_{G,0})$

Take as simulation functions the respective projections π_{S_F} and π_{S_G}.

It is easy to see that the required diagrams in Definition 3.3.14 do commute. For all $(s, s') \in S_H$, we have $h^{\mathrm{rdt}} \circ \pi_{S_F}(s, s') = f^{\mathrm{rdt}}(s)$ by defintion of h^{rdt}. Also, for all $(s, s') \in S_H$, $h^{\mathrm{rdt}} \circ \pi_{S_G}(s, s') = f^{\mathrm{rdt}}(s) = g^{\mathrm{rdt}}(s')$ because there exists some

stream $\overline{x^{\text{in}}}$ such that F results in s and G results in s'; besides, as $F \equiv G$, we have $F\left(\overline{x^{\text{in}}}\right) = G\left(\overline{x^{\text{in}}}\right)$, which implies $f^{\text{rdt}}(s) = g^{\text{rdt}}(s')$. Consequently, the diagrams commute and H simulates both F and G.

□

Lemma 3.3.19

Let $F, G \in \mathrm{DS}(X)$. If $F \vdash G$ then $F \equiv G$.

Proof. Follows by induction on the length of an input stream $(x_i^{\text{in}})_{i \in n} \in \widehat{X^{\text{in}}}$. □

Theorem 3.3.20

$\forall F, G \in \mathrm{DS}(X), F \equiv G \iff F \sim G$; or equivalently: $\equiv = \sim$.

Proof. To prove $\equiv = \sim$, we need $\equiv \subseteq \sim$ and $\sim \subseteq \equiv$.

Suppose first that $F, G \in \mathrm{DS}(X)$ are dynamical systems such that $F \equiv G$. According to Lemma 3.3.18, there exists a $H \in \mathrm{DS}(X)$ such that $H \vdash F$ and $H \vdash G$, and hence $F \sim G$ by Definition 3.3.16. This establishes $\equiv \subseteq \sim$.

We now show that $\sim \subseteq \equiv$. According to Proposition 3.3.13, $\equiv$ is an equivalence relation, and according to Lemma 3.3.19, $\equiv$ contains $\vdash$. Theorem 3.3.17 states that $\sim$ is the smallest equivalence relation that contains $\vdash$; necessarily, we have $\sim \subseteq \equiv$. □

The goal of this chapter is to show that the behaviour of a general discrete system can be emulated by some specific wiring of some other discrete system, chosen with constraints (for example, on its internal structure). As far as we know, this result cannot be obtained with a pure equality. However, we have a description of what it means to be equivalent, both from an internal and from an external point of view, with the assurance that, seen as a blackbox, the "inhabited" box remains unchanged.

As we are not using real equalities, we need to define relations between sets that correspond to the usual inclusion and equality.

Definition 3.3.21 (Inclusion/equality up to equivalence)

Let $D, E \subseteq \mathrm{DS}(X)$. We consider the equivalence relation $\sim$ from Definition 3.3.16 (or, equivalently, in Definition 3.3.12).
We say that D *is a subset of* E *up to equivalence*, and we write $D \sqsubseteq E$, when $\forall d \in D, \exists e \in E, d \sim e$.
We say that D *is equal to* E *up to equivalence*, or D *is equivalent to* E, and we write $D \approx E$, when $D \sqsubseteq E$ and $D \sqsupseteq E$.

If $A, B : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ are functors, then we write $A \sqsubseteq B$ when, for all box X, we have $A(X) \sqsubseteq B(X)$. We write $A \approx B$, when $A \sqsubseteq B$ and $A \sqsupseteq B$.
If $M, N : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ are mappings (not necessarily functors), then we write $M \subseteq N$ when, for all boxes X and Y, we have $M(X) \subseteq N(X)$ and $\mathrm{Mor}_{\mathbf{Sets}}(M(X), M(Y)) \subseteq \mathrm{Mor}_{\mathbf{Sets}}(N(X), N(Y))$.

3.4 Main results

Before we introduce the actual results of the chapter, we need a few more notions.

3.4.1 Algebras and closures

Definition 3.4.1 (Algebra)

Given a monoidal category $\mathscr{C}$, a functor $F : \mathscr{C} \to \mathbf{Sets}$ is called an *algebra over* $\mathscr{C}$ when it is a lax monoidal functor.

In our case, $\mathscr{C} = \mathscr{W}_{\mathbf{Sets}}$, and DS is an algebra by Theorem 3.3.8.

Definition 3.4.2 (Subalgebra)

Let $A : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ be an algebra over $\mathscr{W}_{\mathbf{Sets}}$. Let $\sigma_{X,Y} : A(X) \times A(Y) \to A(X \boxplus Y)$ denote its first coherence map (we recall that $\boxplus$ is the parallel composition of boxes (cf. Definition 3.2.16)).
A functor $B : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ is called a *subalgebra of* A when:

- $\forall X \in \mathscr{W}_{\mathbf{Sets}},\ B(X) \subseteq A(X)$
- $\forall X, Y \in \mathscr{W}_{\mathbf{Sets}}, \forall F \in B(X), \forall G \in B(Y), \sigma_{X,Y}(F, G) \in B(X \boxplus Y)$
- $\forall \varphi : X \to Y \in \mathscr{W}_{\mathbf{Sets}}, \forall F \in B(X), A(\varphi)(F) \in B(Y)$

Here, A and B are functors that transform boxes into sets. In our setting, the conditions can be interpreted as follows:

- (First item) Discrete systems generated by B are included in those generated by A;
- (Second item) The parallel composition of two discrete systems F and G generated by B is also generated by B.
- (Third item) B is stable through wiring diagrams: wiring a discrete system generated by B gives another discrete system generated by B.

Note that a subalgebra is itself an algebra.

Definition 3.4.3 (Closure)

Let $A : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ be an algebra over $\mathscr{W}_{\mathbf{Sets}}$.
Let $B : \mathrm{Ob}_{\mathscr{W}_{\mathbf{Sets}}} \to \mathrm{Ob}_{\mathbf{Sets}}$ be any map such that $\forall X \in \mathscr{W}_{\mathbf{Sets}},\ B(X) \subseteq A(X)$.
The closure of B, denoted $\mathrm{Clos}(B)$, is the intersection of all subalgebras of A that contain $B(X)$ for all $X \in \mathscr{W}_{\mathbf{Sets}}$. (Any intersection of subalgebras is a subalgebra.)

The closure of a map B can be understood as the minimal lax monoidal functor (or algebra) containing B.

3.4.2 Memoryless systems

Our first main result concerns the subclass of discrete systems that we call *memoryless*. We show that wiring together memoryless systems can lead to systems that have memory.

Definition 3.4.4 (Memoryless discrete systems)

Let $X = (X^{\mathrm{in}}, X^{\mathrm{out}})$ be a box.
A *memoryless discrete system for the box* X, or *memoryless discrete system* for short, is a discrete system $F = (S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}) \in \mathrm{DS}(X)$ such that f^{upd} immediately discards the previous state and uses only the current input; more precisely, such that f^{upd} factors as

$$f^{\mathrm{upd}} = \widehat{X^{\mathrm{in}}} \times S_F \xrightarrow{\pi_{\widehat{X^{\mathrm{in}}}}} \widehat{X^{\mathrm{in}}} \xrightarrow{f^u} S_F$$

for some $f^u : \widehat{X^{\mathrm{in}}} \to S_F$.
We denote by $\mathrm{DS}^{\mathrm{ML}}(X)$ the set of all memoryless discrete systems for the box X:

$$\mathrm{DS}^{\mathrm{ML}}(X) = \left\{ \left(S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}\right) \in \mathrm{DS}(X) \;\middle|\; \begin{array}{l} \exists f^u : \widehat{X^{\mathrm{in}}} \to S_F, \\ f^{\mathrm{upd}} = f^u \circ \pi_{\widehat{X^{\mathrm{in}}}} \end{array} \right\}$$

We call these discrete systems *memoryless* because we see the states as a kind of memory (as in Example 3.3.4). The discrete systems defined above transition from one state to another without checking their current state, i.e. without checking their memory.

The following definition is a natural restriction of memoryless discrete systems; as these systems are memoryless, the only goal of their states is to produce the output via their readout function. The simplest case is when the readout function is the identity.

Definition 3.4.5 (Direct-output discrete systems)

Let $X = (X^{\text{in}}, X^{\text{out}})$ be a box.
A *direct-output memoryless discrete system for the box* X, or *direct-output discrete system* for short, is a discrete system $F = (S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}) \in \text{DS}(X)$ such that:

- $S_F = \widehat{X^{\text{out}}}$
- $f^{\text{rdt}} = \text{id}_{\widehat{X^{\text{out}}}}$
- $f^{\text{upd}} = f^u \circ \pi_{\widehat{X^{\text{in}}}}$ for some $f^u : \widehat{X^{\text{in}}} \to \widehat{X^{\text{out}}}$

We denote by $\text{DS}^{\text{ML}}_{\text{out}}(X)$ the set of all direct-output discrete systems for the box X:

$$\text{DS}^{\text{ML}}_{\text{out}}(X) = \left\{ \left(S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}\right) \in \text{DS}^{\text{ML}}(X) \mid S = \widehat{X^{\text{out}}}, f^{\text{rdt}} = \text{id}_{\widehat{X^{\text{out}}}} \right\}$$
$$= \left\{ \left(\widehat{X^{\text{out}}}, \text{id}_{\widehat{X^{\text{out}}}}, f^{\text{upd}}, S_0\right) \in \text{DS}(X) \mid \exists f^u : \widehat{X^{\text{in}}} \to \widehat{X^{\text{out}}},\ f^{\text{upd}} = f^u \circ \pi_{\widehat{X^{\text{in}}}} \right\}$$

Remark 3.4.6. The maps $\text{DS}^{\text{ML}} : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ and $\text{DS}^{\text{ML}}_{\text{out}} : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ are not functors, because they are not closed under wiring. Indeed, the whole point is that the result of wiring together memoryless systems is not necessarily memoryless.

We can now prove one of the main results of this chapter, which is that every discrete system can be obtained (up to equivalence) by a memoryless system and a feedback loop. The feedback loop is responsible for holding the state that was originally in the discrete system.

Here is the formal statement.

Theorem 3.4.7

$\text{Clos}\left(\text{DS}^{\text{ML}}_{\text{out}}\right) \approx \text{DS}$.

Proof. We have $\text{DS}^{\text{ML}}_{\text{out}} \subseteq \text{DS}^{\text{ML}} \subseteq \text{DS}$, so $\text{Clos}\left(\text{DS}^{\text{ML}}_{\text{out}}\right) \subseteq \text{DS}$, thus $\text{Clos}\left(\text{DS}^{\text{ML}}_{\text{out}}\right) \sqsubseteq \text{DS}$. We need the opposite inclusion (up to equivalence) $\text{Clos}\left(\text{DS}^{\text{ML}}_{\text{out}}\right) \sqsupseteq \text{DS}$.

Let $Y = (Y^{\text{in}}, Y^{\text{out}}) \in \mathscr{W}_{\mathbf{Sets}}$, and let $G = (S_G, g^{\text{rdt}}, g^{\text{upd}}, s_{G,0}) \in \text{DS}(Y)$. We will

find $X \in \mathscr{W}_{\mathbf{Sets}}$, $F \in \mathrm{DS}^{\mathrm{ML}}_{\mathrm{out}}(X)$ and $\varphi : X \to Y$ such that $\mathrm{DS}(\varphi)(F) \sim G$.

Let $\delta_{S_G} = \langle S_G \rangle \in \mathbf{TFS}_{\mathbf{Sets}}$ be the list with one element, S_G, and consider the box $(\delta_{S_G}, \delta_{S_G})$ with only that port on the left and the right. We define X as the parallel composition of this box $(\delta_{S_G}, \delta_{S_G})$ and Y, that is:

$$\begin{aligned} X &= \left(X^{\mathrm{in}}, X^{\mathrm{out}}\right) \\ &= (\delta_{S_G}, \delta_{S_G}) \boxplus Y \\ &= \left(\delta_{S_G} + Y^{\mathrm{in}}, \delta_{S_G} + Y^{\mathrm{out}}\right) \end{aligned}$$

Note that $\widehat{X^{\mathrm{in}}} = S_G \times \widehat{Y^{\mathrm{in}}}$ and $\widehat{X^{\mathrm{out}}} = S_G \times \widehat{Y^{\mathrm{out}}}$. Thus, if $\overline{x^{\mathrm{in}}} \in \widehat{X^{\mathrm{in}}}$, then $\overline{x^{\mathrm{in}}} = \left(s, \overline{y^{\mathrm{in}}}\right)$. Similarly, if $\overline{x^{\mathrm{out}}} \in \widehat{X^{\mathrm{out}}}$, then $\overline{x^{\mathrm{out}}} = \left(s, \overline{y^{\mathrm{out}}}\right)$.

We choose $\varphi : X \to Y$ as the pair $(\varphi^{\mathrm{in}}, \varphi^{\mathrm{out}})$ of coproduct inclusions:

— $\varphi^{\mathrm{in}} : \begin{cases} \delta_{S_G} + Y^{\mathrm{in}} & \longrightarrow & Y^{\mathrm{in}} + \delta_{S_G} + Y^{\mathrm{out}} \\ x & \longmapsto & x \end{cases}$

— $\varphi^{\mathrm{out}} : \begin{cases} Y^{\mathrm{out}} & \longrightarrow & \delta_{S_G} + Y^{\mathrm{out}} \\ x & \longmapsto & x \end{cases}$

It follows from 3.2.23 that their dependent products $\widehat{\varphi^{\mathrm{in}}} : \widehat{Y^{\mathrm{in}}} \times S_G \times \widehat{Y^{\mathrm{out}}} \to S_G \times \widehat{Y^{\mathrm{in}}}$ and $\widehat{\varphi^{\mathrm{out}}} : S_G \times \widehat{Y^{\mathrm{out}}} \to \widehat{Y^{\mathrm{out}}}$ are projections.

Recall that the goal is to find a discrete system $F = (S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}) \in \mathrm{DS}^{\mathrm{ML}}_{\mathrm{out}}(X)$ such that $\mathrm{DS}(\varphi)(F) \sim G$. So define F as follows:

— $S_F = \widehat{X^{\mathrm{out}}} = S_G \times \widehat{Y^{\mathrm{out}}}$

— $f^{\mathrm{rdt}} = \mathrm{id}_{\widehat{X^{\mathrm{out}}}}$

— $f^{\mathrm{upd}}\left(\overline{x^{\mathrm{in}}}, \overline{x^{\mathrm{out}}}\right) = f^{\mathrm{upd}}\left(s, \overline{y^{\mathrm{in}}}, s', \overline{y^{\mathrm{out}}}\right) = \left(g^{\mathrm{upd}}\left(\overline{y^{\mathrm{in}}}, s\right), g^{\mathrm{rdt}}\left(g^{\mathrm{upd}}\left(\overline{y^{\mathrm{in}}}, s\right)\right)\right)$

— $s_{F,0} = \left(s_{G,0}, g^{\mathrm{rdt}}\left(s_{G,0}\right)\right)$

It is easy to see that F is in $\mathrm{DS}^{\mathrm{ML}}_{\mathrm{out}}(X)$ because f^{rdt} and f^{upd} have the correct form. So let $(S_H, h^{\mathrm{rdt}}, h^{\mathrm{upd}}, s_{H,0}) = \mathrm{DS}(\varphi)(F)$; we need to show it is equivalent to G. We compute each part of $\mathrm{DS}(\varphi)(F)$ according to Definition 3.3.5.

Its state set is as follows:

$$S_H = S_F = \widehat{X^{\mathrm{out}}} = S_G \times \widehat{Y^{\mathrm{out}}}$$

Its readout function is defined on an arbitrary x^{out} as follows:

$$\begin{aligned} h^{\mathrm{rdt}}\left(\overline{x^{\mathrm{out}}}\right) &= h^{\mathrm{rdt}}\left(s, \overline{y^{\mathrm{out}}}\right) \\ &= \widehat{\varphi^{\mathrm{out}}}\left(f^{\mathrm{rdt}}\left(s, \overline{y^{\mathrm{out}}}\right)\right) \\ &= \pi_{\widehat{Y^{\mathrm{out}}}}\left(\mathrm{id}_{\widehat{X^{\mathrm{out}}}}\left(s, \overline{y^{\mathrm{out}}}\right)\right) \\ &= \overline{y^{\mathrm{out}}} \end{aligned}$$

Its update function is defined on an arbitrary $\left(\overline{y^{\mathrm{in}}}, s, \overline{y^{\mathrm{out}}}\right)$ as follows:

$$\begin{aligned} h^{\mathrm{upd}}\left(\overline{y^{\mathrm{in}}}, s, \overline{y^{\mathrm{out}}}\right) &= f^{\mathrm{upd}}\left(\widehat{\varphi^{\mathrm{in}}}\left(\overline{y^{\mathrm{in}}}, f^{\mathrm{rdt}}\left(s, \overline{y^{\mathrm{out}}}\right)\right), \left(s, \overline{y^{\mathrm{out}}}\right)\right) \\ &= f^{\mathrm{upd}}\left(s, \overline{y^{\mathrm{in}}}, s, \overline{y^{\mathrm{out}}}\right) \\ &= \left(g^{\mathrm{upd}}\left(\overline{y^{\mathrm{in}}}, s\right), g^{\mathrm{rdt}}\left(g^{\mathrm{upd}}\left(\overline{y^{\mathrm{in}}}, s\right)\right)\right) \end{aligned}$$

Finally, its start state is as follows:

$$s_{F,0} = \left(s_{G,0}, g^{\mathrm{rdt}}\left(s_{G,0}\right)\right)$$

Consequently, the following diagram commutes:

$$\begin{array}{lccccc} G: & \widehat{Y^{\mathrm{in}}} \times S_G & \xrightarrow{g^{\mathrm{upd}}} & S_G & \xrightarrow{g^{\mathrm{rdt}}} & \widehat{Y^{\mathrm{out}}} \\ & \Big\downarrow{\scriptstyle \mathrm{id}_{\widehat{X^{\mathrm{in}}}} \times \alpha} & & \Big\downarrow{\scriptstyle \alpha} & & \Big\updownarrow{\scriptstyle =} \\ \mathrm{DS}\left(\varphi\right)\left(F\right): & \widehat{Y^{\mathrm{in}}} \times S_G \times \widehat{Y^{\mathrm{out}}} & \xrightarrow[h^{\mathrm{upd}}]{} & S_G \times \widehat{Y^{\mathrm{out}}} & \xrightarrow[h^{\mathrm{rdt}}]{} & \widehat{Y^{\mathrm{out}}} \end{array}$$

where $\alpha = \left(\mathrm{id}_{S_G}, g^{\mathrm{rdt}}\right)$. This yields $G \vdash \mathrm{DS}\left(\varphi\right)\left(F\right)$ and hence $\mathrm{DS}\left(\varphi\right)\left(F\right) \sim G$, which concludes the proof. □

Corollary 3.4.8

$\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}\right) \approx \mathrm{DS}$.

Corollary 3.4.9

For all $G \in \mathrm{DS}\left(Y\right)$, if G has finite state set, then there exists $H \in \mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}\right)\left(Y\right)$ with finite state set such that $H \sim G$.

Proof. In the proof of Theorem 3.4.7, take $H = \mathrm{DS}(\varphi)(F)$, but instead of $S_F = S_G \times \widehat{Y^{\mathrm{out}}}$, take $S_F = S_G \times g^{\mathrm{rdt}}(S_G) \subseteq S_G \times \widehat{Y^{\mathrm{out}}}$. If S_G is finite, so is S_F.

In that case, H is no more in $\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}_{\mathrm{out}}\right)(Y)$ but in $\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}\right)(Y)$. □

Corollary 3.4.10

(Assuming the axiom of choice) For all $G \in \mathrm{DS}(Y)$, if G has an infinite state set, then there exists $H \in \mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}\right)(Y)$ with a state set of the same cardinality as G, such that $H \sim G$.

Proof. In the proof of Theorem 3.4.7, take $H = \mathrm{DS}(\varphi)(F)$, but instead of $S_F = S_G \times \widehat{Y^{\mathrm{out}}}$, take $S_F = S_G \times g^{\mathrm{rdt}}(S_G) \subseteq S_G \times \widehat{Y^{\mathrm{out}}}$. The axiom of choice gives $\mathrm{card}(S_F) = \mathrm{card}(S_G) \times \mathrm{card}(g^{\mathrm{rdt}}(S_G)) = \mathrm{card}(S_G)$.

In that case, H is no more in $\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}_{\mathrm{out}}\right)(Y)$ but in $\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}\right)(Y)$. □

Theorem 3.4.7 states that systems without memory can be wired together to form systems with memory. In fact, the result is more subtle. It states that for any discrete system, we can find (or build) a memoryless discrete system with the certain wiring such that both systems are equivalent as stream transducers. The internal equivalence relation described in Theorem 3.3.20 is instrumental to prove Theorem 3.4.7, while the result is stated with regard to the external equivalence relation.

3.4.3 Finite-state systems

The second result is a refinement of Theorem 3.4.7, and is somewhat similar to it. We show that wiring together two-state discrete systems can generate a finite-state system with memory.

We can view the result as the generalisation of transistors being wired together in order to build a computer, or a system of neurons wired together to form a brain with finite memory.

Definition 3.4.11 (Finite-state systems)

Let $X = (X^{\mathrm{in}}, X^{\mathrm{out}})$ be a box.
A *finite-state discrete system for the box* X, or *finite-state system* for short, is a discrete system $F = (S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}) \in \mathrm{DS}(X)$ such that S_F is a finite set.
We denote by $\mathrm{DS}_{\mathrm{Fin}}(X)$ the set of all finite-state discrete systems for the box

$X : \mathrm{DS}_{\mathrm{Fin}}(X) = \{(S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}) \in \mathrm{DS}(X) \mid \mathrm{card}(S_F) \in \mathbb{N}\}$. For a wiring diagram ϕ, we set $\mathrm{DS}_{\mathrm{Fin}}(\varphi) = \mathrm{DS}(\varphi)$.

It is easy to see that:

Proposition 3.4.12

The map $\mathrm{DS}_{\mathrm{Fin}} : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ is a subalgebra of DS.

Proof. Follows from Definition 3.4.2. □

Definition 3.4.13 (Boolean systems)

Let $X = (X^{\mathrm{in}}, X^{\mathrm{out}})$ be a box.
A *boolean memoryless discrete system for the box* X, or *boolean system* for short, is a discrete system $F = (S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}) \in \mathrm{DS}_{\mathrm{Fin}}(X)$ such that F is memoryless and $S_F = 2^n = \{0,1\}^n$.
We denote by $\mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}}(X)$ the set of all boolean memoryless discrete systems for the box $X : \mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}}(X) = \left\{(S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0}) \in \mathrm{DS}^{\mathrm{ML}}(X) \mid S_F = \{0,1\}^n\right\}$.

Remark 3.4.14. The map $\mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}} : \mathscr{W}_{\mathbf{Sets}} \to \mathbf{Sets}$ is not a functor, for the same reason as in Remark 3.4.6.

Lemma 3.4.15

$\mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}} \approx \mathrm{DS}^{\mathrm{ML}} \cap \mathrm{DS}_{\mathrm{Fin}}$.

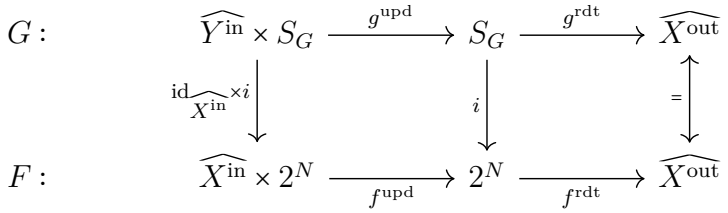
Proof. By construction, $\mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}} \subseteq \mathrm{DS}^{\mathrm{ML}} \cap \mathrm{DS}_{\mathrm{Fin}}$, so $\mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}} \sqsubseteq \mathrm{DS}^{\mathrm{ML}} \cap \mathrm{DS}_{\mathrm{Fin}}$. We need to show the other inclusion, so let $G = (S_G, g^{\mathrm{rdt}}, g^{\mathrm{upd}}, s_{G,0}) \in \mathrm{DS}^{\mathrm{ML}}(X) \cap \mathrm{DS}_{\mathrm{Fin}}(X)$, and it suffices to show that there is $F \in \mathrm{DS}^{\mathrm{ML}}_{\mathrm{Bool}}(X)$ with $G \sim F$.

We have $g^{\mathrm{upd}} = g^u \circ \pi_{\widehat{X^{\mathrm{in}}}}$ and S_G finite. Let $N = \lceil \log_2(\mathrm{card}(S_G)) \rceil$. There exists an injection $i : S_G \to 2^N$ and a surjection $p : 2^N \to S_G$ such that $p \circ i = \mathrm{id}_{S_G}$. This is just a binary encoding of S_G.

Define $F = (S_F, f^{\mathrm{rdt}}, f^{\mathrm{upd}}, s_{F,0})$ such that:

— $S_F = 2^N$
— $f^{\mathrm{rdt}} = g^{\mathrm{rdt}} \circ p$
— $f^{\mathrm{upd}} = g^{\mathrm{upd}} \circ p$
— $s_{F,0} = i(s_{G,0})$

Then the following diagram commutes:



We have $F \in \text{DS}^{\text{ML}}_{\text{Bool}}(X)$ and $G \vdash F$ (with i as simulation function), so $F \sim G$, hence the result. □

Lemma 3.4.16

$\text{DS}_{\text{Fin}} \approx \text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}}$.

Proof. Observe that $\text{DS}_{\text{Fin}} = \text{DS} \cap \text{DS}_{\text{Fin}}$. By Corollary 3.4.8, we have:

$$\text{DS} \approx \text{Clos}\left(\text{DS}^{\text{ML}}\right)$$

In particular, $\text{Clos}\left(\text{DS}^{\text{ML}}\right) \subseteq \text{DS}$, so $\text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}} \subseteq \text{DS} \cap \text{DS}_{\text{Fin}}$. This gives one inclusion, $\text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}} \sqsubseteq \text{DS} \cap \text{DS}_{\text{Fin}}$.

As for the reverse inclusion (up to equivalence), let X be a box and let F be a discrete system in $(\text{DS} \cap \text{DS}_{\text{Fin}})(X)$. By Corollary 3.4.8, there exists $G \in \text{Clos}\left(\text{DS}^{\text{ML}}\right)$ such that $G \sim F$. By Corollary 3.4.9, we can choose G so that $G \in \text{DS}_{\text{Fin}}(X)$, hence the result. □

Theorem 3.4.17

$\text{Clos}\left(\text{DS}^{\text{ML}}_{\text{Bool}}\right) \approx \text{DS}_{\text{Fin}}$.

Proof. Clearly, $\text{Clos}\left(\text{DS}^{\text{ML}}_{\text{Bool}}\right) \sqsubseteq \text{DS}_{\text{Fin}}$. By Lemma 3.4.15, in order to prove the reverse inclusion, it suffices to prove $\text{DS}_{\text{Fin}} \sqsubseteq \text{Clos}\left(\text{DS}^{\text{ML}} \cap \text{DS}_{\text{Fin}}\right)$.

Furthermore, since $\text{DS}_{\text{Fin}} \approx \text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}}$ (Lemma 3.4.16), we reduce to proving that: $\text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}} \sqsubseteq \text{Clos}\left(\text{DS}^{\text{ML}} \cap \text{DS}_{\text{Fin}}\right)$.

Let $Y \in \mathscr{W}_{\mathbf{Sets}}$ be a box, and let $G \in \left(\text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}}\right)(Y)$. We have $G \in \text{Clos}\left(\text{DS}^{\text{ML}}\right)(Y)$, so according to Corollary 3.4.8, there exist a box $X \in \mathscr{W}_{\mathbf{Sets}}$, a wiring diagram $\varphi : X \to Y$, and a $F \in \text{DS}^{\text{ML}}(X)$ such that $\text{DS}(\varphi)(F) \sim G$. Furthermore, according to Corollary 3.4.9, we can choose F with finite state set. Finally, $F \in \left(\text{DS}^{\text{ML}} \cap \text{DS}_{\text{Fin}}\right)(X)$, and $\text{DS}(\varphi)(F) \sim G$, so $\text{Clos}\left(\text{DS}^{\text{ML}}\right) \cap \text{DS}_{\text{Fin}} \sqsubseteq$

$\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}} \cap \mathrm{DS}_{\mathrm{Fin}}\right)$, hence the result.

□

3.5 Conclusion

Boxes are empty frames that condition the inputs and outputs of their content, a generalisation of automata called discrete systems. Such systems come with a state set that represents their memory of previous inputs. In a sense, discrete systems can learn. However, we can define a subclass of discrete systems that do not store any experience of their past. We see these as reactive, in the sense that they still react to any input, but their past experience does not influence that reaction. Unlike typical discrete systems, they do not keep a memory of the previous inputs.

In this chapter, we use a category-theoretic framework to give a constructive proof that any discrete system with memory can be simulated by some correctly-wired memoryless system. This result can be understood as a phenomenon of emergence in a complex system.

This construction opens a number of new questions. A possible question might consist in finding the "best" memoryless system, where "best" could depend on the definition of some valuation function, e.g. the most parsimonious in terms of state set. A similar question could be asked with respect to wiring diagrams, whose number of feedback loops could be bounded by a cost function.

Possible extensions of this work could concern dynamical systems other than DS. For instance, can we establish the same kind of results when considering measurable or continuous dynamical systems?

Part III

Clusters and multiplicity

Chapter 4

Mais dis-moi Jamy, c'est quoi un cluster ?

Eh bien Fred, c'est très simple...

Jamy, dans à peu près tous les épisodes de *C'est Pas Sorcier*.

4.1 Introduction

In Ehresmann and Vanbremeersch's theory [12], a system, be it biological, neurocognitive, social or mechanical, is described as an Evolutive System, that is, a family of categories indexed by time, related by partial functors. The configuration of the system at time t is represented by a category. A component of the system at t corresponds to an object of this category, and the arrows from a given object to another are seen as its interactions. A subsystem at t is then seen as a diagram in the configuration at t, and the interactions between subsystems are transcribed as a set of arrows called clusters. All this framework sets a language whose goal is to describe and predict possible properties of the whole system, e.g. its resilience.

The book, written by a mathematician and a physician, is intended for multidisciplinary audience, resulting sometimes in somehow informal definitions that may leave a mathematician in doubt, but generally followed by their strict mathematical formulation.

Definition 4.1.1 (Cluster [12, Chapter 3, Section 2.1, page 81])

Given two patterns P and Q in a category, a cluster from Q to P is a maximal set G of links between components of these patterns satisfying the following conditions:

1. For each index k of Q, the component Q_k of Q has at least one link to a component of P; and if there are several such links, they are correlated by a zigzag of distinguished links of P.

2. The composite of a link of the cluster with a distinguished link of P, or of a distinguished link of Q with a link of the cluster, also belongs to the cluster.

This definition is a rewording of the first mention of clusters in [16, Appendix, page 47]:

Definition 4.1.2 (Cluster [16, Appendix, page 47])

We consider a category $\mathscr{K}$, and diagrams $A : SA \to \mathscr{K}$ and $B : SB \to \mathscr{K}$
A cluster from B to A consists of links in $\mathscr{K}$ between (indexed) components of B and of A satisfying conditions (S1), (S2), (S3).

(S1) For each vertex u of B: there exists at least one link h in the cluster from B, to a component Ai of A; and if $h*$ is another one (to Aj), there exists a zig-zag of commutative triangles in $\mathscr{K}$, from h to $h*$, whose bases are the images of a zigzag from i to j in the [domain] SA.

(S2) If b is an arrow from v to u in SB and h a link from B, to Ai in the cluster, then the link $B(b) \circ h$ from By to Ai is in the cluster.

(S3) Maximality: it is not possible to add more links to the cluster so that (S1) and (S2) still be satisfied.

[In fact, if a family of links satisfies (S1) and (S2), we may construct a unique cluster which contains it.]

In [12, Chapter 3, Section 2.2, page 83], the authors define a category, which they call $\mathbf{Ind}(\mathscr{C})$, whose objects are all diagrams to $\mathscr{C}$, and whose arrows are the clusters defined above. They explain that, though called $\mathbf{Ind}(\mathscr{C})$, this category extends the standard definition of ind-categories. In fact, most of the litterature focuses on a restriction to a certain kind of diagrams, called *filtered*, which we will define later.

This chapter aims at bringing clarification and formalism to the notion of cluster. We first need to introduce the notions of connected components in categories (Section 4.2). In Section 4.3, we go back to the basics and introduce the ind-category, as introduced by Grothendieck. We give a natural isomorphism between arrows of an ind-category and what we believe are the formal clusters described in Definition 4.1.1. Section 4.4 explicits the formal clusters as sets of arrows verifying certain conditions, correcting the definition given in [20]. We give two new definitions with different sets of conditions, and we prove their equivalence. From here, the next two sections are independent. In Section 4.5, we convert a definition of arrows of pro-categories, adapt it to ind-categories, and find a natural isomorphism between the result and clusters. Finally, in Section 4.6, we give two new characterisations of

clusters: first, in terms of functions between connected components of certain categories, and secondly, in terms of a certain type of functor. We then conclude with an overview and a diagram summarizing the main isomorphisms.

Remark 4.1.3. Let us give a preliminary remark. The work "cluster" is often used in artificial intelligence, more precisely in clustering of data. The original term, in French, is *gerbe*, which translates to *sheaf*, which already has a widely-spread meaning in category theory. The proposed alternative translation is that of *cluster*. However, these clusters have nothing to do with the clusters from artificial intelligence!

The French term of *gerbe* is also the reason why the usual symbol for clusters is a G and not a C.

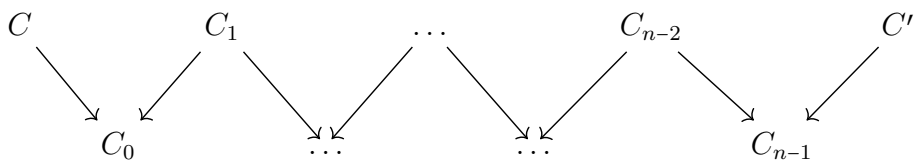
4.2 Detour by connected components

In the following, we introduce some background required for this section and Section 4.6. We will often refer to connected components of comma-categories. We introduce here connected components, and comma-categories.

Definition 4.2.1 (Zigzag, connected components)

Let $\mathscr{C}$ be a category.
We define the *zigzag equivalence relation on* $\mathscr{C}$ as the equivalence relation on objects of $\mathscr{C}$ generated by: $C \sim C'$ iff there is an arrow $C \to C'$.
A *connected component of* $\mathscr{C}$ is an equivalence class of objects of $\mathscr{C}$ under the zigzag equivalence relation.

The name "zigzag" is not random. In fact, the equivalence relation generated by some other relation often takes the form of a zigzag, and it is the case here for $C \to C'$. Visually, C and C' are equivalent under zigzag whenever there is a zigzag of arrows between them:



We sketch the proof of this statement. First, if $C \to C'$, then $C \sim C'$. As an equivalence relation is symmetric, we also have $C' \sim C$. If $C \to C_0$ and $C' \to C_0$, then $C \sim C_0 \sim C'$, because an equivalence relation is transitive. If there is a zigzag

$C \to C_0 \leftarrow C_1 \to \cdots \leftarrow C_{n-2} \to C_{n-1} \leftarrow C'$, then by transitivity, $C \sim C'$. We thus have zigzag $\subset\sim$. Then, $\sim$ is the smallest equivalence relation containing $C \to C'$, and zigzag is an equivalence relation containing $C \to C'$, so $\sim\subset$ zigzag, and $\sim=$ zigzag.

We will also refer to the comma-category. A comma-category is a common construction often used in category theory.

Definition 4.2.2 (Comma-category)

Let $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$ be two functors to $\mathscr{C}$.
The *comma-category* $(P \mid Q)$ is the following category:

Objects: Objects are triples (p, g, q) such that $p \in \mathscr{P}$, $q \in \mathscr{Q}$ and $g : P(p) \to Q(q) \in \mathscr{C}$

Morphisms: Morphisms $(p, g, q) \to (p', g', q')$ are pairs (a, b), where $a : p \to p' \in \mathscr{P}$ and $b : q \to q' \in \mathscr{Q}$, and the following square commutes:

$$\begin{array}{ccc} P(p) & \xrightarrow{P(a)} & P(p') \\ {\scriptstyle g}\downarrow & \checkmark & \downarrow{\scriptstyle g'} \\ Q(q) & \xrightarrow[Q(b)]{} & Q(q') \end{array}$$

Identities: Identities are pairs $(\mathrm{id}_p, \mathrm{id}_q)$

Composition: The composition law is pairwise: $(a', b') \circ (a, b) = (a' \circ a, b' \circ b)$

Comma-categories are the placeholder for sets of "correlated" arrows, in the sense that those arrows are linked by commuting squares.

For the sake of conciseness, the objects (p, g, q) of $(P \mid Q)$ will be denoted by their arrow $g : P(p) \to Q(q)$, where p and q are implicitly given in the description of g.

In the present chapter, we will often consider the connected components of a comma-category $(P \mid Q)$. A zigzag of arrows between two objects $P(p) \to Q(q)$ and

$P(p') \to Q(q')$ takes the following form:

$$\begin{array}{ccccccccccccc}
P(p) & \longrightarrow & P(p_1) & \longleftarrow & P(p_2) & \longrightarrow & \dots & \longleftarrow & P(p_{n-2}) & \longrightarrow & P(p_{n-1}) & \longleftarrow & P(p') \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow & & \downarrow \\
Q(q) & \longrightarrow & Q(q_1) & \longleftarrow & Q(q_2) & \longrightarrow & \dots & \longleftarrow & Q(q_{n-2}) & \longrightarrow & Q(q_{n-1}) & \longleftarrow & Q(q')
\end{array}$$

where each square commutes.

In the following, the most useful restriction of $(P \mid Q)$ is $(P(p) \mid Q)$, which we describe as:

Objects: Objects are pairs (g, q) such that $q \in \mathscr{Q}$ and $g : P(p) \to Q(q) \in \mathscr{C}$

Morphisms: Morphisms $(g, q) \to (g', q')$ are arrows $b : q \to q' \in \mathscr{Q}$, such that the following triangle commutes:

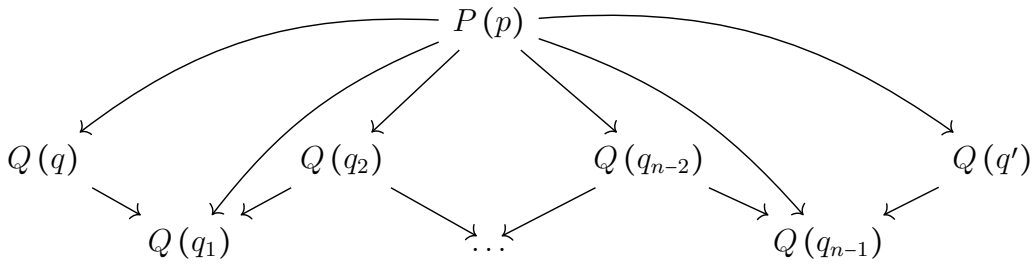
$$\begin{array}{ccc}
 & P(p) & \\
{\scriptstyle g}\swarrow & \checkmark & \searrow{\scriptstyle g'} \\
Q(q) & \xrightarrow[Q(b)]{} & Q(q')
\end{array}$$

Identities: Identities are the identities of $\mathscr{Q}$: id_q

Composition: The composition law is that of $\mathscr{Q}$

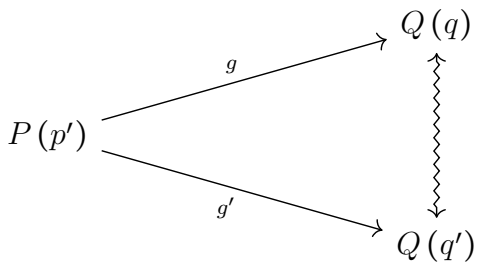
In short, $(P(p) \mid Q)$ is a shortcut for $(\Delta(P(p)) \mid Q)$. It is also the subcategory of $(P \mid Q)$, with only the objects (p, g, q) and pairs of arrows of the form (id_p, b).

A zigzag of arrows in $(P(p) \mid Q)$ takes the following form:



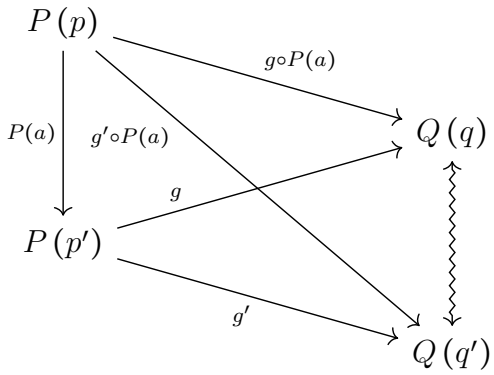
where each triangle commutes.

An important remark that we will use in Sections 4.4 and 4.6. Consider two arrows $g : P(p') \to Q(q)$ and $g' : P(p') \to Q(q)$ in the same connected component of $(P(p') \mid Q)$. For the sake of readability, we do not draw the entire zigzag:



Here we assume that we have a zigzag of commuting triangles, just like above.

