SYMMETRY

Am Anfang war die Symmetrie – In the beginning was symmetry!

Werner Heisenberg, *Der Teil und das Ganze*:
Gespräche im Umkreis der Atomphysik, 1969,
English translation, *Physics and Beyond*, 1971.

by

Marc Bezem
Ulrik Buchholtz
Pierre Cagne
Bjørn Ian Dundas
Daniel R. Grayson

Book version: `a0aa1dd` (2021-10-07)

This book is available at: https://unimath.github.io/SymmetryBook/book.pdf

To cite the book, the following BibTeX code may be useful:

```
@misc{Symmetry,
   title         = {Symmetry},
   author        = {Marc Bezem and Ulrik Buchholtz and Pierre Cagne
                    and Bjørn Ian Dundas and Daniel R. Grayson},
   date          = {2021-10-07},
   howpublished  = {\url{https://github.com/UniMath/SymmetryBook}},
   note          = {Commit: \texttt{a0aa1dd}}
}
```

Short contents

Short contents · iii

Contents · v

1 Introduction to the topic of this book · 1

2 An introduction to univalent mathematics · 4

3 The universal symmetry: the circle · 50

4 Groups · 81

5 Subgroups · 126

6 Symmetry · 145

7 Finitely generated groups · 155

8 Finite group theory · 159

9 Euclidean geometry · 166

10 Geometry (first look) · 171

11 Vector spaces and linear groups · 172

12 Field theory · 173

13 Classification of wallpaper groups(†) · 174

14 Affine geometry · 175

15 Bilinear forms · 176

16 Inversive geometry (Möbius) · 177

17 Projective geometry · 178

18 Minkowski space-time geometry · 179

19 Kleinian geometries · 180

20 *Galois theory* · 181

21 *Impossible constructions* · 184

22 *Possible constructions* · 185

23 *Witt theory, SOSs, Artin-Schreier* · 186

24 *Dual numbers and split-complex numbers* · 187

A *Historical remarks* · 188

B *Metamathematical remarks* · 189

Bibliography · 194

Glossary · 195

Index · 197

Contents

Short contents · iii

Contents · v

1 *Introduction to the topic of this book* · 1

2 *An introduction to univalent mathematics* · 4

2.1 What is a type? · 4
2.2 Types, elements, families, and functions · 5
2.3 Universes · 8
2.4 The type of natural numbers · 9
2.5 Identity types · 11
2.6 Product types · 14
2.7 Identifying elements in members of families of types · 16
2.8 Sum types · 17
2.9 Equivalences · 18
2.10 Identifying pairs · 21
2.11 Binary products · 22
2.12 More inductive types · 23
2.13 Univalence · 26
2.14 Heavy transport · 26
2.15 Propositions, sets and groupoids · 28
2.16 Propositional truncation and logic · 31
2.17 More on equivalences; surjections and injections · 32
2.18 Decidability, excluded middle and propositional resizing · 35
2.19 The replacement principle · 35
2.20 Predicates and subtypes · 36
2.21 Pointed types · 39
2.22 Operations that produce sets · 40
2.23 More on natural numbers · 43
2.24 The type of finite types · 45
2.25 Type families and maps · 46
2.26 Higher structure: stuff, structure, and properties · 47

3 *The universal symmetry: the circle* · 50

3.1 The circle and its universal property · 50
3.2 The integers · 52
3.3 Set bundles · 53
3.4 The symmetries in the circle · 57
3.5 A reinterpretation of the circle · 59
3.6 Connected set bundles over the circle · 62

3.7 The m^{th} root: set bundles over the components of Cyc · 69
3.8 Getting our cycles in order · 73
3.9 Old material yet to be integrated · 75

4 *Groups* · 81
4.1 The type of groups · 82
4.2 Abstract groups · 90
4.3 Homomorphisms · 94
4.4 ∞-groups · 100
4.5 G-sets · 101
4.6 The classifying type is the type of torsors · 104
4.7 Groups; concrete vs. abstract · 106
4.8 Homomorphisms; abstract vs. concrete · 109
4.9 Monomorphisms and epimorphisms · 111
4.10 Abelian groups · 114
4.11 G-sets vs abs(G)-sets · 119
4.12 Sums of groups · 121

5 *Subgroups* · 126
5.1 Subgroups · 126
5.2 Images, kernels and cokernels · 129
5.3 The action on the set of subgroups · 134
5.4 Normal subgroups · 135
5.5 The pullback · 140
5.6 The Weyl group · 142

6 *Symmetry* · 145
6.1 Cayley diagram · 145
6.2 Actions · 145
6.3 Heaps (†) · 146
6.4 Semidirect products · 147
6.5 Orbit-stabilizer theorem · 150
6.6 The isomorphism theorems · 150
6.7 (the lemma that is not) Burnside's lemma · 151
6.8 More about automorphisms · 152
6.9 Orbit type as a groupoid completion(*) · 153

7 *Finitely generated groups* · 155
7.1 Cayley diagrams · 156
7.2 Free groups · 156
7.3 Examples · 156
7.4 Subgroups of free groups · 156
7.5 Intersecting subgroups · 157
7.6 Connections with automata (*) · 157

8 *Finite group theory* · 159
8.1 Lagrange's theorem, counting version · 160
8.2 Cauchy's theorem · 161
8.3 Sylow's Theorems · 163
8.4 cycle decompositions · 165

8.5 Lagrange · 165
8.6 Sylow stuff? · 165

9 *Euclidean geometry* · 166
9.1 Inner product spaces · 166
9.2 Euclidean spaces · 167
9.3 Geometric objects · 168
9.4 The icosahedron · 170

10 *Geometry (first look)* · 171
10.1 incidence geometries and the Levi graph · 171
10.2 euclidean planes · 171
10.3 ruler and compass constructions · 171
10.4 affine planes and Pappus' law · 171
10.5 projective planes · 171

11 *Vector spaces and linear groups* · 172
11.1 the algebraic hierarchy: groups, abelian groups, rings, fields · 172
11.2 vector spaces · 172
11.3 the general linear group as automorphism group · 172
11.4 determinants(†) · 172

12 *Field theory* · 173
12.1 examples: rationals, polynomials, adding a root, field extensions · 173
12.2 ordered fields, real-closed fields, pythagorean fields, euclidean fields · 173
12.3 complex fields, quadratically closed fields, algebraically closed fields · 173
12.4 Diller-Dress theorem(†) · 173

13 *Classification of wallpaper groups(†)* · 174

14 *Affine geometry* · 175
14.1 affine frames, affine planes · 175
14.2 the affine group as an automorphism group · 175
14.3 the affine group as a semidirect product · 175
14.4 affine properties (parallelism, length ratios) · 175

15 *Bilinear forms* · 176

16 *Inversive geometry (Möbius)* · 177
16.1 residue at a point is affine · 177
16.2 Miquel's theorem · 177

17 *Projective geometry* · 178
17.1 projective frames · 178
17.2 the projective group and projectivities · 178
17.3 projective properties (cross-ratio) · 178

17.4 fundamental theorem of projective geometry · 178

18 *Minkowski space-time geometry* · 179

19 *Kleinian geometries* · 180

19.1 conics and dual conics · 180

19.2 elliptic geometry · 180

20 *Galois theory* · 181

20.1 Covering spaces and field extensions · 181

20.2 Intermediate extensions and subgroups · 183

20.3 separable/normal/etc. · 183

20.4 fundamental theorem · 183

21 *Impossible constructions* · 184

21.1 doubling the cube · 184

21.2 trisecting the angle · 184

21.3 squaring the circle · 184

21.4 7-gon · 184

21.5 quintic equations · 184

22 *Possible constructions* · 185

22.1 5-gon and the icosahedron · 185

22.2 17-gon and 257-gon · 185

22.3 cubics and quartics · 185

23 *Witt theory, SOSs, Artin-Schreier* · 186

23.1 quadratic forms · 186

23.2 Grothendieck-Witt ring · 186

24 *Dual numbers and split-complex numbers* · 187

24.1 minkowski and galilaean spacetimes · 187

A *Historical remarks* · 188

B *Metamathematical remarks* · 189

B.1 Definitional equality · 189

B.2 The Limited Principle of Omniscience · 191

B.3 Topology · 191

Bibliography · 194

Glossary · 195

Index · 197

1

Introduction to the topic of this book

ch::intro

> *Poincaré sagte gelegentlich, dass alle Mathematik eine Gruppengeschichte war. Ich erzählte ihm dann über dein Programm, das er nicht kannte.*
>
> *Poincaré was saying that all of mathematics was a tale about groups. I then told him about your program, which he didn't know about.*
>
> (Letter from Sophus Lie to Felix Klein, October 1882)

This book is about symmetry and its many manifestations in mathematics. There are many kinds of symmetry and many ways of studying it. Euclidean plane geometry is the study of properties that are invariant under rigid motions of the plane. Other kinds of geometry arise by considering other notions of transformation. Univalent mathematics gives a new perspective on symmetries: Motions of the plane are forms of identifying the plane with itself in possibly non-trivial ways. It may also be useful to consider different presentations of planes (for instance as embedded in a common three-dimensional space) and different identifications between them. For instance, when drawing images in perspective we identify planes in the scene with the image plane, not in a rigid Euclidean way, but rather via a perspectivity (see Fig. ?). This gives rise to projective geometry.

Does that mean that a plane from the point of view of Euclidean geometry is not the same as a plane from the point of view of projective or affine geometry? Yes. These are of different types, because they have different notions of identification, and thus they have different properties.

Here we follow Quine's dictum: No entity without identity! To know a type of objects is to know what it means to identify representatives of the type. The collection of self-identifications (self-transformations) of a given object form a *group*.

Group theory emerged from many different directions in the latter half of the 19th century. Lagrange initiated the study of the invariants under permutations of the roots of a polynomial equation $f(x) = 0$, which culminated in the celebrated work of Abel and Galois. In number theory, Gauss had made detailed studies of modular arithmetic, proving for instance that the group of units of $\mathbb{Z}/p\mathbb{Z}$ is cyclic. Klein was bringing order to geometry by considering groups of transformation, while Lie was applying group theory in analysis to the study of differential equations.

Galois was the first to use the word "group" in a technical sense, speaking of collections of permutations closed under composition. He realized that the existence of resolvent equation is equivalent to the existence of a normal subgroup of prime index in the group of the equation.

Groupoids vs groups. The type of all squares in a euclidean plane form a groupoid. It is connected, because between any two there exist identifications between them. But there is no canonical identification.

When we say "the symmetry group of the square", we can mean two things: 1) the symmetry group of a particular square; this is indeed a group, or 2) the connected groupoid of all squares; this is a "group up to conjugation".

Vector spaces. Constructions and fields. Descartes and cartesian geometry.

Klein's EP:

> Given a manifold and a transformation group acting on it, to investigate those properties of figures on that manifold that are invariant under transformations of that group.

and

> Given a manifold, and a transformation group acting on it, to study its *invariants*.

Invariant theory had previously been introduced in algebra and studied by Clesch and Jordan.

(Mention continuity, differentiability, analyticity and Hilbert's 5th problem?)

Any finite automorphism group of the Riemann sphere is conjugate to a rotation group (automorphism group of the Euclidean sphere). [Dependency: diagonalizability] (Any complex representation of a finite group is conjugate to a unitary representation.)

All of mathematics is a tale, not about groups, but about ∞-groupoids. However, a lot of the action happens already with groups.

Glossary of coercions

MOVE TO BETTER PLACE Throughout this book we will use the following coercions to make the text more readable.

- If X is the pointed type (A, a), then $x : X$ means $x : A$.
- On hold, lacking context: If p and q are paths, then (p, q) means $(p, q)^=$.
- If e is a pair of a function and a proof, we also use e for the function.
- If e is an equivalence between types A and B, we use $\bar{e}$ for the identification of A and B induced by univalence.
- If $p : A = B$ with A and B types, then we use $\tilde{p}$ for the canonical equivalence from A to B (also only as function).
- If X is $(A, a, \ldots)$ with $a : A$, then pt_X and even just pt mean a.

How to read this book

. . .

A word of warning. We include a lot of figures to make it easier to follow the material. But like all mathematical writing, you'll get the most out of it, if you maintain a skeptical attitude: Do the pictures really accurately represent the formal constructions? Don't just believe us: Think about it!

The same goes for the proofs: When we say that something *clearly* follows, it should be *clear to you*. So clear, in fact, that you could go and convince a proof assistant, should you so desire.

Acknowledgement

The authors acknowledge the support of the Centre for Advanced Study (CAS) at the Norwegian Academy of Science and Letters in Oslo, Norway, which funded and hosted the research project Homotopy Type Theory and Univalent Foundations during the academic year 2018/19.

2

An introduction to univalent mathematics

ch:univalent-mathematics

sec:what-is-a-type

2.1 What is a type?

In some computer programming languages, all variables are introduced along with a declaration of the type of thing they will refer to. Knowing the type of thing a variable refers to allows the computer to determine which expressions in the language are *grammatically well formed*[1], and hence valid. For example, if s is a string[2] and x is a real number, we may write $1/x$, but we may not write $1/s$.[3]

To enable the programmer to express such declarations, names are introduced to refer to the various types of things. For example, the name Bool may be used to declare that a variable is a Boolean value[4], String may refer to 32 bit integers, and Real may refer to 64 bit floating point numbers[5].

Types occur in mathematics, too, and are used in the same way: all variables are introduced along with a declaration of the type of thing they will refer to. For example, one may say "consider a real number x", "consider a natural number n", "consider a point P of the plane", or "consider a line L of the plane". After that introduction, one may say that the *type* of n is *natural number* and that the *type* of P is *point of the plane*. Just as in a computer program, type declarations such as those are used to determine which mathematical statements are grammatically well formed. Thus one may write "P lies on L" or $1/x$, but not "L lies on P" nor $1/L$.[6]

Often ordinary English writing is good enough for such declarations in mathematics expositions, but, for convenience, mathematicians usually introduce symbolic names to refer to the various types of things under discussion. For example, the name $\mathbb{N}$ is usually used when declaring that a variable is a natural number, the name $\mathbb{Z}$ is usually used when declaring that a variable is an integer, and the name $\mathbb{R}$ is usually used when declaring that a variable is a real number. Ways are also given for constructing new type names from old ones: for example, the name $\mathbb{R} \times \mathbb{R}$ may be used when declaring that a variable is a point of the plane, for it conveys the information that a point of the plane is a pair of real numbers.

Once one becomes accustomed to the use of names such as $\mathbb{N}$ in mathematical writing and speaking, it is natural to take the next step and regard those names as denoting things that exist. Thus, we shall refer to $\mathbb{N}$ as the *type of all natural numbers*, and we will think of it as a mathematical object in its own right. Intuitively and informally, it is a collection whose members (or *elements*) are the natural numbers.

[1] The grammar of a programming language consists of all the language's rules. A statement or expression in a programming language is grammatically well formed if it follows all the rules.

[2] A *string* is a sequence of characters, such as "abcdefgh".

[3] In a programming language, the well formed expression $1/x$ may produce a run-time error if x happens to have the value 0.

[4] A Boolean value is either *true* or *false*

[5] An example of a *floating point number* is $.625 \times 2^{33}$ – the *mantissa* $.625$ and the *exponent* 33 are stored inside the floating point number. The "point", when the number is written in base 2 notation, is called "floating", because its position is easily changed by modifying the exponent.

[6] In mathematics there are no "run-time" errors; rather, it is legitimate to write the expression $1/x$ only if we already know that x is a non-zero real number.

Once we view the various types as existing as mathematical objects, they become worthy of study. The language of mathematics is thereby improved, and the scope of mathematics is broadened. For example, we can consider statements such as "ℕ is infinite" and to try to prove it.

Historically, there was some hesitation[7] about introducing the collection of all natural numbers as a mathematical object, perhaps because if one were to attempt to build the collection from nothing by adding numbers to it one at a time, it would take an eternity to complete the assembly. We won't regard that as an obstacle.

We have said that the types of things are used to determine whether mathematical statements are well formed. Therefore, if we expect "ℕ is infinite" to be a well formed statement, we'll have to know what type of thing ℕ is, and we'll have to have a name for that type. Similarly, we'll have to know what type of thing that type is, and we'll have to have a name for it, and so on forever. Indeed, all of that is part of what will be presented in this chapter.

2.2 *Types, elements, families, and functions*

In this section we build on the intuition imparted in the previous section.

univalent-mathematics

In *univalent mathematics*,[8] types are used to classify all mathematical objects. Every mathematical object is an *element* (or a *member*) of some (unique) *type*. Before one can talk about an object of a certain type, one must introduce the type itself. There are enough ways to form new types from old ones to provide everything we need to write mathematics.

One expresses the declaration that an object a is an element of the *type* X by writing $a : X$.[9]

Using that notation, each variable x is introduced along with a declaration of the form $x : X$, which declares that x will refer to something of type X, but provides no other information about x. The declared types of the variables are used to determine which statements of the theory are grammatically well formed.

After introducing a variable $x : X$, it may be possible to form an expression T representing a type, all of whose components have been already been given a meaning. (Here the variable x is regarded also as having already been given a meaning, even though the only thing known about it is its type.) To clarify the dependence of T on x primarily, we may write $T(x)$ instead of T. Such an expression will be called a *family of types* parametrized by the variable x of type X. Such a family provides a variety of types, for, if a is any expression denoting an object of X, one may replace all occurrences of x by a in T, thereby obtaining a new expression representing a type, which may be regarded as a member of the family, and which may be denoted by $T(a)$.

Naturally, if the expression T doesn't actually involve the variable x, then the members of the family are all the same, and we'll refer to the family as a *constant family* of types.

Here's an example of a family of types: we let n be a natural number and P_n be the type of n-sided polygons in the plane. It gives a family of types parametrized by the natural numbers. One of the members of the family is the type $T(5)$ of all pentagons in the plane.

[7] TO DO : Include some pointers to discussions of potential infinity and actual infinity, perhaps.

[8] The term "univalent" is a word coined by Vladimir Voevodsky, who introduced it to describe his principle that types that are *equivalent* in a certain sense can be identified with each other. The principle is stated precisely in Principle 2.13.2. As Voevodsky explained, the word comes from a Russian translation of a mathematics book, where the English mathematical term "faithful" was translated into Russian as the Russian word that sounds like "univalent". He also said "Indeed these foundations seem to be faithful to the way in which I think about mathematical objects in my head."

[9] The notation in mathematics based on *set theory* that corresponds (sort of) to this is $a \in X$.

A family of types may be parametrized by more than one variable. For example, after introducing a variable $x : X$ and a family of types T parametrized by x, we may introduce a variable $t : T$. Then it may be possible to form an expression S representing a type that involves the variables x and t. Such an expression will be called a family of types parametrized by x and t, and we may write $S(x, t)$ instead of S to emphasize the dependence on x and t. The same sort of thing works with more variables.

After introducing a variable $x : X$ and a family of types T, it may be possible to form an expression e of type T, all of whose components have been already been given a meaning. Such an expression will also be called a *family of elements of* T parametrized by the elements of X, when we wish to focus on the dependence of e (and perhaps T) on the variable x. To clarify the dependence of e on x primarily, we may write $e(x)$ (or e_x) instead of e. Such a family provides a variety of elements of T, for, if a is any expression denoting an object of X, one may replace all occurrences of x by a in e and in T, thereby obtaining an element of $T(a)$, which may be regarded as a *member* of the family and which will be denoted by $e(a)$.

Naturally, if the expressions e and T don't actually involve the variable x, then the members of the family are all the same, and we'll refer to the family as a *constant family* of elements.

Here's an example of a family of elements in a constant family of types: we let n be a natural number and consider the real number $\sqrt{n}$. It gives a family of real numbers parametrized by the natural numbers. (The family may also be called a *sequence* of real numbers). One of the members of the family is $\sqrt{11}$.

Here's an example of a family of elements in a (non-constant) family of types: we let n be a natural number and P_n be the type of n-sided polygons in the plane, as we did above. Then we consider the regular n-sided polygon p_n of radius 1 with a vertex on the x-axis. We see that $p_n : P_n$. One of the members of the family is the regular pentagon p_5 of radius 1 with a vertex on the x-axis; it is of type P_5.

The type X containing the variable for a family of types or a family of elements is called the *parameter type* of the family.

Just as a family of types may depend on more than one variable, a family of elements may also depend on more than one variable.

Families of elements can be enclosed in mathematical objects called *functions* (or *maps*), as one might expect. Let e be a family of elements of a family of types T, both of which are parametrized by the elements x of X. We use the notation $x \mapsto e$ for the function that sends an element a of X to the element $e(a)$ of $T(a)$; the notation $x \mapsto e$ can be read as "x maps to e" or "x goes to e". (Recall that $e(a)$ is the expression that is obtained from e by replacing all occurrences of x in e by a.) If we name the function f, then that element of T will be denoted by $f(a)$. The *type* of the function $x \mapsto e$ is called a *product type* and will be denoted by $\prod_{x:X} T$; if T is a constant family of types, then the type will also be called a *function type* and will be denoted by $X \to T$. Thus when we write $f : X \to T$, we mean that f is an element of the type $X \to T$, and we are saying that f is a function from X to T.

An example of a function is the function $n \mapsto \sqrt{n}$ of type $\mathbb{N} \to \mathbb{R}$.

Another example of a function is the function $n \mapsto p_n$ of type $\prod_{n:\mathbb{N}} P_n$, where P_n is the type of polygons introduced above, and p_n is the polygon introduced above.

Another example of a function is the function $m \mapsto (n \mapsto m + n)$ of type $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$. It is a function that accepts a natural number as argument and returns a function as its value. The function returned is of type $\mathbb{N} \to \mathbb{N}$. It accepts a natural number as argument and returns a natural number as value.

The reader may wonder why the word "product" is used when speaking of product types. To motivate that, we consider a simple example informally. We take X to be a type with just two elements, b and c. We take $T(x)$ to be a family of types parametrized by the elements of X, with $T(b)$ being a type with 5 elements and $T(c)$ being a type with 11 elements. Then the various functions f of type $\prod_{x:X} T(x)$ are plausibly obtained by picking a suitable element for $f(b)$ from the 5 possibilities in $T(b)$ and by picking a suitable element for $f(c)$ from the 11 possibilities in $T(c)$. The number of ways to make both choices is 5×11, which is a *product* of two numbers. Thus $\prod_{x:X} T(x)$ is sort of like the product of $T(b)$ and $T(c)$, at least as far as counting is concerned.

The reader may wonder why we bother with functions at all: doesn't the expression e serve just as well as the function $x \mapsto e$, for all practical purposes? The answer is no. One reason is that the expression e doesn't inform the reader that the variable under consideration is x. Another reason is that we may want to use the variable x for elements of a different type later on: then $e(x)$ is no longer well formed. For example, imagine first writing this: "For a natural number n we consider the real number $\sqrt{n}$" and then writing this: "Now consider a triangle n in the plane." The result is that $\sqrt{n}$ is no longer usable, whereas the function $n \mapsto \sqrt{n}$ has enclosed the variable and the family into a single object and remains usable.[10]

Once a family e has been enclosed in the function $x \mapsto e$, the variable x is referred to as a *dummy variable* or as a *bound variable*.[11] This signifies that the name of the variable no longer matters, in other words, that $x \mapsto e(x)$ and $t \mapsto e(t)$ may regarded as identical. Moreover, the variable x that occurs inside the function $x \mapsto e$ is regarded as unrelated to variables x which may appear elsewhere in the discussion.

We have mentioned above the possibility of giving a name to a function. We expand on that now by introducing notation for making and for using *definitions*.

The notation $x :\equiv z$ will be an announcement that we are defining the expression x to be the expression z, all of whose components have already been given a meaning; in that case, we will say that x has been *defined* to be (or to mean) z. The forms allowed for the expression x will be made clear by the examples we give.

For example, after writing $n :\equiv 12$, we will say that n has been defined to be 12.

For another example, the function f that we named above may be introduced by writing $f :\equiv (x \mapsto e(x))$. Alternatively and more traditionally, we may write $f(x) :\equiv e(x)$.

The notation $b \equiv c$ will denote the statement that the expressions b and c become the same thing if all the subexpressions within b or c are

[10] Students of trigonometry are already familiar with the concept of function, as something enclosed this way. The sine and cosine functions, sin and cos, are examples.

[11] Students of calculus are familiar with the concept of dummy variable and are accustomed to using identities such as $\int_a^b f(t)\,dt = \int_a^b f(x)\,dx$.

expanded according to their definitions, if any; in that case, we will say that b and c are *the same by definition*. For example, after writing $n :\equiv 12$ and $m :\equiv n$, we may say that $j + 12 \equiv j + m$ and that $m \times 11 \equiv 12 \times 11$.

Whenever two expressions are the same by definition, we may replace one with the other inside any other expression, because the expansion of definitions is regarded as trivial and transparent.

We proceed now to the promised example. Consider functions $f : X \to Y$ and $g : Y \to Z$. We define the *composite* function $g \circ f : X \to Z$ by setting $g \circ f :\equiv (a \mapsto g(f(a)))$. In other words, it is the function that sends an arbitrary element a of X to $g(f(a))$ in Z. (The expression $g \circ f$ may be read as "g circle f" or as "g composed with f".) The composite function $g \circ f$ may also be denoted simply by gf. We also may use the following alternative notation for the composite function, which includes the three types explicitly:

$$\left(X \xrightarrow{f} Y \xrightarrow{g} Z \right) :\equiv g \circ f$$

Now consider functions $f : X \to Y$, $g : Y \to Z$, and $h : Z \to W$. Then $(h \circ g) \circ f$ and $h \circ (g \circ f)$ are the same by definition, since applying the definitions within expands both expressions to $a \mapsto h(g(f(a)))$. In other words, we have established that $(h \circ g) \circ f \equiv h \circ (g \circ f)$. Thus, we may write $h \circ g \circ f$ for either expression, without danger of confusion.