The crucial remark is the following. Consider the arrow $a : p \to p' \in \mathscr{P}$. Observe the following diagram:

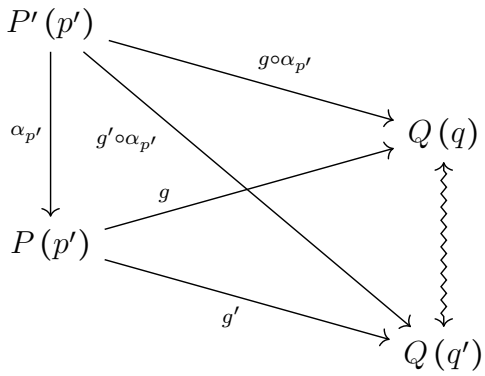


We observe that the existing zigzag between g and g' in $(P(p') \mid Q)$ also exists between $g \circ P(a)$ and $g' \circ P(a)$. We deduce that if $g \sim g'$ in $(P(p') \mid Q)$, then $g \circ P(a) \sim g' \circ P(a)$ in $(P(p) \mid Q)$.

In a sense, we "compose" the connected component of g and g' to obtain (part of) the connected component of $g \circ P(a)$ and $g' \circ P(a)$.

In the same vein, if we have a natural transformation $\alpha : P' \to P$ and $g \sim g'$ in

$(P(p') \mid Q)$, then $g \circ \alpha_{p'} \sim g' \circ \alpha_{p'}$ in $(P'(p') \mid Q)$:



From these observations, we derive the following two lemmas, where $[g]_p$ is the connected component of $g : P(p) \to Q(q)$ in $(P(p) \mid Q(q))$

Lemma 4.2.3 (Composition does not alter connected components)

Let $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$ be two diagrams. Let $g, g' \in (P(p) \mid Q)$ such that $[g]_p = [g']_p$.

1. If $a : p' \to p \in \mathscr{P}$, then $[g \circ P(a)]_{p'} = [g' \circ P(a)]_{p'}$.
2. If $\alpha : P' \to P$ is a natural transformation, then $[g \circ \alpha_p]_p = [g' \circ \alpha_p]_p$, where we consider the equivalence classes in $(P'(p) \mid Q)$.

Lemma 4.2.4 (Composition does not alter connected components (dual))

Let $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$ be two diagrams. Let $g : P(p) \to Q(q), g' : P(p) \to Q(q') \in (P(p) \mid Q)$ such that $[g]_p = [g']_p$.

1. If $b : q \to q'' \in \mathscr{Q}$, then $[g]_p = [Q(b) \circ g]_p$.
2. If $\beta : Q \to Q'$ is a natural transformation, then $[\beta_q \circ g]_p = [\beta_{q'} \circ g]_{p'}$.

4.3 Grothendieck's legacy

In this section, we derive the definition of an arrow in the ind-category in order to find a formal definition of a cluster. Ind-objects are first introduced in [35, Exposé 1, section 8.2, page 67]. We give here a brief overview of this notion and see how it relates to our clusters.

4.3.1 Introduction to ind-categories

First, ind-categories have a certain kind of diagrams as objects. Here, we remind Definition 2.4.7 and add some more terminology.

Definition 4.3.1 (Filtered category, filtered diagram)

A *filtered category* $\mathscr{P}$ is a category such that every finite diagram $\mathscr{D} \to \mathscr{P}$ has at least one cocone in $\mathscr{P}$. A *filtered diagram* is a diagram $\mathscr{P} \to \mathscr{C}$ whose domain category $\mathscr{P}$ is filtered.

Ind-categories were introduced in [35, Exposé 1, section 8.2, page 67], using the notion of filtered diagrams.

To each filtered diagram $P : \mathscr{P} \to \mathscr{C}$, we associate the following presheaf [35, Exposé 1, section 8.2, page 68, equations (8.2.2.1) and (8.2.2.2)]:

$$L(P) : \left\{ \begin{array}{ccc} \mathscr{C}^{\mathrm{op}} & \longrightarrow & \mathbf{Sets} \\ X & \longmapsto & \underset{p \in \mathscr{P}}{\mathrm{Colim}}\ \mathrm{Hom}_{\mathscr{C}}(X, P(p)) \end{array} \right.$$

This is a presheaf $\mathscr{C}^{\mathrm{op}} \to \mathbf{Sets}$, so it is a functor, so there may be natural transformations $\alpha : L(P) \to L(Q)$. We then define the ind-category using this fact:

Definition 4.3.2 (Ind-category)

Let $\mathscr{C}$ be a small category.
The ind-category of $\mathscr{C}$, denoted by $\mathbf{Ind}(\mathscr{C})$, is the following category:

Objects: Objects are filtered diagrams $P : \mathscr{P} \to \mathscr{C}$

Morphisms: An arrow $P \to Q$ is a natural transformation $L(P) \to L(Q)$

Identities: The identities are the identities of presheaves.

Composition: The composition of arrows is the composition of natural transformations

Remark 4.3.3. The choice of filtered categories may sound restrictive. In fact, nothing from the definition of ind-categories actually requires those categories to be filtered.

The first point to notice is that it is *not* a restriction, but rather, a generalisation. In fact, Grothendieck remarks in [35, Exposé 1, section 8.1, paragraph 8.1.5] that the initial use of preorders is more a constraint than anything:

> Classiquement, on se bornait même à des catégories associées à des ensembles préordonnés i.e. dans lesquelles il existe au plus une flèche de

> source et de but donnés). II apparaît cependant que cette restriction est gênante dans les applications, les catégories filtrantes "naturelles" qui s'introduisent dans de nombreuses applications n'étant pas des catégories ordonnées. ([35, Exposé 1, section 8.1, paragraph 8.1.5])

Besides, in [35, Exposé 1, section 8.3, théorème 8.3.3], a functor of the form $L(P)$ can be characterised in terms of a specific filtered comma-category. Finally, accessible categories rely on the notion of filtered categories, and are of the form $\mathbf{Ind}(\mathscr{C})$ for some $\mathscr{C}$ [36].

As said in the Introduction, in [12, Chapter 3, Section 2.2, page 83], the authors never mentioned filtered diagrams. In fact, they call $\mathbf{Ind}(\mathscr{C})$ the category of clusters which contains the clusters between all filtered and non-filtered diagrams. This idea comes from [37, Section 0, definition 0.1, page 475], where the authors study a generalised version of the dual of the ind-category, called pro-category. Also, it was proven in [38, Proposition, paragraph 199.1, comments on /102/, page 371] that such a pro-category is the free completion of $\mathscr{C}$ (and dually, the corresponding ind-category is the free cocompletion of $\mathscr{C}$). See also Section 4.5 for more discussion on pro-categories and clusters.

Let us calculate the equivalents of the arrows of $\mathbf{Ind}(\mathscr{C})$.

$$\mathrm{Hom}_{\mathbf{Ind}(\mathscr{C})}(P,Q) = \mathrm{Hom}_{\mathbf{PSh}(\mathscr{C})}(L(P),L(Q)) \tag{4.1}$$

$$= \mathrm{Hom}_{\mathbf{PSh}(\mathscr{C})}\left(\operatorname*{Colim}_{p\in\mathscr{P}} \mathrm{Hom}_{\mathscr{C}}(-,P(p)), \operatorname*{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(-,Q(q))\right) \tag{4.2}$$

$$\cong \operatorname*{Lim}_{p\in\mathscr{P}} \mathrm{Hom}_{\mathbf{PSh}(\mathscr{C})}\left(\mathrm{Hom}_{\mathscr{C}}(-,P(p)), \operatorname*{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(-,Q(q))\right) \tag{4.3}$$

$$\cong \operatorname*{Lim}_{p\in\mathscr{P}} \operatorname*{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q)) \tag{4.4}$$

Equations 4.1 and 4.2 follow directly from the definition; Equation 4.3 is due to the cocontinuity of the Hom-set functor in its first variable; Equation 4.4 is simply the Yoneda Lemma [22, Lemma 8.2, section 8.3, pages 188-189] applied to the functor $X \mapsto \operatorname*{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(X,Q(q))$.

The last equality is often used as the definition of arrows in $\mathbf{Ind}(\mathscr{C})$ (for example in [39, Definition as diagrams, section 2]).

Lemma 4.3.4 (Cluster - Arrows in Ind $(\mathscr{C})$)

Define $\mathrm{LC}_{\mathscr{C}}(P,Q) = \lim_{p\in\mathscr{P}} \operatorname{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q))$ for the sake of conciseness.
An arrow $G : P \to Q$ is equivalent to an element of $\mathrm{LC}_{\mathscr{C}}(P,Q)$.

4.3.2 Explicit computation

Our goal is to relate clusters with arrows of the ind-category. Let us see if we can retrieve a set of arrows from the formula given in Equation 4.4. For this purpose, we use the explicit formula from Theorem 2.4.42. The explicit formula is obtained by routine algebra, but we give here the explicit computation.

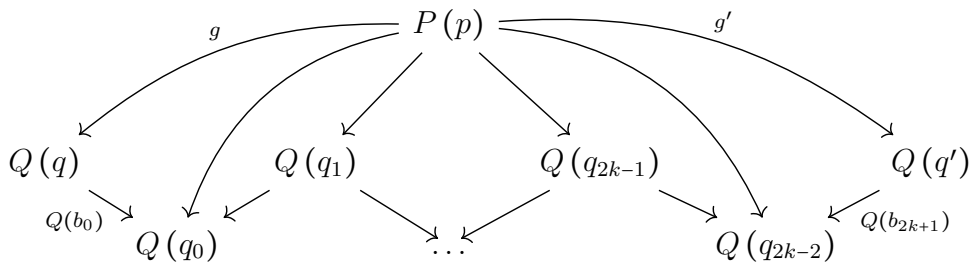
We first consider the functor:

$$M : p \mapsto \operatorname*{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q))$$

Following Theorem 2.4.42, we consider the relation on $\sum_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q))$ defined by: $g \sim g'$ iff there exists a $b \in \mathscr{Q}$ such that $g' = Q(b)\circ g$. This relation generates the "zigzag" equivalence relation: $g \sim g'$ iff there exists a sequence $(b_i)_{i\in 2k}$ and a sequence $(g_i : P(p) \to Q(b_i))_{i\in 2k}$ such that:

$$\begin{aligned}
g_0 &= Q(b_0)\circ g\\
g_0 &= Q(b_1)\circ g_1\\
g_2 &= Q(b_2)\circ g_3\\
&\dots\\
g_{2i-2} &= Q(b_{2i-1})\circ g_{2i-1}\\
g_{2i} &= Q(b_{2i})\circ g_{2i+1}\\
g_{2i} &= Q(b_{2i+1})\circ g_{2i+1}\\
g_{2i+2} &= Q(b_{2i+2})\circ g_{2i+3}\\
&\dots\\
g_{2k-2} &= Q(b_{2k-2})\circ g_{2k-2}\\
g_{2k-2} &= Q(b_{2k-1})\circ g'
\end{aligned}$$

Diagrammatically, this is represented:



For the sake of readability, we do not label all the arrows. Also note that formally, the elements of the coproduct $\sum_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$ are pairs $(q, g : P(p) \to Q(q))$, that is, a labelled arrow.

In short, M is the functor:

$$M : \begin{cases} \mathscr{P}^{\mathrm{op}} & \longrightarrow & \mathbf{Sets} \\ p & \longmapsto & \sum_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q)) \,/\sim \\ a : p \to p' & \longmapsto & M(a) : [g] \mapsto [g \circ P(a)] \end{cases} \tag{4.5}$$

where $\sim$ is the zigzag equivalence relation, that is, the equivalence relation on $\sum_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$ generated by $g \sim g'$ iff $\exists b : q \to q' \in \mathscr{Q}$, $g = Q(b) \circ g'$ (see Section 4.2).

Besides, $M(a)$ takes an equivalence class of $\sum_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$ under zigzag, "extracts" a representative g, compose it with $P(a)$, and returns the equivalence class of $g \circ P(a)$.

Then, we have to compute $\varprojlim_{p \in \mathscr{P}}(M(p)) = \varprojlim_{p \in \mathscr{P}} \varinjlim_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$. By Theorem2.4.42, it is a subset:

$$\varprojlim_{p \in \mathscr{P}}(M(p)) = \mathrm{LC}_{\mathscr{C}}(P, Q) \subset \prod_{p \in \mathscr{P}} M(p)$$

such that, for each sequence $([g_p])_{p \in \mathscr{P}}$, and for all $a : p' \to p$, we have: $M(a)([g_p]) = [g_p \circ P(a)]\,[g_{p'}]$. Thus, each equivalent class $[g_p]$ in the tuple is correlated to the others by $P(a)$.

In summary, we obtain:

$$\begin{aligned}\mathrm{LC}_{\mathscr{C}}(P,Q) &= \lim_{p\in\mathscr{P}} \operatorname*{Colim}_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q)) \\ &= \left\{ ([g_p])_{p\in\mathscr{P}} \in \prod_{p\in\mathscr{P}} M(p) \;\middle|\; \forall a: p'\to p, M(a)([g_p]) = [g_{p'}] \right\} \\ &= \lim_{p\in\mathscr{P}} M(p)\end{aligned} \tag{4.6}$$

where M is defined in Equation 4.5.

The computation gives a functor:

$$\mathrm{LC}_{\mathscr{C}}(-,-) : \left(\mathscr{C}^{\mathscr{P}}\right)^{\mathrm{op}} \times \mathscr{C}^{\mathscr{Q}} \to \mathbf{Sets}$$

If $\alpha : P \to P'$ is a natural transformation between functors $\mathscr{P} \to \mathscr{C}$, then $\mathrm{LC}_{\mathscr{C}}(\alpha, Q)$ is the function:

$$\mathrm{LC}_{\mathscr{C}}(\alpha,Q) : \left\{ \begin{array}{ccc} \mathrm{LC}_{\mathscr{C}}(P',Q) & \longrightarrow & \mathrm{LC}_{\mathscr{C}}(P,Q) \\ ([g_p])_{p\in\mathscr{P}} & \longmapsto & ([g_p\circ\alpha_p])_{p\in\mathscr{P}} \end{array} \right.$$

Similarly, if $\beta : Q \to Q'$ is a natural transformation between functors $\mathscr{Q} \to \mathscr{C}$, then $\mathrm{LC}_{\mathscr{C}}(P,\beta)$ is:

$$\mathrm{LC}_{\mathscr{C}}(P,\beta) : \left\{ \begin{array}{ccc} \mathrm{LC}_{\mathscr{C}}(P,Q) & \longrightarrow & \mathrm{LC}_{\mathscr{C}}(P,Q') \\ ([g_p])_{p\in\mathscr{P}} & \longmapsto & \left([\beta_{\mathrm{cod}\, g_p}\circ g_p]\right)_{p\in\mathscr{P}} \end{array} \right.$$

So, roughly, $\mathrm{LC}_{\mathscr{C}}(P,Q)$ is a set of sets of arrows of $\mathscr{C}$. As the different equivalence classes are disjoint or equal, we can define the following function:

$$U_{P,Q} : (m_p)_{p\in\mathscr{P}} \mapsto G = \biguplus_{p\in\mathscr{P}} m_p \tag{4.7}$$

Definition 4.3.5

We define $\mathrm{Clstr}(P,Q)$ to be the image of $\mathrm{LC}_{\mathscr{C}}(P,Q)$ by $U_{P,Q}$.

The mapping $\mathrm{Clstr}(-,-)$ thus defined is a functor; its actions on arrows $\alpha : P \to P'$ and $\beta : Q \to Q'$ are:

$$\mathrm{Clstr}\,(\alpha, Q) : \begin{cases} \mathrm{Clstr}\,(P', Q) & \longrightarrow & \mathrm{Clstr}\,(P, Q) \\ \biguplus\limits_{p \in \mathscr{P}} [g_p] & \longmapsto & \biguplus\limits_{p \in \mathscr{P}} [g_p \circ \alpha_p] \end{cases}$$

$$\mathrm{Clstr}\,(P, \beta) : \begin{cases} \mathrm{Clstr}\,(P, Q) & \longrightarrow & \mathrm{Clstr}\,(P, Q') \\ \biguplus\limits_{p \in \mathscr{P}} [g_p] & \longmapsto & \biguplus\limits_{p \in \mathscr{P}} \left[\beta_{\mathrm{cod}\, g_p} \circ g_p\right] \end{cases}$$

4.3.3 A natural isomorphism

Now that everything is explicitly defined, we can prove the main theorem of this section. The following proposition is a step forward to making Definition 4.1.1 formal:

Theorem 4.3.6

The family of arrows $U = (U_{P,Q})_{P,Q}$ with components:

$$U_{P,Q} : \begin{cases} \lim\limits_{p \in \mathscr{P}} \operatorname{colim}\limits_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}\,(P(p), Q(q)) & \longrightarrow & \mathrm{Clstr}\,(P, Q) \\ ([g_p])_{p \in \mathscr{P}} & \longmapsto & \biguplus\limits_{p \in \mathscr{P}} [g_p] \end{cases}$$

form a natural isomorphism $U : \lim\limits_{p \in \mathscr{P}} \operatorname{colim}\limits_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}\,(-,-) \to \mathrm{Clstr}\,(-,-)$.

Proof. By construction, $U_{P,Q}$ is definitely surjective.

Let $([g_p])_{p \in \mathscr{P}}$ and $\left([g'_p]\right)_{p \in \mathscr{P}}$ such that $G = \biguplus\limits_{p \in \mathscr{P}} [g_p] = G' = \biguplus\limits_{p \in \mathscr{P}} [g'_p]$. For all $p \in \mathscr{P}$, $G(p) = [g_p]$ is non-empty, so there exists $g : P(p) \to Q(q) \in G$. We have $g \in [g_p]$ and also, $g \in G'$, because $G = G'$. As the union is disjoint, $g \in [g'_p]$. As $[g_p]$ and $[g'_p]$ are equivalence classes, we have $g \in [g_p] \cap [g'_p]$, and then $[g_p] = [g'_p]$. Therefore, $U_{P,Q}$ is bijective.

Let us check its naturality. We check the naturality in P; the naturality in Q is similar but easier due to the covariant-ness.

Let $\alpha : P \to P'$ be a natural transformation. We want the following diagram to commute:

$$\begin{array}{ccc} P & & \mathrm{LC}_{\mathscr{C}}\,(P', Q) \xrightarrow{U_{P',Q}} \mathrm{Clstr}\,(P', Q) \\ \downarrow{\scriptstyle \alpha} & \rightsquigarrow & \downarrow{\scriptstyle \mathrm{LC}_{\mathscr{C}}(\alpha,Q)} \quad ? \quad \downarrow{\scriptstyle \mathrm{Clstr}(\alpha,Q)} \\ P' & & \mathrm{LC}_{\mathscr{C}}\,(P, Q) \xrightarrow[U_{P,Q}]{} \mathrm{Clstr}\,(P, Q) \end{array}$$

Now, take a tuple $\left([g'_p]\right)_{p\in\mathscr{P}} \in \mathrm{LC}_{\mathscr{C}}(P', Q)$; we use

$$\begin{aligned}\mathrm{Clstr}(\alpha, Q) \circ U_{P',Q}\left(\left([g'_p]\right)_{p\in\mathscr{P}}\right) &= \mathrm{Clstr}(\alpha, Q)\left(\biguplus_{p\in\mathscr{P}} [g_p]\right)\\ &= \biguplus_{p\in\mathscr{P}} \left([g_p \circ \alpha_p]\right)\\ &= U_{P,Q}\left(\left([g_p \circ \alpha_p]\right)_{p\in\mathscr{P}}\right)\\ &= U_{P,Q} \circ \mathrm{LC}_{\mathscr{C}}(\alpha, Q)\left(\left([g'_p]\right)_{p\in\mathscr{P}}\right)\end{aligned}$$

So the diagram commutes, hence the naturality of $U_{P,Q}$ in P. The naturality in Q is similar. □

We define clusters as follows:

Definition 4.3.7 (Cluster - formal)

A *cluster* $G : P \to Q$ is an element of $\mathrm{Clstr}(P, Q) = U_{P,Q}(\mathrm{LC}_{\mathscr{C}}(P, Q))$, that is, $G = \biguplus_{p\in\mathscr{P}} [g_p]$ where $([g_p])_{p\in\mathscr{P}} \in \mathrm{LC}_{\mathscr{C}}(P, Q)$.

From here, the reader may follow on the next section, which explores a bit further another definition of arrows based on the dual of ind-categories, or skip to the section after the next one, that gives explicit definitions of clusters in terms of sets of arrows.

4.4 Descriptive definitions of clusters

In this section, we introduce two equivalent descriptions of clusters, seen as sets of arrows verifying certain properties. Some of these properties rely on notions of connected components of comma-categories, which we introduced in Section 4.2.

As stated before, Definition 4.3.7 seems to make our base definition formal. In this section, we exhibit two equivalent definitions of clusters.

First of all, we introduce the following notation.
If G is a subset of $\sum_{p\in\mathscr{P}} \biguplus_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$, we define $G(p)$ as the set:

$$G(p) = G \cap \left(\bigcup_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))\right) \tag{4.8}$$

In other words, $G(p)$ is the subset of G consisting of all the arrows labelled with

p.

We need the following definition:

Definition 4.4.1 (Precluster)

A set

$$G \subset \sum_{p \in \mathscr{P}} \bigcup_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}\left(P(p), Q(q)\right)$$

of arrows is called a *precluster* $G : P \to Q$ if it verifies:

(CLU-1) for all $p \in \mathscr{P}$, $G(p) \neq \varnothing$

(CLU-2) for all $p \in \mathscr{P}$, $G(p)$ is included in exactly one connected component of the comma-category $(P(p) \mid Q)$

(CLU-3) if $g : P(p) \to Q(q) \in G(p)$ and $b : q \to q' \in \mathscr{Q}$, then $Q(b) \circ g \in G(p)$

(CLU-4) if $a : p' \to p \in \mathscr{P}$ and $g : P(p) \to Q(q) \in G(p)$, then $g \circ P(a) \in G(p')$

Using this definition as a base, we introduce a fixed definition of cluster:

Proposition 4.4.2

Let G be a set of arrows $P(p) \to Q(q)$.
G is a cluster in the sense of Definition 4.3.7 $\Leftrightarrow$ G is a precluster that verifies:

(CLU-5) G is maximal for $\subset$ among the preclusters $P \to Q$

Proof. [**Proof of** $\Rightarrow$] Let $G = \biguplus_{p \in \mathscr{P}} m_p \in \mathrm{Clstr}\left(P, Q\right)$. For each $p \in \mathscr{P}$, we have a (non-empty!) equivalence class of arrows $P(p) \to Q(q)$, so G verifies (CLU-1). As for (CLU-2), $G(p)$ is exactly m_p (due to the disjoint union in Proposition 4.3.6), which is an equivalence class under zigzag, so it is exactly a connected component of the comma-category $(P(p) \mid Q)$. (CLU-3) comes from the fact that $G(p) = m_p$ is a connected component, and if $g \in m_p$, then $Q(b) \circ g \in m_p = G(p)$. (CLU-4) translates the fact that $m_p = m_{p'} \circ P(a)$, which is a condition imposed by the "limit" part of Lemma 4.3.4. As a first conclusion, G is a precluster.

Let $G' \supset G$ be another precluster. Let $p \in \mathscr{P}$, let $g' : P(p) \to Q(q) \in G'(p)$ and $g : P(p) \to Q(q) \in G(p) \subset G'(p)$. As G' verifies (CLU-2), there is a zigzag between g and g'. Besides, as $G(p) = m_p$ is an equivalence class, if there is a zigzag between g and g', then g' is in the equivalence class, and thus $G'(p) \subset G(p) \subset G'(p)$, from which we deduce $G(p) = G'(p)$ and $G = G'$. Thus, G is maximal.

[**Proof of** $\Leftarrow$] Let G be a precluster verifying (CLU-5). Let $p \in \mathscr{P}$. Let us prove that $G(p)$ is an equivalence class under zigzag (i.e., that it is an m_p). Let

$g_0 : P(p) \to Q(q)$. Either there is a $g \in G(p)$ such that $g \sim g_0$, or there is none.

If $g_0 \sim g$, then g_0 is in the same connected component of $(P(p) \mid Q)$ as g. Consider G_0 to be the set of arrows that contains G and g_0, and all the arrows of the form $g' \circ P(a)$ and $Q(b) \circ g'$ so that (CLU-3) and (CLU-4) hold. By construction, G_0 also verifies (CLU-1) (because G does and $G_0 \supset G$). We need to check if G_0 verifies (CLU-2).

We let the reader check that if $g_0 \sim g$, then $g_0 \circ P(a) \sim g \circ P(a)$ and $Q(b) \circ g_0 \sim Q(b) \circ g \sim g \sim g_0$. If other words, if $G(p)$ is contained in a unique connected component of $(P(p) \mid Q)$, by construction, $G_0(p)$ also is.

We deduce that G_0 is a precluster. It contains G, which is maximal, thus $G_0 = G$, and $g_0 \in G(p)$.

If there is no $g \in G$ such that $g_0 \sim g$, then $g_0 \notin G(p)$, otherwise (CLU-2) would not hold.

So, we have: for all $g_0 : P(p) \to Q(q)$, $\exists g \in G(p),\ g \sim g_0 \Leftrightarrow g_0 \in G(p)$, so $G(p)$ is an equivalence class under $\sim$ (zigzag). □

The problem with Proposition 4.4.2 is that it refers explicitly to other preclusters. As we wish for a minimal, self-sufficient definition, we introduce the following proposition.

Proposition 4.4.3

Let G be a set of arrows $P(p) \to Q(q)$.
G is a cluster in the sense of Definition 4.3.7 $\Leftrightarrow$ G is a precluster that verifies:

(CLU-2b) $G(p)$ is a connected component of $(P(p) \mid Q)$

So, in fact, (CLU-2b) contains (CLU-2). Instead of being *included in* a connected component of the comma-category $(P(p) \mid Q)$, $G(p)$ *is* a connected component.

Proof. By Proposition 4.4.2, a cluster in the sense of Definition 4.3.7 is equivalently a maximal precluster.

[**Proof of** $\Rightarrow$] Let G be a maximal precluster. G verifies (CLU-1) to (CLU-5). Let $g : P(p) \to Q(q) \in G$, and let $g' : P(p) \to Q(q') \in \mathscr{C}$ such that g and g' are in the same connected component of $(P(p) \mid Q)$. Let G' be the set of arrows containing G, g' and all the missing arrows so that (CLU-3) and (CLU-4) are verified. Then G' is a precluster, containing G, which is maximal. Thus, $G = G'$, and g' is necessarily in G, hence the result.

[**Proof of** $\Leftarrow$] Let G be a precluster that verifies (CLU-2b). Let $G' \supset G$ be a bigger precluster. Suppose that $G \subsetneq G'$, then let $g' : P(p) \to Q(q') \in G' \backslash G$. As G

verifies (CLU-1), there is a $g : P(p) \to Q(q)$ with the same domain as g'. Also, by (CLU-2b), $G(p)$ is a connected component of $(P(p) \mid Q)$. However, $g' \notin G$, so $g' \in G'(p) \backslash G(p)$. This means that g' and g are not in the same connected component of $(P(p) \mid Q)$, and $G'(p)$ is inside at least two connected components of $(P(p) \mid Q)$. Therefore, G' does not verify (CLU-2), which contradicts the assumption that G' is a precluster. G' cannot strictly contain G, which makes G maximal. □

We can give a concise version of the definition of cluster.

Proposition 4.4.4

Let G be a subset of $\sum_{p \in \mathscr{P}} \bigcup_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$.
G is a cluster $\Leftrightarrow$ G verifies (CLU-2b) and (CLU-4).

Proof. The sense $\Rightarrow$ follows from Proposition 4.4.3.

The converse derives from the fact that (CLU-2b) implies (CLU-1) (as a connected component is necessarily non-empty), (CLU-2), (CLU-3) and (CLU-5). □

Although simpler, this definition will not be used in the next chapter, because it does not split the problem enough.

However, it is then obvious that this definition is a rewording of Definition 4.1.2 from [16, Appendix, page 47]. We have then retrieved the original definition.

4.5 Pro-categories and ind-categories

This section is not essential for the rest of this manuscript. The reader may skip it with no loss of understanding of the whole approach.

We mentioned the pro-category as a dual of the ind-category in Section 4.3. Let us give brief overview of how those two relate.

4.5.1 Pro-categories as dual to ind-categories

In [37, Section 0, definition 0.1, page 475], the pro-category of a category $\mathscr{C}$ is the following category:

Definition 4.5.1 (Pro-category)

Let $\mathscr{C}$ be a small category.
The pro-category of $\mathscr{C}$, denoted by $\mathbf{Pro}(\mathscr{C})$, is the category whose objects are diagrams $P : \mathscr{P} \to \mathscr{C}$ and whose hom-sets are defined as:

$$\mathrm{Hom}_{\mathbf{Pro}(\mathscr{C})}(P, Q) = \operatorname*{Colim}_{q \in \mathscr{Q}} \operatorname*{Lim}_{p \in \mathscr{P}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$$

The definition of a pro-category looks like the definition of an ind-category. In fact, both are linked.

Theorem 4.5.2 ([35, Equation 8.10.2, section 8.10, page 119])

$\mathbf{Pro}(\mathscr{C}) \cong (\mathbf{Ind}(\mathscr{C}^{\mathrm{op}}))^{\mathrm{op}}$.

Equivalently, $\mathbf{Ind}(\mathscr{C}) \cong (\mathbf{Pro}(\mathscr{C}^{\mathrm{op}}))^{\mathrm{op}}$.

Remark 4.5.3. In [40, Remark at the end of section 2], which cites [41, Proposition 6.1.12, chapter 6, page 134], $\mathbf{Pro}(\mathscr{C})$ and $(\mathbf{Ind}(\mathscr{C}^{\mathrm{op}}))^{\mathrm{op}}$ are only said to be equivalent; this is because , as stated in [42, very first line]:

> The concept of equivalence of categories is the correct category theoretic notion of "sameness" of categories. (nLab authors, [42])

So, for most usages, being equivalent is strong enough. However, we prefer here the notion of being isomorphic.

Note that Definition 4.5.1 doesn't require the notion of cofiltered diagrams. In fact, in [37, Section 0, definition 0.1, page 475], the pro-category is defined for general diagrams, just as in [12, Chapter 3, Section 2.2, page 83] for the ind-category. As said in Section 4.3, it was stated in [38, Note 2, section 199.1, page 369] that the pro-category of $\mathscr{C}$ is the free completion of $\mathscr{C}$; thus, from Theorem 4.5.2, we deduce that the ind-category is its free cocompletion. This fact is also obtained as a consequence of [12, Part (ii) of Theorem 1, page 83].

In [37, Equations (0.2), (0.3), (0.$\bar{4}$), (0.5), (0.6) and (0.7), pages 473-475] or [43, section 1, pages 10-12], the authors give an explicit description of the arrows of the pro-category. In this section, using Theorem 4.5.2, we will use this description to describe the arrows of the ind-category, and thus, make parallel with clusters. We relate here the description in a non-verbatim way.

Description of the arrows of $\mathbf{Pro}(\mathscr{C})$. Let $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$ be diagrams. We define $\mathrm{PA}(Q, P)$ ("partial arrows", for reasons that will be obvious in

the following) as the subset of the set:

$$\begin{aligned}&\left\{\left(\varphi,\left(f_p\right)_{p\in\mathscr{P}}\right)\;\middle|\;\varphi:\mathrm{Ob}\left(\mathscr{P}\right)\to\mathrm{Ob}\left(\mathscr{Q}\right),\ \forall p\in\mathscr{P}, f_p:Q\left(\varphi(p)\right)\to P(p)\in\mathscr{C}\right\}\\ &\subsetneq \mathrm{Ob}\left(\mathscr{Q}\right)^{\mathrm{Ob}(\mathscr{P})}\times\prod_{p\in\mathscr{P}}\sum_{q\in\mathscr{Q}}\mathrm{Hom}_{\mathscr{C}}\left(Q(q),P(p)\right)\end{aligned}$$

such that each element $\left(\varphi,\left(f_p\right)_{p\in\mathscr{P}}\right)\in\mathrm{PA}\left(Q,P\right)$ verifies the following condition: for all finite set of arrows $\left(a_i:p\to p_i\right)_{i\in n}\in\mathscr{P}$, there exists $q\in\mathscr{Q}$, $b:q\to\varphi(p)\in\mathscr{Q}$ and $\left(b_i:q\to\varphi\left(p_i\right)\right)_{i\in n}\in\mathscr{Q}$, such that for all $i\in n$, the following diagram commutes:

$$\begin{array}{ccccc} & & Q\left(\varphi\left(p\right)\right) & \xrightarrow{f_p} & P\left(p\right) \\ & \overset{Q(b)}{\nearrow} & & & \\ Q\left(q\right) & & \checkmark & & \Big\downarrow{\scriptstyle P(a_i)} \\ & \underset{Q(b_i)}{\searrow} & & & \\ & & Q\left(\varphi\left(p_i\right)\right) & \xrightarrow[f_{p_i}]{} & P\left(p_i\right) \end{array} \qquad (\text{☺})$$

A morphism $f:Q\to P$ in $\mathbf{Pro}\left(\mathscr{C}\right)$ is an equivalence class on $\mathrm{PA}\left(Q,P\right)$ under the equivalence relation $\simeq$ generated by: $\left(\varphi,\left(f_p\right)_{p\in\mathscr{P}}\right)\sim\left(\varphi',\left(f'_p\right)_{p\in\mathscr{P}}\right)$ iff for all $p\in\mathscr{P}$, there exist $b:q\to\varphi(p)$ and $b':q\to\varphi'(p)$ in $\mathscr{Q}$ such that the following diagram commutes:

$$\begin{array}{ccccc} & & Q(\varphi(p)) & & \\ & \overset{Q(b)}{\nearrow} & & \searrow^{f_p} & \\ Q(q) & & \checkmark & & P(p) \\ & \underset{Q(b')}{\searrow} & & \nearrow_{f'_p} & \\ & & Q(\varphi'(p)) & & \end{array}$$

Then, $\mathrm{Hom}_{\mathbf{Pro}(\mathscr{C})}\left(Q,P\right)=\mathrm{PA}\left(Q,P\right)/\simeq$.

Let's make a few steps back to see what we are actually manipulating. An element $\left(\varphi,\left(f_p\right)_{p\in\mathscr{P}}\right)\in\mathrm{PA}\left(Q,P\right)$ is a pair consisting of a function $\varphi:\mathrm{Ob}\left(\mathscr{P}\right)\to\mathrm{Ob}\left(\mathscr{Q}\right)$ and a set of arrows $f_p:Q\left(\varphi(p)\right)\to P(p)$ that make a certain diagram commute. Note that the set of arrows look like a natural transformation, due to the fact that φ sends an object of $\mathscr{P}$ to an object of $\mathscr{Q}$. In fact, if $\mathscr{Q}=\mathscr{P}$, pairs $\left(\mathrm{id}_{\mathscr{Q}},\alpha\right)$ with $\alpha:Q\to P$ being a natural transformation, are elements of $\mathrm{PA}\left(Q,P\right)$, but they may not be the only ones. Besides, $\mathrm{PA}\left(Q,P\right)$ may be non-empty, with none of its elements being a natural transformation.

One may see the arrows in $\mathbf{Pro}(\mathscr{C})$ as an equivalence class of generalisations of natural transformations.

Question 4.5.4. Is there any case where natural transformations are the only elements of $\mathrm{PA}(Q, P)$? At first thought, they are most probably not the only ones, natural transformations are too specific.

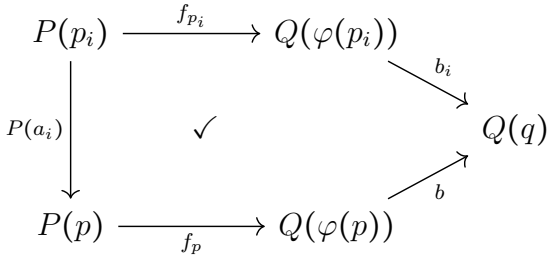
Then, using Theorem 4.5.2, we deduce the form of arrows in $\mathbf{Ind}(\mathscr{C})$.

Description of the arrows of $\mathbf{Ind}(\mathscr{C})$. Let us spend some time describing the arrows and adding more terminology.

Definition 4.5.5 (Partial arrow)

A *partial arrow* $P \to Q$ is a pair $f = \left(\varphi, (f_p)_{p \in \mathscr{P}}\right)$ where:

1. $\varphi : \mathrm{Ob}(\mathscr{P}) \to \mathrm{Ob}(\mathscr{Q})$ is a function
2. for all $p \in \mathscr{P}$, f_p is an arrow $f_p : P(p) \to Q(\varphi(p)) \in \mathscr{C}$
3. the following condition holds: for all finite set of arrows $(a_i : p_i \to p)_{i \in n} \in \mathscr{P}$, there exists $q \in \mathscr{Q}$, $b : \varphi(p) \to q \in \mathscr{Q}$ and $(b_i : \varphi(p_i) \to q)_{i \in n} \in \mathscr{Q}$, such that for all $i \in n$, the following diagram commutes in $\mathscr{C}$:



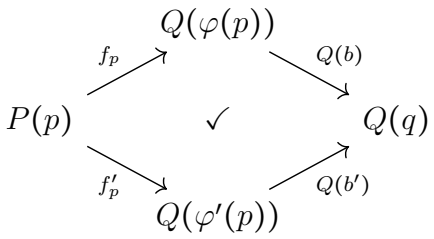
Denote by $\mathrm{PA}^{\mathrm{op}}(P, Q)$ the set of these pairs.

For the sake of both simplicity and readability, we shorten the pair $\left(\varphi, (f_p)_{p \in \mathscr{P}}\right)$ into f.

We then define the equivalence relation $\simeq$ generated by:

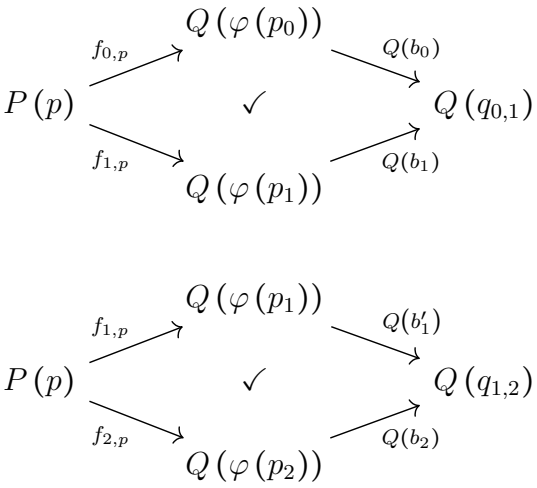
$\left(\varphi, (f_p)_{p \in \mathscr{P}}\right) \sim \left(\varphi', \left(f'_p\right)_{p \in \mathscr{P}}\right)$ iff for all $p \in \mathscr{P}$, there exist $b : \varphi(p) \to q$ and $b' :$

$\varphi'(p) \to q$ in $\mathscr{Q}$ such that the following diagram commutes:



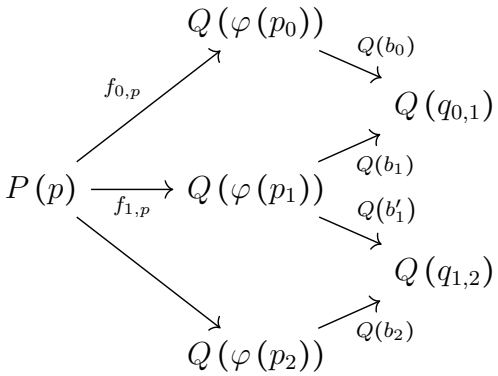
(4.9)

Note that the relation $\sim$ described in Diagram 4.9 is reflexive and symmetric. It only misses the transitivity. To see what the generated equivalence relation $\simeq$ adds to $\sim$, assume we have $(\varphi_0, (f_{0,p})) \sim (\varphi_1, (f_{1,p}))$ and $(\varphi_1, (f_{1,p})) \sim (\varphi_2, (f_{2,p}))$, that is, for all $p \in \mathscr{P}$, the following two diagrams commute:



(4.10)

The obvious way to introduce transitivity here, would be to glue both diagrams on the common arrow $f_{1,p}$, just like this:



It is immediate to prove:

Proposition 4.5.6

The equivalence relation $\simeq$ generated by $\sim$, is such that $(\varphi, (f_p)) \simeq \left(\varphi', \left(f'_p\right)\right)$ iff there exists a finite sequence $(\varphi_i, (f_{i,p}))_{i \in n+1} \in \mathrm{PA}^{\mathrm{op}}(P, Q)$ such that:

1. $(\varphi_0, (f_{0,p})) = (\varphi, (f_p))$
2. $(\varphi_n, (f_{n,p})) = \left(\varphi', \left(f'_p\right)\right)$
3. for all $i \in n$, $(\varphi_i, (f_{i,p})) \sim (\varphi_{i+1}, (f_{i+1,p}))$

Note that the third condition exactly translates the fact that, for all $p \in \mathscr{P}$, f_p and f'_p are in the same connected component of $(P(p)|Q)$.

Definition 4.5.7 (DH-arrow)

An *arrow of Deleanu-Hilton* $P \to Q$, or *DH-arrow* $P \to Q$ for short, is an equivalence class of elements of $\mathrm{PA}^{\mathrm{op}}(P, Q)$ under the relation $\simeq$ defined above. We define $\mathrm{DH}(P, Q) = \mathrm{PA}^{\mathrm{op}}(P, Q) / \simeq$.

In [37] and [43], the duals of these arrows are used as the arrows of $\mathbf{Pro}(\mathscr{C})$. Therefore, we have another definition of the arrows in $\mathbf{Ind}(\mathscr{C})$: $\mathrm{Hom}_{\mathbf{Ind}(\mathscr{C})}(P, Q) = \mathrm{DH}(P, Q)$. As we will see in Proposition 4.5.11, this does not overwrite the previous definition of arrows in $\mathbf{Ind}(\mathscr{C})$ seen in Definition 4.3.2, as both definitions are isomorphic.