One may define the identity function $\mathrm{id}_X : X \to X$ by setting $\mathrm{id}_X :\equiv (a \mapsto a)$. Application of definitions shows that $f \circ \mathrm{id}_X$ is the same by definition as $a \mapsto f(a)$, which, by a standard convention, which we adopt, is to be regarded as the same as f. In other words, we have established that $f \circ \mathrm{id}_X \equiv f$. A similar computation applies to $\mathrm{id}_Y \circ f$.

The convention that $f \equiv (a \mapsto f(a))$ is referred to as the *η-rule* in the jargon of type theory.

page:idofetaf

In the following sections we will present various other elementary types and elementary ways to make new types from old ones.

2.3 *Universes*

sec:universes

In Section 2.2 we have introduced the objects known as *types*. They have *elements*, and the type an element belongs to determines the type of thing that it is. At various points in the sequel, it will be convenient for types also to be elements, for that will allow us, for example, to define families of types just as easily as we define families of elements. To achieve this convenience, we introduce types that are *universes*. Some care is required, for the first temptation is to posit a single new type $\mathcal{U}$ called *the universe*, so that every type is realized as an element of $\mathcal{U}$. This universe would be "the type of all types", but introducing it would lead to an absurdity, for roughly the same reason that introduction of a "set of all sets" leads to the absurdity in traditional mathematics known as Russell's paradox.[12] Some later approaches to set theory included the notion of a *class*, with the collection of all sets being the primary example of a class. Classes are much like sets, but they are bigger and there are more of them. Then one may wonder what sort of thing the collection of all classes would be. Such musings are resolved in univalent mathematics as follows.

(1) There are some types called *universes*.

(2) If $\mathcal{U}$ is a universe, and $X : \mathcal{U}$ is an element of $\mathcal{U}$, then X is a type.

[12] In fact, type theory can trace its origins to Russell's paradox, announced in a 1902 letter to Frege as follows:

> There is just one point where I have encountered a difficulty. You state that a function too, can act as the indeterminate element. This I formerly believed, but now this view seems doubtful to me because of the following contradiction. Let w be the predicate: to be a predicate that cannot be predicated of itself. Can w be predicated of itself? From each answer its opposite follows. Therefore we must conclude that w is not a predicate. Likewise there is no class (as a totality) of those classes which, each taken as a totality, do not belong to themselves.

To which Frege replied:

> Incidentally, it seems to me that the expression "a predicate is predicated of itself" is not exact. A predicate is as a rule a first-level function, and this function requires an object as argument and cannot have itself as argument (subject).

Russell then quickly added *Appendix B* to his *Principles of Mathematics* (1903), in which he said that "it is the distinction of logical types that is the key to the whole mystery", where types are the *ranges of significance* of variables. For more on the history of type theory, see Coquand[13].

[13] Thierry Coquand. "Type Theory". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Metaphysics Research Lab, Stanford University, 2018. URL: `https://plato.stanford.edu/archives/fall2018/entries/type-theory/`.

(3) If X is a type, then it appears as an element in some universe $\mathcal{U}$. Moreover, if X and Y are types, then there is a universe $\mathcal{U}$ containing both of them.

(4) If $\mathcal{U}$ and $\mathcal{U}'$ are universes, $\mathcal{U} : \mathcal{U}'$, X is a type, and $X : \mathcal{U}$, then also $X : \mathcal{U}'$. (Thus we may regard $\mathcal{U}'$ as being *larger* than $\mathcal{U}$.)

(5) There is a particular universe $\mathcal{U}_0$, which we single out to serve as a repository for certain basic types to be introduced in the sequel. Moreover, $\mathcal{U}_0 : \mathcal{U}$ for every universe $\mathcal{U}$, and thus $\mathcal{U}_0$ is the *smallest* universe.

It follows from the properties above that there are an infinite number of universes, for each one is an element of a larger one.

Now suppose we have a type X and a family $T(x)$ of types parametrized by a variable x of type X. Choose a universe U with $T(x) : U$. Then we can make a function of type $X \to U$, namely $f :\equiv (x \mapsto T(x))$. Conversely, if f' is a function of type $X \to \mathcal{U}$, then we can make a family of types parametrized by x, namely $T' :\equiv f'(x)$. The flexibility offered by this correspondence between families of types in $\mathcal{U}$ and functions to $\mathcal{U}$ will often be used.

2.4 *The type of natural numbers*

sec:natural-numbers

Here are Peano's rules[14] for constructing the natural numbers in the form that is used in type theory.

P1 (P1) there is a type called $\mathbb{N}$ in the universe $\mathcal{U}_0$ (whose elements will be called *natural numbers*);

P2 (P2) there is an element of $\mathbb{N}$ called 0, called *zero*;

P3 (P3) if m is a natural number, then there is also a natural number $\mathrm{succ}(m)$, called the *successor* of m;

P4 (P4) suppose we are given:

a) a family of types $X(m)$ parametrized by a variable m of type $\mathbb{N}$;

b) an element a of $X(0)$; and

c) a family of functions $g_m : X(m) \to X(\mathrm{succ}(m))$.

Then from those data we are provided with a family of elements $f(m) : X(m)$.

The first three rules present few problems for the reader. They provide us with the smallest natural number $0 : \mathbb{N}$, and we may introduce as many others as we like with the following definitions.

$$\begin{aligned} 1 &:\equiv \mathrm{succ}(0) \\ 2 &:\equiv \mathrm{succ}(1) \\ 3 &:\equiv \mathrm{succ}(2) \\ &\vdots \end{aligned}$$

[14] Giuseppe Peano. *Arithmetices principia: nova methodo*. See also https://github.com/mdnahas/Peano_Book/ for a parallel translation by Vincent Verheyen. Fratres Bocca, 1889. URL: https://books.google.com/books?id=z80GAAAAYAAJ.

[15] Rule (P4) and our logical framework are stronger than in Peano's original formulation, and this allows us to omit some rules that Peano had to include: that different natural numbers have different successors; and that no number has 0 as its successor. Those omitted rules remain true in this formulation and can be proved from the other rules, after we have introduced the notion of equality in our logical framework.

You may recognize rule (P4) as "the principle of mathematical induction" or as "defining a function by recursion".[15] We will refer to it simply as "induction on $\mathbb{N}$". The resulting family f may be regarded as having been defined inductively by the two declarations $f(0) :\equiv a$ and $f(\mathrm{succ}(m)) :\equiv g_m(f(m))$, and indeed, we will often simply write such a pair of declarations as a shorthand way of applying rule (P4). The two declarations cover the two ways of introducing elements of $\mathbb{N}$ via the use of the two rules (P2) and (P3). (In terms of computer programming, those two declarations amount to the code for a recursive subroutine that can handle any incoming natural number.)

With that notation in hand, speaking informally, we may regard (P4) above as defining the family f by the following infinite sequence of definitions.

$$\begin{aligned} f(0) &:\equiv a \\ f(1) &:\equiv g_0(a) \\ f(2) &:\equiv g_1(g_0(a)) \\ f(3) &:\equiv g_2(g_1(g_0(a))) \\ &\vdots \end{aligned}$$

(The need for the rule (P4) arises from our inability to write down an infinite sequence of definitions in a finite amount of space, and from the need for $f(m)$ to be defined when m is a variable of type $\mathbb{N}$, and thus is not known to be equal to 0, nor to 1, nor to 2, etc.)

We may use induction on $\mathbb{N}$ to define of *iteration* of functions. Let Y be a type, and suppose we have a function $e : Y \to Y$. We define by induction on $\mathbb{N}$ the m-fold *iteration* $e^m : Y \to Y$ by setting $e^0 :\equiv \mathrm{id}_Y$ and $e^{\mathrm{succ}(m)} :\equiv e \circ e^m$. (Here we apply rule (P4) with the the type $Y \to Y$ as the family of types $X(m)$, the identity function id_Y for a, and the function $d \mapsto e \circ d$ for the family $g_m : (Y \to Y) \to (Y \to Y)$ of functions.)

We may now define addition of natural numbers by induction on $\mathbb{N}$. For natural numbers n and m we define $n + m : \mathbb{N}$ by induction on $\mathbb{N}$ with respect to the variable m by setting $n + 0 :\equiv n$ and $n + \mathrm{succ}(m) :\equiv \mathrm{succ}(n + m)$. (The reader should be able to extract the family $X(m)$, the element a, and the family of functions g_m from that pair of definitions.) Application of definitions shows, for example, that $2 + 2$ and 4 are the same by definition, and thus we may write $2 + 2 \equiv 4$, because both expressions reduce to succ(succ(succ(succ(0)))).

Similarly we define the product $m \cdot n : \mathbb{N}$ by induction on m by setting setting $0 \cdot n :\equiv 0$ and $\mathrm{succ}(m) \cdot n :\equiv (m \cdot n) + n$.

Alternatively (and equivalently) we may use iteration of functions to define addition and multiplication, by setting $n + m :\equiv \mathrm{succ}^m(n)$ and $m \cdot n := (i \mapsto i + n)^m(0)$.

Finally, we may define the factorial function $\mathrm{fact} : \mathbb{N} \to \mathbb{N}$ by induction on $\mathbb{N}$, setting $\mathrm{fact}(0) :\equiv 1$ and $\mathrm{fact}(\mathrm{succ}(m)) :\equiv \mathrm{succ}(m) \cdot \mathrm{fact}(m)$. (One can see that this definition applies rule (P4) with $X(m) :\equiv \mathbb{N}$, with 1 for a, and with the function $n \mapsto \mathrm{succ}(m) \cdot n$ for g_m.) Application of the definitions shows, for example, that $\mathrm{fact}(3) \equiv 6$, as the reader may verify.

2.5 *Identity types*

sec:identity-types

One of the most important types is the *identity type*, which implements the intuitive notion of equality; the reader may be more comfortable if we call it the *equality type*, at least initially. Identity (or equality) between two elements may be considered only when the two elements are of the same type; we shall have no need to compare elements of different types.

Here are the rules for constructing and using equality types.

rules-for-equality E1

(E1) for any type X and for any elements a and b of it, there is a type $a =_X b$; moreover, if X is an element of a universe $\mathcal{U}$, then so is $a =_X b$.

E2

(E2) for any type X and for any element a of it, there is an element refl_a of type $a =_X a$ (the name refl comes from the word "reflexivity")

E3

(E3) suppose we are given:

a) a type X and an element $a : X$;

b) a family of types $P(b, e, \ldots)$ parametrized by a variable b of type X, a variable e of type $a =_X b$, and perhaps some further variables; and

c) an element p of $P(a, \mathrm{refl}_a, \ldots)$.

Then from those data we are provided with a family of elements $f(b, e, \ldots) : P(b, e, \ldots)$. Moreover, $f(a, \mathrm{refl}_a, \ldots) \equiv p$.

The type $a =_X b$ may be called an *equality type*, an *identity type*, or simply an *equation*).

When there is no risk of confusion, we will write $a = b$ instead of $a =_X b$.

(Good motivation for the form of the equal sign is provided by a remark made by Robert Recorde in 1557: "to avoid the tedious repetition of these words *is equal to*, I will substitute . . . a pair of parallels or twin lines of the same length, thus: =, because no two things can be more equal.")

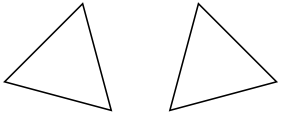
We will refer to an element i of $a = b$ as an *identification* of a with b, or simply as an *identity* or an *equality*. Since the word "identification" is a long one, we may also refer to i as a *path* from a to b – this has the advantage of incorporating the intuition that an identification may proceed gradually through intermediate steps.

In certain circumstances, to be discussed later, we will refer to an element i of $a = b$ as a *proof*[16] that a is equal to b.

The need to record, using the element i, the way we identify a with b may come as a surprise, since normally, in mathematics, one is accustomed to regarding a as either equal to b or not. However, this reflects a situation commonly encountered in geometry when *congruence* of geometric figures is considered. For example, in Euclidean space, two equilateral triangles of the same size are congruent in six (different) ways.[17] The chief novelty of univalent mathematics is that the basic logical notion of equality, as implemented by the types $a = b$, is carefully engineered to accommodate notions of congruence and symmetry from diverse areas of mathematics, including geometry. Exposing that point of view in the context of geometry is the main point of this book.

[16] Here we override the use of the word "proof" that is traditional in metamathematics; such a thing would now be a formal expression denoting an element of $a = b$.

[17] Six, since we allow reflections, otherwise there are only three.



In light of the analogy with geometry just introduced, we will refer to an element i of $a = a$ as a *symmetry* of a. Think, for example, of a congruence of a triangle with itself. An example of a non-trivial symmetry will be seen in Exercise 2.13.3.

Consider the equation $\mathrm{fact}(2) = 2$, where fact denotes the factorial function defined in Section 2.4. Expansion of the definitions in the equation $\mathrm{fact}(2) = 2$ simplifies it to $\mathrm{succ}(\mathrm{succ}(0)) = \mathrm{succ}(\mathrm{succ}(0))$, so we see from rule (E2) that $\mathrm{refl}_{\mathrm{succ}(\mathrm{succ}(0))}$ serves as an element of it.[18] We may also write either refl_2 or $\mathrm{refl}_{\mathrm{fact}(2)}$ for that element. A student might want a more detailed derivation that fact(2) may be identified with 2, but as a result of our convention above that definitions may be applied without changing anything, the application of definitions, including inductive definitions, is normally regarded as a trivial operation, and the details are usually omitted.

We will refer to rule (E3) as "induction for equality". To signal that we wish to apply it, we may announce that we argue *by induction on* e.

The family f resulting from an application of rule (E3) may be regarded as having been completely defined by the single declaration $f(a, \mathrm{refl}_a) :\equiv p$, and indeed, we will often simply write such a declaration as a shorthand way of applying rule (E3). The rule says that to construct something from every identification e of a with something else, it suffices to consider the special case where the identification e is $\mathrm{refl}_a : a = a$.[19]

Intuitively, the induction principle for equality amounts to saying that the element refl_a "generates" the system of types $a = b$, as b ranges over elements of A.[20]

Equality is *symmetric*, in the sense that an identification of a with b may be reversed to give an identification of b with a. In order to produce an element of $b = a$ from an element e of $a = b$, for any b and e, we argue by induction. We let $P(b, e)$ be $b = a$ for any b of type X and for any e of type $a = b$, for use in rule (E3) above. This reduces us to the case where b is a and p is refl_a, and our task is now to produce an element of $a = a$; we choose refl_a for it.

Equality is also *transitive*, and is established the same way. For each $a, b, c : X$ and for each $p : a = b$ and for each $q : b = c$ we want to produce an element of type $a = c$. By induction on q we are reduced to the case where c is b and q is refl_b, and we are to produce an element of $a = b$. The element p serves the purpose.

Now we state our symmetry result a little more formally.

def:eq-symm

Definition 2.5.1. For any type X and for any $a, b : X$, let

$$\mathrm{symm}_{a,b} : (a = b) \to (b = a)$$

be the function defined by induction by setting $\mathrm{symm}_{a,a}(\mathrm{refl}_a) :\equiv \mathrm{refl}_a$.

This operation on paths is called *path inverse*, and we may abbreviate $\mathrm{symm}_{a,b}(p)$ as p^{-1}. ⌟

Similarly, we formulate transitivity a little more formally, as follows.

def:eq-trans

Definition 2.5.2. For any type X and for any $a, b, c : X$, let

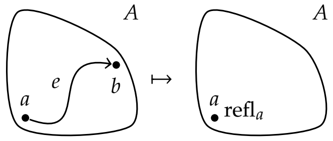
$$\mathrm{trans}_{a,b,c} : (a = b) \to ((b = c) \to (a = c))$$

be the function defined by induction by setting $(\mathrm{trans}_{a,b,b}(p))(\mathrm{refl}_b) :\equiv p$.

[18] We will see later that numbers don't have symmetries, as one would expect, so the possibility that there are other ways to identify fact(2) with 2 doesn't arise.

[19] Notice that the single special case in such an induction corresponds to the single way of introducing elements of equality types via rule (E2), and compare that with (P4), which dealt with the two ways of introducing elements of $\mathbb{N}$.

[20] We can also use a geometric intuition: when b "freely ranges" over elements of A, together with a path $e : a = b$, while we keep the element a fixed, we can picture e as a piece of string winding through A, and the "freeness" of the pair (b, e) allows us to pull the string e, and b with it, until we have the constant path at a, refl_a.



Conversely, we can imagine b starting at a and e starting out as refl_a, and then think of b roaming throughout A, pulling the string e along with it, until it finds every path from a to some other element.

This binary operation is called *path composition* or *path concatenation*, and we may abbreviate $(\mathrm{trans}_{a,b,c}(p))(q)$ as either $p * q$, or as $q \cdot p$, qp, or $q \circ p$, see Fig. 2.1. ⌟

The notation $q \circ p$ for path composition, with p and q in reverse order, fits our intution particularly well when the paths are related to functions and the composition of the paths is related to the composition of the related functions in the same order, as happens, for example, in connection with *transport* (defined below in Definition 2.5.4) in Exercise 2.5.5.

The types of $\mathrm{symm}_{a,b}$ and $\mathrm{trans}_{a,b,c}$ express that equality is symmetric and transitive. Another view of $\mathrm{symm}_{a,b}$ and $\mathrm{trans}_{a,b,c}$ is that they are operations on identifications, namely reversing an identification and concatenating two identifications. As operations, they satisfy many equations, such as $(p^{-1})^{-1} = p$ and associativity of concatenation, which may all be proven by induction. We formulate some of them in the following exercise.

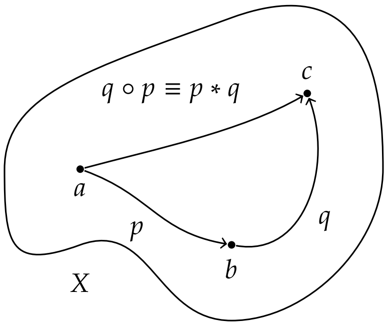


FIGURE 2.1: Composition (also called concatenation) of paths in X

fig:path-concatenation

xca:path-groupoid-laws

EXERCISE 2.5.3. Let X be a type and let $a, b, c, d : X$ be elements.

(1) For $p : a = b$, construct an element of type $p * \mathrm{refl}_b = p$.

(2) For $p : a = b$, construct an element of type $\mathrm{refl}_a * p = p$.

(3) For $p : a = b$, $q : b = c$, and $r : c = d$, construct an element of type $(p * q) * r = p * (q * r)$.

(4) For $p : a = b$, construct an element of type $p^{-1} * p = \mathrm{refl}_b$.

(5) For $p : a = b$, construct an element of type $p * p^{-1} = \mathrm{refl}_a$.

(6) For $p : a = b$, construct an element of type $(p^{-1})^{-1} = p$. ⌟

Concatenation can be used to define powers p^n of $p : a = a$ by induction on $n : \mathbb{N}$, setting $p^0 :\equiv \mathrm{refl}_a$ and $p^{n+1} :\equiv p \cdot p^n$. Negative powers p^{-n} are defined as $(p^{-1})^n$.[21]

One frequent use of elements of identity types is in *substitution*, which is the logical principle that supports our intuition that when x can by identified with y, we may replace x by y in mathematical expressions at will. A wrinkle new to students will likely be that, in our logical framework where there may be various ways to identify x with y, one must specify the identification used in the substitution. Thus one may prefer to speak of using an identification to *transport* properties and data about x to properties and data about y.

Here is a geometric example: if x is a triangle of area 3 in the plane, and y is congruent to x, then y also has area 3.

Here is another example: if x is a right triangle in the plane, and y is congruent to x, then y is also a right triangle, and the congruence informs us which of the 3 angles of y is the right angle.

Now we introduce the notion more formally.

Let X be a type, and let $T(x)$ be a family of types parametrized by a variable $x : X$ (as discussed in Section 2.2). Suppose $a, b : X$ and $e : a = b$. Then there is a function of type $T(a) \to T(b)$. We define one specific such function by induction on e, by taking its value on refl_a of type $a = a$ to be the identity function on $T(a)$. We record that definition as follows.

def:transport

DEFINITION 2.5.4. The function

$$\mathrm{trp}^T_e : T(a) \to T(b)$$

[21] For now, this is just a convention, but after we introduce the set of integers Z below in Definition 3.2.1, we'll be justified in writing p^z for any $z : \mathrm{Z}$.

is defined by induction setting $\mathrm{trp}^{T}_{\mathrm{refl}_a} :\equiv \mathrm{id}_{T(a)}$. ⌟

The function thus defined may be called *the transport function in the type family T along the path e*, or, less verbosely, *transport*.[22] We may also simplify the notation to just trp_e. The transport functions behave as expected: there is an identification of type $\mathrm{trp}_{e'\circ e} = \mathrm{trp}_{e'} \circ \mathrm{trp}_e$. In words: transport along the composition $e \circ e'$ can be identified with the composition of the two transport functions. This may be proved by induction in the following exercise.

xca:trp-compose

Exercise 2.5.5. Let X be a type, and let $T(x)$ be a family of types parametrized by a variable $x : X$. Suppose we are given elements $a, b, c : X$, $e : a = b$, and $e' : b = c$. Show there is an identification of type

$$\mathrm{trp}_{e'\circ e} = \mathrm{trp}_{e'} \circ \mathrm{trp}_e. \quad ⌟$$

Yet another example of good behavior is given in the following exercise.

xca:trp-nondep

Exercise 2.5.6. Let X, Y be types. As discussed in Section 2.2, we may regard the expression Y as a constant family of types parametrized by a variable $x : X$. Produce an identification of type $\mathrm{trp}^{Y}_{p} = \mathrm{id}_Y$, for any path $p : a = b$. ⌟

In Section 2.15 below we will discuss what it means for a type to have at most one element. When the types $T(x)$ may have more than one element, we may regard an element of $T(x)$ as providing additional *structure* on x. In that case, we will refer to the transport function $\mathrm{trp}_e : T(a) \to T(b)$ as *transport of structure* from a to b.

Take, for example, $T(x) :\equiv (x = x)$. Then trp_e is of type $a = a \to b = b$ and transports a symmetry of a to a symmetry of b.

By contrast, when the types $T(x)$ have at most one element, we may regard an element of $T(x)$ as providing a proof of a property of x. In that case, the transport function $\mathrm{trp}_e : T(a) \to T(b)$ provides a way to establish a claim about b from a claim about a, so we will refer to it as *substitution*. In other words, elements that can be identified have the same properties.

2.6 *Product types*

sec:product-types

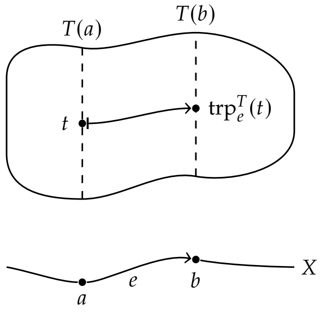
Functions and product types have been introduced in Section 2.2, where we have also explained how to create a function by enclosing a family of elements in one. In this section we treat functions and product types in more detail.

Recall that if X is a type and $Y(x)$ is a family of types parametrized by a variable x of type X, then there is a *product type* $\prod_{x:X} Y(x)$ whose elements f are functions that provide elements $f(a)$ of type $Y(a)$, one for each $a : X$. We will refer to X as the *parameter type* of the product. By contrast, if Y happens to be a constant family of types, then $\prod_{x:X} Y$ will also be denoted by $X \to Y$, and it will also be called a *function type*.

If X and $Y(x)$ are elements of a universe $\mathcal{U}$, then so is $\prod_{x:X} Y(x)$.

Functions preserve equality, and we will use this frequently later on. More precisely, functions induce maps on identity types, as the following definition makes precise.

[22] We sometimes picture this schematically as follows: We draw X as a (mostly horizontal) line, and we draw each type $T(x)$ as a vertical line lying over $x : X$. As x moves around in X, these lines can change shape, and taken all together they form a 2-dimensional blob lying over X. The transport functions map points between the vertical lines.



def:ap

Definition 2.6.1. For all types X, Y, functions $f : X \to Y$ and elements $x, x' : X$, the function

$$\mathrm{ap}_{f,x,x'} : (x = x') \to (f(x) = f(x'))$$

is defined by induction by setting $\mathrm{ap}_{f,x,x}(\mathrm{refl}_x) :\equiv \mathrm{refl}_{f(x)}$. ⌟

The function $\mathrm{ap}_{f,x,x'}$, for any elements x and x' of X, is called an *application* of f to paths or to identities, and this explains the choice of the symbol ap in the notation for it. It may also be called the function (or map) *induced* by f on identity types.

When x and x' are clear from the context, we may abbreviate $\mathrm{ap}_{f,x,x'}$ by writing ap_f instead. For convenience, we may abbreviate it even further, writing $f(p)$ for $\mathrm{ap}_f(p)$.