4.5.2 A natural isomorphism

We can now compare this official definition of clusters, with the ones we gave before. It seems easier for us to compare this definition with Proposition 4.4.2; the main idea is that, gathering all the arrows of a given equivalence class, we should find similar sets of arrows. This is what we are going to prove through the following lemmas and theorem.

An element of $\mathrm{DH}(P, Q)$ is an equivalence class $[\varphi, (f_p)]$ of specific arrows (see the description above). Just as we did to define $\mathrm{Clstr}(P, Q)$ (Definition 4.3.7), we will gather these arrows and see what happens.

Define:

$$\Phi_{P,Q} : \begin{cases} \mathrm{DH}(P,Q) & \longrightarrow \quad \mathcal{P}\left(\biguplus_{p\in\mathscr{P}} \bigcup_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q))\right) \\ [\varphi_r,(f_{r,p})] & \longmapsto \quad \{(p,f_p) \mid p\in\mathscr{P}, (\varphi,(f_p))\in[\varphi_r,(f_{r,p})]\} \end{cases}$$

Lemma 4.5.8

If $Q(b)\circ g \in \Phi_{P,Q}([f_r])$, then $g\in\Phi_{P,Q}([f_r])$.

Proof. In fact, if f_{p_0} is in any diagram, then replacing it by $Q(b_0)\circ g_0$ does not change anything related to commutativity. Moreover, it is easy to see that f', defined by $f'_p = f_p$ if $p\neq p_0$ and $f'_{p_0} = g_0$ otherwise, is a partial arrow equivalent to f. The detail is left to the reader. □

Lemma 4.5.9

If $\mathscr{Q}$ is filtered, $\Phi_{P,Q}$ is a function $\mathrm{DH}(P,Q)\to\mathrm{Clstr}(P,Q)$.

Proof. We have to prove that $\Phi_{P,Q}([f_r])$ verifies the five conditions of a cluster.

$\Phi_{P,Q}([f_r])$ obviously verifies (CLU-1) due to the definition of an element $f\in[f_r]$. The rest is not trivial.

For (CLU-2), define:

$$\Phi_{P,Q}([f_r])(p) = \{f_p \mid (p,f_p)\in\Phi_{P,Q}([f_r])\}$$

Let $g: P(p)\to Q(q)$ and $g': P(p)\to Q(q')\in\Phi_{P,Q}([f_r])(p)$. Those two arrows are some $f_p = g$ and $f'_p = g'$ for some $f, f'\in[f_r]$. As they are in the same equivalence class, we know that $f\simeq f'$ for the equivalence relation $\simeq$ defined in Proposition 4.5.6. Thus, for all p, f_p and f'_p are in the same connected component of $(P(p)|Q)$. So, $\Phi_{P,Q}([f_r])$ verifies (CLU-2).

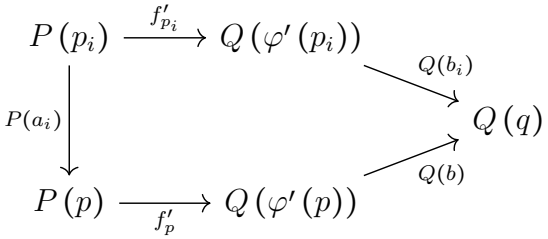
Now, let $a_0: p_0\to p'_0\in\mathscr{P}$ and $f_{p'_0}: P(p'_0)\to Q(q)\in\Phi_{P,Q}([f_r])$. We want to see if $f_{p'_0}\circ P(a_0)\in\Phi_{P,Q}([f_r])$. For that, we define a new partial arrow $f' = \left(\varphi', \left(f'_p\right)_{p\in\mathscr{P}}\right)$ such that:

$$\varphi'(p) = \begin{cases} \varphi(p'_0) & \text{if } p = p_0 \\ \varphi(p) & \text{otherwise} \end{cases}$$

$$f'_p = \begin{cases} f_{p_0}\circ P(a_0) & \text{if } p = p_0 \\ f_p & \text{otherwise} \end{cases}$$

Before anything, we have to check that f' actually is a partial arrow.

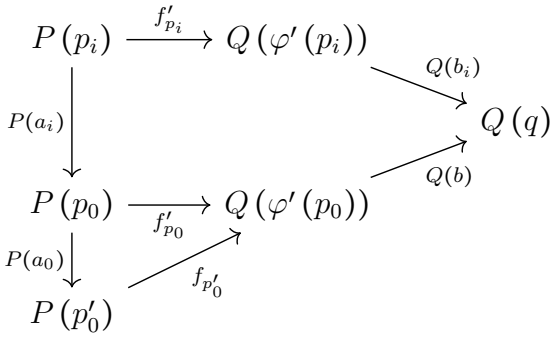
Let $(a_i : p_i \to p)_{i\in n}$ be a finite set of arrows in $\mathscr{P}$. We want to find q and a set of arrows $(b_i : \varphi'(p_i) \to q)_{i\in n}$ such that, for all $i \in n$, the following diagram commutes:



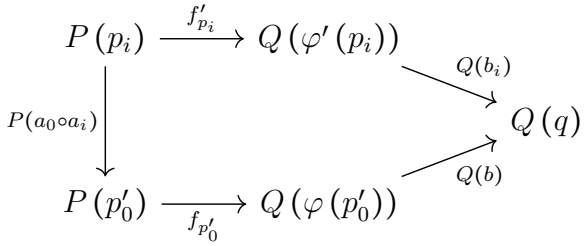
(4.11)

Let us consider three cases:

1. If none of the p_i and p is p_0, then take the same q and $(b_i)_{i\in n}$ as for f
2. If $p = p_0$, Diagram 4.11 becomes:

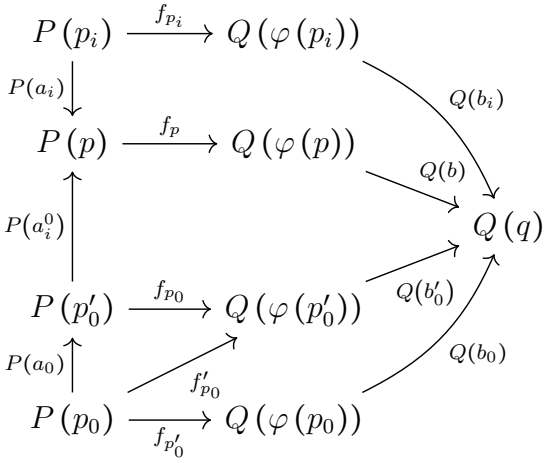


Consider the finite set of arrows $(a_0 \circ a_i : p_i \to p'_0) \in \mathscr{P}$. We then have the following diagram:



If none of the p_i is p_0, as f is a partial arrow, there exist such q, b, and b_i's such that the previous diagram commutes. If some of the p_i's are p_0, then go to the next case.

3. If any of the p_i's are p_0 but $p \neq p_0$, Diagram 4.11 becomes:

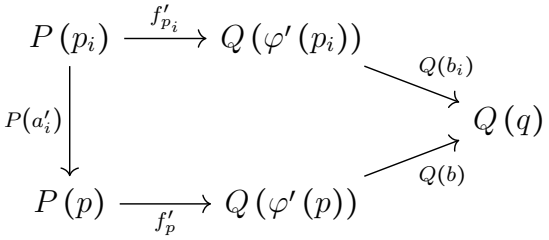


(4.12)

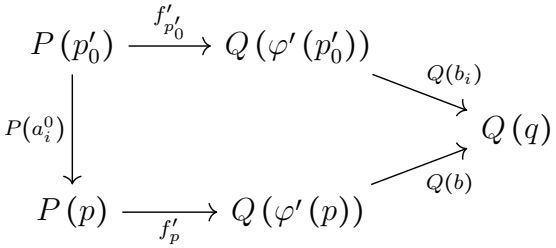
Consider now the finite set of arrows:

$$(a'_i)_{i\in n'} = (a_i : p_i \to p)_{p_i \neq p_0} \cup \left(a_i^0 : p_0' \to p\right)_{p_i = p_0} \cup \left(a_i^0 \circ a_0 : p_i \to p\right)_{p_i = p_0}$$

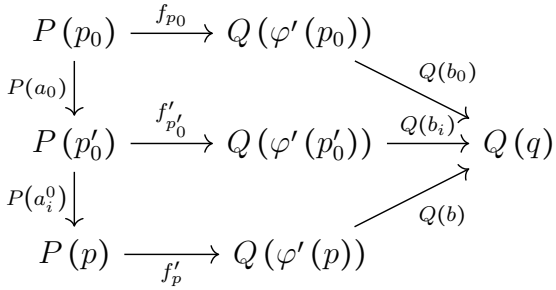
As f is a partial arrow, there are q, b and b_i's such that Diagram 4.11 commutes, with (a'_i) instead of (a_i), and f instead of f'. Then, depending on the case, the commutativity here translates to the commutativity in Diagram 4.12. In more detail; if $a'_i = a_i$, then the following diagram commutes, because $f'_p = f_p$:



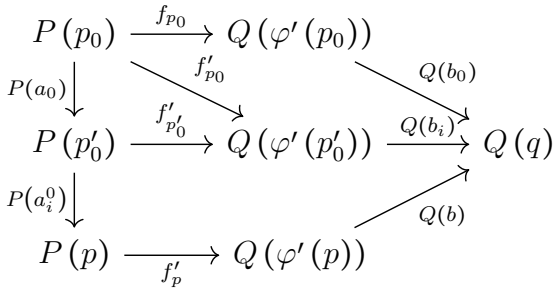
If $a'_i = a_i^0 : p_0' \to p$, then the following diagram commutes, because $f_p = f'_p$ and $f'_{p_0'} = f_{p_0'}$:



Finally, if $a'_i = a^0_i \circ a_0 : p_0 \to p$, then the following diagram commutes:



The bottom diagram commutes due to the previous case: because $f'_{p'_0} = f_{p'_0}$ and $f'_p = f_p$. The top diagram commutes because $f'_{p'_0} = f_{p'_0}$, which leads us to the following diagram:



Putting all diagrams together, we finally deduce that Diagram 4.12 commutes.

Therefore, f' defined as equal to f, except for p_0 where $f'_{p_0} = f_{p'_0} \circ P(a_0)$, is also a partial arrow. It is trivial to verify that $f' \simeq f$, and thus $f' \in [f_r]$. Therefore, $f'_{p_0} = f_{p'_0} \circ P(a_0) \in \Phi_{P,Q}([f_r])$, and $\Phi_{P,Q}([f_r])$ verifies (CLU-4).

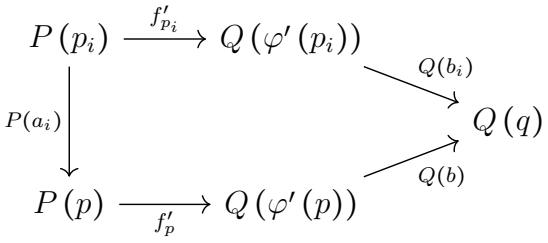
As for (CLU-3), consider $f_{p_0} : P(p_0) \to Q(\varphi(p_0)) \in \Phi_{P,Q}([f_r])$ and $b_0 : \varphi(p_0) \to q_0 \in \mathscr{Q}$; consider f' defined by:

$$\varphi'(p) = \begin{cases} q_0 & \text{if } p = p_0 \\ \varphi(p) & \text{otherwise} \end{cases}$$

$$f'_p = \begin{cases} Q(b_0) \circ f_{p_0} & \text{if } p = p_0 \\ f_p & \text{otherwise} \end{cases}$$

Just as for (CLU-4), we have to check that this actually defines a partial arrow. Let $(a_i : p_i \to p)_{i \in n}$ be a finite set of arrows, we want to check if there exist q and

b_i's such that the following diagram commutes for all i:



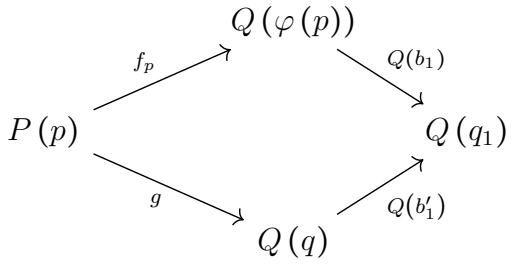
There are two subcases:

1. if p_0 is neither any of the p_i's nor p
2. if $p = p_0$ or $p_i = p_0$ for some i

Case 1 is obvious, as it amounts to verifying that f is a partial arrow (which it is by assumption). Due to the assumption that $\mathscr{Q}$ is filtered, case 2 becomes trivial: as $\mathscr{Q}$ is filtered, there are q, $b : \varphi'(p) \to q$ and arrows $(b_i : \varphi(p_i) \to q)_{i \in n}$, and the diagram necessarily commutes by composition.

Finally, we have to check that $\Phi_{P,Q}([f_r])$ is maximal. Let $f_p \in \Phi_{P,Q}([f_r])$ and $g : P(p) \to Q(q)$ are in the same connected component of $(P(p) \mid Q)$. We will use the previous observation and (CLU-3) to prove that $g \in \Phi_{P,Q}([f_r])$.

There is a zigzag between f_p and g in $(P(p) \mid Q)$. Consider the simple zigzag below:



We know that $f_p \in \Phi_{P,Q}([f_r])$. By (CLU-3), $Q(b_1) \circ f_p \in \Phi_{P,Q}([f_r])$. But $Q(b_1) \circ f_p = Q(b'_1) \circ g$, and as $Q(b_1) \circ f_p \in \Phi_{P,Q}([f_r])$, by Lemma 4.5.8, we have $g \in \Phi_{P,Q}([f_r])$. If the zigzag is longer, then each transitional arrow is in $\Phi_{P,Q}([f_r])$ as well. □

Lemma 4.5.10

Let $f = \left(\varphi, (f_p)_{p \in \mathscr{P}}\right) \in \mathrm{PA}^{\mathrm{op}}(P, Q)$ be a partial arrow.
There is at most one cluster containing f.

Proof. Let G_1 and G_2 be two clusters containing f. For each p, we have $f_p \in G_1(p)$

and $f_p \in G_2(p)$. By Proposition 4.4.3, $G_1(p)$ and $G_2(p)$ are connected components of $(P(p) \mid Q)$. Thus, they are equivalence classes, and they share one element, which means that $G_1(p) = G_2(p)$ and thus, $G_1 = G_2$. □

Theorem 4.5.11

If $\mathscr{Q}$ is filtered, $\mathrm{Clstr}(P, Q) \cong \mathrm{DH}(P, Q)$, naturally in P and Q.

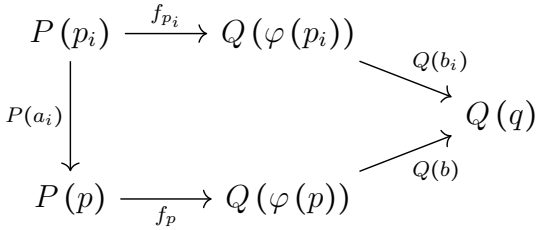
Proof. [**Proof of bijectivity**] Let $f_{r,1}$ and $f_{r,2}$ be two partial arrows such that:

$$\Phi_{P,Q}([f_{r,1}]) = \Phi_{P,Q}([f_{r,2}])$$

Let $f_1 \in [f_{r,1}]$; all its arrows are in both $\Phi_{P,Q}([f_{r,1}])$ and $\Phi_{P,Q}([f_{r,2}])$. Thus, for all $p \in \mathscr{P}$, the arrow $f_{1,p}$ is in the same connected component of $(P(p) \mid Q)$ as both $f_{r,1,p}$ and $f_{r,2,p}$. We deduce that $f_{r,1,p}$ and $f_{r,2,p}$ are in the same connected component of $(P(p) \mid Q)$, which means that they are representatives of the same equivalence class. Thus, $[f_{r,1}] = [f_{r,2}]$ and $\Phi_{P,Q}$ is injective.

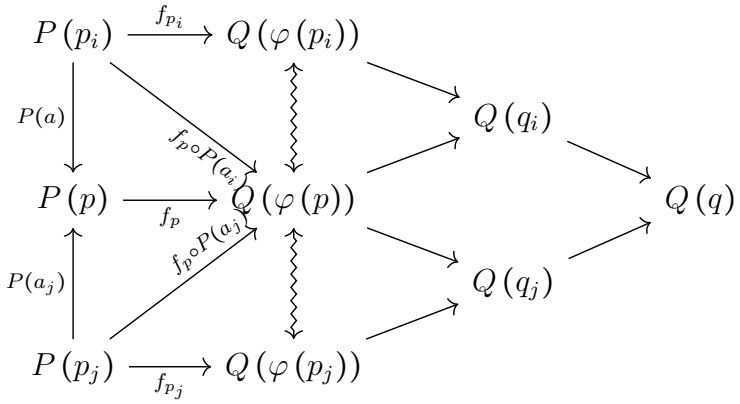
Let G be a cluster. For all $p \in \mathscr{P}$, take one arrow $f_p : P(p) \to Q(q)$ in G. Then for each chosen $f_p : P(p) \to Q(q)$, define $\varphi(p) = q$. We have to check that the pair $\left(\varphi, (f_p)_{p \in \mathscr{P}}\right)$ is partial arrow.

Let $(a_i : p_i \to p)_{i \in n} \in \mathscr{P}$ be a finite set of arrows. We want a q, an arrow $b : \varphi(p) \to q$, and a finite set of arrows $(b_i : \varphi(p_i) \to q)_{i \in n}$ such that for all $i \in n$, the following diagram commutes:



For all $i \in n$, the arrow $f_p \circ P(a_i)$ is in G as well. As $G(p_i)$ is a connected component of $(P(p_i) \mid Q)$, we deduce that, f_{p_i} and $f_p \circ P(a_i)$ are in that connected component, and thus, there is a zigzag between them. Then, as $\mathscr{Q}$ is filtered, for all i, there is a cocone from that zigzag to q_i in $\mathscr{Q}$. As cocones are preserved by functors, this gives a cocone from a zigzag to $Q(q_i)$. Similarly, as $\mathscr{Q}$ is filtered, there is also a cocone to the whole (finite) diagram containing the zigzags and the q_i's,

hence the result.



So, the pair f is a partial arrow. Also note that a partial arrow is a (1)-protocluster. By Lemma 4.5.10, there is at most one cluster containing f. As we know from Lemma4.5.9 that $\Phi_{P,Q}([f])$ is a cluster, and G also is, and both contain all the arrows of f, we deduce that $G = \Phi_{P,Q}([f])$. Therefore, $\Phi_{P,Q}$ is surjective, and thus bijective.

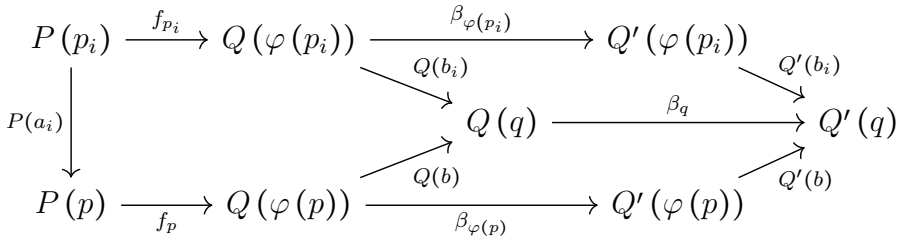
[**Proof of naturality in** Q] The naturality in P will be similar but reversed, as both functors are contravariant in P.

Let $\beta : Q \to Q'$ be a natural transformation. Let us first describe the action of $\mathrm{DH}(P, -)$ on β.

Let $f = \left(\varphi, (f_p)_{p \in \mathscr{P}}\right)$ be a partial arrow. Define $\beta \circ f$ to be the pair:

$$\beta \circ f = \left(\varphi, \left(\beta_{\varphi(p)} \circ f_p\right)_{p \in \mathscr{P}}\right) \tag{4.13}$$

As the following diagram commutes:



we can say that $\beta \circ f$ actually is a partial arrow. It represents an equivalence class $[\beta \circ f]$.

Therefore, $\mathrm{DH}(P,\beta)$ is the following function:

$$\mathrm{DH}(P,\beta): \begin{cases} \mathrm{DH}(P,Q) & \longrightarrow & \mathrm{DH}(P,Q') \\ [f] & \longmapsto & [\beta \circ f] \end{cases}$$

We want the following diagram to commute:

$$\begin{array}{ccc} \mathrm{DH}(P,Q) & \xrightarrow{\Phi_{P,Q}} & \mathrm{Clstr}(P,Q) \\ \Big\downarrow{\scriptstyle \mathrm{DH}(P,\beta)} & ? & \Big\downarrow{\scriptstyle \mathrm{Clstr}(P,\beta)} \\ \mathrm{DH}(P,Q') & \xrightarrow[\Phi_{P,Q'}]{} & \mathrm{Clstr}(P,Q') \end{array}$$

Let $[f] \in \mathrm{DH}(P,Q)$:

$$\mathrm{Clstr}(P,\beta) \circ \Phi_{P,Q}([f]) = \biguplus_{p \in \mathscr{P}} \beta \circ \Phi_{P,Q}([f])(p) \tag{4.14}$$

where $\beta \circ \Phi_{P,Q}([f])(p)$ is the connected component of $(P(p) \mid Q)$ that contains the arrows $\beta_{\varphi(p)} \circ f'_p$, for f'_p the arrow with index p of a partial arrow $f' \in [f]$.

Also:

$$\begin{aligned} \Phi_{P,Q'} \circ \mathrm{DH}(P,\beta)([f]) &= \Phi_{P,Q'}([\beta \circ f]) \\ &= \biguplus_{p \in \mathscr{P}} \Phi_{P,Q'}([\beta \circ f])(p) \end{aligned}$$

where $[\beta \circ f]$ is the equivalence class containing the partial arrow $\left(\varphi, \left(\beta_{\varphi(p)} \circ f_p\right)_{p \in \mathscr{P}}\right)$ described above. It is easy to see that it is the same cluster as in Equation 4.14. □

Discussion. We could not achieve a result in the general case, which leads us to the conclusion that the generalised pro-category described in [37, Section 0, Definition 0.1, page 475] might be different to what a generalised pro-category would be.

Specifically, the authors say:

> Of course, in the extended context of Definition 0.1 we no longer claim that our explicit description may be summarized by (0.1); [...] ([37])

Here, (0.1) is the usual definition of the arrows of $\mathbf{Pro}(\mathscr{C})$:

$$\lim_{p\in\mathscr{P}} \operatorname*{Colim}_{q\in\mathscr{Q}} \operatorname{Hom}_{\mathscr{C}}(Q(q), P(p))$$

which of course holds for cofiltered diagrams. The remark of the authors stresses out that the arrows they describe (and we dualize here) may not coincide as the extension of this set to general diagrams.

What this says, is that the generalised pro-category from [37] might not coincide with the dual of our generalised ind-category. That pro-category might not even have a completion property (dual of the cocompletion).

4.6 Clusters as functors, clusters as functions

In this section, we give a new characterisation of clusters, together with a new natural isomorphism. The later result also allows us to give an upper bound to the number of clusters between any two diagrams. We also give a conjecture on a tighter upper bound and disprove another.

This section relies on the notions introduced in Section 4.2 (comma-categories and connected components).

4.6.1 Clusters as functors

The following definition will be useful throughout this section.

Definition 4.6.1 (Presheaf of connected components)

Consider two diagrams $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$.
The *presheaf of connected components of P and Q*, denoted $\mathrm{CC}_{P,Q}$, or simply CC if there is no ambiguity, is the following functor:

$$\mathrm{CC}_{P,Q} : \begin{cases} \mathscr{P}^{\mathrm{op}} & \longrightarrow & \mathbf{Sets} \\ p & \longmapsto & \{C \subset (P(p) \mid Q) \mid C \text{ is a connected component}\} \\ a : p \to p' & \longmapsto & \mathrm{CC}_{P,Q}(a) : \begin{cases} \mathrm{CC}_{P,Q}(p') & \longrightarrow & \mathrm{CC}_{P,Q}(p) \\ [g] & \longmapsto & [g \circ P(a)] \end{cases} \end{cases}$$

Above, the notation $[g]$ refers to the connected component of $(P(p) \mid Q)$ of which g is an element; g is equivalently refered to as the representative of its connected component $[g]$. Thus, $\mathrm{CC}_{P,Q}(a)$ is the function that sends the

connected component with representative g to the connected component with representative $g \circ P(a)$.

Here, $\mathscr{P}$ and $\mathscr{Q}$ are assumed small, so each connected component is a set, and the $\mathrm{CC}(p)$'s are sets.

In the following, we will refer to CC instead of $\mathrm{CC}_{P,Q}$. Also note that CC is contravariant.

Now consider Definition 4.4.1 (precluster); let us give a weaker version.

Definition 4.6.2 (Protocluster)

Let P and Q be two diagrams in $\mathscr{C}$.
A *protocluster* $P \to Q$ is a subset of the objects of $(P \mid Q)$.
If $\bar{n}$ is a tuple of integers between 1 and 5 (both included), then a *$\bar{n}$-protocluster* $P \to Q$ is a protocluster $P \to Q$ that verifies (CLU-n) of Proposition 4.4.2 for all $n \in \bar{n}$.

For example, a $(3,4)$-protocluster $P \to Q$ is a set of arrows $P(p) \to Q(q)$ that verifies (CLU-3) and (CLU-4). A $(1,2,3,4)$-protocluster is a precluster and a $(1,2,3,4,5)$-protocluster is a cluster.

Fact 4.6.3

If G is a protocluster, then $G = \sum_{p \in \mathscr{P}} G(p)$.

Definition 4.6.4 (Target of an object)

Let $G : P \to Q$ be a protocluster, and let $p \in \mathscr{P}$.
The *target of p in G*, denoted by $\mathrm{Tgt}_G(p)$, is the following set:

$$\mathrm{Tgt}_G(p) = \{C \in \mathrm{CC}(p) \mid G \cap C \neq \varnothing\}$$

In other words, the target of p is the set of connected components of $(P(p) \mid Q)$ that have elements in G (roughly, the connected components "targeted" by $p \in \mathscr{P}$).

As a proof-of-concept use of these notions, we have:

Proposition 4.6.5

Let $G : P \to Q$ be a protocluster.

1. G verifies (CLU-1) $\Leftrightarrow$ for all $p \in \mathscr{P}$, $\mathrm{card}\left(\mathrm{Tgt}_G(p)\right) \geqslant 1$
2. G verifies (CLU-2) $\Leftrightarrow$ for all $p \in \mathscr{P}$, $\mathrm{card}\left(\mathrm{Tgt}_G(p)\right) \leqslant 1$
3. G verifies (CLU-4) $\Rightarrow$ for all $a : p \to p'$, $\mathrm{CC}(a)\left(\mathrm{Tgt}_G(p')\right) \subset \mathrm{Tgt}_G(p)$

In the third item, the set $\mathrm{CC}(a)\left(\mathrm{Tgt}_G(p')\right)$ is the image of $\mathrm{Tgt}_G(p')$ by $\mathrm{CC}(a)$, that is, the set of elements of the form $\mathrm{CC}(a)(C)$ for $C \in \mathrm{Tgt}_G(p')$.

Proof. Actually, the first two items are obvious. As for the third item, let $a : p \to p'$ and let $C \in \mathrm{CC}(a)\left(\mathrm{Tgt}_G(p')\right)$. There is an arrow $g : P(p') \to Q(q) \in G$ such that $C = [g \circ P(a)]$. As G verifies (CLU-4) and $g \in G$, we have $g \circ P(a) \in G$, and thus $[g \circ P(a)] \in \mathrm{Tgt}_G(p)$. □

Note that the third item is not an equivalence. This is because $[g \circ P(a)] \in \mathrm{Tgt}_G(p)$ does not imply that $g \circ P(a) \in G$.

Remark 4.6.6. It is worth noticing that Item 3 implies some kind of compatibility between the connected components targeted by G. This is a direct application of Remark 4.2.3. This remark is translated in Item 3 using $\mathrm{CC}(-)$ and $\mathrm{Tgt}_G(-)$.

We immediately have:

Corollary 4.6.7

Let G be a $(1, 2, 4)$-protocluster.
For all $a : p \to p'$, we have $\mathrm{CC}(a)\left(\mathrm{Tgt}_G(p')\right) = \mathrm{Tgt}_G(p)$.

This gives a hint that we may define a functor based on Tgt_G whenever G is a $(1, 2, 4)$-protocluster:

$$\mathrm{Tgt}_G : \begin{cases} \mathscr{P}^{\mathrm{op}} & \longrightarrow & \mathbf{Sets} \\ p & \longmapsto & \mathrm{Tgt}_G(p) \\ a : p \to p' & \longmapsto & \mathrm{CC}(a)\big|_{\mathrm{Tgt}_G(p')} \end{cases} \tag{4.15}$$

In other words, Tgt_G becomes a subfunctor of CC (Definition 2.2.29).

Proposition 4.6.8

Let $G : P \to Q$ be protocluster.
G is a cluster $\Leftrightarrow$ Tgt_G is a functor and for all $p \in \mathscr{P}$, $\mathrm{Tgt}_G(p) = \{G(p)\}$.

Proof. Let G be a cluster. Then Tgt_G is a functor by Corollary 4.6.7. Using Proposition 4.4.3, G verifies (CLU-2b). For all p, $G(p)$ is a connected component, and necessarily, $G(p) \in \mathrm{Tgt}_G(p)$.

Assume that G is a protocluster. For all $p \in \mathscr{P}$, $G(p) \in \mathrm{Tgt}_G(p)$, so $G(p)$ is a connected component of $(P(p) \mid Q)$, which means that:

1. G verifies (CLU-2b)
2. Each $G(p)$ is non-empty, so G verifies (CLU-1)
3. If $g : P(p) \to Q(q) \in G$ and $b : q \to q' \in \mathscr{Q}$, then $Q(b) \circ g \in G(p)$, so G verifies (CLU-3)

Finally, (CLU-4) is induced by the functoriality of Tgt_G. In fact, if $g : P(p') \to Q(q)$ and $a : p \to p'$, then $g \in G(p')$ and $[g] = G(p')$. We derive $\mathrm{Tgt}_G(a)(G(p')) = \mathrm{CC}(a)(G(p')) = \mathrm{CC}(a)([g]) = [g \circ P(a)] = G(p)$, which is a connected component that contains $g \circ P(a)$. By Proposition 4.4.3, G is a cluster. □

Proposition 4.6.8 is a huge step towards the next main result. Let us introduce a few definitions before proving it.

Definition 4.6.9 (Mono-subfunctors of CC)

A *mono-subfunctor of* $\mathrm{CC}_{P,Q}$ is a subfunctor $T : \mathscr{P}^{\mathrm{op}} \to \mathbf{Sets}$ of $\mathrm{CC}_{P,Q}$ with $T(p) \cong 1$ for all $p \in \mathscr{P}$. We denote by $\mathrm{MSf}(P,Q)$ the set of mono-subfunctors of $\mathrm{CC}_{P,Q}$.

Proposition 4.6.8 states that if G is a cluster, then Tgt_G is a mono-subfunctor of CC.

Theorem 4.6.10

$\mathrm{MSf}(P,Q) \cong \mathrm{Clstr}(P,Q)$, naturally in both P and Q.

Proof. [**Proof of bijectivity**] Define the following functions:

$$\mathrm{Tgt}_\star : \begin{cases} \mathrm{Clstr}(P,Q) & \longrightarrow & \mathrm{MSf}(P,Q) \\ G & \longmapsto & \mathrm{Tgt}_G \end{cases}$$

$$G_\star : \begin{cases} \mathrm{MSf}(P,Q) & \longrightarrow & \mathrm{Clstr}(P,Q) \\ T & \longmapsto & G_T = \biguplus\limits_{p \in \mathscr{P}} (\bigcup T(p)) \end{cases}$$

In the definition of $G_\star$, the notation $\bigcup T(p)$ is set-theoretic. In fact, if t_p (a

connected component of $(P(p) \mid Q)$) is the unique element of $T(p)$, in other words, if $T(p) = \{t_p\}$, then $\bigcup T(p) = t_p$ (this is a connected component of $(P(p) \mid Q)$).

Proposition 4.6.8 makes it easy to see that $\mathrm{Tgt}_\star$ is well-defined. We let the reader check that, for $T \in \mathrm{MSf}(P,Q)$, G_T satisfies the axioms of Proposition 4.4.3. On the one hand, by Proposition 4.6.8:

$$G_\star \circ \mathrm{Tgt}_\star(G) = G_{\mathrm{Tgt}_G} = \biguplus_{p \in \mathscr{P}} \left(\bigcup \mathrm{Tgt}_G(p)\right) = \biguplus_{p \in \mathscr{P}} G(p) = G$$

On the other hand, for any p and $a : p \to p'$ in $\mathscr{P}$:

$$\begin{aligned} \mathrm{Tgt}_{G_T}(p) &= \{G_T(p)\} = \{\bigcup T(p)\} = T(p) \\ \mathrm{Tgt}_{G_T}(a) &= \mathrm{CC}(a)\big|_{T(p')} = T(a) \end{aligned}$$

hence $\mathrm{Tgt}_\star \circ G_\star(T) = T$. It follows that $G_\star = {\mathrm{Tgt}_\star}^{-1}$ because T is a subfunctor of CC.

[**Proof of naturality in** Q] Here we treat the naturality in Q. The naturality in P is similar, but contravariant. For $\beta : Q \to Q'$ any natural transformation and $T \in \mathrm{MSf}(P,Q)$, we define:

$$"\beta \circ T" : \left\{ \begin{array}{ccc} \mathscr{P}^{\mathrm{op}} & \longrightarrow & \mathbf{Sets} \\ p & \longmapsto & \{\beta \circ t_p\} \end{array} \right.$$

where t_p is the unique element of $T(p)$, and $\beta \circ t_p$ is defined as the connected component of $(P(p) \mid Q')$ that contains the arrows of the form $\beta_q \circ g$, for $g : P(p) \to Q(q) \in t_p$. By routine algebra, in the same vein as that of Section 4.3.2, we deduce that $\mathrm{MSf}(P,\beta)$ is written as:

$$\mathrm{MSf}(P,\beta) : \left\{ \begin{array}{ccc} \mathrm{MSf}(P,Q) & \longrightarrow & \mathrm{MSf}(P,Q') \\ T & \longmapsto & "\beta \circ T" \end{array} \right.$$

Using the definition of $\mathrm{Clstr}(P,\beta)$, it is very easy to see, after some calculation left to the reader, that the following diagram commutes:

$$\begin{array}{ccc} \mathrm{MSf}(P,Q) & \xrightarrow{G_\star^{P,Q}} & \mathrm{Clstr}(P,Q) \\ \Big\downarrow{\scriptstyle \mathrm{MSf}(P,\beta)} & \checkmark & \Big\downarrow{\scriptstyle \mathrm{Clstr}(P,\beta)} \\ \mathrm{MSf}(P,Q') & \xrightarrow[G_\star^{P,Q'}]{} & \mathrm{Clstr}(P,Q') \end{array}$$

□

4.6.2 Discussion about the cardinality of clusters

It is easy to derive a first upper bound to the cardinal of clusters, using Theorem 4.6.10:

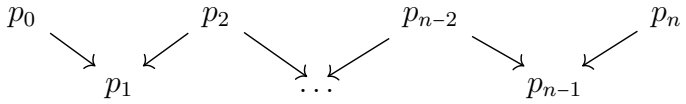
Corollary 4.6.11

$$\operatorname{card}\left(\operatorname{Clstr}\left(P,Q\right)\right) \leqslant \prod_{p\in\mathscr{P}} \operatorname{card}\left(\operatorname{CC}\left(\left(P(p) \mid Q\right)\right)\right)$$

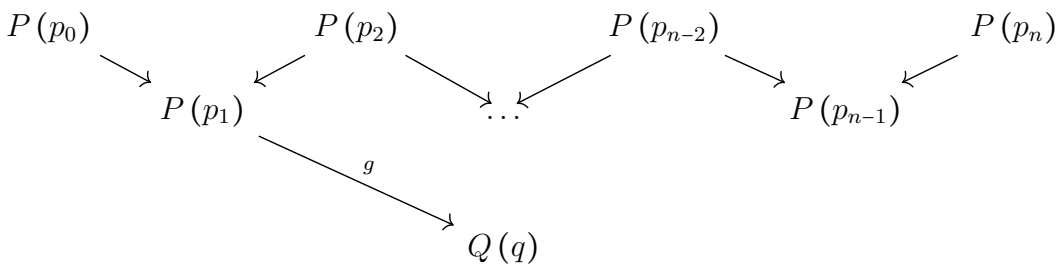
This estimation is rough. We conjecture that the study of the connected components of $(P \mid Q)$ ant those of $\mathscr{P}$ can make this estimation more precise.

Let us explain.

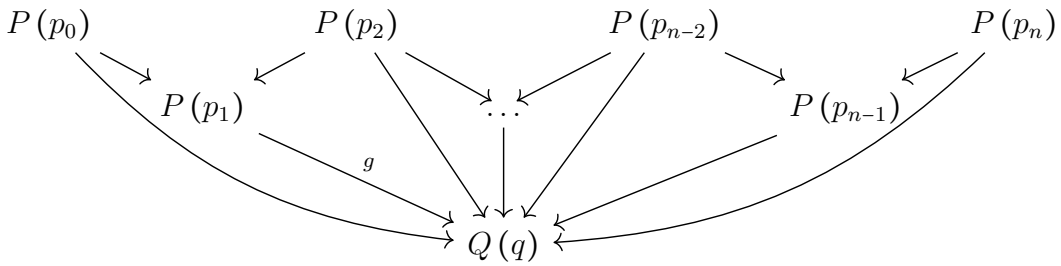
As stated in Remark 4.6.6, and as a consequence of Remark 4.2.3, (CLU-4) is a condition that forces a cluster to be "coherent", or "compatible" in terms of connected components, in the sense of a functor. In this respect, Proposition 4.6.8 roughly states that G is a cluster if and only if, whenever p and p' are in the same connected component of $\mathscr{P}$, then the connected components $G(p)$ and $G(p')$ should be "compatible", in the sense of the functoriality of Tgt_G. Or, in other words, in the sense that, if we have a zigzag of objects in $\mathscr{P}$:



and an arrow $g : P(p_1) \to Q(q) \in G$:



then all the other objects should have an arrow to this $Q(q)$:

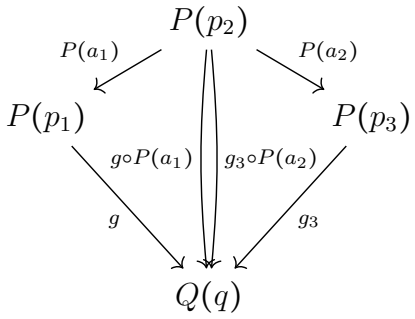


(4.16)

which defines, imposes the connected component targetted by all the objects connected to p_1. In other words, the connected components targetted by $P(p_1)$ is (almost) the "same" as the connected component targetted by $P(p_0)$, $P(p_2)$, ... and $P(p_n)$.

Remark 4.6.12. Note that, above, all the triangles do not necessarily commute.

Let us "extract" the following square from Diagram 4.16:



Let us consider p_3, which was not represented in Diagram 4.16. Let us give labels to the arrows: $a_1 : p_2 \to p_1$, $a_2 : p_2 \to p_3$, $g_3 : P(p_3) \to Q(q)$.

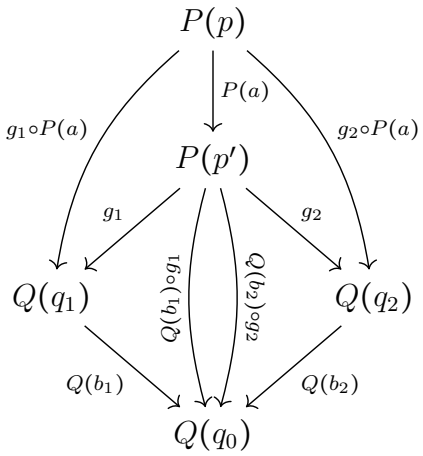
Then we do not necessarily have $g \circ P(a_1) = g_3 \circ P(a_2)$, because nothing implies that $g \circ P(a_1)$ factors through g_3. However, $g \circ P(a_1)$ and $g_3 \circ P(a_2)$ are necessarily in the same connected component of $(P(p_2) \mid Q)$.

In summary, if one arrow $P(p) \to Q(q)$ exists, then that arrow "contaminates" every object in the connected component of p in $\mathscr{P}$. We already observed that clusters $P \to Q$ had something to do with the connected components of the categories $(P(p) \mid Q)$ (Proposition 4.6.5), but the previous consideration gives a hint that they may also have some link with those of $\mathscr{P}$.