The following lemma shows that ap_f is compatible with composition.

lem:apcomp

Lemma 2.6.2. *Given a function $f : X \to Y$, and elements $x, x', x'' : X$, and paths $p : x = x'$ and $p' : x' = x''$, there is an identification of type $\mathrm{ap}_f(p' \cdot p) = \mathrm{ap}_f(p') \cdot \mathrm{ap}_f(p)$.*

Proof. By induction on p and p', one reduces to producing an element of type

$$\mathrm{ap}_f(\mathrm{refl}_x \cdot \mathrm{refl}_x) = \mathrm{ap}_f(\mathrm{refl}_x) \cdot \mathrm{ap}_f(\mathrm{refl}_x).$$

Both sides of the equation are equal to $\mathrm{refl}_{f(x)}$ by definition, so the element $\mathrm{refl}_{\mathrm{refl}_{f(x)}}$ has that type. □

In a similar way one shows that ap_f is compatible with inverse, $\mathrm{ap}_f(p^{-1}) = (\mathrm{ap}_f(p))^{-1}$, and that $\mathrm{ap}_{\mathrm{id}}$ is the identity on paths.

xca:trp-ap

Exercise 2.6.3. Let X be a type, and let $T(x)$ be a family of types parametrized by a variable $x : X$. Furthermore, let A be a type, let $f : A \to X$ be a function, and let $p : a = a'$ be a path. Show that there is an identification of type $\mathrm{trp}^{T \circ f}_p = \mathrm{trp}^T_{\mathrm{ap}_f(p)}$ between these two functions, which are of type $T(f(a)) \to T(f(a'))$. ⌟

If two functions f and g of type $\prod_{x:X} Y(x)$ can be identified, then their values can be identified, i.e., for every element x of X, we may produce an identification of type $f(x) = g(x)$, which can be constructed by induction, as follows.

def:ptw

Definition 2.6.4. Let $f, g : \prod_{x:X} Y(x)$. Define the function

$$\mathrm{ptw}_{f,g} : (f = g) \to \left(\prod_{x:X} f(x) = g(x)\right),$$

by induction by setting $\mathrm{ptw}_{f,f}(\mathrm{refl}_f) :\equiv x \mapsto \mathrm{refl}_{f(x)}$. [23] ⌟

Conversely, given $f, g : \prod_{x:X} Y(x)$, from a basic axiom called *function extensionality*, postulated in Principle 2.9.17, an identity $f = g$ can be produced from a family of identities of type $f(x) = g(x)$ parametrized by the elements x of X.

def:naturality-square-commutativity

Definition 2.6.5. Let X, Y be types and $f, g : X \to Y$ functions. Given an element h of type $\prod_{x:X} f(x) = g(x)$, elements x and x' of X, and a path $p : x = x'$, we can define by induction on p an element

$$\mathrm{ns}(h, p) : h(x') \cdot \mathrm{ap}_f(p) =_{f(x)=g(x')} \mathrm{ap}_g(p) \cdot h(x),$$

[23] The notation ptw is chosen to remind the reader of the word "point-wise", because the identities are provided just for each point x. An alternative approach goes by considering, for any $x : X$, the evaluation function $\mathrm{ev}_x : \left(\prod_{x:X} Y(x)\right) \to Y(x)$ defined by $\mathrm{ev}_x(f) :\equiv f(x)$. Then one could define $\mathrm{ptw}_{f,g}(p, x) :\equiv \mathrm{ap}_{\mathrm{ev}_x}(p)$. The functions provided by these two definitions are not equal by definition, but they can be identified, and one can easily be used in place of the other.

by induction, by setting $\mathrm{ns}(h, \mathrm{refl}_x)$ to be some previously constructed element of $h(x) \cdot \mathrm{refl}_{f(x)} = h(x)$. The type of $\mathrm{ns}(h, p)$ can be depicted as a square[24] and $\mathrm{ns}(h, p)$ is called a *naturality square*. ⌟

2.7 *Identifying elements in members of families of types*

If $Y(x)$ is a family of types parametrized by a variable x of type X, and a and a' are elements of type X, then after identifying a with a' it turns out that it is possible to "identify" an element of $Y(a)$ with an element of $Y(a')$, in a certain sense. That is the idea of the following definition.

def:pathsoverpaths

Definition 2.7.1. Suppose we are given a type X in a universe $\mathcal{U}$ and a family of types $Y(x)$, also in $\mathcal{U}$, parametrized by a variable x of type X. Given elements $a, a' : X$, $y : Y(a)$, and $y' : Y(a')$ and a path $p : a = a'$, we define a new type $y =^Y_p y'$ in $\mathcal{U}$ as follows. We proceed by induction on a' and p, which reduces us to the case where a' is a and p is refl_a, rendering y and y' of the same type $Y(a)$ in $\mathcal{U}$, allowing us to define $y =^Y_{\mathrm{refl}_a} y'$ to be $y = y'$, which is also in $\mathcal{U}$. ⌟

An element $q : y =^Y_p y'$ is called a *path* from y to y' *over* p.[25]

ft:path-over-pic

The following definition identifies the type of paths over p with a type of paths using transport along p.

def:pathover-trp

Definition 2.7.2. In the context of Definition 2.7.1, define by induction on p an identification $\mathrm{po}_p : \left(y =^Y_p y'\right) = \left(\mathrm{trp}^Y_p(y) = y'\right)$ in $\mathcal{U}$, by setting $\mathrm{po}_{\mathrm{refl}_x} :\equiv \mathrm{refl}_{y=y'}$. ⌟

Many of the operations on paths have counterparts for paths over paths. For example, we may define composition of paths over paths as follows.

def:pathovercomposition

Definition 2.7.3. Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. Suppose also that we have elements $x, x', x'' : X$, a path $p : x = x'$, and a path $p' : x' = x''$. Suppose further that we have elements $y : Y(x)$, $y' : Y(x')$, and $y'' : Y(x'')$, with paths $q : y =^Y_p y'$ over p and $q' : y' =^Y_{p'} y''$ over p'. Then we define the *composite* path $q' \mathbin{\underline{\circ}} q : y =^Y_{p' \circ p} y''$ over $p' \circ p$ as follows. First we apply path induction on x'' and p' to reduce to the case where x'' is x' and p' is $\mathrm{refl}_{x'}$. That also reduces the type $y' =^Y_{p'} y''$ to the identity type $y' = y''$, so we may apply path induction on y'' and q' to reduce to the case where y'' is y' and q' is $\mathrm{refl}_{y'}$. Now observe that $p' \circ p$ is p, so q provides the element we need. ⌟

Similarly, one can define the inverse of a path over a path, writing $q^{-1} : b' =^B_{p^{-1}} b$ for the inverse of $q : b =^B_p b'$. These operations on paths over paths satisfy many of the laws satisfied by the corresponding operations on paths, after some modification. We will state these laws when we need them.[26]

The following construction shows how to handle application of a dependent function f to paths using the definition above.

def:apd

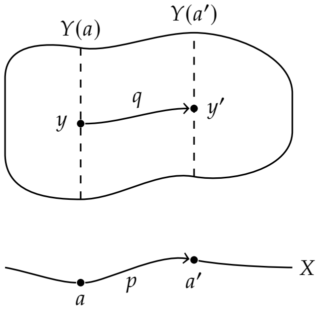
Definition 2.7.4. Suppose we are given a type X, a family of types $Y(x)$ parametrized by the elements x of X, and a function $f : \prod_x Y(x)$. Given elements $x, x' : X$ and a path $p : x = x'$, we define

$$\mathrm{apd}_f(p) : f(x) \overset{Y}{\underset{p}{=}} f(x')$$

[24] Like this, where we add little arrows to the equal signs to indicate their direction:

$$\begin{tikzcd} f(x) \arrow[r, equal, "\mathrm{ap}_f(p)", "\rightarrow"'] \arrow[d, equal, "h(x)"', "\downarrow"] & f(x') \arrow[d, equal, "h(x')", "\downarrow"'] \\ g(x) \arrow[r, equal, "\rightarrow", "\mathrm{ap}_g(p)"'] & g(x') \end{tikzcd}$$

[25] We picture this as follows: the path from y to y' over p travels through the vertical lines representing the types $Y(x)$ as $x : X$ moves along the path p in X from a to a':



[26] Exercise: Try to state some of these laws yourself.

by induction on p, setting

$$\mathrm{apd}_f\,(\mathrm{refl}_x) :\equiv \mathrm{refl}_{f(x)}. \qquad \lrcorner$$

The function apd_f is called *dependent application* of f to paths.[27] For convenience, we may abbreviate $\mathrm{apd}_f(p)$ to $f(p)$, when there is no risk of confusion.

The following construction shows how functions of two variables may be applied to paths over paths.

def:app1fun2

Definition 2.7.5. Suppose we are given a type X, a family of types $Y(x)$ parametrized by the elements x of X, and a type Z. Suppose also we are given a function $g : \prod_{x:X}(Y(x) \to Z)$ of two variables. Given elements $x, x' : X$, $y : Y(x)$, and $y' : Y(x')$, a path $p : x = x'$, and a path $q : y =^Y_p y'$ over p, we may construct a path

$$\mathrm{apap}_g(p)(q) : g(x)(y) = g(x')(y')$$

by induction on p and q, setting

$$\mathrm{apap}_g(\mathrm{refl}_x)(\mathrm{refl}_y) :\equiv \mathrm{refl}_{g(x)(y)}. \qquad \lrcorner$$

The function $p \mapsto q \mapsto \mathrm{apap}_g(p)(q)$ is called *application* of g to paths over paths. For convenience, we may abbreviate $\mathrm{apap}_g(p)(q)$ to $g(p)(q)$.

The following simple lemma will be useful later.

def:app1fun2comp

Definition 2.7.6. Suppose we are given a type X, a family of types $Y(x)$ parametrized by the elements x of X, and a type Z. Suppose also we are given a function $g : \prod_{x:X}(Y(x) \to Z)$ of two variables. Given an element $x : X$, elements $y, y' : Y(x)$, and an identity $q : y = y'$, then we define an identification of type $\mathrm{apap}_g(\mathrm{refl}_x)(q) = \mathrm{ap}_{g(x)}(q)$, by induction on q, thereby reducing to the case where y' is y and q is refl_y, rendering the two sides of the equation equal, by definition, to $\mathrm{refl}_{g(x)(y)}$. $\lrcorner$

2.8 *Sum types*

sec:sum-types

There are *sums* of types. By this we mean if X is a type and $Y(x)$ is a family of types parametrized by a variable x of type X, then there will be a type[28] $\sum_{x:X} Y(x)$ whose elements are all pairs (a, b), where $a : X$ and $b : Y(a)$. Since the type of b may depend on a we also call such a pair a *dependent* pair. We may refer to X as the *parameter type* of the sum.[29]

If X and $Y(x)$ are elements of a universe $\mathcal{U}$, then so is $\sum_{x:X} Y(x)$.

Proving something about (or constructing something from) every element of $\sum_{x:X} Y(x)$ is done by performing the construction on elements of the form (a, b), for every $a : X$ and $b : Y(a)$. Two important examples of such constructions are:

(1) *first projection*, $\mathrm{fst} : (\sum_{x:X} Y(x)) \to X$, $\mathrm{fst}(a, b) :\equiv a$;

it:second-projection

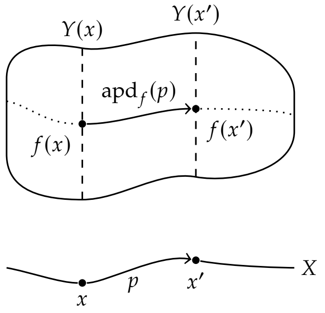
(2) *second projection*, $\mathrm{snd}(a, b) : Y(a)$, $\mathrm{snd}(a, b) :\equiv b$.

In (2), the type of snd is, in full, $\prod_{z : \sum_{x:X} Y(x)} Y(\mathrm{fst}(z))$.

iterated-sums

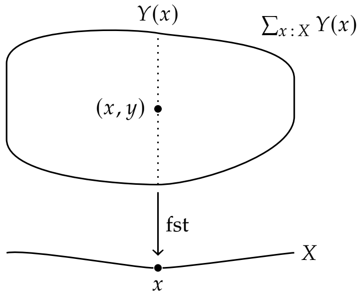
Remark 2.8.1. One may consider sums of sums. For example, suppose X is a type, suppose $Y(x)$ is a family of types parametrized by a variable x of type X, and suppose $Z(x, y)$ is a family of types parametrized by variables $x : X$ and $y : Y(x)$. In this case, the *iterated sum* $\sum_{x:X} \sum_{y:Y(x)} Z(x, y)$

[27] We picture f via its *graph* of the values $f(x)$ as x varies in X. The dependent application of f to p is then the piece of the graph that lies over p:



[28] Also known as a *Sigma-type*.

[29] We also call $\sum_{x:X} Y(x)$ the *total type* of the family, and we picture it, in the style of the pictures above, as the entire blob lying over X. (Each $Y(x)$ is a vertical line over $x : X$, and a point $y : Y(x)$ becomes a point (x, y) in the blob.)



consists of pairs of the form $(x,(y,z))$. For simplicity, we introduce the notation $(x,y,z) :\equiv (x,(y,z))$, and refer to (x,y,z) as a *triple* or as a 3-*tuple*.

That process can be repeated: suppose X_1 is a type, suppose $X_2(x_1)$ is a family of types parametrized by a variable x_1 of type X_1, suppose $X_3(x_1,x_2)$ is a family of types parametrized by variables $x_1 : X_1$ and $x_2 : X_2(x_1)$, and so on, up to a family $X_n(x_1,\ldots,x_{n-1})$ of types. In this case, the *iterated sum*

$$\sum_{x_1 : X_1} \sum_{x_2 : X_2(x_1)} \cdots \sum_{x_{n-1} : X_{n-1}(x_1,\ldots,x_{n-2})} X_n(x_1,\ldots,x_{n-1})$$

consists of elements of the form $(x_1,(x_2,(\ldots(x_{n-1},x_n)\ldots)))$; each such element is a pair whose second member is a pair, and so on, so we may refer to it as an *iterated pair*. For simplicity, we introduce the notation $(x_1,x_2,\ldots,x_n)$ for such an iterated pair, and refer to it as an *n-tuple*. ⌟

2.9 *Equivalences*

sec:equivalence

Using a combination of sum, product, and identity types allows us to express important notions, as done in the following definitions.

The property that a type X has "exactly one element" may be made precise by saying that X has an element such that every other element is equal to it. This property is encoded in the following definition.

def:contractible

Definition 2.9.1. Given a type X, define a type $\mathrm{isContr}(X)$ by setting

$$\mathrm{isContr}(X) :\equiv \sum_{c:X} \prod_{x:X} (c = x). \qquad ⌟$$

If $(c,h) : \mathrm{isContr}(X)$, then c will be called the *center* of the the *contraction* h, and we call the type X *contractible*.

By path composition, one sees that any element $x : X$ can serve as the center of a contraction of a contractible type X.

The following lemma gives an important example of a contractible type.

Give a type X and an element a of X, the *singleton type* $\sum_{x:X}(a = x)$ consists of pairs (x,i) with $i : a = x$. The following lemma shows that a singleton type has exactly one element, justifying the name.

lem:thepathspaceiscontractible

Lemma 2.9.2. *For any type X and $a : X$, the singleton type $\sum_{x:X}(a = x)$ is contractible.*

Proof. Take as center the pair (a,refl_a). We have to produce, for any element x of X and for any identification $i : a = x$, an identification of type $(a,\mathrm{refl}_a) = (x,i)$. This is done by path induction on x and i, which reduces us to producing an identity of type $(a,\mathrm{refl}_a) = (a,\mathrm{refl}_a)$; reflexivity provides one, namely $\mathrm{refl}_{(a,\mathrm{refl}_a)}$. □

def:fiber

Definition 2.9.3. Given a function $f : X \to Y$ and an element $y : Y$, the *fiber* (or *preimage*) $f^{-1}(y)$ is encoded by defining

$$f^{-1}(y) :\equiv \sum_{x:X} (y = f(x)).$$

In other words, an element of the fiber $f^{-1}(y)$ is a pair consisting of an element x of X and an identification of type $y = f(x)$. ⌟

In set theory, a function $f : X \to Y$ is a bijection if and only if all preimages $f^{-1}(y)$ consist of exactly one element. This we can also express in type theory, in a definition due to Voevodsky.

def:equivalence

Definition 2.9.4. A function $f : X \to Y$ is called an *equivalence* if $f^{-1}(y)$ is contractible for all $y : Y$. The condition is encoded by the type

$$\mathrm{isEquiv}(f) :\equiv \prod_{y:Y} \mathrm{isContr}(f^{-1}(y)).$$

⌟

We may say that X and Y are *equivalent* if there is an equivalence between them.

def:type-of-equivalences

Definition 2.9.5. We define the type $X \simeq Y$ of equivalences from X to Y by the following definition.

$$(X \simeq Y) :\equiv \sum_{f:X\to Y} \mathrm{isEquiv}(f).$$

⌟

Suppose $f : X \to Y$ is an equivalence, and let $t(y) : \mathrm{isContr}(f^{-1}(y))$, for each $y : Y$, be the corresponding witness to contractibility. Using t we can define an inverse function $g : Y \to X$ by setting $g(y) :\equiv \mathrm{fst}(\mathrm{fst}(t(y)))$.[30]

Clearly, $f(g(y)) = y$ by unfolding all definitions. Moreover, we have $(x, \mathrm{refl}_{f(x)}) : f^{-1}(f(x))$, with the latter as the fiber that $t(f(x))$ proves contractible. Hence the center of contraction $\mathrm{fst}(t(f(x))$ is equal to $(x, \mathrm{refl}_{f(x)})$, and so $g(f(x)) \equiv (\mathrm{fst}(\mathrm{fst}(t(f(x))) = x$.

We have shown that f and g are inverse functions. When it won't cause confusion with the notation for the fibers of f, we will write f^{-1} for g.

For any type X, the identity function id_X is an equivalence from X to X: for every element a in X, $\mathrm{id}_X^{-1}(a)$ is a singleton type and hence contractible. This simple observation combined with the fact that $\mathrm{trp}^T_{\mathrm{refl}_x} \equiv \mathrm{id}_{T(x)}$ gives that the function trp^T_e from Definition 2.5.4 is an equivalence from $T(x)$ to $T(y)$, for all $e : x = y$.

xca:equivalence-invers

Exercise 2.9.6. Make sure you understand the two applications of fst in the definition $f^{-1}(y) :\equiv \mathrm{fst}(\mathrm{fst}(t(y)))$ above. Show that f^{-1} is an equivalence from Y to X. Give a function $(X \simeq Y) \to (Y \simeq X)$. ⌟

xca:equivalence-comp

Exercise 2.9.7. Give a function $(X \simeq Y) \to ((Y \simeq Z) \to (X \simeq Z))$. ⌟

xca:2-out-of-3

Exercise 2.9.8. Consider types A, B, and C, functions $f : A \to B$, $g : A \to C$ and $h : B \to C$, together with an element $e : hf = g$. Prove that if two of the three functions are equivalences, then so is the third one. ⌟

The following lemma gives an equivalent characterization of equivalence that is sometimes easy to use.

lem:weq-iso

Lemma 2.9.9. *Let X, Y be types. A function $f : X \to Y$ is an equivalence if and only if there exists a function $g : Y \to X$ such that for all $x : X$ we have $g(f(x)) = x$ and for all $y : Y$ we have $f(g(y)) = y$.*

Proof. If $f : X \to Y$ is an equivalence we can take $g = f^{-1}$. For the converse, see Chapter 4 of the HoTT Book,[31] or `isweq_iso`. □

We put Lemma 2.9.9 immediately to good use.

[30] Note that $\mathrm{fst}(t(y)) : f^{-1}(y)$, so $\mathrm{fst}(\mathrm{fst}(t(y))) : X$ with $\mathrm{snd}(\mathrm{fst}(t(y))) : y = f(\mathrm{fst}(\mathrm{fst}(t(y))))$.

[31] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

lem:contract-away

Lemma 2.9.10. *Let X be a type with element a, and let $B(x,i)$ be a type for all $x:X$ and $i:a=x$. Define $f(x,i):B(x,i)\to B(a,\mathrm{refl}_a)$ by induction on i, setting $f(a,\mathrm{refl}_a,b):\equiv b$ for all $b:B(a,\mathrm{refl}_a)$. Then f defines an equivalence*

$$f\ :\ \sum_{x:X}\sum_{i:a=x}B(x,i)\quad\to\quad B(a,\mathrm{refl}_a).$$

Proof. We can also define $g:B(a,\mathrm{refl}_a)\to\sum_{x:X}\sum_{i:a=x}B(x,i)$ mapping $b:B(a,\mathrm{refl}_a)$ to (a,refl_a,b). Clearly $f(g(b))=b$ for all $b:B(a,\mathrm{refl}_a)$. Moreover, $g(f(x,i,b))=(x,i,b)$ is clear by induction on i, for all $b:B(x,i)$. By Lemma 2.9.9 it follows that f is an equivalence. □

The above lemma clearly reflects the contractibility of the singleton type $\sum_{x:X}(a=x)$.[32] For this reason we call application of this lemma 'to contract away' the prefix $\sum_{x:X}\sum_{i:a=x}$, in order to obtain a simpler type. It is often applied in the following simpler form.

cor:contract-away

Corollary 2.9.11. *With conditions as above, but with B not depending on i, the same f establishes an equivalence*

$$\sum_{x:X}((a=x)\times B(x))\ \ \simeq\ \ B(a).$$

In the direction of further generality, we offer the following exercise.

xca:sum-equiv-base

Exercise 2.9.12. Suppose X,Y are types connected by an equivalence $f:X\to Y$. Let $B(x)$ be a type for all $x:X$. Construct an equivalence between $\sum_{x:X}B(x)$ and $\sum_{y:Y}B(f^{-1}(y))$. ⌟

We proceed now to define the notion of fiberwise equivalence.

def:fiberwise

Definition 2.9.13. Let X be a type, and let $Y(x),Z(x)$ be families of types parameterised by $x:X$. A map f of type $\prod_{x:X}(Y(x)\to Z(x))$ can be viewed as a family of maps $f(x):Y(x)\to Z(x)$ and is called a *fiberwise* map. The *totalization* of f is defined as

$$\mathrm{tot}(f):\Big(\sum_{x:X}Y(x)\Big)\to\sum_{x:X}Z(x),$$

setting $\mathrm{tot}(f)(x,y):\equiv(x,f(x)(y))$. ⌟

lem:fiberwise

Lemma 2.9.14. *Let conditions be as in Definition 2.9.13. If $f(x):Y(x)\to Z(x)$ is an equivalence for every $x:X$ (we say that f is a* fiberwise *equivalence), then* $\mathrm{tot}(f)$ *is an equivalence.*

Proof. If $f(x):Y(x)\to Z(x)$ is an equivalence for all x in X, then the same is true of all $f(x)^{-1}:Z(x)\to Y(x)$. Then we have the totalization $\mathrm{tot}(x\mapsto f(x)^{-1})$, which can easily be proved to be an inverse of $\mathrm{tot}(f)$ (see the next exercise). Now apply Lemma 2.9.9. □

xca:fiberwise

Exercise 2.9.15. Complete the details of the proof of Lemma 2.9.14. ⌟

The converse to Lemma 2.9.14 also holds.

lem:fiberwise-equiv-from-tot

Lemma 2.9.16. *Continuing with the setup of Definition 2.9.13, if* $\mathrm{tot}(f)$ *is an equivalence, then f is a fiberwise equivalence.*

For a proof see Theorem 4.7.7 of the HoTT Book[33].

Yet another application of the notion of equivalence is to postulate axioms.

[32] In fact, an alternative proof would go as follows: First, we use Lemma 2.9.9 to show associativity of sum types, i.e., $\sum_{x:X}\sum_{y:Y(x)}Z(x,y)\simeq\sum_{w:(\sum_{x:X}Y(x))}Z(\mathrm{fst}\,w,\mathrm{snd}\,w)$, where X is a type, $Y(x)$ is a family of types depending on $x:X$, and $Z(x,y)$ is a family of types depending on $x:X$ and $y:Y(x)$. Then, we show for any contractible type X and for any family of types $Y(x)$ depending on $x:X$, that there is an equivalence between $\sum_{x:X}Y(x)$ and $Y(c)$, where c is the center of contraction.