Remark 4.6.13. It is worth noticing that the connected components of $(P \mid Q)$ may be totally unrelated with those of the $(P(p) \mid Q)$'s. Two objects g and g' might be

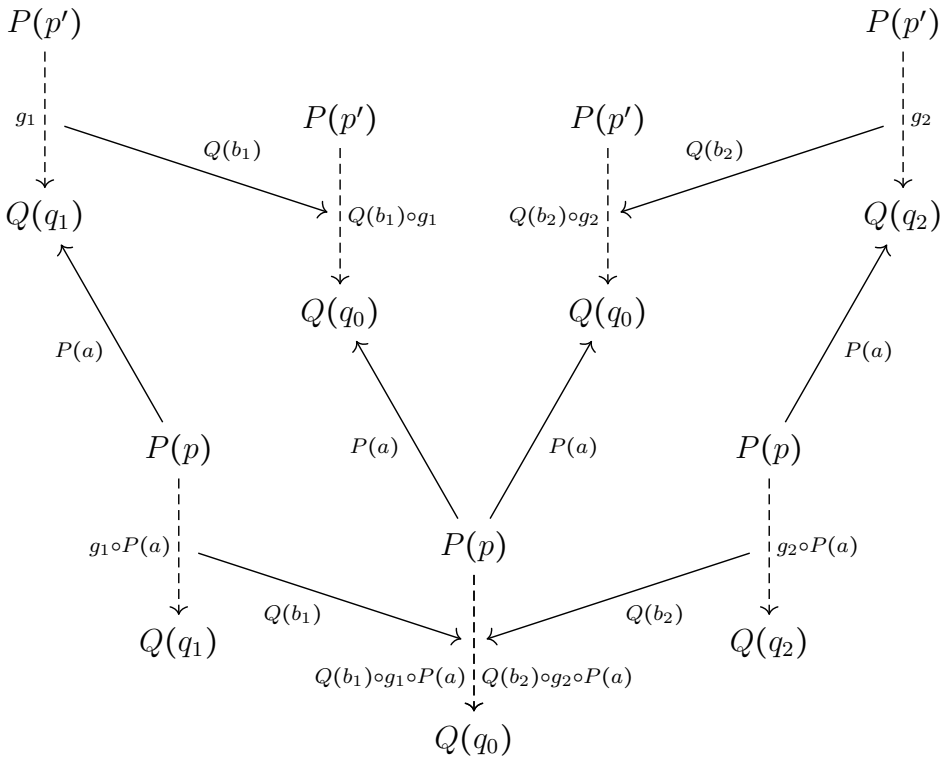
disconnected in $(P(p) \mid Q)$ but connected in $(P \mid Q)$.

Let P and Q be diagrams to $\mathscr{C}$ such that $\mathscr{C}$ is:

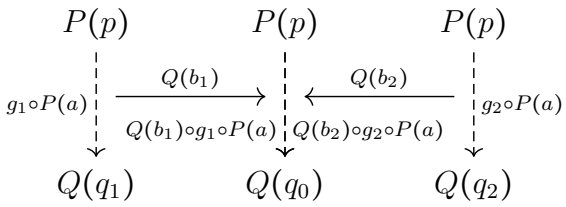


where the two arrows $Q(b_1) \circ g_1$ and $Q(b_2) \circ g_2$ are different, but at the same time, $Q(b_1) \circ g_1 \circ P(a) = Q(b_2) \circ g_2 \circ P(a)$. This may happen if $P(a)$ "equalises" $Q(b_1) \circ g_1$ and $Q(b_2) \circ g_2$ just like an equaliser would.

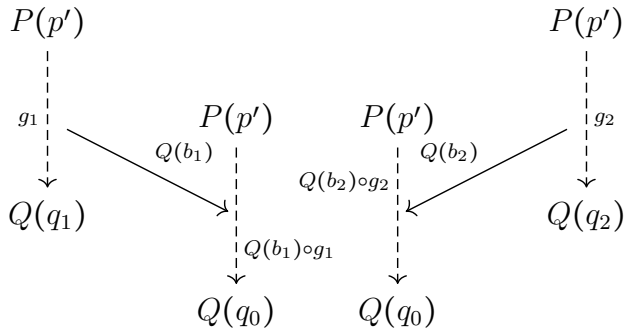
The comma-category $(P \mid Q)$ is:



while the comma-categories $(P(p) \mid Q)$ and $(P(p') \mid Q)$ are respectively:



and:



We have g_1 and g_2 connected in $(P \mid Q)$, but disconnected in $(P(p') \mid Q)$.

In this subsection, we did not achieve any consistent result on the cardinality of clusters. However, the analysis given seem to point out the key role played by the connected components of $\mathscr{P}$ (cf. the contamination of arrows).

We have the following conjecture:

Conjecture 4.6.14. $\operatorname{card}(\operatorname{Clstr}(P,Q)) \leqslant \left(\max_{p \in \mathscr{P}} \operatorname{card}(\operatorname{CC}(P(p) \mid Q))\right)^{\operatorname{card}(\operatorname{CC}(\mathscr{P}))}$.

This conjecture is left for further work.

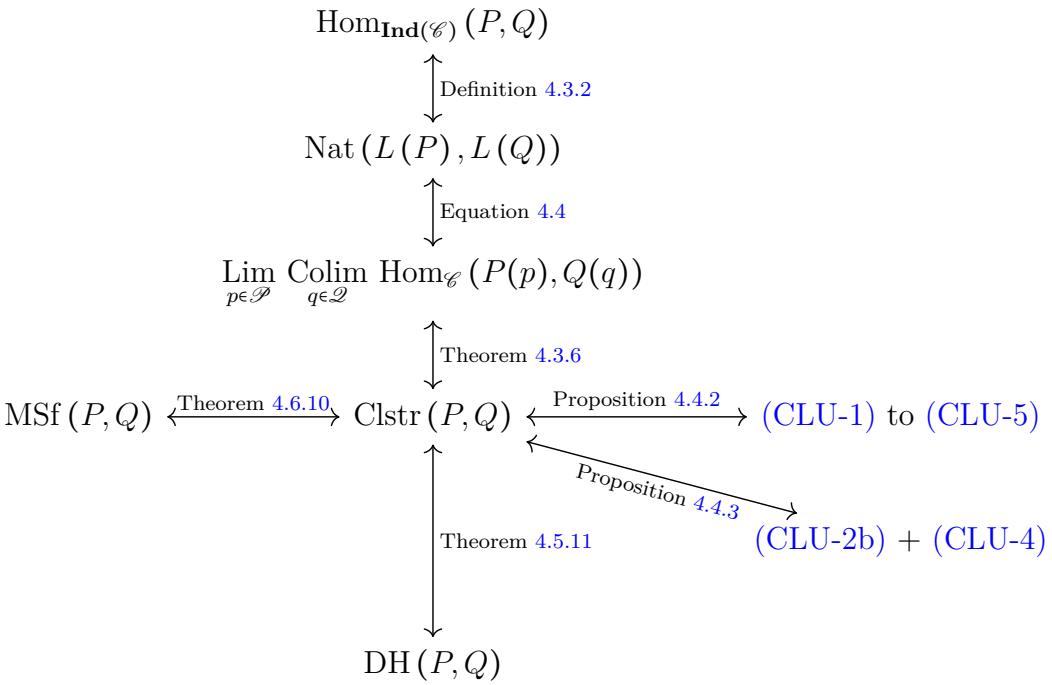
4.7 Conclusion

We saw the following equivalent definitions of clusters. A cluster $P \to Q$ is, equivalently, any of the following:

1. An element of $\operatorname{LC}_{\mathscr{C}}(P,Q) = \varprojlim_{p \in \mathscr{P}} \varinjlim_{q \in \mathscr{Q}} \operatorname{Hom}_{\mathscr{C}}(P(p), Q(q))$ (Lemma 4.3.4)
2. An element of $\operatorname{Clstr}(P,Q)$ (Equation 4.7, Definition 4.3.7)
3. A maximal precluster (Definition 4.4.1 and Proposition 4.4.4)
4. A disjoint union of connected components (Proposition 4.4.3)

5. An arrow of Deleanu-Hilton, element of $\mathrm{DH}(P,Q)$ (Definition 4.5.7, if $\mathscr{Q}$ is filtered)
6. A mono-subfunctor of the CC-functor, element of $\mathrm{MSf}(P,Q)$ (Definition 4.6.9)

These equivalences are summarized in the following diagram:



Among these definitions, the equivalence between the first three was known since their creation [35, Exposé 1, section 8.2, page 67]. To the best of our knowledge, our formalisation of the definition of clusters seen in [12, Chapter 3, Section 2.1, page 81], is new, as well as the two other equivalent definitions, and both characterisations in terms of connected-component-aware functions and mono-subfunctors of the connected-component functor.

In the context of the study of interactions between subsystems, the best definitions to use are, in our opinion, either Proposition 4.4.2 or Proposition 4.4.3. The former introduces clusters as maximal preclusters, leading to a total of five conditions to verify, while the latter prefers four conditions, among which (CLU-2b) may turn difficult to verify in practice.

Although this chapter's goal was to be exhaustive, we could not find a natural isomorphism between Deleanu-Hilton arrows and clusters in the general case, i.e. when $\mathscr{Q}$ is not assumed filtered.

Chapter 5

Construction of clusters

If you tell a categorist that you co-constructed a theory, they will understand that you destroyed it.

Erwan Beurier.

5.1 Introduction

In the theory of Memory Evolutive Systems by Ehresmann and Vanbremeersch [44] [12] [45], systems, be they biological, social or artificial, are seen as hierarchical categories evolving through time. Those categories become more and more complex in their structure, due to a property called the *Multiplicity Principle* (MP for short) [12, chapter 4, section 4.3, page 92]. The MP is a property of categories that present two diagrams with isomorphic cocones, such that no cluster verifies a certain property. The purpose of the next chapter is the study of the MP in the context of thin categories, so we will not develop here the property. However, the MP may occur under two circumstances: when there is no cluster between the two diagrams, or there are clusters but they do not verify the given property.

It turns out that the case when two diagrams do not have a cluster can be addressed very generally, i.e. the results stated below hold for all categories. The property to be verified appears to be too complicated for the general case, so in the next chapter, we limits ourselves to the special case of preorders.

In the last chapter, we proved several equivalences between definitions of clusters. In this chapter, we focus on the definition given in Proposition 4.4.2 in order to study when clusters cannot be constructed.

5.2 Generating clusters from the full protocluster

The notion of protocluster was already introduced before (Definition 4.6.2). It is a weaker version of cluster, that is, a set of arrows that verify only certain conditions of a cluster. We remind it here:

Definition 5.2.1 (Protocluster)

Let P and Q be two diagrams in $\mathscr{C}$.
A *protocluster* $P \to Q$ is a subset of the objects of $(P \mid Q)$.
If $\bar{n}$ is a tuple of integers between 1 and 5 (both included), then a *$\bar{n}$-protocluster* $P \to Q$ is a protocluster $P \to Q$ that verifies (CLU-n) of Proposition 4.4.2 for all $n \in \bar{n}$.

For example, a $(3,4)$-protocluster $P \to Q$ is a set of arrows $P(p) \to Q(q)$ that verifies (CLU-3) and (CLU-4). A $(1,2,3,4)$-protocluster is a precluster and a $(1,2,3,4,5)$-protocluster is a cluster.

Just like clusters and preclusters, if $G : P \to Q$ is a protocluster, we define $G(p)$ to be the subset $G \cap \left(\bigcup_{q \in \mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q)) \right)$. We remind here a tiny result that will prove helpful:

Fact 5.2.2

If G is a protocluster, then $G = \sum_{p \in \mathscr{P}} G(p)$.

Among all the protoclusters $P \to Q$, one seems to stand out. As we will see soon, the following protocluster is key to the study of clusters:

Definition 5.2.3 (Full protocluster)

Let P, Q be two diagrams. The *full protocluster* $F : P \to Q$ is the following protocluster:

$$F = \{ g : P(p) \to Q(q) \mid p \in \mathscr{P}, q \in \mathscr{Q} \}$$

That is, F is the set of all arrows of the ambiant category of the form $P(p) \to Q(q)$.

We straightforwardly deduce from the definition that:

Proposition 5.2.4

Let P and Q be two diagrams.

1. There is exactly one full protocluster $P \to Q$
2. The full protocluster verifies (CLU-3) and (CLU-4)

3. The full protocluster is the set $\sum_{p\in\mathscr{P}} \bigcup_{q\in\mathscr{Q}} \mathrm{Hom}_{\mathscr{C}}(P(p), Q(q))$
4. Every protocluster $P \to Q$ is a subset of the full protocluster $P \to Q$

A full protocluster might still not be a cluster. In fact, it may fail to verify (CLU-1) or (CLU-2). By the way:

Proposition 5.2.5 (No-target condition)

No protocluster $G : P \to Q$ verifies (CLU-1) $\Leftrightarrow$ the full protocluster $F : P \to Q$ does not verify (CLU-1).

Proof. [**Proof of** $\Rightarrow$] By Proposition 5.2.4, F is itself a protocluster.

[**Proof of** $\Leftarrow$] Assume there is no arrow with domain $P(p)$ in F. By Proposition 5.2.4, every protocluster G is a subset of F; thus G does not contain any arrow with domain $P(p)$. □

Proposition 5.2.5 is a first example of the importance of the full protocluster in this chapter. When looking at the full protocluster, we may be able to deduce properties of all the protoclusters, and why not, clusters. In fact, if no protocluster verifies (CLU-1), then there is no cluster at all between the two diagrams.

While we are at it, let us continue. The fact that every protocluster is a subset of the full protocluster, together with the fact that the full protocluster contains all possible arrows between two diagrams, give us a hint that clusters must be special subsets. In fact, they are, but we will need to introduce and study a few more notions before.

The problem of the full protocluster, is its "full-ness". Let us explain. As it contains all the arrows between two diagrams, it contains probably "too many" arrows, in the sense that $F(p)$ might be bigger than a connected component of $(P(p) \mid Q)$; it may contain several of them. The natural thought is then to delete some of these arrows, in order to select only one connected component per p, and make a cluster. However, as stated in Section 4.6.1, this needs to be done in such a way that the connected components should be compatible. For a given full protocluster, this might not happen easily.

In the following, we assume that we are given two diagrams P and Q such that the full protocluster $F : P \to Q$ is a $(1, 3, 4)$-protocluster. If (CLU-1) does not hold, then Proposition 5.2.5 states that there is no need to look for clusters. Here, we want to find the clusters inside the full protocluster.

In order to ease the reading of this section, we remind here some notions seen in Section 4.6.1.

Definition 5.2.6 (Presheaf of connected components)

Consider two diagrams $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$.
The *presheaf of connected components of P and Q*, denoted $\mathrm{CC}_{P,Q}$, or simply CC if there is no ambiguity, is the following functor:

$$\mathrm{CC}_{P,Q} : \begin{cases} \mathscr{P}^{\mathrm{op}} & \longrightarrow & \textbf{Sets} \\ p & \longmapsto & \{C \subset (P(p) \mid Q) \mid C \text{ is a connected component}\} \\ a : p \to p' & \longmapsto & \mathrm{CC}_{P,Q}(a) : \begin{cases} \mathrm{CC}_{P,Q}(p') & \longrightarrow & \mathrm{CC}_{P,Q}(p) \\ [g] & \longmapsto & [g \circ P(a)] \end{cases} \end{cases}$$

Above, the notation $[g]$ refers to the connected component of $(P(p) \mid Q)$ that contains g; g is equivalently referred to as the representative of its connected component $[g]$. Thus, $\mathrm{CC}_{P,Q}(a)$ is the function that sends the connected component with representative g to the connected component with representative $g \circ P(a)$.

In the following, we will get rid of the indices and refer to CC instead of $\mathrm{CC}_{P,Q}$, as P and Q are fixed.

Also note that CC is a presheaf; e.g. it is contravariant.

The notion of Target was introduced before (Definition 4.6.4); we remind it here for convenience:

Definition 5.2.7 (Target of an object)

Let $G : P \to Q$ be a protocluster, and let $p \in \mathscr{P}$.
The *target of p in G*, denoted by $\mathrm{Tgt}_G(p)$, is the following set:

$$\mathrm{Tgt}_G(p) = \{C \in \mathrm{CC}(p) \mid \exists g \in G \text{ such that } g \in C\}$$

In the rest of this section, we will focus on how to pick connected components from a given target.

Definition 5.2.8 (Choice of connected components)

Let $G : P \to Q$ be a (1)-protocluster. A *choice of connected components for G*, or *choice function* for short if there is no ambiguity, is a function f of the

form:

$$f : \begin{cases} \mathrm{Ob}(\mathscr{P}) & \longrightarrow & \bigcup\limits_{p \in \mathscr{P}} \mathrm{Tgt}_G(p) \\ p & \longmapsto & C \in \mathrm{Tgt}_G(p) \end{cases}$$

A choice of connected component takes an object $p \in \mathscr{P}$ and returns some connected component of $\mathrm{Tgt}_G(p)$.

The choice function will be used in order to "fix" protoclusters; the goal is to alter a protocluster so that it verifies (CLU-2).

Definition 5.2.9 (Fixes of a protocluster)

Let $G : P \to Q$ be a $(1,3,4)$-protocluster. Let f be a choice of connected components.
The *f-fix for G* is the following protocluster:

$$G_{\mathrm{fix}} = \sum_{p \in \mathscr{P}} (G(p) \cap f(p))$$

A *fix of G* is a protocluster $G_{\mathrm{fix}} : P \to Q$ defined as the f-fix of G for some choice of connected components f.

In short, a fix of G is a protocluster that will be proved to verify (CLU-2). As we will see in Lemma 5.3.2 in the next section, "fixing" a protocluster might not happen easily - that is, without sacrificing (CLU-4).

But for now, we are looking for clusters inside the full protocluster.

Lemma 5.2.10

Every cluster is a fix of the full protocluster.

Proof. Let $F : P \to Q$ be a full protocluster, and let $G : P \to Q$ be a cluster. For all $p \in \mathscr{P}$, $G(p)$ is a connected component of $(P(p) \mid Q)$ (Proposition 4.4.3). We thus have $\mathrm{Tgt}_G(p) = \{G(p)\}$. Define f to be the choice function that assigns to each $p \in \mathscr{P}$ this unique connected component $G(p) \in \mathrm{Tgt}_G(p) \subset \mathrm{Tgt}_F(p)$, because $G \subset F$. This choice function f comes with its f-fix F_{fix} and by construction:

$$F_{\mathrm{fix}} = \sum_{p \in \mathscr{P}} (F(p) \cap f(p)) = \sum_{p \in \mathscr{P}} (F(p) \cap G(p)) = \sum_{p \in \mathscr{P}} G(p) = G$$

□

Note that the notion of fix only exists if the full protocluster verifies (CLU-1). Thus, Lemma 5.2.10 and Proposition 5.2.5 are perfectly compatible and complement each other.

5.3 When is there no cluster at all?

This section explores conditions for the existence of clusters. From the formula $\underset{p\in\mathscr{P}}{\mathrm{Lim}}\ \underset{q\in\mathscr{Q}}{\mathrm{Colim}}\ \mathrm{Hom}_{\mathscr{C}}(P(p),Q(q))$, it is a hard task to determine whether the set of clusters is empty or not. This chapter aims at finding a more direct way to answer this question.

5.3.1 A quest of compatibility

From Proposition 5.2.5 and Lemma 5.2.10, we directly derive:

Proposition 5.3.1

There is no cluster $P \to Q \Leftrightarrow$ the full protocluster $P \to Q$ does not verify (CLU-1) or no fix of it is a cluster.

We may agree that the second condition of the right-hand part of the equivalence is not actually enlightening, as it does not say anything about the "form" of the full protocluster. In this section, we study how to turn this condition into a test that we can run.

Again, we assume that we are given two diagrams P and Q such that the full protocluster $F : P \to Q$ is a $(1,3,4)$-protocluster. Allowing a tiny bit of generalisation, the rest of this study focuses on $(1,3,4)$-protoclusters.

Lemma 5.3.2 (Duality of (CLU-2) and (CLU-4))

Let $G : P \to Q$ be a protocluster and G_{fix} be a fix of G.
If G is a $(1,3,4)$-protocluster then G_{fix} is a $(1,2,3)$-protocluster.

Proof. Let f be the choice of connected components of G associated with G_{fix}. By construction, G_{fix} verifies:

1. (CLU-1): for all $p \in \mathscr{P}$, we have an arrow $P(p) \to Q(q) \in G_{\mathrm{fix}} \cap f(p)$
2. (CLU-2): for each $p \in \mathscr{P}$, $\mathrm{Tgt}_{G_{\mathrm{fix}}}(p) = \{f(p)\}$ so p targets only one connected component $f(p)$ via G_{fix}

3. (CLU-3): for each $g : P(p) \to Q(q) \in G_{\text{fix}}$, for each $Q(b) : Q(q) \to Q(q')$, we have $g \in f(p) \subset G$. As G verifies (CLU-3), we have $Q(b) \circ g \in G$. As $Q(b) \circ g$ has domain $P(p)$, we have $Q(b) \circ g \in f(p)$, and thus, by Definition 5.2.9, $Q(b) \circ g \in G_{\text{fix}}$.

□

The following result is a direct consequence of Lemma 5.3.2 and the assumption that no cluster exists between P and Q.

Corollary 5.3.3

Let P and Q be two diagrams with no clusters between them.
If G is a $(1,3,4)$-protocluster that does not verify (CLU-2) then G_{fix} is a $(1,2,3)$-protocluster that does not verify (CLU-4).

Proof. We assumed that there was no cluster between P and Q, so G_{fix} is still not a cluster. As it verifies the first three conditions (Lemma 5.3.2), we deduce that the last one, (CLU-4), fails. □

One may interpret Corollary 5.3.3 as follows: if (CLU-2) fails for a hopeful cluster G, it is because there are too many arrows in G, and this "excess" of arrows is due to (CLU-4).

On the other hand, as fixing basically removes arrows from G, the failure of (CLU-4) in the fix means that too many arrows of G were removed.

Another way to look at things is that (CLU-4) imposes some "compatibility" between the connected components targetted by G, in the sense that if the connected component of g is targeted by G, then the connected component of $g \circ P(a)$ should also be targeted. In a sense, (CLU-4) makes sure that there is no "conflict" of connected components. This compatibility is discussed in Section 4.6, as illustrated by Theorem 4.6.10 (clusters are seen as a kind of functor).

One last remark. Lemma 5.3.2 also makes it clear why we are using Proposition 4.4.2 (clusters verify (CLU-1) to (CLU-5)) instead of Proposition 4.4.4 (clusters verify (CLU-2b) and (CLU-4)). This is because the absence of clusters depends on (CLU-1), (CLU-2) and (CLU-4), and those conditions are easier to verify than (CLU-2b).

In the following sections, we give a few theorems resulting for this quest of compatibility. We give them all the name of "Compatible Connected Component Theorem" (abbreviated in CCCT), because they follow the same idea of compatibility between connected components.

5.3.2 A first attempt - Very Weak CCCT

In this section, we introduce a first attempt of a CCCT. The previous sections make this attempt obvious, as it is in the continuity of them; however, its proof requires a tiny lemma that we prove rightaway.

Lemma 5.3.4 (Tiny lemma)

Let $(X_i)_{i\in I}$ be a non-empty set of non-empty sets, such that $\bigcap_{i\in I} X_i = \varnothing$. Then, there exists X_1 and X_2, such that there exists $x_1 \in X_1 \backslash X_2$ and $x_2 \in X_2$. In particular, $\operatorname{card}\left(\bigcup_{i\in I} X_i\right) \geqslant 2$.

Proof. There are two cases.

1. Either there exist X_1 and X_2 such that $X_1 \cap X_2 = \varnothing$; as X_1 and X_2 are non empty, we obtain the wanted result;
2. Or, for all pairs $X_1 \neq X_2$, we have $X_1 \cap X_2 \neq \varnothing$. If we had $X_1 \cap X_2 = X_2$ for all pairs X_1, X_2, then we would also have $X_1 \cap X_2 = X_1$ and $\bigcap_{i\in I} X_i$ would be non-empty and equal to X_1. Thus, there necessarily exist X_1 and X_2 such that $X_1 \cap X_2 \subsetneq X_1$.
   Therefore, $X_1 \cup X_2 = (X_1 \backslash X_2) + X_2 \subset \bigcup_{i\in I} X_i$. As $X_1 \cap X_2 \subsetneq X_1$, we know that $(X_1 \backslash X_2) \neq \varnothing$, hence the result.

□

The first attempt of stating a CCCT is the following, that we call "Very Weak":

Theorem 5.3.5 (Very Weak CCCT)

Let G be a $(1,3,4)$-protocluster.
There exists $p_0 \in \mathscr{P}$ such that $\bigcap_{a:p_0\to p} \mathrm{CC}(a)(\mathrm{Tgt}_G(p)) = \varnothing \Rightarrow G$ does not verify (CLU-2) and none of its fixes verifies (CLU-4).

Proof. We assume that there is a p_0 such that:

$$\bigcap_{a:p_0\to p} \mathrm{CC}(a)(\mathrm{Tgt}_G(p)) = \varnothing$$

By Proposition 4.6.5, as G verifies (CLU-1), for all $p \in \mathscr{P}$, we have:

$$\operatorname{card}\left(\operatorname{Tgt}_G(p)\right) \geqslant 1$$

Therefore, $\operatorname{card}\left(\operatorname{CC}(a)\left(\operatorname{Tgt}_G(p)\right)\right) \geqslant 1$; in particular, each $\operatorname{CC}(a)\left(\operatorname{Tgt}_G(p)\right)$ is non-empty (for any $a : p_0 \to p \in \mathscr{P}$).

Using Lemma 5.3.4, we find that $\operatorname{card}\left(\bigcup_{a:p_0\to p} \operatorname{CC}(a)\left(\operatorname{Tgt}_G(p)\right)\right) \geqslant 2$.

By assumption, G verifies (CLU-4). Also by Proposition 4.6.5, for all $a : p_0 \to p$, $\operatorname{CC}(a)\left(\operatorname{Tgt}_G(p)\right) \subset \operatorname{Tgt}_G(p_0)$. We deduce that $\bigcup_{a:p_0\to p} \operatorname{CC}(a)\left(\operatorname{Tgt}_G(p)\right) \subset \operatorname{Tgt}_G(p_0)$, which gives $\operatorname{card}\left(\operatorname{Tgt}_G(p_0)\right) \geqslant 2$. Therefore, by Proposition 4.6.5, G does not verify (CLU-2).

Now, let G_{fix} be any fix of G.

As G_{fix} verifies (CLU-1) and (CLU-2) (Lemma 5.3.2), it follows from Proposition 4.6.5 that for all $p \in \mathscr{P}$, we have $\operatorname{card}\left(\operatorname{Tgt}_G(p)\right) = 1$. We deduce that for all $a : p_0 \to p$, $\operatorname{card}\left(\operatorname{CC}(a)\operatorname{Tgt}_{G_{\text{fix}}}(p)\right) = 1$. Also, we assumed that $\bigcap_{a:p_0\to p} \operatorname{CC}(a)\operatorname{Tgt}_{G_{\text{fix}}}(p) = \varnothing$. By Lemma 5.3.4 again, we have:

$$\operatorname{card}\left(\bigcup_{a:p_0\to p} \operatorname{CC}(a)\operatorname{Tgt}_{G_{\text{fix}}}(p)\right) \geqslant 2$$

In addition, by Proposition 4.6.5, if G_{fix} verified (CLU-4), we would have, for all $a : p_0 \to p$, $\operatorname{CC}(a)\left(\operatorname{Tgt}_{G_{\text{fix}}}(p)\right) \subset \operatorname{Tgt}_{G_{\text{fix}}}(p_0)$, which can be rewritten as:

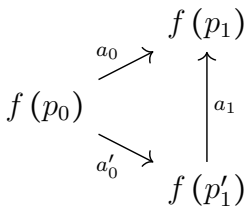
$$\bigcup_{a:p_0\to p} \operatorname{CC}(a)\operatorname{Tgt}_{G_{\text{fix}}}(p) \subset \operatorname{Tgt}_{G_{\text{fix}}}(p_0)$$

which contradicts $\operatorname{card}\left(\operatorname{Tgt}_{G_{\text{fix}}}(p_0)\right) = 1$. Therefore, G_{fix} does not verify (CLU-4). □

Remark 5.3.6 (The converse of the Very Weak CCCT probably does not hold). As stated in the title of this remark, there is no reason for the converse to hold. In order to see why, let us try to construct a cluster from a protocluster G that verifies $\forall p_0 \in \mathscr{P}$, $\bigcap_{a:p_0\to p} \operatorname{CC}(a)\left(\operatorname{Tgt}_G(p)\right) \neq \varnothing$.

We tried to make an algorithm for constructing a fix of G, however, in our few attempts, we could not ensure that the connected components were compatible. We always faced the same problem. Assume that $f(p_0)$ was chosen in $\operatorname{CC}(p_0)$ and $f(p_1)$ and $f(p_1')$ were chosen to be compatible with $f(p_0)$, such that $\operatorname{CC}(a_0)\left(f(p_1)\right) =$

$f(p_0) = \mathrm{CC}(a'_0)(f(p'_1))$.



We could not find a way to choose $f(p_1)$ and $f(p'_1)$ so that they were also compatible, in the sense that $\mathrm{CC}(a_1)(f(p_1)) = f(p_1)$.

The problem is more visible in the case of two parallel arrows:

$$f(p_0) \underset{a'_0}{\overset{a_0}{\rightrightarrows}} f(p_1)$$

The condition stated in the Very Weak CCCT is not enough to tackle this case; in fact, nothing ensures that:

$$\mathrm{CC}(a_0)^{-1}(f(p_0)) \cap \mathrm{CC}(a'_0)^{-1}(f(p_0)) \neq \varnothing$$

and thus, nothing ensures that $f(p_1)$ can be chosen to be compatible with $f(p_0)$ along both arrows a_0 and a'_0.

Thus, the compatibility needs to be stepped up.

5.3.3 Second attempt - Weak CCCT

In this section, we introduce a new functor which will allow for a new weak condition for the non-existence of clusters. This definition uses a transfinite recursion. Briefly, a transfinite recursion is a recursion in three steps instead of two:

1. Step 0: define the starting point S_0 (just like a normal recursion)
2. Step 1: for an ordinal α, if S_α is defined, define $S_{\alpha+1} = f(S_\alpha)$ for a certain function f (just like a normal recursion)
3. Step 2: for a limit ordinal λ, define S_λ as a kind of "limit" of all the S_α with $\alpha < \lambda$. Typically, S_λ will be the intersection or the union of all the S_α's.

For more information on transfinite recursion, on the uniqueness of the thus defined function, see [27, Chapter 1, §9, pages 23-27].

We keep working with $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$, and the connected component functor $\mathrm{CC}_{P,Q}$, written as CC.

We define two functions S and T by transfinite recursion:

1. $S_0 = \mathrm{CC}$
2. If S_α is defined for an ordinal α, then for all $p_0 \in \mathscr{P}$, we set:

$$T_\alpha(p_0) = \bigcap_{a:p\to p_0\in\mathscr{P}} \mathrm{CC}(a)^{-1}(S_\alpha(p))$$

3. If T_α is defined for an ordinal α, then for all $p_0 \in \mathscr{P}$, we set:

$$S_{\alpha+1}(p_0) = \bigcap_{a:p_0\to p\in\mathscr{P}} \mathrm{CC}(T_\alpha(p))$$

4. If λ is a limit ordinal, define $S_\lambda = \bigcap_{\alpha<\lambda} S_\alpha$
5. Define $S = \bigcap_\alpha S_\alpha$
6. Define $T = \bigcap_\alpha T_\alpha$

The recursion continues at most until all the $S_\alpha(p)$'s and $T_\alpha(p)$'s are empty, so it stops at most after $\sup_{p\in\mathscr{P}}(\mathrm{card}(\mathrm{CC}(p)))$ steps (the argument is that we cannot remove infinitely many elements). This ensures that the definition of S and T are sound, because they are intersections over a *set* (and not a *class*) of ordinals.

Before drawing the links between S, T and clusters, let us study those functions and their recursion.

Lemma 5.3.7

For all α, $S_{\alpha+1} \subset T_\alpha \subset S_\alpha$.

Proof. Let α be an ordinal. By definition:

$$S_{\alpha+1}(p_0) = \bigcap_{a:p_0\to p\in\mathscr{P}} \mathrm{CC}(a)(T_\alpha(p)) \subset \mathrm{CC}(\mathrm{id}_{p_0})(T_\alpha(p_0)) \subset T_\alpha(p_0)$$

Similarly:

$$T_\alpha(p_0) = \bigcap_{a:p\to p_0\in\mathscr{P}} \mathrm{CC}(a)^{-1}(S_\alpha(p)) \subset \mathrm{CC}(\mathrm{id}_{p_0})^{-1}(S_\alpha(p_0)) \subset S_\alpha(p_0)$$

□

By a "squeeze argument", we obtain:

Lemma 5.3.8

$S = T$.

Proof. Define $\kappa = \sup_{p \in \mathscr{P}} (\mathrm{card}(\mathrm{CC}(p)))$; κ is the least upper bound of the cardinalities of all $\mathrm{CC}(p)$'s. We already discussed that $S_\kappa = S$ and $T_\kappa = T$ (just below the definition of S and T).

From Lemma 5.3.7, we derive that $S = S_{\kappa+1} \subset T_\kappa = T \subset S_\kappa = S$. □

We can then rewrite S as:

$$S(p_0) = \bigcap_{a : p_0 \to p \in \mathscr{P}} \mathrm{CC}(a)(S(p)) = \bigcap_{a : p \to p_0 \in \mathscr{P}} \mathrm{CC}(a)^{-1}(S(p)) \tag{5.1}$$

Lemma 5.3.9

S is a subfunctor of CC.

Proof. By Lemma 5.3.7, we have $S \subset S_0 = \mathrm{CC}$.

We have to prove that for all $a_0 : p_0' \to p_0$, $S(p_0') = \mathrm{CC}(a_0)(S(p_0))$. From Equation (5.1), we already derive $S(p_0') \subset \mathrm{CC}(a_0)(S(p_0))$ (if a set is a subset of an intersection of sets, then it is a subset of any st in the intersection).

Again, by Equation (5.1), $S(p_0) = \bigcap_{a : p \to p_0 \in \mathscr{P}} \mathrm{CC}(a)^{-1}(S(p)) \subset \mathrm{CC}(a_0)^{-1}(S(p_0'))$ so $\mathrm{CC}(a_0)(S(p_0)) \subset S(p_0')$. □

In short, S is the biggest subfunctor of CC whose components are "always compatible" together. In fact, the definition of T_α in terms of $\mathrm{CC}(a)^{-1}(S_\alpha(p))$ forces the elements of $T_\alpha(p_0)$ (which are connected components of $(P(p_0) \mid Q)$) to be the antecedents of elements of $S_\alpha(p)$. The intersection, again, ensures that these antecedents are always compatible. The same reasoning applies for $S_{\alpha+1}$ in terms of $\mathrm{CC}(a)(T_\alpha(p))$: it forces the elements of $S_{\alpha+1}(p_0)$ to be images of the $T_{\alpha+1}(p)$ in a compatible way.

Lemma 5.3.10

If G is a cluster, then for all p, $G(p) \in S(p)$.

Proof. By Theorem 4.6.10, we can identify a cluster with its target Tgt_G, which is a mono-subfunctor of CC. Then, we observe that, for all $a : p_0 \to p$, $\mathrm{Tgt}_G(p_0) =$

$\mathrm{CC}(a)(\mathrm{Tgt}_G(p))$, which ensures that:

$$\mathrm{Tgt}_G(p_0) = \bigcap_{a:p_0 \to p} \mathrm{CC}(a)(\mathrm{Tgt}_G(p))$$

and also:

$$\mathrm{Tgt}_G(p_0) \subset \bigcap_{a:p \to p_0} \mathrm{CC}(a)^{-1}(\mathrm{Tgt}_G(p))$$

When we relate Tgt_G with S, we have:

1. $\mathrm{Tgt}_G \subset S_0$
2. If $\mathrm{Tgt}_G \subset S_\alpha$, then $\mathrm{CC}(a)^{-1}(\mathrm{Tgt}_G(p)) \subset \mathrm{CC}(a)^{-1}(S_\alpha(p))$ and thus we have the inclusion: $\mathrm{Tgt}_G(p_0) \subset T_\alpha(p_0)$
3. Similarly, if $\mathrm{Tgt}_G \subset T_\alpha$, then $\mathrm{CC}(a)(\mathrm{Tgt}_G(p)) \subset \mathrm{CC}(a)(T_\alpha(p))$ and thus $\mathrm{Tgt}_G(p_0) \subset S_{\alpha+1}(p_0)$
4. Finally, if for all $\alpha < \lambda$, $\mathrm{Tgt}_G \subset S_\alpha$, then $\mathrm{Tgt}_G \subset S_\lambda$

As a consequence, $\mathrm{Tgt}_G \subset S$, hence the result. □

Theorem 5.3.11 (Weak CCCT)

Let $G : P \to Q$ be a $(1,3,4)$-protocluster, define S as in this section.
There exists $p_0 \in \mathscr{P}$ such that $S(p_0) = \varnothing \Rightarrow G$ does not verify (CLU-2) and none of its fixes verifies (CLU-4).

Proof. If a fix of G verified (CLU-4), then it would be a precluster (Lemma 5.3.2) and generate a cluster G_{fix}. By Lemma 5.3.10, $G_{\mathrm{fix}}(p_0) \in S(p_0)$, which contradicts $S(p_0) = \varnothing$. □

Remark 5.3.12. From Remark 5.3.6, we deduced that CC was not enough to characterise the existence, or inexistence of clusters. The new functor S was thought to be a stronger condition, but it seems like it is still not strong enough. In fact, the goal of the recursion was to define S, whose components $S(p)$ only contains connected components of $(P(p) \mid Q)$ that are compatible, not only in the sense that $\mathrm{CC}(a)([g]) = [g \circ P(a)]$, but in the stronger sense that, if there is a zigzag of length n between p_0 and p_n, then if we choose a connected component $[g_0]$ for p_0, then there is necessarily a choice $[g_n]$ that is compatible with it. We could then write an algorithm [1]which would run like this:

1. This algorithm is "rough", because we could not find a way to make it work. The exact algorithm does not seem to matter here, as the main idea never solves the problem that we describe later.

1. Choose a p_0 in $\mathscr{P}$
2. Choose a connected component $[g_0] \in S(p_0)$. Define $U(p_0) = \{[g_0]\}$.
3. If $U(p)$ is defined:
   - 3.1. For all $a_1 : p_1 \to p$, define $U(p_1) = \mathrm{CC}(a_1)(U(p))$
   - 3.2. For all $a_1 : p_0 \to p_1$, define $U(p_1) = \mathrm{CC}(a_1)^{-1}(U(p))$ if $U(p_1)$ is undefined, else $U(p_1) = U(p_1) \cap \mathrm{CC}(a_1)^{-1}(U(p))$

Intuitively, this defines U as the target of a cluster; however we did not manage to prove it. In fact, this algorithm does not seem does not solve the problem of "gluing" different zigzags together; for example again, it does not solve the case of two parallel arrows:

$$[g_0] \underset{a'_0}{\overset{a_0}{\rightrightarrows}} [g_1]$$

If $[g_0]$ is chosen in $S(p_0)$, then we know that:

$$S(p_1) = \bigcap_{a:p\to p_1} \mathrm{CC}(a)^{-1}(S(p)) \subset \mathrm{CC}(a_0)^{-1}(S(p_0)) \cap \mathrm{CC}(a'_0)^{-1}(S(p_0))$$

However, we cannot (again) ensure that $\mathrm{CC}(a_0)^{-1}(\{[g_0]\}) \cap \mathrm{CC}(a'_0)^{-1}(\{[g_0]\})$ is non-empty.

The functor S was created as a way to enforce compatibility between connected components; however, Remark 5.3.12 suggests that this compatibility is not enough. S only ensures a compatibility between sets of connected components, but this compatibility does not seem to reduce to connected components alone (or singletons of them). We conjecture that this is because S ensures a "first order"-like compatibility, that is, a compatibility "in zigzags". Clusters seem to need a "second order"-like compatibility, that is, a compatibility in "squares", that ensures that if there exists two zigzags between two objects of $\mathscr{P}$, then the resulting choices are compatible all the way down the zigzags. This question is out of the scope of this thesis, as no answer was given within the expected time limits.

5.3.4 Third attempt - Transfinite recursion in a transfinite algorithm

In the previous section, S was thought to be a "purified", a "compatibilized" version of CC, removing all the connected components of $(P(p) \mid Q)$ that will fail

to be compatible at some point with another connected component. In other words, S is a smooth version of CC.

The idea of smoothening CC can be used in order to get closer to a third version of the CCCT.

In this section, we consider a function $U : \mathrm{Ob}(\mathscr{P}) \to \mathbf{Sets}$ where $U(p) \subset \mathrm{CC}(p)$. Now we will try to "smoothen" U, in the same way that we smoothened CC in the previous section:

Definition 5.3.13 (Smoothening procedure)

Given a function U such that $U \subset \mathrm{CC}$, the *smoothening of* U is the function V defined by the following transfinite recursion:

1. $V_0 = U$
2. If V_α is defined for an ordinal α, then for all $p_0 \in \mathscr{P}$, we set:
$$W_\alpha(p_0) = \bigcap_{a : p \to p_0 \in \mathscr{P}} \mathrm{CC}(a)^{-1}(V_\alpha(p))$$
3. If W_α is defined for an ordinal α, then for all $p_0 \in \mathscr{P}$, we set:
$$V_{\alpha+1}(p_0) = \bigcap_{a : p_0 \to p \in \mathscr{P}} \mathrm{CC}(W_\alpha(p))$$
4. If λ is a limit ordinal, define $V_\lambda = \bigcap_{\alpha<\lambda} V_\alpha$
5. Define $V = \bigcap_\alpha V_\alpha$
6. Define $W = \bigcap_\alpha W_\alpha$

Remark 5.3.14. The functor S defined in the previous section is obviously the smoothening of CC.

By similarity with the definition of S in the previous section, we deduce:

Lemma 5.3.15

With the same notation as Definition 5.3.13:

1. For all ordinal α, $V_{\alpha+1} \subset W_\alpha \subset V_\alpha$
2. $V \subset U$
3. $W = V$
4. V is a functor

Proof. The proof of Item 1 is the same as Lemma 5.3.7, and Item 2 derives from it. The proof of Item 3 is the same as Lemma 5.3.8 and the proof of Item 4 is the same as Lemma 5.3.9. □

Lemma 5.3.16

With the same notation as Definition 5.3.13, if G is a cluster such that $\mathrm{Tgt}_G \subset U$, then $\mathrm{Tgt}_G \subset V$.

Proof. Let G be a cluster such that $\mathrm{Tgt}_G \subset U$.

For all p_0, we have at the same time:

$$\mathrm{Tgt}_G(p_0) = \bigcap_{a:p_0 \to p \in \mathscr{P}} \mathrm{CC}(\mathrm{Tgt}_G(p))$$
$$\mathrm{Tgt}_G(p_0) \subset \bigcap_{a:p \to p_0 \in \mathscr{P}} \mathrm{CC}(a)^{-1}(\mathrm{Tgt}_G(p))$$

If we assume $\mathrm{Tgt}_G \subset U$. Then, following the description of the transfinite recursion, we have:

1. $\mathrm{Tgt}_G \subset V_0 = U$
2. Assuming $\mathrm{Tgt}_G \subset V_\alpha$ for some ordinal α, then we have:

   $$\mathrm{CC}(a)^{-1}(\mathrm{Tgt}_G(p)) \subset \mathrm{CC}(a)^{-1}(V_\alpha(p))$$

   and thus:

   $$\mathrm{Tgt}_G(p_0) \subset \bigcap_{a:p \to p_0} \mathrm{CC}(a)^{-1}(\mathrm{Tgt}_G(p)) \subset \bigcap_{a:p \to p_0} \mathrm{CC}(a)^{-1}(V_\alpha(p)) = W_\alpha(p_0)$$

3. Assuming $\mathrm{Tgt}_G \subset W_\alpha$ for some ordinal α, we have:

   $$\mathrm{CC}(a)(\mathrm{Tgt}_G(p)) \subset \mathrm{CC}(a)(W_\alpha(p))$$

   and thus:

   $$\mathrm{Tgt}_G(p_0) \subset \bigcap_{a:p_0 \to p} \mathrm{CC}(a)(\mathrm{Tgt}_G(p)) \subset \bigcap_{a:p_0 \to p} \mathrm{CC}(a)(W_\alpha(p)) = V_{\alpha+1}(p_0)$$

4. Assuming $\mathrm{Tgt}_G \subset V_\alpha$ and $\mathrm{Tgt}_G \subset W_\alpha$ for all $\alpha < \lambda$, where λ is a limit ordinal, then we derive $\mathrm{Tgt}_G \subset V_\lambda$ and $\mathrm{Tgt}_G \subset W_\lambda$.

□

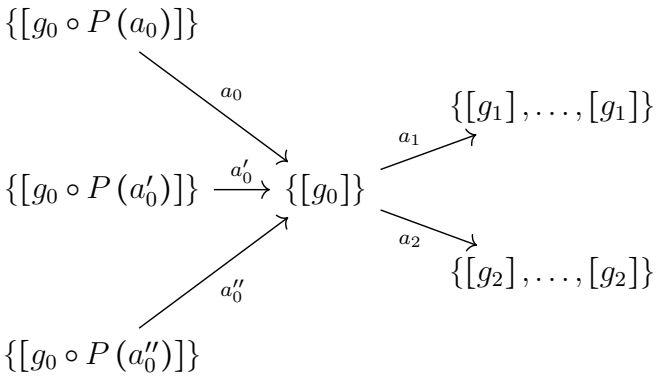
Corollary 5.3.17

With the same notation as Definition 5.3.13, if there exists p such that $V(p) = \varnothing$, then there is no cluster G such that $\mathrm{Tgt}_G \subset U$.

The idea behind the smoothening is to construct clusters from CC, by replacing some $\mathrm{CC}(p_0)$ with a given singleton consisting of a connected component. Let us describe the complete procedure. We will then explain it informally, and finally, we will prove its properties.

For $p_0 \in \mathscr{P}$, define $\mathcal{S}(p_0, 1) = \{p \in \mathscr{P} \mid \mathrm{Hom}_{\mathscr{C}}(p, p_0) \neq \varnothing \text{ or } \mathrm{Hom}_{\mathscr{C}}(p_0, p) \neq \varnothing\}$. In short, $\mathcal{S}(p_0, 1)$ a sphere with radius 1, centered on p_0.

The idea behind this procedure is choose a p_0, and then a connected component in $\mathrm{CC}(p_0)$, written $U(p_0) = \{[g_0]\}$, and then we try to spread this singleton like this:



If $[g_0]$ is chosen for p_0, then for each arrow $a_0 : p \to p_0$, the only compatible choice is $U(p) = \{[g_0 \circ a_0]\}$. Then, if $a_1 : p_0 \to p_1$, then there might be several connected components $[g_1] \in \mathrm{CC}(p_1)$ such that $[g_1 \circ P(a_1)] = [g_0]$. Instead of choosing among them, we just keep them all: $U(p_1) = \{[g] \in \mathrm{CC}(p_1) \mid [g \circ P(a_1)] = [g_0]\}$. Then, we spread again this new choice, along every zigzag starting or finishing with p_0. This spread is formally described by the use of the smoothening function.

If this spread results in an empty set for some p, then there is no cluster G with $G(p_0) = [g_0]$. If the spread finishes with a function U whose components have cardinality 1, then it is a cluster. If the spread stops but the resulting function has some components with cardinality > 1, then we try again. For some p_0' such that $\mathrm{card}(U(p_0')) > 1$, we try and set $U(p_0) = \{[g_0]\}$ but also $U(p_0') = \{[g_0']\}$. We then have two connected components to spread. If the resulting smoothening is empty, then there is no cluster G with $G(p_0) = [g_0]$ and $G(p_0') = [g_0']$. If the resulting

Input: A function $U \subset \mathrm{CC}_{P,Q}$, a set A of pairs $(p, [g] \in \mathrm{CC}(p))$
Output: The set of clusters $P \to Q$

foreach $p \in \mathscr{P}$ **do**
  if *there is a pair* $(p, [g]) \in A$ **then**
    Set $V(p) = \{[g]\}$;
  else
    Set $V(p) = U(p)$;
  end
end
Replace V with its smoothening (Definition 5.3.13)
if *there is a* p *such that* $V(p) = \varnothing$ **then**
  `/* There will be no cluster */`
  Return $\varnothing$;
end
if *for all* $p \in \mathscr{P}$, $V(p) \cong 1$ **then**
  `/* There is only one cluster that corresponds to the choice of connected components given in` A `*/`
  Return $\{V\}$;
end
if *there is a* $p \in \mathscr{P}$ *such that* $V(p) > 1$ **then**
  `/* The smoothening might yield many clusters, the choice of` A `was not enough, there might be several clusters with those connected components */`
  Let Res $= \varnothing$;
  foreach p_1 *such that* $V(p_1) > 1$, *for each* $[g_1] \in V(p_1)$ **do**
    Add EnumerateClusters$(V, A \cup \{(p_1, [g_1])\})$ to Res
  end
  Return Res
end

Procedure EnumerateClusters(U, A)

function has all $U(p)$ with cardinality 1, then U is (the target of) a cluster. If again the resulting U has some components $U(p)$ with cardinality greater than 1, then we repeat the algorithm, this time with three choices of connected components, and so on.

We now prove the stated properties of the algorithm.

Definition 5.3.18 (Cluster complying with partial choice)

A *partial choice* A is a partial function $\mathrm{Ob}(\mathscr{P}) \to \prod_{p \in \mathscr{P}} \mathrm{CC}(p)$ such that for each p with an image, $A(p) \in \mathrm{CC}(p)$.
A cluster G *complies with a partial choice* A when for each p for which $A(p)$ exists, $A(p) = G(p)$.
Denote by $\mathrm{Clstr}^A(P,Q)$ the set of clusters $P \to Q$ that comply with A.

In the algorithm, A is thought as the partial version of the choice of connected components (Definition 5.2.8). Just like a choice function may or may not yield a cluster (its fix), a partial choice may or may not be completed to make a cluster; that is what the algorithm is about.

For convenience, in the description of the algorithm, we write A as a set of pairs. The use of the notion of partial function makes implicit the fact that only one connected component is chosen per p.

Lemma 5.3.19

Let V be the smoothening of a subfunction $U \subset \mathrm{CC}$.
For all $G \in \mathrm{Clstr}^A(P,Q)$, $\mathrm{Tgt}_G \subset U \Rightarrow$ for all $G \in \mathrm{Clstr}^A(P,Q)$, $\mathrm{Tgt}_G \subset V$.

Proof. Direct consequence of Lemma 5.3.16. □

Lemma 5.3.20

$$\mathrm{Clstr}^A(P,Q) = \bigcup_{A \subset A'} \mathrm{Clstr}^{A'}(P,Q).$$

Proof. If $G \in \mathrm{Clstr}^A(P,Q)$, then $G \in \mathrm{Clstr}^{\mathrm{Tgt}_G}(P,Q) \subset \bigcup_{A \subset A'} \mathrm{Clstr}^{A'}(P,Q)$.

As for the converse, it is clear that if G complies with $A' \supset A$, then G complies with A. □

Lemma 5.3.21

Let A be a partial choice, and consider $A' = A \cup \{(p', [g'])\}$ such that A' is also a partial choice (p' is such that $A(p')$ is undefined).
Then EnumerateClusters$(U, A') \subset$ EnumerateClusters(U, A).

Proof. This is clear from the algorithm. □

Lemma 5.3.22

If $A \subset A'$ then EnumerateClusters$(U, A') \subset$ EnumerateClusters(U, A).

Proof. Follows directly from Lemma 5.3.21. □

Lemma 5.3.23

$$\text{EnumerateClusters}(U, A) = \bigcup_{A \subset A'} \text{EnumerateClusters}(U, A').$$

Proof. The inclusion $\supset$ follows directly from Lemma 5.3.22.

Let $G \in$ EnumerateClusters(U, A). Then Tgt_G can be seen as a partial choice, and $A \subset \text{Tgt}_G$. Thus $G \in$ EnumerateClusters$\,(U, \text{Tgt}_G)$. □

Lemma 5.3.24

$$\text{EnumerateClusters}(\text{CC}, \text{A}) = \{G\} \Leftrightarrow \text{Clstr}^A\,(P, Q) = \{G\}.$$

Proof. The algorithm clearly gives: EnumerateClusters$(\text{CC}, \text{A}) \subset \text{Clstr}^A\,(P, Q)$.

Assume that $\text{Clstr}^A\,(P, Q) = \{G\}$. Then either EnumerateClusters(CC, A) is empty, or equal to $\{G\}$. However, we know that:

$$\text{EnumerateClusters}(\text{CC}, \text{A}) = \bigcup_{A \subset A'} \text{EnumerateClusters}(U, A')$$

and as $A \subset \text{Tgt}_G$ as a partial choice, we have $\{G\} \subset$ EnumerateClusters$\,(U, \text{Tgt}_G) \subset$ EnumerateClusters$(\text{CC}, \text{A}) = \{G\}$.

Assume that EnumerateClusters$(\text{CC}, \text{A}) = \{G\}$. Then let $G' \in \text{Clstr}^A\,(P, Q)$. We have $A \subset \text{Tgt}_{G'}$, and thus:

$$\text{EnumerateClusters}(\text{CC}, \text{Tgt}_{\text{G}}) \subset \text{EnumerateClusters}(\text{CC}, \text{A})$$

by Lemma 5.3.22, and thus $G = G'$. □

Theorem 5.3.25

For any partial choice A, $\mathrm{EnumerateClusters}(\mathrm{CC}, \mathrm{A}) = \mathrm{Clstr}^A(P, Q)$.

Proof. The algorithm clearly gives: $\mathrm{EnumerateClusters}(\mathrm{CC}, A) \subset \mathrm{Clstr}^A(P, Q)$.

By Lemma 5.3.24, $\mathrm{EnumerateClusters}(\mathrm{CC}, A) = \mathrm{Clstr}^A(P, Q)$ whenever either has cardinal 1.

Then, by Lemma 5.3.23, we have:

$$\begin{aligned}
\mathrm{EnumerateClusters}(\mathrm{CC}, A) &= \bigcup_{A \subset A'} \mathrm{EnumerateClusters}(\mathrm{CC}, A') \\
&\supset \bigcup_{G \in \mathrm{Clstr}^A(P,Q)} \mathrm{EnumerateClusters}\left(\mathrm{CC}, \mathrm{Tgt}_G\right) \\
&= \bigcup_{G \in \mathrm{Clstr}^A(P,Q)} \mathrm{Clstr}^{\mathrm{Tgt}_G}(P, Q) \\
&= \mathrm{Clstr}^A(P, Q)
\end{aligned}$$

hence the result. □

We thus obtain an algorithm allowing us to enumerate all the clusters. This algorithm essentially "filters out" the connected components to try, and then tries every combination of connected components.

We would have preferred a condition, rather than an algorithm, but the following theorem will be our last attempt (in this thesis) of finding a CCCT:

Theorem 5.3.26 (Expensive CCCT)

Let $G : P \to Q$ be a $(1, 3, 4)$-protocluster.
$\mathrm{EnumerateClusters}(\mathrm{CC}, \varnothing) = \varnothing \Leftrightarrow G$ does not verify (CLU-2) and none of its fixes verifies (CLU-4).

5.3.5 More properties and an example

In this section, we give several results related to protoclusters and clusters. Some of these results will be useful in justifying later remarks or examples. I thought it would be too bad not to have them, because I spent some time proving them anyway.

Lemma 5.3.27

Let G be any protocluster.
There exists a $(3,4)$-protocluster G^* containing G, such that:

1. If G verifies (CLU-1) then G^* also does
2. If G does not verify (CLU-2) then G^* does not either

Proof. For $n \in \mathbb{N}$, consider:

$$\begin{aligned} G_0 &= G \\ G_{n+1} &= \{\, (p', g \circ P(a)) \in \mathscr{P} \times \mathscr{C} \mid\ g : P(p) \to Q(q) \in G_n, a : p' \to p \in \mathscr{P} \,\} \\ &\quad \cup \{\, (p, Q(b) \circ g) \in \mathscr{P} \times \mathscr{C} \mid\ g : P(p) \to Q(q) \in G_n, b : q \to q' \in \mathscr{Q} \,\} \end{aligned}$$

Then, $G^* = \bigcup_{n \in \mathbb{N}} G_n$ is a $(3,4)$-protocluster containing G.

The stated properties of G^* straightforwardly derive from the fact that, for all p, $G(p) \subset G^*(p)$. □

Lemma 5.3.28

If G is a cluster, then each $G(p)$ is a connected component of $(P(p) \mid Q)$.

Proof. This directly derives from Proposition 4.4.3. Let us give another proof.

Assume the contrary. Let p be such that $G(p)$ is not a connected component. Let $C \subset (P(p) \mid Q)$ be the connected component that strictly contains $G(p)$. Let $g : P(p) \to Q(q) \in C \backslash G(p)$, and define $G_0 = G + \{g\}$. By Lemma 5.3.27, let G_0^* be the $(3,4)$-protocluster that contains G_0. G_0^* verifies (CLU-1) because $G \subset G_0 \subset G_0^*$.

We have to check that G_0^* still verifies (CLU-2) despite having added arrows to verify (CLU-3) and (CLU-4).

Let $a : p' \to p \in \mathscr{P}$. There is a $g' : P(p) \to Q(q) \in G(p)$ such that $[g] = [g'] = C \supsetneq G(p)$. Then, $[g \circ P(a)] = [g' \circ P(a)] \supset G(p')$: it is still a unique connected component of $(P(p') \mid Q)$. Similarly, if $b : q \to q' \in \mathscr{Q}$, then $[g] = [Q(b) \circ g] \supset G(p)$ and it is also a unique connected component of $(P(p) \mid Q)$.

Then, $G \subsetneq G_0^*$, and G_0^* is a precluster by construction, which contradicts the maximality of G. □

The following lemma is a generalisation of Lemma 4.5.10:

Lemma 5.3.29

Let $G : P \to Q$ be a (1)-protocluster.
There is at most one cluster G^* containing G.

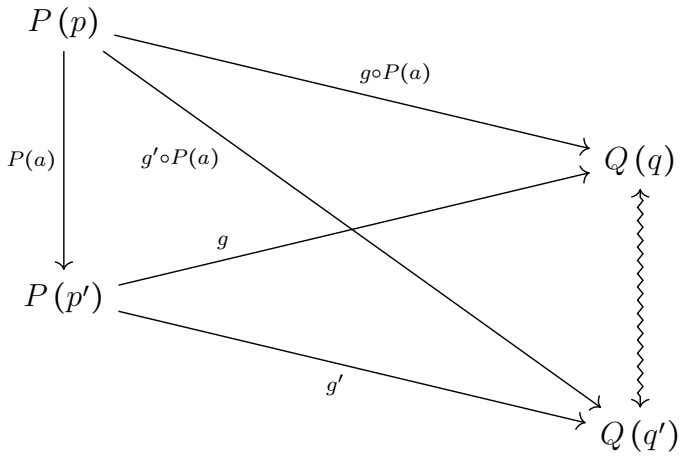
Proof. Let G_1^* and G_2^* be two clusters containing G. For each p, there is a $g : P(p) \to Q(q) \in G$ such that $g \in G_1^*(p)$ and $g \in G_2^*(p)$. By Lemma 5.3.28, $G_1^*(p)$ and $G_2^*(p)$ are connected components of $(P(p) \mid Q)$. Thus, they are equivalence classes, and they share one element, which means that $G_1^*(p) = G_2^*(p)$ and thus, $G_1^* = G_2^*$. □

A protocluster that does not verify (CLU-1), may be included in more than one cluster or no cluster at all. However:

Lemma 5.3.30

Let $G : P \to Q$ be a precluster.
There is exactly one cluster $G^* : P \to Q$ containing G.

Proof. For each $p \in \mathscr{P}$, consider the connected component C_p of $(P(p) \mid Q)$ such that $G(p) \subset C_p$ (C_p exists by (CLU-2)). Then, define $G^* = \sum_{p \in \mathscr{P}} C_p$ (as suggested by Fact 5.2.2). Defined as such, we have $G^*(p) = C_p$. We trivially have $G \subset G^*$. As G verifies (CLU-1) and (CLU-2), by construction, G^* also does. As G^* contains whole connected components, G^* trivially verifies (CLU-3). Now, if $a : p \to p' \in \mathscr{P}$ and $g : P(p') \to Q(q) \in G^*$, then $g \circ P(a)$ is an object of $(P(p) \mid Q)$. Besides, $g \in G^*$ is in $C_{p'}$. By definition of $C_{p'}$, there exists some $g' : P(p') \to Q(q') \in G(p') \subset C_{p'}$. As G is a precluster, $g' \circ P(a) \in G(p) \subset C_p$. By composition, $g \circ P(a)$ is in the same connected component as $g' \circ P(a)$, which is C_p, as shown in the diagram below:



(5.2)

Finally, for G^* contains the whole connected component, we have $g \circ P(a) \in G^*$, and thus G^* verifies (CLU-4).

G^* is trivially maximal by being a union of connected components, and it is unique by Lemma 5.3.29. □

Proposition 5.3.31

There is no precluster $P \to Q \Leftrightarrow$ there is no cluster $P \to Q$.

Proof. The sense $\Rightarrow$ is obvious, as a cluster is a precluster. The reciprocal derives from Lemma 5.3.30. □

Proposition 5.3.31 is an invitation to consider preclusters instead of clusters whenever useful. The constraint of maximality makes most proofs harder and trickier.

We finish the section with an example of usage of Theorem 5.3.5.

Example 5.3.32 (Clusters in **Sets**). Consider the category **Sets**, and two diagrams $P : \mathscr{P} \to \mathbf{Sets}$ and $Q : \mathscr{Q} \to \mathbf{Sets}$. As usual, denote by $F : P \to Q$ the full protocluster.

Sets has a terminal object $1 = \{0\}$, so there is a cocone $\alpha^1 : P \to \Delta(1)$. Then, if for some $q \in \mathscr{Q}$, $Q(q)$ is a non-empty set, the hom-set $\mathrm{Hom}_{\mathbf{Sets}}(1, Q(q))$ is non-empty. Let $u : 1 \to Q(q)$. The composite $\Delta(u) \circ \alpha^1 : P \to \Delta(Q(q))$ is a cocone from P to $Q(q)$.

This means that, for all p_0, not only is $F(p_0)$ non-empty, because it contains $u \circ \alpha^1_{p_0}$, but also, $\bigcap_{a : p_0 \to p} \mathrm{CC}(a)(\mathrm{Tgt}_F(p))$ contains at least the connected component of that arrow $u \circ \alpha^1_{p_0}$.

Finally, note that in **Sets**, there are always functions between any two sets $X \to Y$, except if X is non-empty and Y is empty. Thus, if there is a $q \in \mathscr{Q}$ such that $Q(q)$ has a non-empty set, there is always a cocone from P to that $Q(q)$. In particular, the full protocluster $P \to Q$ always verifies (CLU-1).

If P and Q are diagrams to **Sets** and at least one of the $Q(q)$ is a non-empty set, by Theorem 5.3.5, then there may always be clusters between P and Q.

5.4 The special case of preorders

To be fair, the CCCT, be it very weak (Theorem 5.3.5) or weak (Theorem 5.3.11) are most probably intractable in general categories, because it requires knowledge about the comma-category, which is generally not easy to grasp. In this section, we

study the special case of preorders and give a necessary (but not sufficient) condition that will simplify the use of the CCCT.

The next definition will be helpful:

Definition 5.4.1 (Protocluster subcategory)

Let $G : P \to Q$ be a protocluster between two diagrams P and Q in $\mathscr{C}$.
The *G-subcategory*, denoted by $\mathrm{Sub}_{\mathscr{C}}(G)$, is the subcategory of $\mathscr{C}$ defined by:

Objects: The objects are the $P(p)$'s and $Q(q)$'s for $p \in \mathscr{P}$ and $q \in \mathscr{Q}$

Morphisms: The arrows are the arrows of the form $P(a) : P(p) \to P(p')$ for $a : p \to p' \in \mathscr{P}$, of the form $Q(b) : Q(q) \to Q(q')$ for $b : q \to q' \in \mathscr{Q}$, of the form $g : P(p) \to Q(q)$ for $g \in G$, and every composite $g \circ P(a)$ and $Q(b) \circ g$ when the composition law of $\mathscr{C}$ allows it

Identities: The identities are $P(\mathrm{id}_p)$ and $Q(\mathrm{id}_q)$

Composition: The composition law is the composition law in $\mathscr{C}$

Let us add some more terminology.

If $P : \mathscr{P} \to \mathscr{C}$ is a diagram, we call *image of* P and denote by $\mathrm{Im}(P)$ the category consisting of objects of the form $P(p)$ for $p \in \mathscr{P}$, and of arrows $P(a)$ where $a : p \to p' \in \mathscr{P}$, and, if necessary, the missing compositions of arrows.

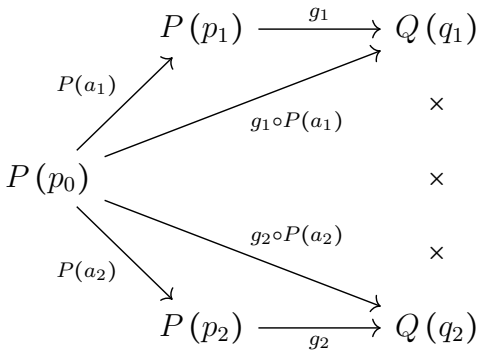
If G is a protocluster, then $\mathrm{Im}(P)$ and $\mathrm{Im}(Q)$ embed into $\mathrm{Sub}_{\mathscr{C}}(G)$.

The G-subcategory is simply the image of the diagrams P and Q (as subcategories of $\mathscr{C}$) together with the arrows of G between these diagrams. Thus, the G-subcategory may, or may not be a full subcategory of $\mathscr{C}$. Its intent is to merge the structure of $\mathscr{P}$ (domain of P), the structure of $\mathscr{Q}$ (domain of Q) and the protocluster.

Definition 5.4.2 (Clawitzer diagram)

Let $G : P \to Q$ be a protocluster.
A *Clawitzer*[2] *diagram*, if it exists, is the subcategory of $\mathrm{Sub}_{\mathscr{C}}(G)$ of the follow-

ing form:



We say that *the Clawitzer diagram includes into* G when:

1. g_1, $g_1 \circ P(a_1)$, g_2 and $g_2 \circ P(a_2)$ are in G
2. The two triangles commute
3. $g_1 \circ P(a_1)$ and $g_2 \circ P(a_2)$ are in different connected components of $(P(p_0) \mid Q)$
4. There is no arrow $P(p_1) \to Q(q_2)$ nor $P(p_2) \to Q(q_1)$ in $\mathrm{Sub}_{\mathscr{C}}(G)$

Clawitzer is a Pokémon inspired from a shrimp with a big, big, right-hand pincer.

The Clawitzer diagram is a first clue that a $(1,3,4)$-protocluster cannot be fixed, but it is not a proof. We will give an example below the following proposition:

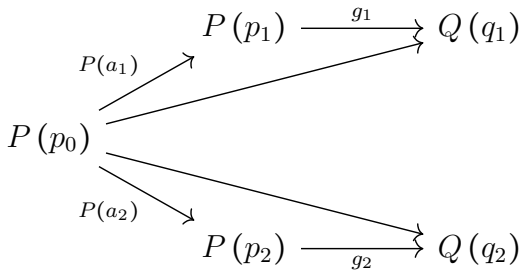
Proposition 5.4.3 (CCCT implies Clawitzer diagram)

Let G be a $(1,3,4)$-protocluster in a preorder $\mathscr{C}$.
There is a $p_0 \in \mathscr{P}$ such that $\bigcap_{a:p_0 \to p} \mathrm{CC}(a)(\mathrm{Tgt}_G(p)) = \varnothing \Rightarrow$ the Clawitzer diagram includes into G.

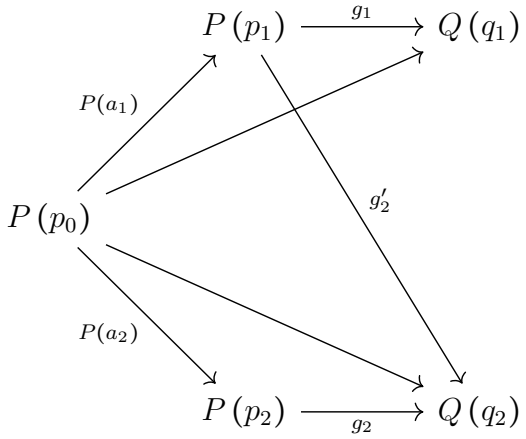
Proof. By Lemma 5.3.4, there are two distinct elements $C_1 \in \mathrm{CC}(a_1)(\mathrm{Tgt}_G(p_1))$ and $C_2 \in \mathrm{CC}(a_2)(\mathrm{Tgt}_G(p_2)) \setminus \mathrm{CC}(a_1)(\mathrm{Tgt}_G(p_1))$, for some arrows $a_1 : p_0 \to p_1$ and $a_2 : p_0 \to p_2$.

For $i = 1, 2$, C_i is in the image of $\mathrm{Tgt}_G(p_i)$ by $\mathrm{CC}(a_i)$, so there exists $g_i : P(p_i) \to$

$Q(q_i)$ such that C_i is of the form $C_i = [g_i \circ P(a_i)]$. We then have the diagram:

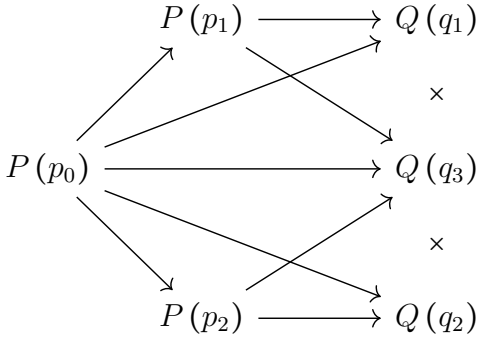


Assume there is an arrow $g_2' : P(p_1) \to Q(q_2)$:



In a preorder, as every diagram commutes, we would have $g_2' \circ P(a_1) = g_2 \circ P(a_2)$. We derive $C_2 = [g_2 \circ P(a_2)] = [g_2' \circ P(a_1)]$ and thus $C_2 \in \mathrm{CC}(a_1)(\mathrm{Tgt}_G(p_1))$, leading to a contradiction. The same reasoning proves that there is no arrow $P(p_2) \to Q(q_1)$, and thus, the Clawitzer diagram embeds into G. □

Example 5.4.4 (Example of the non-sufficiency of the Clawitzer diagram). Consider the following protocluster G:



There is a Clawitzer diagram included in G (consider the arrows $P(p_1) \to Q(q_1)$ and $P(p_2) \to Q(q_2)$). Assuming that every diagram commutes, the protocluster consisting of the arrows to $Q(q_3)$ is in fact a cluster.

Unfortunately, we could not find an equivalence using the Clawitzer diagram. We however have the following conjecture:

Conjecture 5.4.5 (Informal). $\bigcap_{a:p_0 \to p} \mathrm{CC}(a)(\mathrm{Tgt}_G(p)) = \varnothing \Leftrightarrow$ *there is a Clawitzer diagram "every time".*

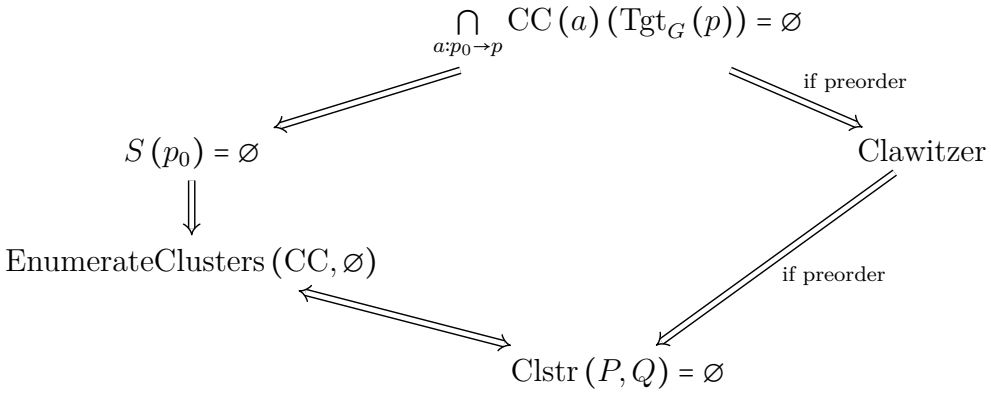
Our intuition is that the Clawitzer diagram should be the typical case stating the absence of clusters. The problem with that conjecture is that we could not determine what formal words could translate the expression "every time".

However, we infer that the Clawitzer diagram, even if it is "typical" in some sense, will probably never be more than a sufficient condition for the absence of clusters.

5.5 Conclusion

In this chapter, we proved a theorem helpful for our purpose. In fact, the absence of cluster can lead to a subcase of Multiplicity Principle, that we study in the next chapter. It is thus crucial to have a tool to (more or less) easily test the existence or absence of clusters between two given diagrams.

We combine the results of this chapter: the results from Sections 5.3.2, 5.3.3, 5.3.4 and 5.4, we have the following graph:



In the next chapter, we study the other subcase of Multiplicity Principle, this time in the subcase of preorders.

Chapter 6

Das Multiplizitätsprinzip in Quasiordnungen

Ne t'inquiète pas, ça va bien se passer. Bien s'passer. Ne t'inquiète pas.

Maître Yoda (201X?) *Maître Yoda CLASH Lorenzo BECKER (Menaces et Révélations)*, vidéo re-postée par Axel le 1er avril 2016, version originale introuvable.

6.1 Introduction

Edelman and Gally define degeneracy as "*the ability of elements that are structurally different to perform the same function or yield the same output*" [17]. Degeneracy is not the same as redundancy, in that redundancy involves identical elements while degeneracy involves structurally different elements. The term might read misleading while it suggests the idea of a certain robustness of the system. This is because one biological element of the system may replace another one that was more appropriate for the concerned function. The replacement is not supposed to be perfect though, leading to a function somewhat accomplished, but not as well as with the appropriate element.

This ambiguity disappears in the categorical translation of degeneracy. In order to provide a categorical framework for describing systems, and in particular biological systems and emergence, Ehresmann and Vanbremeersch incorporated degeneracy in their theory of Memory Evolutive Systems [12, 45]. In this work, a system becomes a category (or rather, a family of categories indexed by time), a subsystem becomes a diagram in that category (in that family of categories), and degeneracy is called the Multiplicity Principle. The Multiplicity Principle (MP) becomes the property of a category to have two (or more) diagrams that have the same cocones (a cocone being seen as a functionality) while being structurally different in a certain sense that calls on the tricky notion of cluster. Besides, the MP is presented as the

backbone of the notion of complexity, which highlights its importance in the theory of Memory Evolutive Systems.

The MP appeared in [20] to account for a specific relation between two classes of statistical tests. Namely, Neyman-Pearson tests [46] and Random Distorsion Tests [47] are shown to share the same functionality (they both tend to perfect tests, called oracles) while being structurally different in a certain sense. The preorder representing tests thus verifies the MP due to these two classes of tests.

The goal of [20] was to find an instance of MP in a preorder, but not study the MP as such. Thus, a sufficient condition for a preorder to verify MP was enough for its purpose.

In the present work, we study the Multiplicity Principle in the special case of thin categories (preorders). Section 6.2 introduces the categorical background originally used to describe the MP in general categories. We introduce first the notions surrounding that of clusters, and the formal notion of MP. We then study the occurrence of MP in categories. In Chapter 5, we give sufficient conditions for two diagrams not to have any clusters from one to the other. It turns out that these conditions concern all categories, not only thin ones. However, the case when two diagrams have clusters between them needs to then be restricted to preorders, leading to two other necessary and sufficient conditions, and finally to the main theorem of this chapter (Section 6.3). We then put the theorem into practice in a few examples of orders, in Section 6.4.

6.2 Background

Let us recall the definition of the notions needed by the Multiplicity Principle.

In the following, we consider a (locally-small) category $\mathscr{C}$ and two diagrams (or patterns) $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$. The definitions are adapted from [12].

Definition 6.2.1 (Homologous diagrams)

Two diagrams P and Q are *(lax-)homologous* when their categories of cocones are isomorphic: $\mathbf{Cocones}(P) \cong \mathbf{Cocones}(Q)$. P and Q are *strictly homologous* when there is an isomorphism $I : \mathbf{Cocones}(P) \to \mathbf{Cocones}(Q)$ that preserves peaks: for all $\alpha \in \mathbf{Cocones}(P)$, $\mathrm{peak}(I(\alpha)) = \mathrm{peak}(\alpha)$.

The notion of strictly homologous diagrams is defined in [12]. That of lax homologous diagrams is neither novel nor difficult, but this terminology will prove useful.

Considering that colimits are the initial elements of the categories of cocones

(Proposition 2.4.14), we easily derive:

Proposition 6.2.2

The following holds:

1. If two diagrams are strictly homologous then they are lax-homologous
2. If two diagrams are lax-homologous and one has a colimit, then they both have a colimit whose peaks need not be isomorphic
3. If two diagrams are strictly homologous and one has a colimit, then they both have the same colimit, with the same peak

If diagrams represent subsystems of the whole system (the whole category), then a colimit represents the "synthetic information" of the diagram. In the case of biological systems, a diagram represents the cells of an organ, and the colimit represent their joint action, that is, the organ itself. Then, the diagram can be seen as a "decomposition", a "detailed view" of the organ.

Again in the study of systems, a cluster between two diagrams represents the interaction between the corresponding two subsystems. In this chapter, we focus on the characterisation by Proposition 4.4.2, that is, the definition of clusters with five conditions (maximal precluster). See Chapter 4 for other definitions.

Definition 6.2.3 (Composition of a cluster with a cocone)

Let $G : P \to Q$ be a cluster and let $\alpha : Q \to \Delta(A)$ be a cocone from Q (the codomain of the cluster).
The *composition of* α *with* G, denoted by $\alpha \circ G$, is the cocone $\alpha \circ G : P \to \Delta(A)$ consisting of all arrows $\alpha_q \circ g$ such that $g : P(p) \to Q(q) \in G$ and $\alpha_q : Q(q) \to A$.

Note that this definition is a special case of the composition of clusters (Figure 6.1). In fact, a cocone is exactly a cluster between a diagram and some diagram of the form $\Delta(A)$. Also note that the result of the composition of a cluster with a cocone is also a cocone.

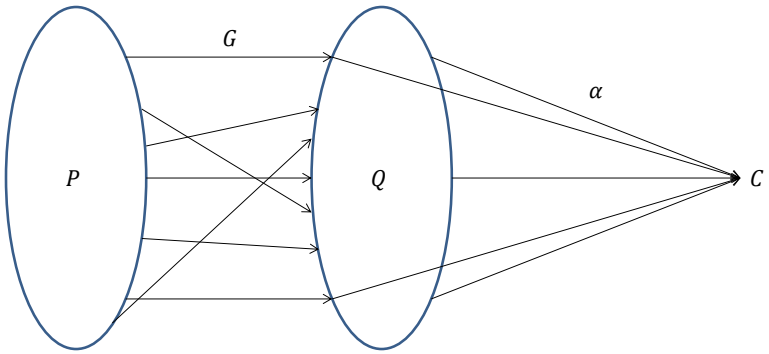


Figure 6.1 – Note that such a composition yields a cocone: a cocone $Q \to \Delta(C)$ composed with a cluster $P \to Q$ yields a cocone $P \to \Delta(C)$.

Consider now two diagrams P and Q with colimits $\mathrm{Colim}(P)$ and $\mathrm{Colim}(Q)$, respectively with cocones $\alpha_P : P \to \Delta(\mathrm{Colim}(P))$ and $\alpha_Q : Q \to \Delta(\mathrm{Colim}(Q))$. Also assume that there is a cluster $P \to Q$. The composite $\alpha_Q \circ G$ is a cluster between P and the diagonal functor $\Delta(\mathrm{Colim}(Q))$; equivalently, it is a cocone from P to $\mathrm{Colim}(Q)$ (cf Figure 6.1). Thus, there is a unique arrow $g : \mathrm{Colim}(P) \to \mathrm{Colim}(Q)$ due to the UMP of $\mathrm{Colim}(P)$. Hence the following notion:

Definition 6.2.4 (Binding of a cluster)

Let $G : P \to Q$ be a cluster between two diagrams P and Q that both have a colimit (hereafter denoted by $\mathrm{Colim}(P)$ and $\mathrm{Colim}(Q)$). Denote by $\alpha_P : P \to \Delta(\mathrm{Colim}(P))$ and $\alpha_Q : Q \to \Delta(\mathrm{Colim}(Q))$ the cocones from the diagrams to their respective colimits.
The *binding of* G is the unique arrow $g : \mathrm{Colim}(P) \to \mathrm{Colim}(Q) \in \mathscr{C}$ such that $\Delta(g) \circ \alpha_P = \alpha_Q \circ G$; that is, the following square commutes (in $\mathrm{Clstr}(\mathscr{C})$):

$$\begin{array}{ccc} \Delta(\mathrm{Colim}(P)) & \xrightarrow{\Delta(g)} & \Delta(\mathrm{Colim}(Q)) \\ \uparrow{\scriptstyle \alpha_P} & \checkmark & \uparrow{\scriptstyle \alpha_Q} \\ P & \xrightarrow[G]{} & Q \end{array}$$

We equivalently say that G *binds to* g. We say that G *binds to an isomorphism* when g is an isomorphism.

With the terminology introduced just above, whenever two diagrams have a

colimit, if there is a cluster between those two diagrams, then the binding of the cluster always exists.

Remark 6.2.5. Note that $\Delta(g)$ is a cluster $\Delta(\mathrm{Colim}(P)) \to \Delta(\mathrm{Colim}(Q))$, so the previous diagram really is a diagram in $\mathrm{Clstr}(\mathscr{C})$.

The following functor is introduced in [12, discussion below definition, Chapter 3, section 1.4, page 79] but is left unnamed.

Definition 6.2.6 (Cluster-composition functor)

Let $G : P \to Q$ be a cluster. We define the *cluster-composition functor* ΩG as:

$$\Omega G : \begin{cases} \mathbf{Cocones}(Q) & \longrightarrow & \mathbf{Cocones}(P) \\ \alpha & \longmapsto & \alpha \circ G \\ u & \longmapsto & u \end{cases}$$

Definition 6.2.7 (Connected diagrams [12, Definition, Chapter 3, section 4.1, page 90])

Two diagrams P and Q are connected when there is a cluster $G : P \to Q$ such that ΩG is an isomorphism.

As we will see later, ΩG may be an isomorphism even if the cluster itself is not invertible (invertible as an arrow in the strict cocompletion of $\mathscr{C}$.

It is easy to see that:

Lemma 6.2.8

If two diagrams are connected, then they are lax-homologous.

Proof. If two diagrams are connected, then there is a cluster G such that ΩG is an isomorphism between their cocone categories. □

The Multiplicity Principle is exactly the complement of this lemma, in the sense described in Figure 6.2.

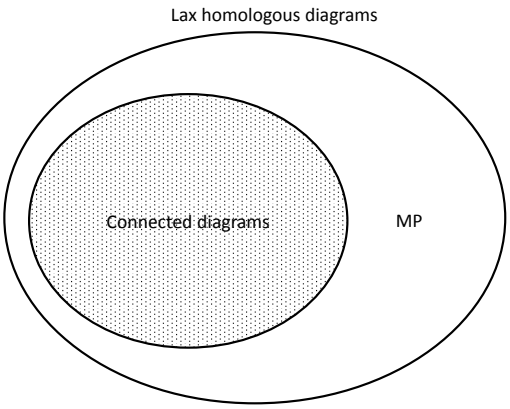


Figure 6.2 – The Multiplicity Principle happens exactly when two diagrams are lax homologous but not connected. Thus, the MP is the complement of Lemma 6.2.8.