[33] Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*.

def:funext

Principle 2.9.17. The axiom of *function extensionality* postulates that the function $\mathrm{ptw}_{f,g} : f = g \to \prod_{x:X} f(x) = g(x)$ in Definition 2.6.4 is an equivalence. Formally, we postulate the existence of an element $\mathrm{funext} : \mathrm{isEquiv}(\mathrm{ptw}_{f,g})$. From that we can construct the corresponding inverse function.

$$\mathrm{ptw}^{-1}_{f,g} : \Big(\prod_{x:X} f(x) = g(x)\Big) \to f = g.$$

Thus two functions whose values can all be identified can themselves be identified. This supports the intuition that there is nothing more to a function than the values it sends its arguments to. ⌟

2.10 *Identifying pairs*

sec:pairpaths

Equality of two elements of $\sum_{x:X} Y(x)$ is inductively defined in Section 2.5, as for any other type, but one would like to express equality between pairs in terms of equalities in the constituent types. This would explain better what it means for two pairs to be equal. We start with a definition.

def:pairtopath

Definition 2.10.1. Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. Consider the function

$$\mathrm{pair} : \prod_{x:X}\Big(Y(x) \to \sum_{x':X} Y(x')\Big)$$

defined by

$$\mathrm{pair}(x)(y) :\equiv (x,y).$$

For any elements (x,y) and (x',y') of $\sum_{x:X} Y(x)$, we define the map

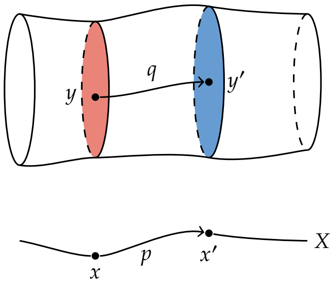
$$\Big(\sum_{p:x=x'} y \overset{Y}{\underset{p}{=}} y'\Big) \to ((x,y) = (x',y'))$$

by

$$(p,q) \mapsto \mathrm{apap}_{\mathrm{pair}}(p)(q).$$

(Refer to Definition 2.7.1 for the meaning of the type $y =^{Y}_{p} y'$, and to Definition 2.7.5 for the definition of apap.) We introduce $\overline{(p,q)}$ as notation for $\mathrm{apap}_{\mathrm{pair}}(p)(q)$. ⌟

We picture paths between pairs much in the same way as paths over paths, cf. Footnote 25. Just as, to give a pair in the sum type $\sum_{x:X} Y(x)$, we need both the point x in the parameter type X as well as the point y in $Y(x)$, to give a path from (x,y) to (x',y'), we need both a path $p : x = x'$ as well as a path $q : y =^{Y}_{p} y'$ over p. Here's a similar picture, where we depict the types in the family as being 2-dimensional for a change.



cor:isEq-pair=

Lemma 2.10.2. *In the situation of Definition 2.10.1, if x' is x, so that we have $(y =^{Y}_{\mathrm{refl}_x} y') \equiv (y = y')$, then for any $q : y = y'$, the identity*

$$\overline{(\mathrm{refl}_x, q)} = \mathrm{ap}_{\mathrm{pair}(x)}\, q$$

holds.

Proof. By induction on q it suffices to establish the identity

$$\overline{(\mathrm{refl}_x, \mathrm{refl}_y)} = \mathrm{ap}_{\mathrm{pair}(x)}(\mathrm{refl}_y),$$

both sides of which are equal to $\mathrm{refl}_{(x,y)}$ by definition. □

The following lemma gives the desired characterization of paths between pairs.

lem:isEq-pair=

Lemma 2.10.3. *Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. For any elements (x,y) and (x',y') of $\sum_{x:X} Y(x)$, the map defined in Definition 2.10.1 defined by*

$$(p,q) \mapsto \overline{(p,q)}$$

is an equivalence of type

$$\left(\sum_{p:x=x'} y \overset{Y}{\underset{p}{=}} y'\right) \simeq ((x,y)=(x',y')).$$

Proof. Call the map Φ. A map the other way,

$$\Psi : ((x,y)=(x',y')) \to \sum_{p:x=x'} y \overset{Y}{\underset{p}{=}} y',$$

can be defined by induction, by setting

$$\Psi(\mathrm{refl}_{(x,y)}) :\equiv (\mathrm{refl}_x, \mathrm{refl}_y).$$

One proves, by induction on paths, the identities $\Psi(\Phi(p,q)) = (p,q)$ and $\Phi(\Psi(r)) = r$, so Ψ and Φ are inverse functions. Applying Lemma 2.9.9, we see that Φ and Ψ are inverse equivalences, thereby obtaining the desired result. □

We often use $\mathrm{fst}(\overline{(p,q)}) = p$ and $\mathrm{snd}(\overline{(p,q)}) = q$, which follow by induction on p and q from the definitions of ap and $\overline{(\_,\_)}$. Similarly, $r = \overline{(\mathrm{fst}(r), \mathrm{snd}(r))}$ by induction on r.

2.11 *Binary products*

sec:binprod-types

There is special case of sum types that deserves to be mentioned since it occurs quite often. Let X and Y be types, and consider the constant family of types $Y(x) :\equiv Y$. In other words, $Y(x)$ is a type that depends on an element x of X that happens to be Y for any such x. (Recall Exercise 2.5.6.) Then we can form the sum type $\sum_{x:X} Y(x)$ as above. Elements of this sum type are pairs (x,y) with x in X and y in $Y(x) \equiv Y$.[34] In this case the type of y doesn't depend on x, and in this special case the sum type is called the *binary product*, or *cartesian product* of the types X and Y, denoted by $X \times Y$.

At first glance, it might seem odd that a sum is also a product, but exactly the same thing happens with numbers, for the sum $5+5+5$ is also referred to as the product 3×5. Indeed, that's one way to define 3×5.

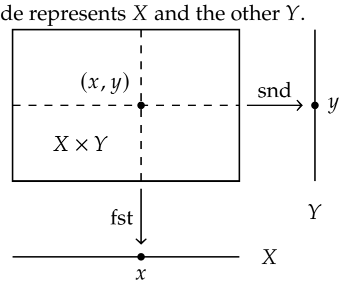
Recall that we have seen something similar with the product type $\prod_{x:X} Y(x)$, which we let $X \to Z$ denote in the case where $Y(x)$ is a constant family of the form $Y(x) :\equiv Z$, for some type Z.

The type $X \times Y$ inherits the functions fst, snd from $\sum_{x:X} Y(x)$, with the same definitions $\mathrm{fst}(x,y) :\equiv x$ and $\mathrm{snd}(x,y) :\equiv y$. Their types can now be denoted in a simpler way as $\mathrm{fst} : (X \times Y) \to X$ and $\mathrm{snd} : (X \times Y) \to Y$, and they are called as before the first and the second projection, respectively.

Again, proving something about (or constructing something from) every element (a,b) of $X \times Y$ is simply done for all $a : X$ and $b : Y$.

There is an equivalence between $(a_1,b_1) = (a_2,b_2)$ and $(a_1 = a_2) \times (b_1 = b_2)$. This follows from Lemma 2.10.3 together with Exercise 2.5.6.

[34] These *cartesian* products we illustrate as usual by rectangles where one side represents X and the other Y.



If $f : X \to Y$ and $f' : X' \to Y'$, then we let $f \times f'$ denote the map of type $(X \times X') \to (Y \times Y')$ that sends (x, x') to $(f(x), f'(x'))$.

The following lemma follows from Lemma 2.10.3, combined with Definition 2.7.2 and Exercise 2.5.6.

lem:isEq-pair-bin=

Lemma 2.11.1. *Suppose we are given type X and Y. For any elements (x, y) and (x', y') of $X \times Y$, the map defined in Definition 2.10.1 defined by*

$$(p, q) \mapsto \overline{(p, q)}$$

is an equivalence of type

$$(x = x') \times (y = y') \simeq ((x, y) = (x', y'))\,.$$

xca:binary-prod-equiv

Exercise 2.11.2. Let X, Y be types in a universe $\mathcal{U}$, and consider the type family $T(z)$ in $\mathcal{U}$ depending on $z : \mathrm{Bool}$ defined by $T(\mathrm{no}) :\equiv X$ and $T(\mathrm{yes}) :\equiv Y$. Show that the function $(\prod_{b : \mathrm{Bool}} T(b)) \to X \times Y$ sending f to $(f(\mathrm{no}), f(\mathrm{yes}))$, is an equivalence. ⌟

2.12 *More inductive types*

sec:inductive-types

There are other examples of types that are conveniently introduced in the same way as we have seen with the natural numbers and the equality types. A type presented in this style shares some common features: there are some ways to create new elements, and there is a way (called *induction*) to prove something about every element of the type (or family of types). We will refer to such types as *inductive* types, and we present a few more of them in this section, including the finite types, and then we present some other constructions for making new types from old ones. For each of these constructions we explain what it means for two elements of the newly constructed type to be equal in terms of equality in the constituent types.

2.12.1 *Finite types*

sec:finite-types

Firstly, there is the *empty* type in the universe $\mathcal{U}_0$, denoted by $\emptyset$ or by False. It is an inductive type, with no way to construct elements of it. The induction principle for $\emptyset$ says that to prove something about (or to construct something from) every element of $\emptyset$, it suffices to consider no special cases (!). Hence, every statement about an arbitrary element of $\emptyset$ can be proven. (This logical principle is traditionally called *ex falso quodlibet*.[35]) As an example, we may prove that any two elements x and y of $\emptyset$ are equal by using induction on x.

An element of $\emptyset$ will be called an *absurdity*. Of course, one expects that there are no real absurdities in mathematics, nor in any logical system (such as ours) that attempts to provide a language for mathematics, but it is important to have such a name so we can discuss the possibility, which might result inadvertently from the introduction of unwarranted assumptions. For example, to assert that a type P has no elements, it would be sensible to assert that an element of P would lead to an absurdity. Providing a function of type $P \to \emptyset$ is a convenient way to make that assertion. Hence we define the *negation* of a type by setting $\neg P :\equiv (P \to \emptyset)$. Using it, we may define the type $(a \neq b) :\equiv \neg(a = b)$; an element of it asserts that a and b cannot be identified.

[35] From falsehood, anything follows.

Secondly, there will also be an inductive type called True in the universe $\mathcal{U}_0$ provided with a single element triv; (the name triv comes from the word "trivial"). Its induction principle states that, in order to prove something about (or to construct something from) every element of True, it suffices to consider the special case where the element is triv. As an example, we may construct, for any element $u : \mathrm{True}$, an identity of type $u = \mathrm{triv}$; we use induction to reduce to the case where u is triv, and then $\mathrm{refl}_{\mathrm{triv}}$ provides the desired element. One may also construct, for any elements x and y of True, an identity of type $x = y$ by using induction both on x and on y.

There is a function $X \to \mathrm{True}$, for any type X, namely: $a \mapsto \mathrm{triv}$. This corresponds, for propositions, to the statement that an implication holds if the conclusion is true.

Thirdly, there will be an inductive type called Bool in the universe $\mathcal{U}_0$, provided with two elements, yes and no. Its induction principle states that, in order to prove something about (or to construct something from) every element of Bool, it suffices to consider two cases: the special case where the element is yes and the special case where the element is no.

We may use substitution to construct an element of type yes $\neq$ no. To do this, we introduce a family of types $P(b)$ in the universe $\mathcal{U}_0$ parametrized by a variable $b : \mathrm{Bool}$. We define $P(b)$ by induction on b by setting $P(\mathrm{yes}) :\equiv \mathrm{True}$ and $P(\mathrm{no}) :\equiv \mathrm{False}$. (The definition of $P(b)$ is motivated by the expectation that we will be able to construct an equivalence between $P(b)$ and $b = \mathrm{yes}$.) If there were an element $e : \mathrm{yes} = \mathrm{no}$, we could substitute no for yes in $\mathrm{triv} : P(\mathrm{yes})$ to get an element of $P(\mathrm{no})$, which is absurd. Since e was arbitrary, we have defined a function $(\mathrm{yes} = \mathrm{no}) \to \emptyset$, as desired.

In the same way, we may use substitution to prove that successors of natural numbers are never equal to 0, i.e., for any $n : \mathbb{N}$ that $0 \neq \mathrm{succ}(n)$. To do this, we introduce a family of types $P(i)$ in $\mathcal{U}_0$ parametrized by a variable $i : \mathbb{N}$. Define P recursively by specifying that $P(0) :\equiv \mathrm{True}$ and $P(\mathrm{succ}(m)) :\equiv \mathrm{False}$. (The definition of $P(i)$ is motivated by the expectation that we will be able to construct an equivalence between $P(i)$ and $i = 0$.) If there were an element $e : 0 = \mathrm{succ}(n)$, we could substitute $\mathrm{succ}(n)$ for 0 in $\mathrm{triv} : P(0)$ to get an element of $P(\mathrm{succ}(n))$, which is absurd. Since e was arbitrary, we have defined a function $(0 = \mathrm{succ}(n)) \to \emptyset$, establishing the claim.

In a similar way we will in Section 2.24 define types $\mathbb{n}$ for any n in $\mathbb{N}$ such that $\mathbb{n}$ is a type (set) of n elements.

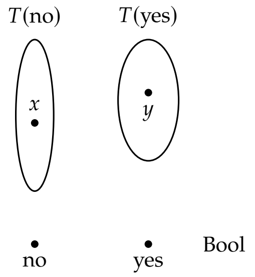
2.12.2 *Binary sums*

sec:binsum-types

For sum types of the form $\sum_{b : \mathrm{Bool}} T(b)$, with $T(b)$ a type depending on b in Bool, there is an equivalence with a simpler type.[36] After all, the type family $T(b)$ is fully determined by two types, namely by the types $T(\mathrm{no})$ and $T(\mathrm{yes})$. The elements of $\sum_{b : \mathrm{Bool}} T(b)$ are dependent pairs (no, x) with x in $T(\mathrm{no})$ and (yes, y) with y in $T(\mathrm{yes})$. The resulting type can be viewed as the *disjoint union* of $T(\mathrm{no})$ and $T(\mathrm{yes})$: from an element of $T(\mathrm{no})$ or an element of $T(\mathrm{yes})$ we can produce an element of $\sum_{b : \mathrm{Bool}} T(b)$.