Definition 6.2.9 (Multiplicity Principle)

A category $\mathscr{C}$ verifies the *lax (resp. strict) Multiplicity Principle* (MP for short) when there are two diagrams P and Q that are lax-homologous (resp. strictly homologous) but not connected.
If P and Q are homologous but not connected, we say, by misuse of language, that *P and Q verify the MP*.

In other words: $\mathscr{C}$ verifies the Multiplicity Principle when there are two diagrams P and Q whose cocone categories are isomorphic, but such that for all cluster $G : P \to Q$, the composition of cocones by G does not define an isomorphism (there may also be no cluster at all). One may also say: the cocone categories are isomorphic, but ΩG is not the witnessing isomorphism.

The Multiplicity Principle was first introduced in [44] in its strict form, and only for diagrams with the same colimit. The cluster-composition functor was first introduced in [12, Chapter 3, section 4.3, page 91] in order to generalise the original MP to all diagrams. Thus, the MP introduced in [12, Chapter 3, section 4.3, page 92] corresponds to our strict MP. Also, although the MP introduced in [20] was meant to be lax, it resulted strict in the preorder of tests. In the present work, we focus on the lax MP, so by default, we will drop the "lax" mention.

The following proposition is necessary for the rest of this chapter, and it makes things clearer.

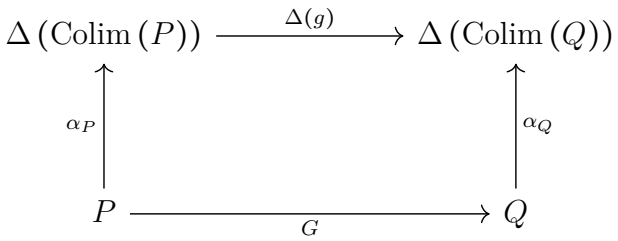
Proposition 6.2.10

Let $G : P \to Q$ be a cluster and assume that P and Q have colimits $\mathrm{Colim}(P)$ and $\mathrm{Colim}(Q)$.
Then, ΩG is an isomorphism $\Leftrightarrow$ G binds to an isomorphism.

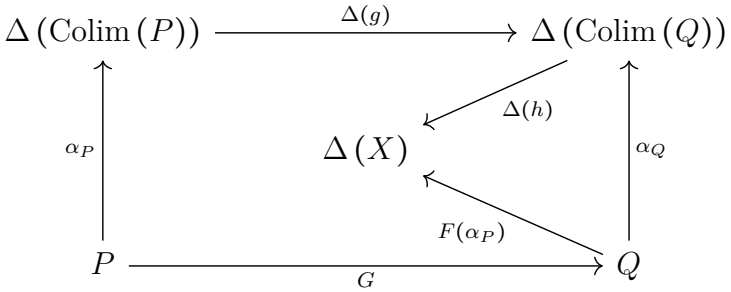
This proposition is probably not new, although it is not found in [12], on which we base this work. We give here a proof.

Proof. [**Proof of** $\Rightarrow$] Suppose ΩG is an isomorphism. Let $g : \mathrm{Colim}(P) \to \mathrm{Colim}(Q)$ be the binding of G; we want to show that g is an isomorphism. Denote by $\alpha_P : P \to \Delta(\mathrm{Colim}(P))$ (resp. $\alpha_Q : Q \to \Delta(\mathrm{Colim}(Q))$) the cocone from P (resp. Q) to its colimit $\mathrm{Colim}(P)$ (resp. $\mathrm{Colim}(Q)$). Denote by $F : \mathbf{Cocones}(P) \to \mathbf{Cocones}(Q)$ the inverse of ΩG.

By definition, we have $\Omega G(\alpha_Q) = \alpha_Q \circ G$. According to Definition 6.2.4, the following diagram commutes:



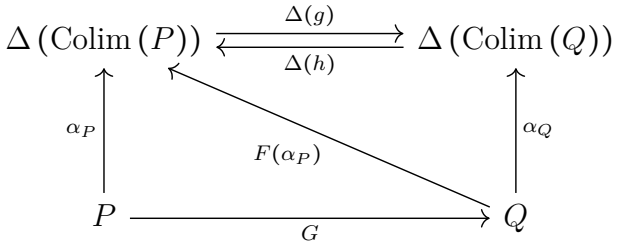
Consider $F(\alpha_P)$. It is a cocone from Q to some object $X = \mathrm{peak}(F(\alpha_P))$. We do not have the expression of F, but using the UMP of $\mathrm{Colim}(Q)$, there exists a similar arrow $h : \mathrm{Colim}(Q) \to X$, such that the following diagram commutes:



We are going to find this X.

Note that the functor ΩG "preserves the peaks": for a cocone $\beta \in \mathbf{Cocones}(Q)$, $\mathrm{peak}(\Omega G(\beta)) = \mathrm{peak}(\beta)$. So, for a cocone $\alpha \in \mathbf{Cocones}(P)$, $\mathrm{peak}(\Omega G(F(\alpha))) = \mathrm{peak}(F(\alpha))$. But also, $\Omega G \circ F(\alpha) = \alpha$, so $\mathrm{peak}(F(\alpha)) = \mathrm{peak}(\alpha)$ (which means that F also "preserves peaks"). In particular, $X = \mathrm{peak}(F(\alpha_P)) = \mathrm{peak}(\alpha_P) = \mathrm{Colim}(P)$

and $h \in \mathrm{Hom}_{\mathscr{C}}(\mathrm{Colim}(Q), \mathrm{Colim}(P))$. The previous diagram thus reduces to:



By unicity of the arrow from the UMP, and using the fact that all triangles in the above diagram commute, we have: $g \circ h = \mathrm{id}_{\mathrm{Colim}(Q)}$ and $h \circ g = \mathrm{id}_{\mathrm{Colim}(P)}$. Thus, the binding of cluster G to g is actually an isomorphism.

[**Proof of** $\Leftarrow$] Suppose G binds to an isomorphism. The existence of a cluster entails the existence of functor ΩG. We have to prove that it is an isomorphism.

Denote by $g : \mathrm{Colim}(P) \to \mathrm{Colim}(Q)$ the binding of G. For now, the following square commutes:

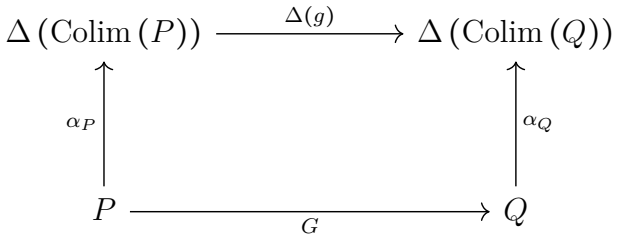
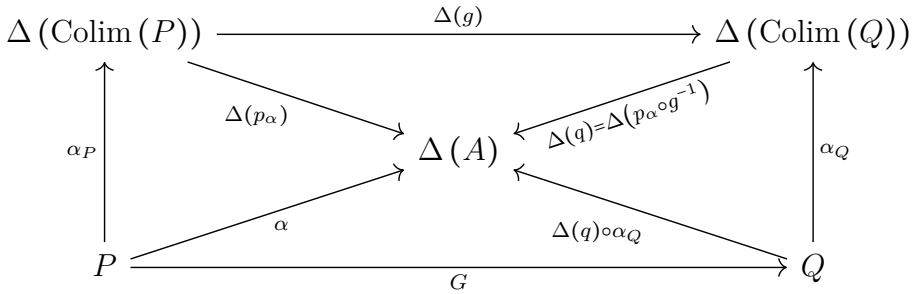


Diagram (6.1) below is explained by the following construction:



(6.1)

Let $\alpha : P \to \Delta(A)$ be a cocone to P. There is a unique arrow $p_\alpha : \mathrm{Colim}(P) \to A$ due to the UMP of $\mathrm{Colim}(P)$, such that $\Delta(p_\alpha) \circ \alpha_P = \alpha$. As $\mathrm{Colim}(Q) \cong \mathrm{Colim}(P)$. This induces the arrow $q = p_\alpha \circ g^{-1} : \mathrm{Colim}(Q) \to A$. Then, $\Delta(q) \circ \alpha_Q : Q \to \Delta(A)$ is a cocone from Q.

This gives a pyramid with top $\Delta(A)$. In Diagram (6.1), the left-hand, upper and

right-hand triangles commute, as well as the base square. Therefore, the last, lower triangle commutes too.

Thus:

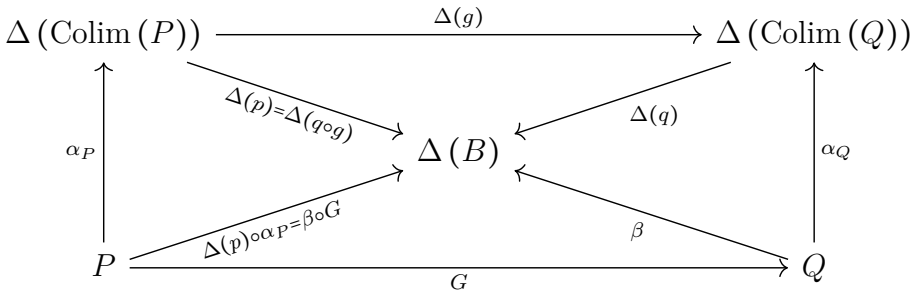
$$\begin{aligned}\alpha &= \Delta(q) \circ \alpha_Q \circ G \\ &= \Delta\left(p_\alpha \circ g^{-1}\right) \circ \alpha_Q \circ G \\ &= \Omega G\left(\Delta\left(p_\alpha \circ g^{-1}\right) \circ \alpha_Q\right)\end{aligned}$$

Define F to be:

$$F : \begin{cases} \mathbf{Cocones}(P) & \longrightarrow & \mathbf{Cocones}(Q) \\ \alpha & \longmapsto & \Delta\left(p_\alpha \circ g^{-1}\right) \circ \alpha_Q \\ u & \longmapsto & u \end{cases}$$

where p_α is the unique arrow $p_\alpha : \mathrm{Colim}(P) \to A$ due to the UMP of $\mathrm{Colim}(P)$.

We already have $\Omega G \circ F = \mathrm{Id}_{\mathbf{Cocones}(P)}$. We now want to prove $F \circ \Omega G = \mathrm{Id}_{\mathbf{Cocones}(Q)}$. The reader may refer to the following diagram in order to follow the proof:



Let $\beta \in \mathbf{Cocones}(Q)$ where $\beta : Q \to \Delta(B)$. There is a unique arrow $q : \mathrm{Colim}(Q) \to B$ (by UMP) such that $\Delta(q) \circ \alpha_Q = \beta$. Also, this q gives rise to $p = q \circ g$. Thus:

$$\begin{aligned} p &= q \circ g \\ \Delta(p) \circ \alpha_P &= \Delta(q \circ g) \circ \alpha_P \\ &= \Delta(q) \circ \Delta(g) \circ \alpha_P \\ &= \Delta(q) \circ \alpha_Q \circ G \\ &= \beta \circ G \\ &= \Omega G(\beta) \end{aligned}$$

The last but one equation also brings that $p = p_{\beta \circ G}$ (where $p_{\beta \circ G}$ is p_α for $\alpha = \beta \circ G$, cf. above; that is, p is the unique arrow from the UMP of $\mathrm{Colim}(P)$ to $\beta \circ G$).

In summary:

$$\begin{aligned} F \circ \Omega G(\beta) &= F(\beta \circ G) \\ &= \Delta\left(p_{\beta \circ G} \circ g^{-1}\right) \circ \alpha_Q \\ &= \Delta\left(p \circ g^{-1}\right) \circ \alpha_Q \\ &= \Delta\left(q \circ g \circ g^{-1}\right) \circ \alpha_Q \\ &= \Delta(q) \circ \alpha_Q \\ &= \beta \end{aligned}$$

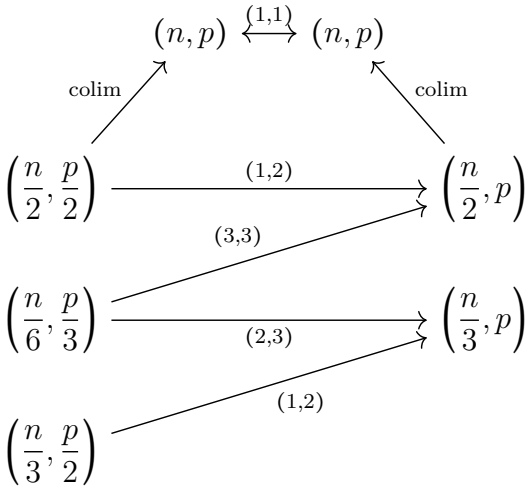
Thus, F is the inverse of ΩG and ΩG is an isomorphism. □

Note that P and Q need to have colimits for this theorem to hold (also note that, under MP, if one has a colimit, the other also does). Otherwise, the notion of binding does not have any sense.

Also note that, if a cluster G binds to an isomorphism, or more generally, if ΩG is an isomorphism, then it says nothing about the invertibility of G in the cluster category. The functor ΩG may be an isomorphism, independently on whether G is, or is not invertible (of course, if G is invertible, then ΩG is an isomorphism).

Example 6.2.11. Consider the preorder $\mathbb{Z} \times \mathbb{Z}$ such that $(m_1, m_2) \leqslant (n_1, n_2)$ iff m_1 divides n_1 and m_2 divides n_2. Then, consider the following two diagrams, together

with their colimits:



Clearly, the arrow between the colimits is the identity, and thus it is an isomorphism. Also clearly, there are two clusters between those two diagrams: the cluster $\{(1,2),(1,2),(3,3)\}$ and the cluster $\{(1,2),(1,2),(2,3)\}$. They both bind to an isomorphism. However, the clusters are clearly not invertible here, because there are no arrow in the category $\mathbb{Z}\times\mathbb{Z}$ from the right-hand diagram to the left-hand diagram.

6.3 A characterisation of preorders with Multiplicity

In this section, we study the structural properties of preorders that give rise to MP. This has been started in [20]. As proven in this section, Proposition 1 in [20, Section 2.2, page 4] is just a particular case of MP in a preorder. However, no equivalence, and no characterisation was given. This section will fulfill this wish.

The goal is to get rid of the tricky notion of cluster in order to define the Multiplicity Principle in the special case of preorders. This characterisation turns the MP in preorders into some structural properties that may, or may not, hold in a preorder.

Note that the categories we consider may be small or large, in which case we should call them *thin categories*, and not *preorders*. The "set-ness" (the size) of preorders does not intervene in the following, so we allow the use an inaccurracy. However, when we mention diagrams, we always mean *small* diagrams.

We enter now the realm of preorders. In this wonderful and peaceful world, categories become simpler, and the following applies:

1. A full diagram is a sub-preorder of the whole preorder
2. Colimits are the same as coproducts (if either exists)
3. The colimit of a diagram is the supremum of its objects (if it exists)
4. Every diagram commutes
5. The peak of a cocone from a diagram is an upper bound of the set of objects of this diagram (if it exists)
6. For a given upper bound, there can be only one cocone from a given diagram to that upper bound
7. Two diagrams P and Q are homologous when their sets of upper bounds are isomorphic

Given a preorder $\mathscr{C}$ and two diagrams $P : \mathscr{P} \to \mathscr{C}$ and $Q : \mathscr{Q} \to \mathscr{C}$ (seen as subsets of the preorder in the following), we want to analyse when those two diagrams verify the MP and when they do not. Suppose P and Q verify the MP. We want to know what that means for P and Q.

They verify the lax MP when they are homologous (their upper bounds are isomorphic), and the two diagrams have no cluster G such that ΩG is an isomorphism (Definition 6.2.6). Being homologous does not require any categorical notion, so we do not need a characterisation of that. We are interested in converting the condition about clusters without categorical notions. There is no cluster G such that ΩG is an isomorphism when (1) there is no cluster G at all, or when (2) there are clusters G but ΩG is never an isomorphism. This gives us two branches to climb, the first of which we have already climbed in the preceding chapter.

Unfortunately, the Expensive CCCT (Theorem 5.3.26) does not seem to have a specific behaviour in preorders. Thus, the non-existence of clusters, in preorders, simply becomes:

Theorem 6.3.1

Let $\mathscr{C}$ be a preorder. Let P and Q be two (small) diagrams to $\mathscr{C}$, and consider their full protocluster $F : P \to Q$.
There is no cluster $P \to Q \Leftrightarrow$ either there is a p such that $\mathrm{Upper}\,(P(p)) \cap \mathrm{Im}\,(Q) = \varnothing$, or $\mathrm{EnumerateClusters}\,(\mathrm{CC}_{P,Q}, \varnothing)$.

In the sequel, we always assume that the two diagrams P and Q are homologous (isomorphic upper bounds). Under the hypothesis that P and Q have clusters, we analyse what it means for them to be not connected, meaning that for all cluster G, ΩG is not an isomorphism.

Note that, under MP, either P and Q have sup, or none of them do (Proposition 6.2.2). This means that the exclusive subcases are: either both have colimits,

or both do not; under MP, there cannot be one diagram having a colimit while the other does not.

We start with the case when both diagrams have a colimit.

Proposition 6.3.2 (Isomorphic-colimits condition)

Let G be a cluster $P \to Q$ in a preorder category. Assume P and Q have a sup.
ΩG is not an isomorphism $\Leftrightarrow \sup P \ncong \sup Q$.

Proof. If P and Q have each one a sup, then Proposition 6.2.10 states that ΩG is not an isomorphism iff G does not bind to an isomorphism. The binding of a cluster always exists when P and Q have a sup (cf. the remark introducing Definition 6.2.4). Therefore, G does not bind to an isomorphism iff the unique arrow $\mathrm{Colim}\,(P) \to \mathrm{Colim}\,(Q)$ is not invertible. In a preorder, there is at most one arrow $\mathrm{Colim}\,(P) \to \mathrm{Colim}\,(Q)$, and that unique arrow is not an isomorphism. □

We now consider the case when P and Q do not have a sup, but still have a cluster $P \to Q$. P and Q are still assumed homologous.

We begin with this very helpful lemma:

Lemma 6.3.3

Let $\mathscr{C}$ be a preorder category, and let $G : P \to Q$ be a cluster.
ΩG is an isomorphism $\Leftrightarrow$ ΩG is a bijection on objects.

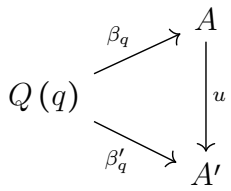
Proof. The direct sense is obvious.

Suppose ΩG defines a bijection between the objects of $\mathbf{Cocones}\,(P)$ and those of $\mathbf{Cocones}\,(Q)$.

Let $u : \alpha \to \alpha' \in \mathbf{Cocones}\,(P)$, where α and α' are cocones from P to A and A', respectively. Thus, u is also an arrow $u : A \to A'$ in $\mathscr{C}$.

As ΩG defines a bijection on objects, there are unique β and $\beta' \in \mathbf{Cocones}\,(Q)$ such that $\alpha = \beta \circ G$ and $\alpha' = \beta' \circ G$. Thus, u is an arrow $u : \beta \circ G \to \beta' \circ G \in \mathbf{Cocones}\,(P)$. We have to check whether u is also an arrow $u : \beta \to \beta' \in \mathbf{Cocones}\,(Q)$. Of course, β and β' are cocones from Q to A and A' respectively, so the question amounts to finding whether $u \circ \beta = \beta'$, or equivalently, whether for all

$q \in \mathscr{Q}$, the following triangle commutes:



In a preorder category, every diagram commutes. So this diagram commutes, and thus u is also in $\mathbf{Cocones}(Q)$. It follows that F defined as:

$$F : \begin{cases} \mathbf{Cocones}(P) & \longrightarrow & \mathbf{Cocones}(Q) \\ \alpha & \longmapsto & (\Omega G)^{-1}(\alpha) \\ u : \alpha \to \alpha' & \longmapsto & u : (\Omega G)^{-1}(\alpha) \to (\Omega G)^{-1}(\alpha') \end{cases}$$

is the inverse of ΩG, which concludes the proof. □

Remark 6.3.4. Lemma 6.3.3 implies that, in a preorder, the lax MP is a strict MP.

Let us add something more to the previous lemma. In a preorder, there can be only one cocone from a diagram to a given object / peak. For example, if $\alpha : P \to \Delta(A)$, then there is no other $\alpha' : P \to \Delta(A)$ such that $\alpha \neq \alpha'$. So, instead of refering to cocones, we may just refer to the peak of these cocones, which are upper bounds (and we identify $\mathbf{Cocones}(P)$ with the full subcategory of $\mathscr{C}$ consisting of all the upper bounds of P). Then, by the previous lemma, we may also ignore the relations (i.e. arrows) between those upper bounds and just consider sets of them.

In the following, we denote by $\mathrm{Upper}(P)$ the class of the peaks of the objects of $\mathbf{Cocones}(P)$. If $\mathbf{Cocones}(P)$ is seen as the sub-preorder of $\mathscr{C}$ consisting of all the (cocones to) upper bounds of the diagram P, then $\mathrm{Upper}(P)$ is simply the class (or set) of upper bounds, without the inner relations (arrows) between them. Note that ΩG then becomes a bijection $\mathrm{Upper}(Q) \to \mathrm{Upper}(P)$. The reduction of $\mathbf{Cocones}(P)$ and $\mathbf{Cocones}(Q)$ to $\mathrm{Upper}(P)$ and $\mathrm{Upper}(Q)$ also implies that the sets $\mathrm{Upper}(P)$ and $\mathrm{Upper}(Q)$ may contain the same elements.

Proposition 6.3.5 (Same-upper-bounds condition)

Let G be a cluster $P \to Q$ in a preorder category. Assume P and Q do not have a sup.
ΩG is not an isomorphism $\Leftrightarrow \mathrm{Upper}(P) \neq \mathrm{Upper}(Q)$.

Proof. Suppose ΩG is an isomorphism $\mathbf{Cocones}(Q) \to \mathbf{Cocones}(P)$, or equivalently, a bijection $\mathrm{Upper}(Q) \to \mathrm{Upper}(P)$ (Lemma 6.3.3). Then, $\Omega G(B) = B$ (the

composition by a cluster does not change the peak of the cocone, cf. Definition 6.2.3), and $\Omega G^{-1}(A) = A$. So, for all $A \in \mathrm{Upper}(P)$, we have $A \in \mathrm{Upper}(Q)$ and for all $B \in \mathrm{Upper}(Q)$, we have $B \in \mathrm{Upper}(P)$. This means that the upper bounds are not only isomorphic, but also equal: $\mathrm{Upper}(P) = \mathrm{Upper}(Q)$.

Conversely, if $\mathrm{Upper}(P) = \mathrm{Upper}(Q)$, then for all $B \in \mathrm{Upper}(Q)$, we have $\Omega G(B) = B$, so ΩG is injective. And, for all $A \in \mathrm{Upper}(P)$, we have $A \in \mathrm{Upper}(Q)$, so $\Omega G(A) = A$ as well; and that means that ΩG is also surjective, so ΩG is bijective, and an isomorphism by Lemma 6.3.3. □

Even if we proved that whenever P and Q do not have a sup, then ΩG is an isomorphism $\Leftrightarrow \mathrm{Upper}(P) = \mathrm{Upper}(Q)$, we prefer to state the lemma in the form used here. In fact, we are looking for necessary and sufficient conditions for ΩG not to be an isomorphism.

Compiling the previous propositions with Theorem 5.3.26, we finally obtain the wanted characterisation.

Theorem 6.3.6 (Characterisation of MP in preorders)

Let P and Q be two diagrams in a preorder category.
P and Q verify MP iff $\mathrm{Upper}(Q) \cong \mathrm{Upper}(P)$ and exactly one of the following occurs:

1. Both full protoclusters $F : P \to Q$ and $F' : Q \to P$ verify at least (MP-1) or (MP-2) below [1]:

   (MP-1) (Incomparable) there is a $P(p)$ that has no upper bound in $\mathrm{Im}(Q)$ or there is a $Q(q)$ that has no upper bound in $\mathrm{Im}(P)$

   (MP-2) (Compatible connected component) The algorithms $\mathrm{EnumerateClusters}(\mathrm{CC}_{P,Q}, \varnothing)$ or $\mathrm{EnumerateClusters}(\mathrm{CC}_{Q,P}, \varnothing)$ yield an empty set

2. Or, at least one of the following holds:

   (MP-3) (Same colimits) if P and Q have sups, then $\sup P \ncong \sup Q$

   (MP-4) (Same upper bounds) if P and Q have no sup, then $\mathrm{Upper}(P) \neq \mathrm{Upper}(Q)$

[1] What this means is the following: F should verify at least one between (MP-1) or (MP-2), and F' should verify at least one between (MP-1) or (MP-2). Note that F and F' do not need to verify the same condition.

Proof. (MP-1) and (MP-2) correspond to Theorem 5.3.26; (MP-3) corresponds to Proposition 6.3.2 and (MP-4) corresponds to Proposition 6.3.5. □

(MP-1) and (MP-2) translate the fact that there is no cluster between P and Q. We can tell there is no cluster at all by analysing the full protocluster and trying to construct cluster from it (Theorem 5.3.26). (MP-3) and (MP-4) translate the fact that no cluster G is such that ΩG is an isomorphism between cocones.

So, a preorder verifies MP when there are two subsets of objects P and Q, such that their upper bounds are isomorphic and, depending on the case, one condition in the table in Figure 6.3.

	If P and Q have colimits	If P and Q have no colimit
If no cluster (ΩG does not exist)	No-upper-bounds condition or compatible connected components condition	No-upper-bounds condition or compatible connected components condition
If clusters (ΩG exists)	The colimits are not isomorphic	The sets of upper bounds are isomorphic but different

Figure 6.3 – A summary of the conditions on P and Q for them to verify MP or not. First, they need to have isomorphic upper bounds. Then, depending on the case (whether P and Q have colimits or not, and whether there are clusters between them), they need to verify the condition in the corresponding cell.

As remarked at the end of Section 5.3.4, the algorithm describing the enumeration of clusters requires heavy calculation. In most practical cases, it is probably sufficient to find a weaker condition, based on Theorem 5.3.5 or Theorem 5.3.11. In the special case of preorders, we will even use the Clawitzer diagram (Proposition 5.4.3), which is the typical case for the absence of clusters:

Theorem 6.3.7 (Simpler characterisation of MP in preorders)

Let P and Q be two diagrams in a preorder category.
If $\mathrm{Upper}(Q) \cong \mathrm{Upper}(P)$ and exactly one of the following occurs:

1. Both full protoclusters $F : P \to Q$ and $F' : Q \to P$ verify at least (MP-1) or (sMP-2) below [2]:

   (MP-1) (Incomparable) there is a $P(p)$ that has no upper bound in $\mathrm{Im}(Q)$ or there is a $Q(q)$ that has no upper bound in $\mathrm{Im}(P)$

   (sMP-2) (Clawitzer condition) The Clawitzer diagram embeds into F or the Clawitzer diagram embeds into F'

2. Or, at least one of the following holds:

   (MP-3) (Same colimits) if P and Q have sups, then $\sup P \not\cong \sup Q$

   (MP-4) (Same upper bounds) if P and Q have no sup, then $\mathrm{Upper}(P) \neq \mathrm{Upper}(Q)$

then P and Q verify MP.

As in Theorem 6.3.6, what this means is the following: F should verify at least one between (MP-1) or (sMP-2), and F' should verify at least one between (MP-1) or (sMP-2). Note that F and F' do not need to verify the same condition.

A word on strict MP Now that we have a characterisation for the lax MP, let us consider the case of the strict MP. For preorders, two patterns are strictly homologous when the isomorphism between their cocone categories preserves peaks.

Lemma 6.3.8

Let P and Q be diagrams to a preorder $\mathscr{C}$.
If P and Q verify the strict MP, then there is no cluster $P \to Q$ or $Q \to P$.

Proof. This is direct application of Theorem 6.3.6.

If P and Q verify the strict MP, then they verify the lax MP. Suppose there are clusters between P and Q. By Theorem 6.3.6, at least one between (MP-3) and (MP-4) holds. (MP-3) states that the colimits should be non-isomorphic, which cannot happen if the diagrams are strictly homologous. (MP-4) that the cocones should be isomorphic but different, which cannot happen either if the diagrams are strictly homologous.

In conclusion, assuming there are clusters between P and Q leads to a contradiction; thus, there cannot be clusters between P and Q. □

It should also be noted that this example is in fact an instance of strict-MP.

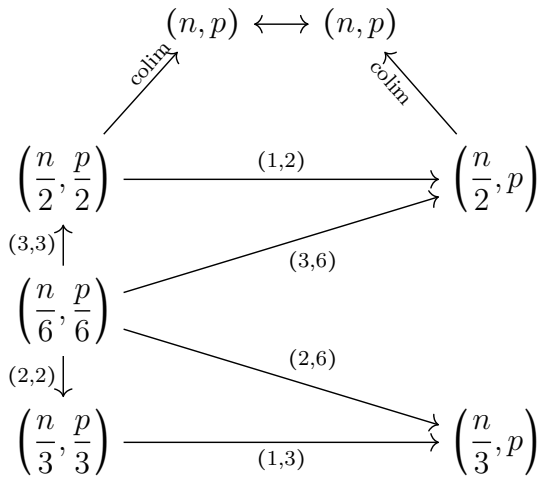
6.4 Other examples of multiplicity

In this section, we want to appply Theorem 6.3.6 in preorders.

6.4.1 Example of MP by (sMP-2)

In this section, we introduce an example of MP by (sMP-2). Consider the same category as 6.2.11, that is, the partial order $\mathbb{Z} \times \mathbb{Z}$, ordered by pairwise divisibility:

$(m_1, n_1) \leqslant (m_2, n_2)$ iff m_1 divides m_2 and n_1 divides n_2.



Why can't there be any cluster between the left and right-hand diagrams?

Let's answer by trying to construct a cluster G.

As always, a cluster needs to have at least one arrow from each object of its domain. There is only one arrow from $\left(\frac{n}{2}, \frac{p}{2}\right)$ and from $\left(\frac{n}{3}, \frac{p}{3}\right)$, so these arrows are in G; $(1,2), (1,3) \in G$. Then, we need to pick at least one arrow between $(3,6)$ and $(2,6)$. We observe that $(3,3)$ belongs to the right-hand diagram, and $(1,2)$ belongs to the cluster. Thus, the composite $(1,2) \circ (3,3) = (3,6)$ needs to be in the cluster (CLU-4). By the same reasonning, $(2,6)$ also needs to be in the cluster, so that the cluster needs to contain every arrow shown in the figure. However, whenever there are two arrows coming from the same object, their targets have to have a zigzag between them, which is not the same for the arrows $(3,6)$ and $(2,6)$. Because of this, the set of arrows G is not a cluster.

There is no cluster between those two diagrams, and these diagrams have the same colimits, so these diagrams verify the Multiplicity Principle (not in an interesting way, though). It is however a case where the Clawitzer diagram is sufficient (Definition 5.4.2).

6.4.2 Preorder on the ordinals

The current section aims at giving an example of MP based in set-theory. Readers non-familiar with set-theory may refer to [27] for definitions of ordinals and cardinals.

Consider a cardinal that is "big enough", for example, a $\aleph_\alpha$ for $\alpha \geqslant 1$. We see it as a well-ordered set (total order such that every subset has a least element).

Proposition 6.4.1

For $\alpha \geqslant 1$, the cardinal $\aleph_\alpha$ verifies the lax MP.

Proof. Consider the category $\mathbb{N} = \omega$ seen as a well-ordered set as well. The diagrams:

$$P : \begin{cases} \omega & \longrightarrow & \aleph_\alpha \\ n & \longmapsto & n \end{cases} \qquad Q : \begin{cases} \omega \times 2 & \longrightarrow & \aleph_\alpha \\ m & \longmapsto & m \end{cases}$$

define functors between well-ordered classes. There is one cluster $G : P \to Q$, defined by all the arrows from $n \in \omega$ to all $m \in \omega \times 2$ with $n \leqslant m$.

The colimits (least upper bounds) of P and Q are respectively ω and $\omega \times 2$, which are not isomorphic in terms of the well-ordering of $\aleph_\alpha$. Thus, G defined above does not bind to an isomorphism.

We have:

$$\mathrm{Upper}\,(P) = \{\beta \in \aleph_\alpha \mid \beta \geqslant \omega\} \quad \mathrm{Upper}\,(Q) = \{\beta \in \aleph_\alpha \mid \beta \geqslant \omega \times 2\}$$

The following functors:

$$U : \begin{cases} \mathrm{Upper}\,(P) & \longrightarrow & \mathrm{Upper}\,(D) \\ \beta & \longmapsto & \begin{cases} \omega + \beta & \text{if } \beta < \omega \times \omega \\ \beta & \text{if } \beta \geqslant \omega \times \omega \end{cases} \end{cases}$$

$$V : \begin{cases} \mathrm{Upper}\,(D) & \longrightarrow & \mathrm{Upper}\,(P) \\ \beta & \longmapsto & \begin{cases} \min\limits_{\omega+\gamma=\beta} (\gamma) & \text{if } \beta < \omega \times \omega \\ \beta & \text{if } \beta \geqslant \omega \times \omega \end{cases} \end{cases}$$

are inverses of each other, thus they constitute an isomorphism between $\mathrm{Upper}\,(P)$ and $\mathrm{Upper}\,(D)$. Note that they are not ΩG, because G does not bind to an isomorphism and thus, ΩG cannot be an isomorphism Proposition 6.2.10.

Hitherto, we have two diagrams with a cluster, with non-isomorphic colimits and isomorphic upper bounds; we then have an instance of MP due to (MP-3). □

Remark 6.4.2. This example also works with $\mathbb{O}\mathrm{n}$, the well-ordered class of ordinals, instead of $\aleph_\alpha$.

So we have an instance of lax MP in a total order. Whence, the next question.

6.4.3 Can a total order verify strict MP?

The previous example was a total order, and it verified the lax MP due to isomorphic, but different, upper bounds. This cannot happen in the first intended meaning of the MP: MP was supposed to be strict. Here, we investigate whether total orders can verify the strict MP.

By Lemma 6.3.8, we already know that if two diagrams verify the strict MP in a preorder, then there cannot be any cluster between them.

Lemma 6.4.3

Let $\mathscr{C}$ be a total order, and P and Q be diagrams to $\mathscr{C}$.
If P and Q verify the strict MP, then both full protoclusters $P \to Q$ and $Q \to P$ verify (CLU-1).

Proof. Otherwise, if for example the full protocluster $P \to Q$ does not verify (CLU-1), then there is a $P(p)$ that has no arrow to any $Q(q)$. Thus, since the order is total, there is necessarily an arrow $Q(q) \to P(p)$ for each $Q(q)$, which induces a cluster $Q \to P$ (it defines a cocone that we can extend to make a cluster), and thus contradicts Lemma 6.3.8. □

We finally deduce:

Theorem 6.4.4

There is no strict MP in a total order.

Proof. By Lemma 6.4.3, the functor $\mathrm{CC}_{P,Q}$ is non-empty.

In a total order, every comma-category $(P(p) \mid Q)$ contains only one connected component. Indeed, let $g : P(p) \to Q(q)$ and $g' : P(p) \to Q(q')$; as $\mathscr{C}$ is a total order, there is an arrow $Q(q) \to Q(q')$ or $Q(q') \to Q(q)$; as $\mathscr{C}$ is a preorder, the triangle necessarily commutes. The same is true about $(Q(q) \mid P)$.

Thus, $\mathrm{CC}_{P,Q}$ has only components with cardinality 1: for all p, $\mathrm{CC}_{P,Q}(p) \cong 1$. As it is a functor, it defines a cluster (Theorem 4.6.10), which contradicts Lemma 6.3.8, hence the absence of strict MP in total orders. □

While the (lax) MP seems to be common among preorders, its strict version appears to be less common.

6.4.4 Non-standard models of Peano

In this section, we give an example of preorder verifying the lax MP as a consequence of the fourth condition of Theorem 6.3.6: two diagrams without colimit, and with isomorphic but different sets of upper bounds. This example is again a total order.

First, let us remind a theorem from model theory and the study of models of Peano arithmetic.

Definition 6.4.5 (DLO)

Let $\mathscr{C}$ be a preorder.
$\mathscr{C}$ is called a *dense, linear order without endpoints* (*DLO* for short) when it is a total order that is dense, unbounded upwards and downwards; that is, a total order that verifies:

1. Dense: $\forall x, y \in \mathscr{C}, x < y \Rightarrow \exists z \in \mathscr{C},\ x < z < y$
2. Unbounded upwards: $\forall x \in \mathscr{C}, \exists z \in \mathscr{C},\ x < z$
3. Unbounded downwards: $\forall x \in \mathscr{C}, \exists z \in \mathscr{C},\ z < x$

The theory of DLO's is a textbook case because of the following:

Proposition 6.4.6 ([48, Section 2.4, page 48])

DLO is ω-categorical.

What that means, is that all countable models of DLO are order-isomorphic, which we often rephrase by saying that the only countable model of DLO is $\mathbb{Q}$.

For example, $\mathbb{R}$ is also a model of DLO but it is not the only one of its cardinality (Shelah's classification theorem [49, Theorem 0.3, §0, Chapter VIII, page 441] implies that there are $2^{\mathbf{c}}$ non-isomorphic models of DLO with cardinality $\mathbf{c} = 2^{\aleph_0}$).

Let us introduce on a specific order. Consider A to be a DLO.

Elements of $\omega + A \times \mathbb{Z}$ are either natural integers $n \in \omega$ or pairs $(a, z) \in A \times \mathbb{Z}$. Its order can be described as follows:

1. $\forall n \in \omega,\ \forall (a, z) \in A \times \mathbb{Z},\ n < (a, z)$
2. The restriction of the order to ω is isomorphic to the usual order on ω
3. The restriction of the order to $A \times \mathbb{Z}$ is isomorphic to the lexicographical order[3]: $(a_0, z_0) < (a_1, z_1) \Leftrightarrow (a_0 < a_1) \vee (a_0 = a_1 \wedge z_0 < z_1)$

3. Beware, in some books, the order type is $\omega + \mathbb{Z} \times A$ with the anti-lexicographical order on

This total order can be seen as follows: we start with a copy of ω. After that ω, we associate to each element $a \in A$ a copy of the integers $\mathbb{Z}$. We then order those slices $\{a\} \times \mathbb{Z}$ by the order on A. All those slices are then put after ω.

We discuss why orders $\omega + A \times \mathbb{Z}$ are of importance after the following proposition.

Proposition 6.4.7

Let $A = \mathbb{Q}$ or $\mathbb{R}$. Then $\omega + A \times \mathbb{Z}$ verifies the MP.

Proof. This total order $\omega + A \times \mathbb{Z}$ will verify (MP-4).

We need two subsets of that order, that will play the role of diagrams. For $x \in A$, consider the following subset:

$$P(x) = \omega + A_{<x} \times \mathbb{Z}$$

where $A_{<x} = \{a \in A \mid a < x\}$.

The set of upper bounds of $P(x)$ is:

$$\mathrm{Upper}\,(P(x)) = A_{\geqslant x} \times \mathbb{Z}$$

There is no least element in $\mathrm{Upper}\,(P(x))$ due to the structure of $\mathbb{Z}$, but for $x, x' \in A$, $A_{\geqslant x} \cong A_{\geqslant x'}$.

For $x \neq x'$, $P(x)$ and $P(x')$ have isomorphic but different upper bounds, and no sup: that is, $P(x)$ and $P(x')$ verify (MP-4), and thus, the MP. □

Note that we have a whole set of diagrams that witness the MP (one per $x \in \mathbb{Q}$ or $\mathbb{R}$).

Remark 6.4.8. The proof of Proposition 6.4.7 does not work for general DLO's. Consider $A =]-\infty, 0] \cup ([0, +\infty[\cap \mathbb{Q})$. Then A is a model of DLO; however, using the notation of the proof of Proposition 6.4.7, the sets $\mathrm{Upper}\,(P(-\pi))$ and $\mathrm{Upper}\,(P(0))$ are obviously not order-isomorphic (one is countable while the other is not).

This order may seem surprising at first. However, it is not a random order. It arises in the study of models of Peano arithmetic (PA for short). For readers not familiar with Peano Arithmetic and its models, the rest of this section may be skipped. We only recall here the bare minimum in order to justify the order $\omega + A \times \mathbb{Z}$.

$\mathbb{Z} \times A$.

Theorem 6.4.9 ([50, Theorem 6.4, section 6.2, page 75])

Nonstandard models of Peano Arithmetic (PA) have order type $\omega + A \times \mathbb{Z}$, where A is a DLO.

Corollary 6.4.10

In particular, $\omega + \mathbb{Q} \times \mathbb{Z}$ is the unique (up to isomorphism) order type of countable nonstandard models of PA.

Remark 6.4.11. This is beyond the scope of this thesis, but for the reader non-familiar with Peano models, we feel the need to clarify one point: any $\omega + A \times \mathbb{Z}$ (with A some DLO) is *not* necessarily the order type of a model of Peano. For instance, no model of PA has order type $\omega + \mathbb{R} \times \mathbb{Z}$. [51, Fact 2, section 1.2, page 13]

Following Proposition 6.4.7, we obtain:

Corollary 6.4.12

Let M be a countable nonstandard model of PA. Then the reduct of M to its order verifies the Multiplicity Principle.

This fact is most likely not provable in PA directly, because the MP is a "global" property of models.

6.5 Conclusion

The biological property of degeneracy led to the categorical notion of Multiplicity Principle (MP). In a first work [20], the goal was to find the MP in a certain preorder category, that really tells something about the involved objects: RDT tests and NP tests are made for different problems and they shouldn't be compared.

In this chapter, we gave a characterisation of preorders that does not rely on categorical notions (and above all, the tricky one of clusters), and illustrated it in several other preorders, suggesting that the MP may occur quite often in preorders.

The point however, is not that the MP carries no information for being a common occurrence, at least in preorders. The MP may carry information, just as in [20], when the diagrams witnessing the MP also make sense.

This chapter also had the purpose of simplifying the topic of MP in preorders, allowing for more occurrences to be found without the need for a detour through category theory.

Chapter 7

Multiplicity in tests

I've seen your kind, time and time again.
Every fleeing man must be caught.
Every secret must be unearthed.
Such is the conceit of the self-proclaimed seeker of truth.
But in the end, you lack the stomach.
For the agony you'll bring upon yourself.

Sir Vilhelm (2016). In *Ashes of Ariandel*, first DLC of *Dark Souls III* (2016).

7.1 Introduction

In [17], Edelman & Gally pointed out degeneracy as the fundamental property allowing for living systems to evolve through natural selection towards more complexity in fluctuating environments. Degeneracy is defined [17] as " ... *the ability of elements that are structurally different to perform the same function or yield the same output*". Degeneracy differs from redundancy since the latter "... occurs when the same function is performed by identical elements" [17].

Degeneracy is an ubiquitous and crucial feature of biological systems, e.g. immune systems and neural networks, at all organization levels. The key point put forward by Edelman and Gally is [17] that "...degeneracy is not a property simply selected by evolution, but rather is a prerequisite for and an inescapable product of the process of natural selection itself."

The Multiplicity Principle (MP) [12, 45], introduced by Ehresmann & Vanbremeersch, is a mathematical formalization of degeneracy in categorical terms. A thorough study of MP can be found in Chapter 6. The consequences of this principle, as treated in [12, 45], underpin Edelman & Gally's conjecture according to which "*complexity and degeneracy go hand in hand*" [17].

Another property of many biological and social systems is their resilience: (i) they can perform in degraded mode, with some performance loss, but without collapsing;

(ii) they can recover their initial performance level when nominal conditions are satisfied again; (iii) they can perform corrections and auto-adaption so as to maintain essential tasks for their survival. In addition, resilience of social or biological systems is achieved via agents with different skills. For instance, cells are simply reactive organisms, whereas social agents have some cognitive properties. Thence the idea that resilience may derive from fundamental properties satisfied by agents, interactions and organizations. Could this fundamental property be a possible consequence of degeneracy [45, Section 3.1, p. 15]?

The notion of resilience remains, however, somewhat elusive, mathematically speaking. In contrast, the notion of robustness has a long history and track record in mathematical statistics [52]. By and large, a statistical method is robust if its performance is not unduly altered in case of outliers or fluctuations around the model for which it is designed. Can we fathom the links between resilience and robustness?

As an attempt to embrace the questions raised above from a comprehensive outlook, the original question addressed in this work-in-progress is the possible connection between MP and robustness to account for emergence of resilience in complex systems. As a first step in our study aimed at casting the notions of robustness, resilience and degeneracy within the same theoretical framework based on MP, we hereafter establish that statistical tests do satisfy MP. The task to perform by the tests is the fundamental problem of detecting a signal in noise. However, to ease the reading of a chapter at the interface between category theory and mathematical statistics, we consider a simplified version of this problem.

The chapter is organized as follows. We begin by specifying notation and notions in mathematical statistics. In Section 7.2, we state MP in categorical words on the basis of [12] and consider the particular case of preorders, which will be sufficient at the present time to establish that statistical tests satisfy MP for detecting signals in noise. In Section 7.3, we set out the statistical detection problem. We will then introduce a preorder that makes it possible to exhibit two types of "structurally different" tests, namely, the Neyman-Pearson tests (Section 7.5) and the RDT tests (Section 7.6). Section 7.7 concludes the chapter by establishing that these two types of tests achieve the MP for the detection problem under consideration. For space considerations, we limit proofs to the minimum making it possible to follow the approach without too much undue effort.

Notation

Before anything, we remind here that we are using the set-theoretic notation for integers. E.g., 2 is exactly the set $2 = \{0, 1\}$.

Random variables. Given two measurable spaces $\mathcal{E}$ and $\mathcal{F}$, $\mathcal{M}(\mathcal{E},\mathcal{F})$ denotes the set of all measurable functions defined on $\mathcal{E}$ and valued in $\mathcal{F}$. The two σ-algebra involved are omitted in the notation because, in the sequel, they will always be obvious from the context. In particular, we will throughout consider a probability space $(\Omega,\mathcal{B},\mathbb{P})$ and systematically endow $\mathbb{R}$ with the Borel σ-algebra, which will not be recalled. Therefore, $\mathcal{M}(\Omega,\mathbb{R})$ designates the set of all real random variables. Given $q \in [0,\infty[$, $\mathcal{B}_\infty(q)$ is the set of all real random variables $\Delta \in \mathcal{M}(\Omega,\mathbb{R})$ such that $|\Delta|_\infty \leqslant q$. As usual, we write $X \sim \mathcal{N}(0,1)$ to mean that $X \in \mathcal{M}(\Omega,\mathbb{R})$ is standard normal.

The cumulative distributive function (cdf) of any $X \sim \mathcal{N}(0,1)$ is denoted by Φ. Given a sequence $(X_n)_{n\in\mathbb{N}} \in \mathcal{M}(\Omega,\mathbb{R})^{\mathbb{N}}$ of real random variables, we write

$$X_1, X_2, \ldots \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$$

to mean that the random variables $X_1, X_2, \ldots$ are independent and identically distributed with common distribution $\mathcal{N}(0,1)$. For any $n \in \mathbb{N}$, the Generalized Marcum function $Q_{n/2}$ is defined for all $\rho \geqslant 0$ and all $\lambda \geqslant 0$ by setting: $Q_{n/2}(\rho,\lambda) = 1 - \mathbb{F}_{\chi^2_n(\rho^2)}(\lambda^2)$, where $\mathbb{F}_{\chi^2_n(\rho^2)}$ is the cdf of the χ^2 law with n degrees of freedom and non-centrality parameter ρ^2.

Decisions et Observations. Throughout, $\mathcal{M}(2\times\Omega,2)$ designates the set of all measurable functions $D : 2\times\Omega \to 2$. Any element of $\mathcal{M}(2\times\Omega,2)$ is called a decision for obvious reasons given below. If $D \in \mathcal{M}(2\times\Omega,2)$ then, for any $\varepsilon \in 2$, $D(\varepsilon)$ denotes the Bernoulli-distributed random variable $D(\varepsilon) : \Omega \to 2$ defined for any given $\omega \in \Omega$ by $D(\varepsilon)(\omega) = D(\varepsilon,\omega)$. An n-dimensional test is hereafter any measurable function $f : \mathbb{R}^n \to 2$ and $\mathcal{M}(\mathbb{R}^n,2)$ stands for the set of all n-dimensional tests. A measurable function $\mathbf{X} : 2\times\Omega \to \mathbb{R}^n$ is hereafter called an observation and $\mathcal{M}(2\times\Omega,\mathbb{R}^n)$ denotes the set of all these observations. Given a test $f \in \mathcal{M}(\mathbb{R}^n,2)$ and $\mathbf{X} \in \mathcal{M}(2\times\Omega,\mathbb{R}^n)$, $D = f(\mathbf{X})$ is trivially a decision: $D \in \mathcal{M}(2\times\Omega,2)$. If $\mathbf{X} \in \mathcal{M}(2\times\Omega,\mathbb{R}^n)$ then, for any $\varepsilon \in 2$, $\mathbf{X}(\varepsilon) = \mathbf{X}(\varepsilon,\cdot) \in \mathbf{X}(\varepsilon) \in \mathcal{M}(\Omega,\mathbb{R}^n)$ is defined for every $\omega \in \Omega$ by $\mathbf{X}(\varepsilon)(\omega) = \mathbf{X}(\varepsilon,\omega)$.

Empirical means. We define the empirical mean of a given sequence $y = (y_n)_{n\in\mathbb{N}}$ of real values as the sequence $(\langle y\rangle_n)_{n\in\mathbb{N}}$ of real values such that, $\forall n \in \mathbb{N}, \langle y\rangle_n = \frac{1}{n}\sum_{i=1}^n y_i$. By extension, the empirical mean of a sequence $Y = (Y_n)_{n\in\mathbb{N}}$ of random variables where each $Y_n \in \mathcal{M}(\Omega,\mathbb{R})$ is the sequence $(\langle Y\rangle_n)_{n\in\mathbb{N}}$ of random variables where, for each $n \in \mathbb{N}$, $\langle Y\rangle_n \in \mathcal{M}(\Omega,\mathbb{R})$ is defined by $\langle Y\rangle_n = \frac{1}{n}\sum_{i=1}^n Y_i$. Therefore, for any $\omega \in \Omega$, $\langle Y\rangle_n(\omega) = \langle Y(\omega)\rangle_n$ with $Y(\omega) = (Y_n(\omega))_{n\in\mathbb{N}}$. If $Y = (Y_n)_{n\in\mathbb{N}}$ is a sequence of observations ($\forall n \in \mathbb{N}$, $Y_n \in \mathcal{M}(2\times\Omega,\mathbb{R})$), we define the empirical mean of Y as the sequence $(\langle Y\rangle_n)_{n\in\mathbb{N}}$ of observations such that, for $\varepsilon \in 2$, $\langle Y\rangle_n \in \mathcal{M}(2\times\Omega,\mathbb{R})$ with$\langle Y\rangle_n(\varepsilon) = \langle Y(\varepsilon)\rangle_n$ and $Y(\varepsilon) = (Y_n(\varepsilon))_{n\in\mathbb{N}}$.

7.2 Multiplicity Principle

The multiplicity principle (MP) comes from [12] and was thoroughly studied in Chapter 6.

The reader should refer to Definition 6.2.6 for the cluster-composition functor ΩG and to Definition 6.2.9 for the Multiplicity Principle.

In the biological model described in [12, Chapter 3, Section 4], MP translates "*the robustness and the adaptability of a complex system*", and is interpreted that way. P and Q having the same cocones translates the property of both systems to accomplish the same function. The absence of clusters between P and Q that define an isomorphism, reflects the structural difference between them, which is key to robustness and adaptability: if the system described by P fails, then Q may replace it.

The main purpose of this chapter is to find a meaningful instance of the multiplicity principle in some preorder. Just as in Chapter 6, we do not distinguish between a preorder and its category.

Proposition 7.2.1 (Simplified MP in a preorder)

Let $(E, \leqslant)$ be a preorder. If there are two disjoint subsets $A, B \subset E$ such that the following conditions hold, then E verifies the multiplicity principle:

(i) A and B have the same sets of upper bounds
(ii) There is an $a \in A$ with no upper bounds in B
(iii) There is a $b \in B$ with no upper bounds in A

Proof. Condition (i) ensures that A and B have isomorphic categories of cocones (in a preorder, cocones are the upper bounds). Conditions (ii) and (iii) respectively ensure that there is no cluster $i_A \to i_B$ nor $i_B \to i_A$ where $i_A : A \hookrightarrow E$ and $i_B : B \hookrightarrow E$ are the inclusion functors. Overall, this is an easy application of Theorem 6.3.6. □

Albeit trivial, the following corollary will be helpful.

Corollary 7.2.2

Given a preordered set $(E, \leq)$, if A and B are two subsets of E such that $\sup(A, (E, \leqslant)) = \sup(B, (E, \leqslant))$ and if $A \times B \cap \leqslant = \varnothing$, then E satisfies MP.

7.3 Statistical detection of a signal in noise

In this section, we introduce the tools from statistical decision testing.

7.3.1 Problem statement

Let $\varepsilon \in 2$ be the unknown indicator value on whether a certain physical phenomenon has occurred ($\varepsilon = 1$) or not ($\varepsilon = 0$). We aim at determining this value. It is desirable to resort to something more evolved than tossing a coin to estimate ε. However, whatever D, the decision is erroneous for any $\omega \in \Omega$ such that $D(\varepsilon, \omega) \neq \varepsilon$. We thus have two distinct cases.

False alarm probability. If $\varepsilon = 0$ and $D(0, \omega) = 1$, we commit an error of the first kind false or false alarm, since we have erroneously decided that the phenomenon has occurred while nothing actually happened. We thus define the false alarm probability (aka size, aka error probability of the first kind) of D as:

$$\mathbb{P}_{\text{FA}}[D] \stackrel{\text{def}}{=} \mathbb{P}[D(0) = 1] \tag{7.1}$$

Detection probability. If $\varepsilon = 1$ and $D(1, \omega) = 0$, we commit an error of the second kind, also called missed detection since, in this case, we have missed the occurrence of the phenomenon. As often in the literature on the topic, we prefer using the probability of correctly detecting the phenomenon by defining the detection probability:

$$\mathbb{P}_{\text{DET}}[D] \stackrel{\text{def}}{=} \mathbb{P}[D(1) = 1] \tag{7.2}$$

7.3.2 Decision with level $\gamma \in [0, 1]$ and oracles

Among all the possible decisions, the omniscient oracle $D^* \in \mathcal{M}(2 \times \Omega, 2)$ is defined for any pair $(\varepsilon, \omega) \in 2 \times \Omega$ by setting $D^*(\varepsilon, \omega) = \varepsilon$. Its probability of false alarm is 0 and its probability of detection is 1: $\mathbb{P}_{\text{FA}}[D^*] = 0$ et $\mathbb{P}_{\text{DET}}[D^*] = 1$. This omniscient oracle has no practical interest since it knows ε, which is pretty unfair and unrealistic. Since it is not possible in practice to guarantee a null false alarm probability, we focus on decisions whose false alarm probabilities are upper-bounded by a real number $\gamma \in [0, 1]$ called level. We state the following definition.

Definition 7.3.1 (Level of a decision)

Given a decision $D \in \mathcal{M}(2 \times \Omega, 2)$, D *has level* γ when γ is an upper bound of its probability of false alarm:

$$\mathbb{P}_{\text{FA}}[D] \leqslant \gamma$$

The set of decisions of level γ is denoted by Dec_γ.

We can easily prove the existence of an infinite number of elements in Dec_γ that all have a detection probability equal to 1. Whence the following definition.

Definition 7.3.2 (Oracle)

Given $\gamma \in [0,1]$, an oracle with level γ is any decision $D \in \text{Dec}_\gamma$ such that $\mathbb{P}_{\text{DET}}[D] = 1$.
The set of all the oracles with level γ is denoted by $\mathbb{O}_\gamma$.

Oracles with level γ have no practical interest either since they require prior knowledge of ε! Therefore, we restrict our attention to decisions in Dec_γ that "approximate" at best the oracles with level γ, without prior knowledge of ε, of course. To this end, we must preorder decisions.

Definition 7.3.3 (Total preorder $(\text{Dec}_\gamma, \preceq)$)

For any given $\gamma \in [0,1]$ and any pair $(D, D') \in \text{Dec}_\gamma \times \text{Dec}_\gamma$, we define a preorder $(\text{Dec}_\gamma, \preceq)$ by setting:

$$D \preceq D' \quad \text{if} \quad \mathbb{P}_{\text{DET}}[D] \leqslant \mathbb{P}_{\text{DET}}[D'].$$

We write $D \cong D'$ if $D \preceq D'$ and $D' \preceq D$.

7.3.3 Observations

In practice, observations help us decide whether the phenomenon has occurred or not. By collecting a certain number of them, we can expect to make a decision. Hereafter, observations are assumed to be elements of $\mathcal{M}(2 \times \Omega, \mathbb{R})$ and corrupted versions of ε. We suppose to have a sequence $(Y_n)_{n \in \mathbb{N}}$ of such random variables. As a first standard model, we could assume that, for any $n \in \mathbb{N}$ and any $(\varepsilon, \omega) \in 2 \times \Omega$, $Y_n(\varepsilon, \omega) = \varepsilon + X_n(\omega)$ with $X_1, X_2, \ldots, X_n, \ldots \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$. In this additive model, X_n models noise on the nth observation. We could make this model more complicated and realistic by considering random vectors instead of variables. However, with re-

spect to our purpose, the significant improvement we can bring to the model is elsewhere. Indeed, we have assumed above that the signal, regardless of noise, is ε. However, from a practical point of view, it is more realistic to assume that the nth observation Y_n captures ε in presence of some interference Δ_n, independent of X_n. In practice, the probability distribution of Δ_n will hardly be known and, as a means to compensate for this lack of knowledge, we assume the existence of a uniform bound on the amplitude of all possible interferences. Therefore, we assume that, for all $(\varepsilon,\omega) \in 2\times\Omega$, $Y_n(\varepsilon,\omega) = \varepsilon + X_n(\omega) + \Delta_n(\omega)$ and the existence of $q \in [0,\infty)$ such that $\Delta_n \in \mathcal{B}_\infty(q)$. After all, this model is standard in time series analysis: ε plays the role of a trend, Δ_n is the seasonal variation and X_n is the measurement noise.

For each $q \in [0,\infty)$, $\mathrm{Seq}^{[q]}$ is hereafter the set of all the sequences $Y^{[q]} = \left(Y_n^{[q]}\right)_{n\in\mathbb{N}}$ where, $\forall n \in \mathbb{N}$ and $\forall(\varepsilon,\omega) \in 2\times\Omega$, $Y_n^{[q]}(\varepsilon,\omega) = \varepsilon + \Delta_n(\omega) + X_n(\omega)$, where $\Delta_n \in \mathcal{B}_\infty(q)$ and $X_n \sim \mathcal{N}(0,1)$ are independant. Therefore, for all $n \in \mathbb{N}$ and all $\varepsilon \in 2$, $Y_n^{[q]}(\varepsilon) = \varepsilon + \Delta_n + X_n$, with $X_1, X_2, \ldots, X_n, \ldots \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$.

7.4 Selectivity, landscapes of tests and preordering

For any $q \in \left[0, \frac{1}{2}\right[$, any $n \in \mathbb{N}$ and any $Y^{[q]} = \left(Y_n^{[q]}\right)_{n\in\mathbb{N}} \in \mathrm{Seq}^{[q]}$, we set:

$$\mathbf{Y}_n^{[q]} = \left(Y_1^{[q]}, Y_2^{[q]}, \ldots, Y_n^{[q]}\right) \tag{7.3}$$

Definition 7.4.1 (Selectivity of a test)

Given any $n \in \mathbb{N}$ and any test $f \in \mathcal{M}(\mathbb{R}^n, 2)$, the selectivity of f at given level $\gamma \in [0,1]$ is defined as the set:

$$\mathrm{Sel}_\gamma(f) = \left\{ q \in \left[0, \frac{1}{2}\right[\;\middle|\; \forall Y^{[q]} \in \mathrm{Seq}^{[q]}, f\left(Y_n^{[q]}\right) \text{ has level } \gamma \right\}$$

Definition 7.4.2 (Landscape of a test)

Given any $n \in \mathbb{N}$ and any test $f \in \mathcal{M}(\mathbb{R}^n, 2)$, the landscape of f at given level $\gamma \in [0,1]$ is the subset of Dec_γ defined by:

$$\mathrm{Lnd}_\gamma(f) = \bigcup_{q\in\mathrm{Sel}_\gamma(f)} \left\{ f\left(\mathbf{Y}_n^{[q]}\right) \;\middle|\; Y^{[q]} \in \mathrm{Seq}^{[q]} \right\}$$

This notion of landscape makes it possible to compare tests via landscapes via the following preorder. More precisely, if f and f' are two elements of $\mathcal{M}(\mathbb{R}^n, 2)$, it is natural to compare them in a uniform sense, according to which the former performs better than the latter if $f\left(\mathbf{Y}_n^{[q]}\right)$ outperforms $f'\left(\mathbf{Y}_n^{[q]}\right)$ for any $Y^{[q]} \in \mathrm{Seq}^{[q]}$. This leads us to introduce a preorder on landscapes, with respect to which it will be possible to find the landscapes of $\mathrm{Lndscps}_\gamma$ that best approximate landscapes of oracles with given level γ. The proofs that the following definition is consistent and that the next lemma holds true are left to the reader.

Definition 7.4.3 (Preorder $(\mathcal{P}(\mathrm{Dec}_\gamma)\ ,\ \preccurlyeq^*\ )$)

Given any level $\gamma \in [0,1]$, we define the preorder $(\mathcal{P}(\mathrm{Dec}_\gamma)\ ,\ \preccurlyeq^*\ )$ via the three following properties:

(P1) $\forall n \in \mathbb{N},\ \forall (f, f') \in \mathcal{M}(\mathbb{R}^n, 2) \times \mathcal{M}(\mathbb{R}^n, 2),\ \mathrm{Lnd}_\gamma(f) \preccurlyeq^* \mathrm{Lnd}_\gamma(f')$ iff they have the same selectivity $\mathrm{Sel}_\gamma(f) = \mathrm{Sel}_\gamma(f')$ and:

$$\forall q \in \mathrm{Sel}_\gamma(f), \forall Y^{[q]} \in \mathrm{Seq}^{[q]}, f\left(\mathbf{Y}_n^{[q]}\right) \leq\ f'\left(\mathbf{Y}_n^{[q]}\right)$$

(P2) $\forall (\mathcal{L}, \mathcal{L}') \in \left(\mathrm{Lndscps}_\gamma \cup \mathrm{Lnd}\mathbb{O}_\gamma\right) \times \mathrm{Lnd}\mathbb{O}_\gamma,\ \ \mathcal{L} \preccurlyeq^* \mathcal{L}'$ with:

$$\mathrm{Lndscps}_\gamma = \bigcup_{n \in \mathbb{N}} \left\{ \mathrm{Lnd}_\gamma(f) : f \in \mathcal{M}(\mathbb{R}^n, 2) \right\} \quad \text{and} \quad \mathrm{Lnd}\mathbb{O}_\gamma = \mathcal{P}(\mathbb{O}_\gamma)$$

(P3) $\forall \mathcal{L} \in \mathcal{P}(\mathrm{Dec}_\gamma) \setminus \left(\mathrm{Lndscps}_\gamma \cup \mathrm{Lnd}\mathbb{O}_\gamma\right),\ \mathcal{L} \preccurlyeq^* \mathcal{L}$

We directly derive from the definition:

Lemma 7.4.4

For all landscapes $\mathcal{L}, \mathcal{L}' \in \mathrm{Lndscps}_\gamma \cup \mathrm{Lnd}\mathbb{O}_\gamma$, we have $\mathcal{L} \preccurlyeq^* \mathcal{L}' \Rightarrow \mathcal{L} \times \mathcal{L}' \subset \leq$.

Henceforth, given a preordered set $(E, \preccurlyeq)$ and $A \subset E$, the set of maximal elements of A is denoted by $\max(A, (E, \preccurlyeq))$, the set of upper bounds is denoted by $\mathrm{Upper}(A, (E, \preccurlyeq))$ and the set of least upper bounds of A in $(E, \preccurlyeq)$ is denoted by $\sup(A, (E, \preccurlyeq))$.

Theorem 7.4.5 (Approximation of oracles in $(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*)$)

Given $\gamma \in [0,1]$, let a set Ξ_γ and a family of tests $(f_{\xi,n})_{\xi\in\Xi_\gamma, n\in\mathbb{N}}$ such that:

(i) $\forall (\xi, n) \in \Xi_\gamma \times \mathbb{N},\ f_{\xi,n} \in \mathcal{M}(\mathbb{R}^n, 2)$;

(ii) $\exists Q_\gamma \subset [0,\infty),\ \forall (\xi, n) \in \Xi_\gamma \times \mathbb{N},\ \mathrm{Sel}_\gamma(f_{\xi,n}) = Q_\gamma$;

(iii) $\forall (\xi, q) \in \Xi_\gamma \times Q_\gamma,\ \forall Y^{[q]} \in \mathrm{Seq}^{[q]},\ \lim_{n\to\infty} \mathbb{P}_{\mathrm{DET}}\left[f_{\xi,n}\left(\mathbf{Y}_n^{[q]}\right)\right] = 1$;

then, by setting $\mathrm{Lndscps}_\gamma{}' = \left\{\mathrm{Lnd}_\gamma(f_{\xi,n}) : n \in \mathbb{N}, \xi \in \Xi_\gamma\right\}$, we have:

$$\begin{aligned}\mathrm{Lnd}\mathbb{O}_\gamma &= \mathrm{Upper}\left(\mathrm{Lndscps}_\gamma{}', \left(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*\right)\right)\\ &= \sup\left(\mathrm{Lndscps}_\gamma{}', \left(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*\right)\right)\end{aligned} \tag{7.4}$$

Proof. For any $(\xi, n) \in \Xi_\gamma \times \mathbb{N}$ and any $\mathcal{L} \in \mathrm{Lnd}\mathbb{O}_\gamma$, (P2) in Definition 7.8.2 straightforwardly implies that $\mathrm{Lnd}_\gamma(f_{\xi,n}) \leqslant^* \mathcal{L}$. As a consequence:

$$\mathrm{Lnd}\mathbb{O}_\gamma \subset \mathrm{Upper}\left(\mathrm{Lndscps}_\gamma{}', \left(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*\right)\right) \tag{7.5}$$

To prove the reverse inclusion, let $\mathcal{L} \in \mathrm{Upper}\left(\mathrm{Lndscps}_\gamma{}', \left(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*\right)\right)$. We thus have $\forall (\xi, n) \in \mathbb{N} \times \Xi_\gamma,\ \mathrm{Lnd}_\gamma(f_{\xi,n}) \leqslant^* \mathcal{L}$.

According to Lemma 7.4.4, we have $\forall (\xi, n) \in \Xi_\gamma \times \mathbb{N},\ \mathrm{Lnd}_\gamma(f_{\xi,n}) \times \mathcal{L} \subset \preceq$. Therefore, $\forall (\xi, n) \in \mathbb{N} \times \Xi_\gamma,\ \forall q \in \mathrm{Sel}_\gamma(f_{\xi,n}),\ \forall Y^{[q]} \in \mathrm{Seq}^{[q]}$ and $\forall D \in \mathcal{L},\ f_{\xi,n}\left(\mathbf{Y}_n^{[q]}\right) \preceq \mathcal{L}$.

It follows from the definition of $\preceq$ and assumption (ii) above that:

$$\forall (\xi, n) \in \mathbb{N} \times \Xi_\gamma, \forall q \in Q_\gamma, \forall Y^{[q]} \in \mathrm{Seq}^{[q]}, \forall D \in \mathcal{L}, \mathbb{P}_{\mathrm{DET}}\left[f_{\xi,n}\left(\mathbf{Y}_n^{[q]}\right)\right] \leqslant \mathbb{P}_{\mathrm{DET}}[D]$$

We derive from assumption (iii) that $\mathbb{P}_{\mathrm{DET}}[D] = 1$ and thus that $D \in \mathbb{O}_\gamma$. It follows that $\mathcal{L} \in \mathrm{Lnd}\mathbb{O}_\gamma$. We thus obtain that $\mathrm{Upper}\left(\mathrm{Lndscps}_\gamma{}', \left(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*\right)\right) \subset \mathrm{Lnd}\mathbb{O}_\gamma$ and therefore, from (7.5), $\mathrm{Lnd}\mathbb{O}_\gamma = \mathrm{Upper}\left(\mathrm{Lndscps}_\gamma{}', \left(\mathcal{P}(\mathrm{Dec}_\gamma), \leqslant^*\right)\right)$. The second inequality in (7.4) is straightforward since the elements of $\mathrm{Lnd}\mathbb{O}_\gamma$ are isomorphic in the sense of $\leqslant^*$. □

For later use, given $J \subset [0,\infty)$, $n \in \mathbb{N}$ and $\mathbb{C} \subset \mathcal{M}(\mathbb{R}^n, 2)$, we hereafter set:

$$\mathrm{Lndscps}_\gamma^J(\mathbb{C}) = \left\{\mathrm{Lnd}_\gamma(f) \in \mathrm{Lndscps}_\gamma : f \in \mathbb{C}, \mathrm{Sel}_\gamma(f) = J\right\} \tag{7.6}$$

7.5 The Neyman-Pearson (NP) solution

When n spans $\mathbb{N}$, the Neyman-Pearson (NP) Lemma makes it possible to pinpoint a maximal element in each $\left(\text{Lndscps}_\gamma^{\{0\}}(\mathbb{C}), \preceq\right)$ with $\mathbb{C} = \mathcal{M}(\mathbb{R}^n, 2)$. These maximal elements are hereafter called NP decisions. Specifically, we have the following result.

Lemma 7.5.1 (Maximality of the NP decisions)

For any $\gamma \in [0,1]$ and any $n \in \mathbb{N}$,

$$\text{Lnd}_\gamma\left(f_n^{\text{NP}(\gamma)}\right) = \max\left(\text{Lndscps}_\gamma^{\{0\}}\left(\mathcal{M}(\mathbb{R}^n, 2)\right), \preccurlyeq^*\right) \tag{7.7}$$

where $f_n^{\text{NP}(\gamma)} \in \mathcal{M}(\mathbb{R}^n, 2)$ is the n-dimensional NP test with size γ defined by:

$$\forall (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n, f_n^{\text{NP}(\gamma)}(y_1, y_2, \ldots, y_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n y_i > \sqrt{n}\Phi^{-1}(1-\gamma) \\ 0 & \text{otherwise} \end{cases}$$

and satisfies, $\forall Y^{[0]} \in \text{Seq}^{[0]}$,

$$\begin{cases} \mathbb{P}_{\text{FA}}\left[f_n^{\text{NP}(\gamma)}\left(\mathbf{Y}_n^{[0]}\right)\right] = \gamma \\ \mathbb{P}_{\text{DET}}\left[f_n^{\text{NP}(\gamma)}\left(\mathbf{Y}_n^{[0]}\right)\right] = 1 - \Phi\left(\Phi^{-1}(1-\gamma) - \sqrt{n}\right) \end{cases}$$

Proof. A direct application of the Neyman-Pearson Lemma [53, Theorem 3.2.1, page 60], followed by some standard algebra to obtain $\mathbb{P}_{\text{DET}}\left[f_n^{\text{NP}(\gamma)}\left(\mathbf{Y}_n^{[0]}\right)\right]$. □

The next result states that it suffices to increase the number of observations to approximate oracles with level γ by NP decisions.

Theorem 7.5.2 (Approximation of oracles by NP tests)

Setting $\text{Lnd}^{\text{NP}(\gamma)} = \left\{\text{Lnd}_\gamma\left(f_n^{\text{NP}(\gamma)}\right) : n \in \mathbb{N}\right\}$ for any $\gamma \in [0,1]$, we have:

$$\begin{aligned} \text{Lnd}\mathbb{O}_\gamma &= \text{Upper}\left(\text{Lnd}^{\text{NP}(\gamma)}, \left(\mathcal{P}\left(\text{Dec}_\gamma\right), \preccurlyeq^*\right)\right) \\ &= \sup\left(\text{Lnd}^{\text{NP}(\gamma)}, \left(\mathcal{P}\left(\text{Dec}_\gamma\right), \preccurlyeq^*\right)\right) \end{aligned}$$

Proof. According to Lemma 7.5.1, $\lim_{n\to\infty} \mathbb{P}_{\text{DET}}\left[f_n^{\text{NP}(\gamma)}\mathbf{Y}_n^{[0]}\right] = 1$. The set $\text{Lnd}^{\text{NP}(\gamma)} \subset \text{Lndscps}_\gamma$ satisfies the conditions of Theorem 7.4.5 for any $\gamma \in [0,1]$ and $\Xi_\gamma = \varnothing$. Thence the result. □

7.6 The RDT solution

7.6.1 An elementary RDT problem

Problem statement. The RDT theoretical framework is exposed in full details in [47, 54]. To ease the reading of the present chapter, we directly focus on the particular RDT problem that can be used in connection with the detection problem at stake.

In this respect, suppose that $\mathbf{Y} = \Theta + \mathbf{X} \in \mathcal{M}(\Omega, \mathbb{R}^n)$, where Θ and $\mathbf{X}$ are independent elements of $\mathcal{M}(\Omega, \mathbb{R}^n)$. In the sequel, we assume that $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_n)$, $\mathbf{I}_n$ being the $n \times n$ identity matrix, and consider the mean testing problem of deciding on whether $|\langle\Theta\rangle_n(\omega)| \leqslant \tau$ (null hypothesis $\mathcal{H}_0$) or $|\langle\Theta\rangle_n(\omega)| > \tau$ (alternative hypothesis $\mathcal{H}_1$), when we are given $\mathbf{Y}(\omega) = \Theta(\omega) + \mathbf{X}(\omega)$, for $\omega \in \Omega$. The idea is that Θ oscillates uncontrollably around 0 and that only sufficient large deviations of the norm should be detected. This is a particular Block-RDT problem, following the terminology and definition given in [54]. This problem is summarized by discarding ω and writing:

$$\begin{cases} \textbf{Observation:} \mathbf{Y} = \Theta + \mathbf{X} \in \mathcal{M}(\Omega, \mathbb{R}^n) \text{ with } \begin{cases} \Theta \in \mathcal{M}(\Omega, \mathbb{R}^n), \mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_N), \\ \Theta \text{ and } \mathbf{X} \text{ are independent,} \end{cases} \\ \mathcal{H}_0 : |\langle\Theta\rangle_n| \leqslant \tau, \\ \mathcal{H}_1 : |\langle\Theta\rangle_n| > \tau. \end{cases} \tag{7.8}$$

Standard likelihood theory [53, 55, 56] does not make it possible to solve this problem. Fortunately, this problem can be solved as follows via the Random Distortion Testing (RDT) framework.

Size and power of tests for mean testing. We seek tests with guaranteed size and optimal power, in the sense specified below.

Definition 7.6.1 (Size for the mean testing problem)

The size of $f \in \mathcal{M}(\mathbb{R}^n, 2)$ for testing the empirical mean of the signals $\Theta \in \mathcal{M}(\Omega, \mathbb{R}^n)$ such that $\mathbb{P}[|\langle\Theta\rangle_n| \leqslant \tau] \neq 0$, given the processes $\Theta + \mathbf{X} \in \mathcal{M}(\Omega, \mathbb{R}^n)$ with $\mathbf{X}$ independent of Θ, is defined by:

$$\alpha^{[n]}(f) = \sup_{\Theta \in \mathcal{M}(\Omega, \mathbb{R}^n) : \mathbb{P}[|\langle\Theta\rangle_n| \leqslant \tau] \neq 0} \mathbb{P}\left[f(\Theta + \mathbf{X}) = 1 \mid |\langle\Theta\rangle_n| \leqslant \tau\right] \tag{7.9}$$

We say that $f \in \mathcal{M}(\mathbb{R}^n, 2)$ has level (resp. size) γ if $\alpha^{[n]}(f) \leqslant \gamma$ (resp. $\alpha^{[n]}(f) = \gamma$). The class of all the tests with level γ is denoted by $\text{Tests}_\gamma^{[n]}$:

$$\text{Tests}_\gamma^{[n]} = \left\{ f \in \mathcal{M}(\mathbb{R}^n, 2) : \alpha^{[n]}(f) \leqslant \gamma \right\}$$

Definition 7.6.2 (Power for the mean testing problem)

The power of $f \in \mathcal{M}(\mathbb{R}^n, 2)$ for testing the empirical mean of $\Theta \in \mathcal{M}(\Omega, \mathbb{R}^n)$ such that $\mathbb{P}[|\langle\Theta\rangle_n| > \tau] \neq 0$ when we are given $\mathbf{Y} = \Theta + \mathbf{X} \in \mathcal{M}(\Omega, \mathbb{R}^n)$, with $\mathbf{X}$ independent of Θ, is defined by:

$$\beta_\Theta^{[n]}(f) = \mathbb{P}\left[f(\Theta + \mathbf{X}) = 1 \middle| |\langle\Theta\rangle_n| > \tau \right] \tag{7.10}$$

The RDT solution. We can easily construct a preorder $\left(\text{Tests}_\gamma^{[n]}, \leqslant^\diamond\right)$ by setting, for all $(f, f') \in \text{Tests}_\gamma^{[n]} \times \text{Tests}_\gamma^{[n]}$:

$$f \leqslant^\diamond f' \text{ iff } \forall \Theta \in \mathcal{M}(\Omega, \mathbb{R}^n), \mathbb{P}[|\langle\Theta\rangle_n| > \tau] \neq 0 \Rightarrow \beta_\Theta^{[n]}(f) \leqslant \beta_\Theta^{[n]}(f')$$

No maximal element exists in $\left(\text{Tests}_\gamma^{[n]}, \leqslant^\diamond\right)$. However, we can exhibit $\mathbb{C}_\gamma^{[n]} \subset \text{Tests}_\gamma^{[n]}$ whose elements satisfy suitable invariance properties with respect to the mean testing problem and prove the existence of a maximal element in $\left(\mathbb{C}_\gamma^{[n]}, \leqslant^\diamond\right)$.

Set $\mathcal{S} = \{\text{id}, -\text{id}\}$ where id is the identity of $\mathbb{R}$. Endowed with the usual composition law $\circ$ of functions, $(\mathcal{S}, \circ)$ is a group. Let $\mathcal{A}$ be the group action that associates to each given $s \in \mathcal{S}$ the map $\mathcal{A}_s : \mathbb{R}^n \to \mathbb{R}^n$ defined for every $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ by $\mathcal{A}_s(x) = (s(x_1), s(x_2), \ldots, s(x_n))$. Readily, the mean testing problem is invariant under the action of $\mathcal{A}$ in that $\mathcal{A}_s(\mathbf{Y}) = \mathcal{A}_s(\Theta) + \mathbf{X}'$ where $\mathbf{X}' = (\mathbf{X}'_1, \mathbf{X}'_2, \ldots, \mathbf{X}'_n) \sim \mathcal{N}(0, \mathbf{I}_n)$ is independent of $\mathcal{A}_s(\Theta)$. Therefore, $\mathcal{A}_s(\mathbf{Y})$ satisfies the same hypotheses as $\mathbf{Y}$. We also have $|\langle\mathcal{A}_s(\Theta)\rangle_n| = |\langle\Theta\rangle_n|$. Hence, the mean testing problem remains unchanged by substituting $\mathcal{A}_s(\Theta)$ for Θ and $\mathbf{X}'$ for $\mathbf{X}$. It is thus natural to seek $\mathcal{A}$-invariant tests, that is, tests $f \in \mathcal{M}(\mathbb{R}^n, 2)$ such that $f(\mathcal{A}_s(\mathbf{x})) = f(\mathbf{x})$ for any $s \in \mathcal{S}$ and any $\mathbf{x} \in \mathbb{R}^n$.

On the other hand, since we can reduce the noise variance by averaging observations, we consider $\mathcal{A}$-invariant integrator tests, that is, $\mathcal{A}$-invariant tests $f \in \mathcal{M}(\mathbb{R}^n, 2)$ for which exists $\overline{f} \in \mathcal{M}(\mathbb{R}^1, 2)$, henceforth called the reduced form of f, such that $f(\mathbf{x}) = \overline{f}(\langle\mathbf{x}\rangle_n)$ for any $\mathbf{x} \in \mathbb{R}^n$. Reduced forms of $\mathcal{A}$-invariant integrator tests are also $\mathcal{A}$-invariant: $\forall x \in \mathbb{R}$, $\forall s \in \mathcal{A}$, $\overline{f}(s(x)) = \overline{f}(x)$. We thus define $\mathbb{C}_\gamma^{[n]} \subset \text{Tests}_\gamma^{[n]}$ of all $\mathcal{A}$-invariant integrator tests. We thus have $f \in \mathbb{C}_\gamma^{[n]}$ if:

1. [**Size**]: $\alpha^{[n]}(f) \leqslant \gamma$;
2. [$\mathcal{A}$**-invariance**]: $\forall (s, \mathbf{x}) \in \mathcal{S} \times \mathbb{R}^n,\ f(\mathcal{A}_s(\mathbf{x})) = f(\mathbf{x})$;
3. [**Integration**]: $\exists \overline{f} \in \mathcal{M}(\mathbb{R}^1, 2),\ \forall \mathbf{x} \in \mathbb{R}^n,\ f(\mathbf{x}) = \overline{f}(\langle \mathbf{x} \rangle_n)$.

The following result derives from the foregoing and [47, 54]. The maximal element put forward in this proposition is called the Block-RDT test for the following reasons: it is basically an RDT test in the sense given to this term in [47, 54] and the adjective "block" reminds that this test exploits the "block" of n components of observation $\mathbf{Y}$. However, for the sake of shortening notation, we limit the notation of this test to remind that it is basically an RDT test.

Proposition 7.6.3 (Maximal element of $\left(\mathbb{C}_\gamma^{[n]},\ \leqslant^\diamond\right)$)

For any $\gamma \in [0,1]$ and any $n \in \mathbb{N}$,

$$\left\{ f_{n,\tau}^{\mathrm{RDT}(\gamma)} \right\} = \max \left(\mathbb{C}_\gamma^{[n]},\ \leqslant^\diamond \right) \tag{7.11}$$

where $f_{n,\tau}^{\mathrm{RDT}(\gamma)} \in \mathcal{M}(\mathbb{R}^n, 2)$ is defined by setting:

$$\forall (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n,\ f_{n,\tau}^{\mathrm{RDT}(\gamma)}(y_1, y_2, \ldots, y_n) = \begin{cases} 1 & \text{if } \left|\sum_{i=1}^n y_i\right| \leqslant \sqrt{n}\lambda_\gamma(\tau\sqrt{n}) \\ 0 & \text{otherwise} \end{cases}$$

and $\lambda_\gamma(\tau\sqrt{n})$ is the unique solution in x to the equation $Q_{1/2}(\tau\sqrt{n}, x) = \gamma$.

RDT and NP tests are structurally different because dedicated to two different testing problems and optimal with respect to two different criteria. This structural difference will be enhanced by coming back to our initial detection problem.

7.6.2 Application to detection

Consider again the problem of estimating $\varepsilon \in 2$, when we have a sequence $Y^{[q]} \in \mathrm{Seq}^{[q]}$ of observations. We thus have:

$$\forall n \in \mathbb{N}, \forall (\varepsilon, \omega) \in 2 \times \Omega, Y_n^{[q]}(\varepsilon, \omega) = \varepsilon + \Delta_n(\omega) + X_n(\omega) \tag{7.12}$$

where $X_1, X_2, \ldots \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $\forall n \in \mathbb{N},\ \Delta_n \in \mathcal{B}_\infty(q)$ with $q \in [0, \infty)$. The empirical mean of $Y^{[q]}$ satisfies: $\forall n \in \mathbb{N}, \langle Y^{[q]} \rangle_n(\varepsilon) = \langle Y^{[q]}(\varepsilon) \rangle_n = \varepsilon + \langle \Delta \rangle_n + \langle X \rangle_n$.

We thus have $|\langle \Delta \rangle_n| \leqslant q$ (a-s). Set $\Theta_n = \varepsilon + \Delta_n$ for every $n \in \mathbb{N}$. In the sequel, we

assume $q < 1/2$ because, in this case, we straightforwardly verify that:

$$\left\{\begin{array}{rcl} \varepsilon = 0 & \Leftrightarrow & |\langle\Theta\rangle_n| \leqslant q \\ \varepsilon = 1 & \Leftrightarrow & |\langle\Theta\rangle_n| \geqslant 1-q \end{array}\right. \tag{7.13}$$

Therefore, when $q \in \left[0, \frac{1}{2}\right[$, deciding on whether ε is zero or not when we are given $\mathbf{Y}_n^{[q]}(\omega)$ amounts to testing whether $|\langle\Theta\rangle_n(\omega)| \leqslant \tau$ or not for $\tau \in [q, 1-q]$. We thus can use the decision $f_{n,\tau}^{\mathrm{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)$, where $f_{n,\tau}^{\mathrm{RDT}(\gamma)}$ is given by Proposition 7.6.3.

We can calculate the false alarm probability of $f_{n,\tau}^{\mathrm{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)$ defined by (7.1) and the theoretical results in [47] yield that $\forall \tau \in [q, 1-q], \mathbb{P}_{\mathrm{FA}}\left[f_{n,\tau}^{\mathrm{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)\right] \leqslant \gamma$.

In the sequel, for the sake of simplifying notation, we assume that both τ and q are in $\left[0, \frac{1}{2}\right[$. In this case, we have:

$$\forall \tau \in \left[0, \frac{1}{2}\right[, \left\{\begin{array}{l} \mathrm{Sel}_\gamma\left(f_{n,\tau}^{\mathrm{RDT}(\gamma)}\right) = [0, \tau] \\ \mathrm{Lnd}_\gamma\left(f_{n,\tau}^{\mathrm{RDT}(\gamma)}\right) = \displaystyle\bigcup_{q \in [0,\tau]} \left\{f_{n,\tau}^{\mathrm{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right) : Y^{[q]} \in \mathrm{Seq}^{[q]}\right\} \end{array}\right.$$

We can then state the following lemma, which is the counterpart to Lemma 7.5.1.

Theorem 7.6.4 (Maximality of RDT decisions)

For any $\gamma \in [0,1]$, any $n \in \mathbb{N}$ and any $0 \leqslant q \leqslant \tau < 1/2$, $\mathrm{Lnd}_\gamma\left(f_{n,\tau}^{\mathrm{RDT}(\gamma)}\right) = \max\left(\mathrm{Lndscps}_\gamma^{[0,\tau]}\left(\mathbb{C}_\gamma^{[n]}\right), \leqslant^*\right)$.

Proof. It results from Definition 7.4.2 that $\mathrm{Lnd}_\gamma(f) = \left\{f\left(\mathbf{Y}_n^{[q]}\right) : Y^{[q]} \in \mathrm{Seq}^{[q]}, q \in [0,\tau]\right\}$.

According to (7.6), we also have $\mathrm{Lndscps}_\gamma^{[0,\tau]}\left(\mathbb{C}_\gamma^{[n]}\right) = \left\{\mathrm{Lnd}_\gamma(f) \in \mathrm{Lndscps}_\gamma : f \in \mathbb{C}_\gamma^{[n]}, \mathrm{Sel}_\gamma(f) = [0,\tau]\right\}$.

Given $q \in [0,\tau]$ and $Y^{[q]} \in \mathrm{Seq}^{[q]}$, set:

$$\left\{\begin{array}{rcl} \mathbf{Y} & = & \mathbf{Y}_n^{[q]} = \left(Y_1^{[q]}, Y_2^{[q]}, \ldots, Y_n^{[q]}\right) \text{ (see (7.3))} \\ \mathbf{X} & = & (X_1, X_2, \ldots, X_n) \sim \mathcal{N}(0, \mathbf{I}_n) \\ \Theta & = & (1+\Delta_1, 1+\Delta_2, \ldots, 1+\Delta_n) \end{array}\right.$$

We basically have $\mathbf{Y} = \Theta + \mathbf{X}$. Consider now the mean testing problem (7.8) with

Θ, $\mathbf{X}$ and $\mathbf{Y}$ defined as above. For any $f \in \mathcal{M}(\mathbb{R}^n, 2)$, it follows from Eqs. (7.12) , (7.13), (7.2) and (7.9) that:

$$\beta_{\Theta}^{[n]}(f) = \mathbb{P}_{\text{DET}}\left[f\left(\mathbf{Y}_n^{[q]}\right)\right] \tag{7.14}$$

Suppose now that $f \in \mathbb{C}_{\gamma}^{[n]}$ with $\text{Sel}_{\gamma}(f) = [0, \tau]$. We derive from Proposition 7.6.3, Equation (7.14) and its application to $f_{n,\tau}^{\text{RDT}(\gamma)}$, that:

$$\mathbb{P}_{\text{DET}}\left[f\left(\mathbf{Y}_n^{[q]}\right)\right] \leqslant \mathbb{P}_{\text{DET}}\left[f_{n,\tau}^{\text{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)\right]$$

Since $q \leqslant \tau < 1/2$ implies that $q \in \text{Sel}_{\gamma}(f)$ and since $\text{Sel}_{\gamma}(f) = \text{Sel}_{\gamma}\left(f_{n,\tau}^{\text{RDT}(\gamma)}\right) = [0, \tau]$, we can rewrite the foregoing equality as $f\left(\mathbf{Y}_n^{[q]}\right) \preceq f_{n,\tau}^{\text{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)$.

This holding true for any $q \in \text{Sel}_{\gamma}(f)$, any $Y^{[q]} \in \text{Seq}^{[q]}$ and since f and $f_{q,n}^{\text{RDT}(\gamma)}$ have same selectivity $[0, \tau]$, we derive from the foregoing and Definition 7.8.2 that $\text{Lnd}_{\gamma}(f) \leqslant^{*} \text{Lnd}_{\gamma}\left(f_{n,\tau}^{\text{RDT}(\gamma)}\right)$. □

We now prove that the oracles with level γ are approximated by RDT decisions.

Lemma 7.6.5 (Approximation of oracles with γ by RDT)

Setting $\text{Lnd}_{\tau}^{\text{RDT}(\gamma)} = \left\{\text{Lnd}_{\gamma}\left(f_{n,\tau}^{\text{RDT}(\gamma)}\right) : n \in \mathbb{N}\right\}$ for any given $\gamma \in [0, 1]$, we have:

$$\begin{aligned}\text{Lnd}\mathbb{O}_{\gamma} &= \text{Upper}\left(\text{Lnd}_{\tau}^{\text{RDT}(\gamma)}, \left(\mathcal{P}(\text{Dec}_{\gamma}), \leqslant^{*}\right)\right) \\ &= \sup\left(\text{Lnd}_{\tau}^{\text{RDT}(\gamma)}, \left(\mathcal{P}(\text{Dec}_{\gamma}), \leqslant^{*}\right)\right)\end{aligned}$$

Proof. According to (7.2) and [47, Theorem 2], we obtain:

$$\forall (q, \tau) \in \left[0, \tfrac{1}{2}\right[\times \left[0, \tfrac{1}{2}\right[, \forall n \in \mathbb{N}, \mathbb{P}_{\text{DET}}\left[f_{n,\tau}^{\text{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)\right] \geqslant Q_{1/2}\left((1-q)\sqrt{n}, \lambda_{\gamma}(\tau\sqrt{n})\right)$$

In [57], we proved that $\lim_{\sigma \to 0} Q_{1/2}(\rho/\sigma, \lambda_{\gamma}(\tau/\sigma)) = \mathbb{1}_{]\tau, \infty[}(\rho)$. Therefore, since $\tau < 1-q$, we have $\lim_{n \to \infty} Q_{1/2}\left(\sqrt{n}(1-q), \lambda_{\gamma}\left(\tau\sqrt{n}\right)\right) = 1$. Thus, $\lim_{n \to \infty} \mathbb{P}_{\text{DET}}\left[f_{n,\tau}^{\text{RDT}(\gamma)}\left(\mathbf{Y}_n^{[q]}\right)\right] = 1$.

The set $\text{Lnd}_{\tau}^{\text{RDT}(\gamma)} \subset \text{Lndscps}_{\gamma}$ satisfies Theorem 7.4.5 conditions for any $\gamma \in [0, 1]$ and $\Xi_{\gamma} = \{\tau\}$. □

7.7 Multiplicity Principle in $(\mathcal{P}(\mathrm{Dec}_\gamma), \preccurlyeq^*)$

To state the MP in $(\mathcal{P}(\mathrm{Dec}_\gamma), \preccurlyeq^*)$, we need the following lemma.

Lemma 7.7.1 (Selectivity of NP tests)

$\forall n \in \mathbb{N}, \mathrm{Sel}_\gamma\left(f_n^{\mathrm{NP}(\gamma)}\right) = \{0\}$

Proof. A consequence of [47, Section B., p. 6.]. □

We have now all the material to state the main result.

Theorem 7.7.2 (Multiplicity Principle in $(\mathcal{P}(\mathrm{Dec}_\gamma), \preccurlyeq^*)$)

For any given $\tau \in \left[0, \frac{1}{2}\right[$, the MP is satisfied in $(\mathcal{P}(\mathrm{Dec}_\gamma), \preccurlyeq^*)$ by the pair $\left(\mathrm{Lnd}^{\mathrm{NP}(\gamma)}, \mathrm{Lnd}_\tau^{\mathrm{RDT}(\gamma)}\right)$.

Proof. According to Theorems 7.5.2 and 7.6.4, the subsets $\mathrm{Lnd}^{\mathrm{NP}(\gamma)}$ and $\mathrm{Lnd}_\tau^{\mathrm{RDT}(\gamma)}$ of $\mathcal{P}(\mathrm{Dec}_\gamma)$ are such that

$$\sup\left(\mathrm{Lnd}^{\mathrm{NP}(\gamma)}, \left(\mathcal{P}(\mathrm{Dec}_\gamma), \preccurlyeq^*\right)\right) = \sup\left(\mathrm{Lnd}_{,}^{\mathrm{RDT}(\gamma)}\left(\mathcal{P}(\mathrm{Dec}_\gamma), \preccurlyeq^*\right)\right) = \mathrm{Lnd}\mathbb{O}_\gamma$$

In addition, Lemma 7.7.1 implies that $\mathrm{Lnd}^{\mathrm{NP}(\gamma)} \times \mathrm{Lnd}_\tau^{\mathrm{RDT}(\gamma)} \cap \preccurlyeq^* = \varnothing$. The conclusion follows from Corollary 7.2.2. □

7.8 MP in a slightly different preorder

The preorder $\preccurlyeq^*$ may seem very hard to grasp. This is because, by essence, tests are functions $\mathbb{R}^n \to 2$, while oracles are decisions, that is, functions $2 \times \Omega \to 2$.

Thus, we need to use a trick in order to compare what is comparable: the landscape of a test f is the set of all its decisions of level γ. And oracles are decisions of level γ. This is where the trick lies.

In this section, we introduce an equivalent preorder that gets rid of this trick, and compares tests and oracles together.

Definition 7.8.1 (SEL)

We denote by SEL_γ the set of tests f such that $\mathrm{Sel}_\gamma(f) \neq \varnothing$.

We then redefine the preorder $\preccurlyeq^*$:

Definition 7.8.2 (Preorder on landscapes)

Let $\mathcal{P}(\mathrm{Dec}_\gamma)$ be the power set of Dec_γ (set of decisions). We define the preorder $\preccurlyeq^*$ over $\mathcal{P}(\mathrm{Dec}_\gamma)$ as: $D \preccurlyeq^* D'$ iff one of the following occurs:

1. $D = D'$
2. $\exists f \in \mathcal{M}(\mathbb{R}^n, 2)$, $D = \mathrm{Lnd}_\gamma(f)$ and $D' = \mathbb{O}_\gamma$
3. $\exists f, f' \in \mathcal{M}(\mathbb{R}^n, 2)$ $D = \mathrm{Lnd}_\gamma(f)$, $D' = \mathrm{Lnd}_\gamma(f')$ such that $\mathrm{Sel}_\gamma(f) = \mathrm{Sel}_\gamma(f')$ and $\mathbb{P}_{\mathrm{FA}}[f] \leqslant \mathbb{P}_{\mathrm{FA}}[f']$

The preorder introduced there makes sense. The goal is to compare tests in a meaningful way (i.e. tests must have the same selectivity before being compared by their probability of detection, meaning that they are made for the same problem).

Note that, in this preorder, the only sets that are comparable are landscapes, and the set of oracles. The other subsets of Dec_γ are not comparable (e.g. intersections or unions of landscapes, sets that contain both oracles and decisions), so it feels like this preorder considers only a small part of the whole set. This might reveal the idea behind this preorder: find a trick to compare tests and oracles.

Denote by $\mathrm{Core}_\gamma = \{D \in \mathcal{P}(\mathrm{Dec}_\gamma) \mid D \preccurlyeq^* \mathbb{O}_\gamma\}$. This is basically the main connected component of the preorder $\mathcal{P}(\mathrm{Dec}_\gamma)$. Note that Core_γ can be thought as a kind of fiber over all the oracles (if $\preccurlyeq^*$ was a function instead of a preorder).

Proposition 7.8.3

Let $D \in \mathcal{P}(\mathrm{Dec}_\gamma)$.
If $D \notin \mathrm{Core}_\gamma$, then the set $\{D' \mid D \preccurlyeq^* D' \text{ or } D' \preccurlyeq^* D\}$ is reduced to the singleton $\{D\}$.

Proof. Let D' such that $D \preccurlyeq^* D'$. By Definition 7.8.2, one of the three conditions happen. 2 and 3 imply that $D \in \mathrm{Core}_\gamma$, which contradicts the assumption. Necessarily $D' = D$. □

In the following, we give simpler preorder that is equivalent (in terms of categories) to Core_γ.

Consider the set $\mathrm{SEL}_\gamma + \mathbb{O}_\gamma$ (disjoint union of tests and oracles), together with the following preorder $\lhd$:

1. for all tests $f, f' \in \mathcal{M}(\mathbb{R}^n, 2)$, $f \lhd f' \Leftrightarrow \mathrm{Sel}_\gamma(f) = \mathrm{Sel}_\gamma(f')$ and for all $Y^{[q]} \in \mathrm{Seq}^{[q]}$, $\mathbb{P}_{\mathrm{FA}}\left[f\left(Y^{[q]}_{[n]}\right)\right] \leqslant \mathbb{P}_{\mathrm{FA}}\left[f'\left(Y^{[q]}_{[n]}\right)\right]$.

2. for all test or oracle $g \in \mathrm{SEL}_\gamma + \mathbb{O}_\gamma$, for all oracle $O \in \mathbb{O}_\gamma$, $g \triangleleft O$

It is easy to see that $f \triangleleft f' \Leftrightarrow \mathrm{Lnd}_\gamma(f) \leqslant^* \mathrm{Lnd}_\gamma(f')$. Therefore, the following functor is full and faithful:

$$I : \begin{cases} \mathrm{SEL}_\gamma + \mathbb{O}_\gamma & \longrightarrow & \mathcal{P}(\mathrm{Dec}_\gamma) \\ f \in \mathrm{SEL}_\gamma & \longmapsto & \mathrm{Lnd}_\gamma(f) \\ o \in \mathbb{O}_\gamma & \longmapsto & \mathbb{O}_\gamma \end{cases} \tag{7.15}$$

Besides, considering that the image of I is Core_γ, I seen as a functor $\mathrm{SEL}_\gamma + \mathbb{O}_\gamma \to \mathrm{Core}_\gamma$ becomes surjective. Thus, I is part of an equivalence of categories .

Theorem 7.8.4

$(\mathrm{SEL}_\gamma + \mathbb{O}_\gamma, \leqslant^*)$ and $(\mathrm{Core}_\gamma, \triangleleft)$ are equivalent.

This says that the two preorders have the same "structure": parallel branches of tests of the same selectivity that meet at their extremum, the oracles. Except that $\triangleleft$ uses only probabilities of detection and selectivity, while $\leqslant^*$ also uses the obscure notion of landscape.

Then, the subpreorder consisting of RDT tests and that of NP tests are homologous (in fact, the cocones are, here, exactly the colimits) and there is no cluster between them because RDT tests and NP tests are on disjoint chains of the preorder, hence:

Theorem 7.8.5

$\mathrm{SEL}_\gamma + \mathbb{O}_\gamma$ verifies the MP.

Note that, in general, the MP does not transfer to equivalent categories. Homologous diagrams in a category do not need to be homologous in an equivalent category. It is the case for the equivalence $\mathrm{SEL}_\gamma + \mathbb{O}_\gamma \sim \mathrm{Core}_\gamma$, because the cocones of the diagrams made by the subpreorder consisting of RDT tests, and NP tests, are exactly the colimits. In other words, in both preorders, the upper bounds are the suprema.

7.9 Conclusions and Perspectives

In this chapter, via the framework provided by the Multiple Principle (MP), which is motivated by the concept of degeneracy in biology, and by introducing the notions of test landscapes and selectivity, we have established that this principle

is satisfied when we consider the standard NP tests and the RDT tests applied to a detection problem. One interest of this result is that it opens prospects on the construction of Memory Evolutive Systems [12, 45] via tests.

Here, the multiplicity suggests a functional equivalence between the two classes of statistical tests, although being designed for different statistical problems and having different properties. The NP tests are optimal among all tests, however they need to be closely adapted to the signal they are searching for. RDT tests are optimal, but for a different property, and they are robust. Thus, this result suggests that a combination of RDT and NP tests should lead to a resilient detection system.

More elaborated statistical decision problems should be considered beyond this preliminary work. Sequential tests are particularly appealing because they collect information till they can decide with guaranteed performance bounds. On the one hand, the Sequential Probability Ratio Test (SPRT) established in [58] is proved to be optimal; on the other hand, in [57], we have exhibited non-optimal tests with performance guarantees in presence of interferences. In the same way as NP and RDT tests satisfy PM, we conjecture that these two types of sequential tests satisfy MP as well.

Part IV

Conclusion

Chapter 8

Conclusion

I am released from this world of servile obligation

Nile (2012) *Supreme humanism of megalomania*, At the Gate of Sethu.

8.1 Summary

This manuscript aimed at presenting my work during the last three years towards the resilience of sensor networks.

This work is mainly based on category theory. As the expected audience of this thesis is multidisciplinary, or from signal processing, we dedicated Chapter 2 to the introduction of all the necessary material from category theory. This chapter can serve as a base for any student or interested reader as a very brief and general tutorial for category theory, as most of the basics were introduced: categories, functors, natural transformations, limits/colimits and common examples. However, this chapter is far from exhaustive, as many notions, such as adjoints, monads, presheaves, topoi, and so on, are not mentioned in these pages, because they were not relevant for this work.

As observed in [15], communication is key in the memory of the immune system. Chapter 3 retrieves this result in the theory of dynamical systems, here seen as automata. Using a categorical framework developed in [19], we prove that any dynamical system is equivalent to a system without memory (without consideration of the current state) with the right connections.

The following three chapters then use a different approach of category theory, following Andrée Ehresmann and Jean-Paul Vanbremeersch's work [12] on the modelling of the complexity of systems. A system is viewed as a set of categories.

This modelling relies on the notion of clusters, which are the arrows of the ind-category, which we remind in Chapter 4. We then give a few equivalent definitions of clusters, one of which we use in the next chapter. Chapter 5 checks how we can construct a cluster. Given two diagrams and the structure of the category, Theorems 5.3.5 and 5.3.11 give us some conditions to determine whether there is

a cluster between those diagrams. This result is interesting by itself, but we also use it in the next chapter. It constitutes a subcase of the Multiplicity Principle, the categorical counterpart of degeneracy, the functional redundancy that occurs at every level in biological systems, and that induces resilience in the whole system. Theorem 5.3.26 and Proposition 5.4.3 are used as part of the main theorems of Chapter 6. That theorem provides a characterisation preorder categories that verify the Multiplicity Principle.

Chapter 7 uses this result to find multiplicity between Neymann-Pearson (NP) tests and RDT tests. Here, the multiplicity shows a functional equivalence between the two classes of statistical tests, although being designed for different statistical problems and having different properties. The NP tests are optimal among tests, however they need to be closely adapted to the signal they are searching for. RDT tests are optimal, but for a different property, and they are robust. Thus, this result suggests that a combination of RDT and NP tests should lead to a resilient detection.

8.2 Contributions

My contributions were already listed in Section 1.4 in the Introduction chapter. We remind them here too. We also mention related work produced during this PhD that lead to drafts published on the HAL platform.

Chapter 2 is a basic introduction to category theory; no result introduced here is new.

Chapter 3 is based on an article we published [18]. The categorical framework for dynamical systems is based on a previous work from one of the coauthors [19]. Section 3.4 only contains new results.

The following chapters tackle the topic of clusters: their definitions, their existence. Chapter 4 is a compilation of existing work from renown categoricists (the link with ind-categories, the description of arrows of pro-categories). The two descriptive definitions of clusters as sets of arrows, and the link with the presheaf of connected components are new. The conjecture on the cardinal of clusters has, as far as I know, never been tackled yet.

Chapters 5 and 6 mainly consist of new results.

Chapter 7 consists of a brief introduction to statistical decision testing, and the results presented here are from our conference article [20].

Furthermore, the new results introduced in Chapters 4, 5 and 6 are summarized in an article to be submitted to the journal *Theory and Applications of Categories*.

My work on the thesis also includes the following.

Chapter 2 is an excerpt of a crash course on category theory that Dominique Pastor and myself started, as a part of my training on the topic. A draft of this crash course can be found online on the HAL platform [59]. This crash course, though incomplete due to being a draft, covers most basics on category theory, including topics we did not tackle in this manuscript: adjoints, monads, topoi. None of the proofs are eluded, and an effort is made on pedagogy: proofs are as explicit as possible, and numerous examples are given. As such, this crash course is suitable for students or scientists willing to discover category theory without prior knowledge.

As we mentioned, the crash course introduced topoi, while topos theory is completely unrelated to the work presented in this manuscript. Topoi ware a former trail of ours due to the logical aspects they induced, and we thought about describing the resilience of a system using logical formulas. This work has not come to a successful conclusion (yet...), but it lead, once again, to a first draft on this topic [60]. In this draft, we study the logic behind the presheaves induced by p-values.

Here are all my publications:

E. Beurier, D. Pastor, D. I. Spivak , "Memoryless Systems Generate the Class of all Discrete Systems ," *International Journal of Mathematics and Mathematical Sciences* , 2019. Available online: https://www.hindawi.com/journals/ijmms/2019/6803526/

D. Pastor, E. Beurier, A. C. Ehresmann, R. Waldeck , " Interfacing biology, category theory and mathematical statistics ," *ACT 2019: Applied Category Theory Conference* , Oxford, United Kingdom, July, 2019. Available online: http://eptcs.web.cse.unsw.edu.au/paper.cgi?ACT2019.9.pdf

E. Beurier, D. Pastor, D. I. Spivak, R. Waldeck , "Memoryless systems generate the class of all discrete systems - Extended abstract ," *ACT 2019: Applied Category Theory Conference* , Oxford, United Kingdom, July, 2019. Preprint available online: https://hal-imt-atlantique.archives-ouvertes.fr/hal-02173177

D. Pastor, E. Beurier, A. Ehresmann, R. Waldeck , "A mathematical approach to resilience ," *iTWIST'20 (international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques)* , Jun 2020, Nantes, France. Available online: https://arxiv.org/abs/2009.09666v1

E. Beurier, D. Pastor, A. Ehresmann, R. Waldeck , " On the definition and existence of clusters, and their application to the Multiplicity Principle ," *To be submitted to Theory and Applications of Categories* , Not available yet.

8.3 Perspectives

A good part of this thesis was dedicated to the study of clusters and the Multiplicity Principle. Although the results in Chapters 4 and 5 hold in general categories, the results of Chapter 6 focus only on preorder categories. A characterisation of MP in the same vein could be given for other classes of categories, among which: filtered categories, finite categories, free categories over graphs.

Although minor, in Chapter 4, we give a conjecture on the cardinal of clusters between two diagrams, depending on the cardinal of connected components of two categories. This conjecture seems fairly accessible.

In Chapter 5, we give a conjecture on the equivalence between the Clawitzer diagram and the absence of clusters. This result would make it very easy to check for clusters between two diagrams. However, the conjecture needs a a further study, not only to prove or disprove it, but also, on the formalisation.

A few conjectures are left unanswered too; there remains to find an easier characterisation of the existence of clusters (in the same vein of the so called CCCT's - Theorems 5.3.5, 5.3.11). The algorithm described in Theorem 5.3.26 is requires heavy calculations and is impractical.

This work originated from the intention of providing multidisciplinary theoretical tools for the study and design of resilient systems. The study of clusters and MP provided here should serve as a base for future work on the study of resilience, furthermore for our first goal, the resilience of sensor networks. More generally, our work could be used in the design and study of more general resilient systems, be they immune, social, or biological in a broad sense. The resilience of sensor networks could also be adapted to the study of resilient systems for cybersecurity. In other words, and if the reader allows me the temerity of the following words, the goal of this thesis was to explore and bring the foundations of a new branch of mathematics we may call the mathematics of resilience.

Index

Nomenclature

$A \times_C B$ pullback of $f : A \to C$ and $g : B \to C$ p. 59

$B +_A C$ pushout of $f : A \to B$ and $g : A \to C$ p. 61

$X \boxplus Y$ parallel composition of $\mathscr{C}$-boxes X and Y p. 82

$\varphi \boxplus \psi$ parallel composition of wiring diagrams φ and ψ p. 83

$\square$ closed box with no input nor output ports p. 83

$F_0 \boxtimes F_1$ parallel composition of the discrete systems F_0 and F_1 p. 88

$\mathbf{DS}(X_0) \boxtimes \mathbf{DS}(X_1)$ parallel composition of the sets of discrete systems $\mathrm{DS}(X_0)$ and $\mathrm{DS}(X_1)$ p. 88

$F \equiv G$ transducer equivalence between F and G (equivalence in output streams) p. 90

$F \vdash G$ simulation relation between F and G; F simulates G p. 91

$F \sim G$ internal equivalence relation between F and G; F simulates G and conversely p. 92

$A \sqsubseteq B$ inclusion of A into B up to equivalence (functors or mappings) p. 93

$A \approx B$ equality of A and B up to equivalence (functors or mappings) p. 93

$D \sqsubseteq E$ inclusion of D into E up to equivalence (subset) p. 93

$D \approx E$ equality of D and E up to equivalence (subset) p. 93

$C \sim C'$ zigzag equivalence between C and C' p. 107

(a, b) tuple representing an arrow of the comma-category $(P \mid Q)$ p. 108

$\mathscr{C}$ the general category pp. 30, 107

cod codomain function p. 30

$\mathscr{C}^{\mathbf{op}}$ opposite category p. 37

$\mathbf{Cocones}(P)$ category of cocones from P pp. 47, 174

$\mathbf{Cones}(P)$ category of cones from P p. 47

$\mathrm{Colim}(P)$ colimit cocone of the diagram P p. 48

cP peak of the colimit cocone of the diagram P p. 48

$\mathbf{Clos}(B)$ closure of B: the smallest subalgebra cointaining B p. 94

dom domain function p. 30

Δ diagonal functor p. 46

$\mathbf{DS}$ functor associating to a box all the discrete systems that fit in p. 85

$\mathbf{DS}^{\mathbf{ML}}(X)$ set of memoryless systems that fit in the box X p. 95

$\mathbf{DS}^{\mathbf{ML}}_{\mathbf{out}}(X)$ set of direct-output systems that fit in the box X p. 96

$\mathbf{DS}_{\mathbf{Fin}}(X)$ set of finite-state systems that fit in the box X p. 99

Bibliography

[1] S. Eilenberg and S. MacLane, "General theory of natural equivalences," *Transactions of the American Mathematical Society*, vol. 58, pp. 231–294, 1945. [Online]. Available: https://www.ams.org/journals/tran/1945-058-00/S0002-9947-1945-0013131-6/S0002-9947-1945-0013131-6.pdf

[2] J.-P. Marquis, "Category theory and the foundations of mathematics: Philosophical excavations," *Synthese*, vol. 103, no. 3, pp. 421–447, 1995. [Online]. Available: https://doi.org/10.1007/BF01089735

[3] D. Ernadote, "An ontology mindset for system engineering," in *2015 IEEE International Symposium on Systems Engineering (ISSE)*, 2015, pp. 454–460.

[4] M. A. Mabrok and M. J. Ryan, "Category theory as a formal mathematical foundation for model-based systems engineering," *Applied Mathematics and Information Sciences*, vol. 11, no. 1, pp. 43–51, Jan 2017. [Online]. Available: http://dx.doi.org/10.18576/amis/1101

[5] D. I. Spivak, "The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits," ePrint online: www.arXiv.org/abs/arXiv:1305.0297, 2013.

[6] C. A. R. Hoare, "Notes on an approach to category theory for computer scientists," in *Constructive Methods in Computing Science*, M. Broy, Ed., vol. 55. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 245–305.

[7] B. Milewski, "Category theory for programmers," Last version of unofficial pdf available at: https://github.com/hmemcpy/milewski-ctfp-pdf/releases/download/v1.3.0/category-theory-for-programmers.pdf, based on Bartosz Milewski's blog https://bartoszmilewski.com/2014/10/28/category-theory-for-programmers-the-preface/, 2019.

[8] The Haskell Community, "Haskell. An advanced, purely functional programming language," Available at https://www.haskell.org/ (last visited 2020-05-01), 2020.

[9] R. Brown and T. Porter, "Category theory and higher dimensional algebra: potential descriptive tools in neuroscience," 2008.

[10] R. Rosen, "The representation of biological systems from the standpoint of the theory of categories," *The bulletin of mathematical biophysics*, vol. 20, pp. 317–341, December 1958.

[11] ——, "A relational theory of biological systems II," *The bulletin of mathematical biophysics*, vol. 21, pp. 109–128, June 1959.

[12] A. Ehresmann and J.-P. Vanbremeersch, *Memory Evolutive Systems; Hierarchy, Emergence, Cognition*, 1st ed., ser. Studies in multidisciplinarity. Elsevier, 2007, vol. 4.

[13] R. Seaton, "Why category theory matters," Post on his blog: https://rs.io/why-category-theory-matters/, 2014.

[14] D. I. Spivak, *Category Theory for the Sciences*. MIT Press, 2014.

[15] B. Bellier, V. Thomas-Vaslin, M.-F. Saron, and D. Klatzmann, "Turning immunological memory into amnesia by depletion of dividing t cells," *Proceedings of the National Academy of Sciences*, vol. 100, no. 25, pp. 15 017–15 022, 2003. [Online]. Available: https://www.pnas.org/content/100/25/15017

[16] A. Ehresmann and J.-P. Vanbremeersch, "Hierarchical evolutive systems: A mathematical model for complex systems," *Bulletin of Mathematical Biology*, vol. 49, no. 1, pp. 13 – 50, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0092824087800332

[17] G. M. Edelman and J. A. Gally, "Degeneracy and complexity in biological systems," *Proceedings of the National Academy of Sciences*, vol. 98, no. 24, pp. 13 763–13 768, 2001. [Online]. Available: https://www.pnas.org/content/98/24/13763

[18] E. Beurier, D. Pastor, and D. I. Spivak, "Memoryless Systems Generate the Class of all Discrete Systems," *International Journal of Mathematics and Mathematical Sciences*, p. 10, 2019. [Online]. Available: https://www.hindawi.com/journals/ijmms/2019/6803526/

[19] D. I. Spivak, "The steady states of coupled dynamical systems compose according to matrix arithmetic," Available online: https://arxiv.org/abs/1512.00802, 2016.

[20] D. Pastor, E. Beurier, A. Ehresmann, and R. Waldeck, "Interfacing biology, category theory and mathematical statistics," in Proceedings *Applied Category Theory 2019,* University of Oxford, UK, 15-19 July 2019, ser. Electronic Proceedings in Theoretical Computer Science, J. Baez and B. Coecke, Eds., vol. 323. Open Publishing Association, 2020, pp. 136–148.

[21] G. Moore and N. Seiberg, "Classical and quantum conformal field theory," *Communications in Mathematical Physics*, vol. 123, no. 2, pp. 177–254, 1989. [Online]. Available: https://projecteuclid.org:443/euclid.cmp/1104178762

[22] S. Awodey, *Category Theory*, 2nd ed., ser. Oxford Logic Guides. Oxford University Press, Oxford, 2010, vol. 52.

[23] E. Riehl, *Category theory in context*, 1st ed. Cambridge University Press, 2014.

[24] T. Leinster, *Basic Category Theory*, ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2016, vol. 143. [Online]. Available: www.arXiv.org/abs/arXiv:1612.09375

[25] M. Barr and C. Wells, *Category Theory for Computing Science*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[26] S. MacLane, *Categories for the Working Mathematician*, 2nd ed., ser. Graduate Texts in Mathematics. Springer-Verlag, New York, 1998, vol. 5.

[27] K. Kunen, *Set theory - An introduction to independence proofs*, 7th ed., ser. Studies in logic and the foundations of mathematics. North-Holland Publishing Company, 1999, vol. 102.

[28] D. E. Speyer, "What's a reasonable category that is not locally small?" Question asked by aorq, replied by David E. Speyer; last accessed: 08-november-2018: https://mathoverflow.net/questions/3278/whats-a-reasonable-category-that-is-not-locally-small, 2009.

[29] F. Borceux, *Handbook of Categorical Algebra*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1994, vol. 2.

[30] nLab authors, "universal construction," http://ncatlab.org/nlab/show/universal%20construction, Mar. 2020, Revision 24.

[31] S. J. Russel and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed., ser. Prentice Hall series in artifical intelligence. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010.

[32] M. Behrisch, S. Kerkhoff, R. Pöschel, F. M. Schneider, and S. Siegmund, "Dynamical systems in categories," *Applied Categorical Structures*, vol. 25, no. 1, pp. 29–57, Feb 2017. [Online]. Available: https://doi.org/10.1007/s10485-015-9409-8

[33] M. Sipser, *Introduction to the theory of computation*, 3rd ed. Cengage Learning, 2012.

[34] J. Adamek and V. Trnkova, *Automata and Algebras in Categories*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1990.

[35] A. Grothendieck and J. L. Verdier, *Théorie des Topos et Cohomologie Etale des Schémas (Séminaire de Géométrie algébrique IV-1)*, 2nd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1972, vol. 1.

[36] nLab authors, "accessible category," http://ncatlab.org/nlab/show/accessible%20category, Mar. 2020, Revision 47.

[37] A. Deleanu and P. Hilton, "Borsuk shape and a generalization of grothendieck's definition of pro-category," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 79, no. 3, p. 473–482, 1976.

[38] A. Ehresmann, "On /102/ : Sur l'existence de structures libres et de foncteurs adjoints." *Charles Ehresmann: oeuvres complètes et commentées*, vol. IV-1, pp. 361–377, 1981.

[39] nLab authors, "ind-object," http://ncatlab.org/nlab/show/ind-object, Dec. 2019, Revision 64.

[40] ——, "pro-object," http://ncatlab.org/nlab/show/pro-object, Jan. 2020, Revision 50.

[41] M. Kashiwara and P. Schapira, *Categories and Sheaves*, 1st ed., ser. Grundlehren der mathematischen Wissenschaften. Springer-Verlag Berlin Heidelberg, 2006, vol. 332.

[42] nLab authors, "equivalence of categories," http://ncatlab.org/nlab/show/equivalence%20of%20categories, Jan. 2020, Revision 39.

[43] T. Porter, "Essential properties of pro-objects in grothendieck categories," *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, vol. 20, no. 1, pp. 3–57, 1979. [Online]. Available: http://www.numdam.org/item/CTGDC_1979___20_1_3_0

[44] A. C. Ehresmann and J.-P. Vanbremeersch, "Multiplicity principle and emergence in memory evolutive systems," *Syst. Anal. Model. Simul.*, vol. 26, no. 1-4, pp. 81–117, Dec. 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=246318.246332

[45] A. Ehresmann and J.-P. Vanbremeersch, "MES: A mathematical model for the revival of natural philosophy," *Philosophies*, vol. 4, no. 1, pp. 9–0, 2019. [Online]. Available: https://www.mdpi.com/2409-9287/4/1/9

[46] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, pp. 289–337, 1933. [Online]. Available: http://www.jstor.org/stable/91247

[47] D. Pastor and Q.-T. Nguyen, "Random distortion testing and optimality of thresholding tests," *IEEE Transactions on Signal processing*, vol. 61, no. 16, pp. 4161–4171, 2013.

[48] D. Marker, *Model theory: an introduction*, 1st ed., ser. Graduate Texts in Mathematics. Springer-Verlag New York, 2002, vol. 217.

[49] S. Shelah, *Classification Theory and the Number of Non-Isomorphic Models*, 2nd ed., ser. Studies in Logic and the Foundations of Mathematics. North-Holland, 1990, vol. 92.

[50] R. Kaye, *Models of Peano Arithmetic*, ser. Oxford Logic Guides. Clarendon Press, 1991, vol. 15.

[51] A. Bovykin, "On order-types of models of arithmetic," PhD thesis, available online at https://logic.pdmi.ras.ru/~andrey/phd.pdf, 2000.

[52] F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel, *Robust Statistics: the Approach based on Influence Functions*. John Wiley and Sons, New York, 1986.

[53] E. L. Lehmann and J. P. Romano, *Testing statistical hypotheses*, 3rd ed. Springer, 2005.

[54] D. Pastor and F.-X. Socheleau, "Random distortion testing with linear measurements," *Signal Processing*, vol. 145, pp. 116 – 126, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168417304127

[55] A. A. Borovkov, *Mathematical statistics*. Gordon and Breach Science Publishers, 1998.

[56] M. L. Eaton, *Multivariate statistics. A vector space approach*. Wiley, 1983. [Online]. Available: https://projecteuclid.org/euclid.lnms/1196285102

[57] P. Khanduri, D. Pastor, V. Sharma, and P. K. Varshney, "Sequential Random Distortion Testing of Non-Stationary Processes," *IEEE Transactions on Signal Processing (accepted with minor revisions)*, 2019.

[58] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 06 1945. [Online]. Available: http://www.jstor.org/stable/2235829

[59] D. Pastor and E. Beurier, "A crash course on Category Theory," IMT Atlantique, Research Report IMTA-RR-2019-01-SC, Jul. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02190027

[60] E. Beurier and D. Pastor, "p-value presheaves," IMT Atlantique, Research Report IMTA-RR-2019-02-SC, Jul. 2019, cette nouvelle version introduit de nouveaux éléments techniques, comme indiqué dans la nouvelle introduction. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02190029

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Caractérisation des organisations pour la détection résiliente de menaces

Mot clés : Résilience, théorie des catégories, cluster, multiplicité, composantes connexes

Résumé : Le point de départ de cette thèse est les réseaux de capteurs, et comment les rendre résilients. Notre approche utilise le langage de la théorie des catégories.

Nous abordons en premier lieu l'usage de systèmes dynamiques, et leur composition. Il s'avère que chaque système dynamique peut être décomposé en systèmes plus simples, dits réactifs, qui pourraient être des capteurs.

Dans une deuxième partie, nous cherchons à utiliser un langage catégorique utilisé pour la description de systèmes biologiques, naturellement résilients. Les systèmes biologiques présentent une forme de redondance fonctionnelle et non-structurelle. Cette propriété s'appelle *degeneracy*, et sa traduction catégorique, le *principe de multiplicité* (PM). Le PM nous semble donc être à la base de la résilience. Cependant, le PM requiert la notion de cluster, qui est en fait le nom des flèches dans les ind-catégories. Nous étudions donc la notion de cluster, exhibant de nouvelles propriétés et définitions (utilisant les composantes connexes de comma-catégories) que nous utiliserons pour trouver une caractérisation non-catégorique du PM dans le cas simple, mais important, des préordres.

Title: Characterisation of organisations for resilient detection of threats

Keywords: Resilience, category theory, cluster, multiplicity, connected components

Abstract: The starting point of this thesis is sensor networks, and how to instigate resilience in them. Our approach relies on category theory.

We first tackle the use of dynamical systems and their composition. We prove that every dynamical system may be decomposed into simpler, reactive systems, that could be seen as sensors.

In a second part, we use a categorical language first meant for biological systems, that are resilient by nature. Biological systems enjoy a form of functional, non-structural redundancy that biologists call *degeneracy*. Category theorists translate it into the *multiplicity principle* (MP). MP seems to constitute a fertile ground for resilience. However, MP relies on the notion of cluster, which are the arrows of ind-categories. We thus study that notion of a cluster, exhibit some new properties and definitions, which use the connected components of the comma-cateogry, and that we use to find a non-categorical characterisation of MP in the special, simpler, but important case of preorders.