These disjoint union types can be described more clearly in the following way. The *binary sum* of two types X and Y, denoted $X \amalg Y$,

[36] In a case like this, we can thicken up the lines denoting $T(\mathrm{no})$ and $T(\mathrm{yes})$ in our picture, if we like:



is an inductive type with two constructors: $\mathrm{inl} : X \to X \amalg Y$ and $\mathrm{inr} : Y \to X \amalg Y$.[37] Proving a property of any element of $X \amalg Y$ means proving that this property holds of any inl_x with $x : X$ and any inr_y with $y : Y$. In general, constructing a function f of type $\prod_{z : X \amalg Y} T(z)$, where $T(z)$ is a type depending on z, is done by defining $f(\mathrm{inl}_x)$ for all x in X and $f(\mathrm{inr}_y)$ for all y in Y.

xca:binary-sum-equiv

Exercise 2.12.3. Let X, Y be types in a universe $\mathcal{U}$, and consider the type family $T(z)$ in $\mathcal{U}$ depending on $z : \mathrm{Bool}$ defined by induction on z by $T(\mathrm{no}) :\equiv X$ and $T(\mathrm{yes}) :\equiv Y$. Show that the map $f : X \amalg Y \to \sum_{b : \mathrm{Bool}} T(b)$, defined by $f(\mathrm{inl}_x) :\equiv (\mathrm{no}, x)$ and $f(\mathrm{inr}_y) :\equiv (\mathrm{yes}, y)$, is an equivalence. ⌟

Identification of two elements a and b in $X \amalg Y$ is only possible if they are constructed with the same constructor. Thus $\mathrm{inl}_x = \mathrm{inr}_y$ is always empty, and there are equivalences of type $(\mathrm{inl}_x = \mathrm{inl}_{x'}) \simeq (x = x')$ and $(\mathrm{inr}_y = \mathrm{inr}_{y'}) \simeq (y = y')$.

xca:binary-sum-id

Exercise 2.12.4. Prove these statements using Exercise 2.12.3, Lemma 2.10.3, and a characterization of the identity types of Bool. ⌟

2.12.5 *Unary sums*

sec:unary-sum-types

Sometimes it is useful to be able to make a copy of a type X: A new type that behaves just like X, though it is not definitionally equal to X. The *unary sum* or *wrapped copy* of X is an inductive type $\mathrm{Copy}(X)$ with a single constructor, $\mathrm{in} : X \to \mathrm{Copy}(X)$.[38] Constructing a function $f : \prod_{z : \mathrm{Copy}(X)} T(z)$, where $T(z)$ is a type depending on $z : \mathrm{Copy}(X)$, is done by defining $f(\mathrm{in}_x)$ for all $x : X$. Taking $T(z)$ to be the constant family at X, we get a function, $\mathrm{out} : \mathrm{Copy}(X) \to X$, called the *destructor*, with $\mathrm{out}(\mathrm{in}_x) :\equiv x$ for $x : X$, and the induction principle implies that $\mathrm{in}_{\mathrm{out}(z)} = z$ for all $z : \mathrm{Copy}(X)$, so there is an equivalence of type $\mathrm{Copy}(X) \simeq X$, as expected. In fact, we will assume that the latter equation even holds definitionally. It follows that there are equivalences of type $(\mathrm{in}_x = \mathrm{in}_{x'}) \simeq (x = x')$ and $(\mathrm{out}(z) = \mathrm{out}(z')) \simeq (z = z')$.

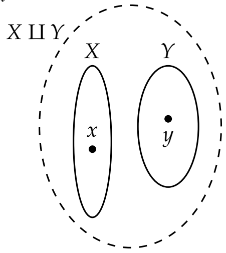
Note that we can make several copies of X that are not definitionally equal to each other, for instance, by picking different names for the constructor. We write $\mathrm{Copy}_{\mathrm{con}}(X)$ for a copy of X whose constructor is

$$\mathrm{con} : X \to \mathrm{Copy}_{\mathrm{con}}(X).$$

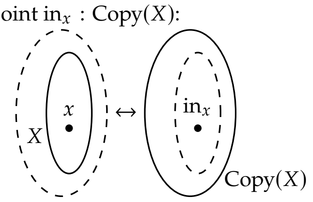
exa:nm

Example 2.12.6. Here's an example to illustrate why it can be useful to make such a wrapped type: We introduced the natural numbers $\mathbb{N}$ in Section 2.4. Suppose we want a type consisting of negations of naturals numbers, $\{\ldots, -2, -1, 0\}$, perhaps as an intermediate step towards building the set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.[39] Of course, the type $\mathbb{N}$ itself would do, but then we would need to pay extra attention to whether $n : \mathbb{N}$ is supposed to represent n as an integer or its negation. So instead we take the wrapped copy $\mathbb{N}^- :\equiv \mathrm{Copy}_-(\mathbb{N})$, with constructor $- : \mathbb{N} \to \mathbb{N}^-$. There is no harm in also writing $- : \mathbb{N}^- \to \mathbb{N}$ for the destructor. This means that there is an equivalence of type $\mathbb{N}^- \simeq \mathbb{N}$, for the elements of $\mathbb{N}^-$ are of the form $-n$ for $n : \mathbb{N}$. Indeed, $-(-n) \equiv n$ for n an element of either $\mathbb{N}$ or $\mathbb{N}^-$, and there is an equivalence of type $(-n = -n') \simeq (n = n')$. ⌟

[37] Be aware that in a picture, the same point may refer either to x in X or to inl_x in the sum $X \amalg Y$:



[38] A point $x : X$ corresponds to the point $\mathrm{in}_x : \mathrm{Copy}(X)$:



Note that $\mathrm{Copy}(X)$ can alternatively be defined as $\sum_{z : \mathrm{True}} X$.

[39] We implement this in Definition 3.2.1.

2.13 *Univalence*

sec:univax

The univalence axiom, to be presented in this section, greatly enhances our ability to produce identities between the two types and to use the resulting identities to transport (in the sense of Definition 2.5.4) properties and structure between the types. It asserts that if $\mathcal{U}$ is a universe, and X and Y are types in $\mathcal{U}$, then there is an equivalence between identities between X and Y and equivalences between X and Y.

We now define the function that the univalence axiom postulates to be an equivalence.

def:idtoeq

Definition 2.13.1. For types X and Y in a universe $\mathcal{U}$ and a path $p : X = Y$, we define an equivalence $\mathrm{cast}_{X,Y}(p) : X \simeq Y$ by induction on Y and p, setting $\mathrm{cast}_{X,X}(\mathrm{refl}_X) :\equiv \mathrm{id}_X : X \simeq X$. The result is a function

$$\mathrm{cast}_{X,Y} : (X = Y) \to (X \simeq Y). \qquad \lrcorner$$

In expressions such as $\mathrm{cast}_{X,Y}(p)$ we may abbreviate $\mathrm{cast}_{X,Y}$ to cast if no confusion will result. We may also write $\mathrm{cast}(p)$ more briefly as $\tilde{p}$, which we also use to denote the corresponding function from X to Y.[40]

Let T be a variable of type $\mathcal{U}$; then we may view T as a family of types parametrized by $\mathcal{U}$, of the sort required for use with transport as defined in Definition 2.5.4. One may construct an identity of type $\mathrm{cast}(p)(x) = \mathrm{trp}^T_p(x)$, for $x : X$, by induction on Y and p. As a corollary, one sees that the function trp^T_p is an equivalence.

We are ready to state the univalence axiom.

def:univalence

Principle 2.13.2 (Univalence Axiom). Voevodsky's *univalence axiom* postulates that $\mathrm{cast}_{X,Y}$ is an equivalence for all $X, Y : \mathcal{U}$. Formally, we postulate the existence of an element

$$\mathrm{ua}_{X,Y} : \mathrm{isEquiv}(\mathrm{cast}_{X,Y}). \qquad \lrcorner$$

For an equivalence $f : X \simeq Y$, we will adopt the notation $\mathrm{ua}(f) : X = Y$ to denote $\mathrm{cast}^{-1}_{X,Y}(f)$, the result of applying the inverse function of $\mathrm{cast}_{X,Y}$ to f, if no confusion will result. Thus there are identities of type $\mathrm{cast}(\mathrm{ua}(f)) = f$ and $\mathrm{ua}(\mathrm{cast}(p)) = p$.

We may also write $\mathrm{ua}(f)$ more briefly as $\bar{f}$. Thus there are identities of type $\bar{\tilde{p}} = p$ and $\tilde{\bar{f}} = f$. There are also identities of type $\overline{\mathrm{id}_X} = \mathrm{refl}_X$ and $\overline{g\,f} = \bar{g}\,\bar{f}$ if $g : Y \simeq Z$.

xca:C2

Exercise 2.13.3. Prove that $\mathrm{Bool} = \mathrm{Bool}$ has exactly two elements, $\mathrm{refl}_{\mathrm{Bool}}$ and twist (where twist is given by univalence from the equivalence $\mathrm{Bool} \to \mathrm{Bool}$ interchanging the two elements of Bool), and that $\mathrm{twist} \cdot \mathrm{twist} = \mathrm{refl}_{\mathrm{Bool}}$. $\lrcorner$

2.14 *Heavy transport*

sec:heavy-transport

In this section we collect useful results on transport in type families that are defined by a type constructor applied to families of types. Typical examples of such 'structured' type families are $Y(x) \to Z(x)$ and $x = x$ parametrized by $x : X$.

def:function-type-families

Definition 2.14.1. Let X be a type, and let $Y(x)$ and $Z(x)$ be families of types parametrized by a variable $x : X$. Define $Y \to Z$ to be the type family with $(Y \to Z)(x) :\equiv Y(x) \to Z(x)$. $\lrcorner$

[40] *Cast* is here used in the sense of "arranging into a suitable form". A more theatrical description would be that an element x of X is cast in the *role* of an element of Y as *directed* by the path $p : X = Y$. But beware that some programming languages use *cast* in a different sense: to take a bit-pattern representing an object of type X and simply *reinterpreting* the same bits as an object of type Y.

Recall from Definition 2.9.13 that an element $f : \prod_{x:X}(Y \to Z)(x)$ is called a fiberwise map, and f is called a fiberwise equivalence, if $f(x) : Y(x) \to Z(x)$ is an equivalence for all $x : X$.

lem:trp-in-function-type

Lemma 2.14.2. *Let X be a type, and let $Y(x)$ and $Z(x)$ be types for every $x : X$. Then we have for every $x, x' : X$, $e : x = x'$, $f : Y(x) \to Z(x)$, and $y' : Y(x')$:*

$$\mathrm{trp}_e^{Y\to Z}(f)(y') = \mathrm{trp}_e^Z\left(f\left(\mathrm{trp}^Y_{e^{\text{-}1}}(y')\right)\right).$$

Proof. By induction on $e : x = x'$. For $e \equiv \mathrm{refl}_x$, we have $e^{-1} \equiv \mathrm{refl}_x$, and all transports are identity functions of appropriate type. □

An important special case of the above lemma is with $\mathcal{U}$ as parameter type and type families $Y :\equiv Z :\equiv \mathrm{id}_{\mathcal{U}}$. Then $Y \to Z$ is $X \to X$ as a type depending on $X : \mathcal{U}$. Now, if $A : \mathcal{U}$ and $e : A = A$ comes by applying the univalence axiom to some equivalence $g : A \to A$, then the above lemma combined with function extensionality yields that for any $f : A \to A$

$$\mathrm{trp}_e^{X\mapsto(X\to X)}(f) = g \circ f \circ g^{-1}.$$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "g"'] & A \arrow[d, "g"] \\
A \arrow[r, "g\circ f\circ g^{-1}", "\mathrm{trp}_{\mathrm{ua}(g)}(f)"'] & A
\end{tikzcd}$$

This equation is phrased as 'transport by conjugation'. The following lemma is proved by induction on $e : x = x'$.

lem:trp-in-fx=Ygx

Lemma 2.14.3. *Let X, Y be types, $f, g : X \to Y$ functions, and let $Z(x) :\equiv (f(x) = g(x))$ for every $x : X$. Then for all x, x' in X, $e : x = x'$, and $i : f(x) = g(x)$ we have:*

$$\mathrm{trp}_e^Z(i) = \mathrm{ap}_g(e) \cdot i \cdot \mathrm{ap}_f(e)^{-1}.$$

xca:trp-in-a/x=b/x

Exercise 2.14.4. Prove the following special cases of Lemma 2.14.3, where $Y \equiv X$ and a, b members of X:

trp-in-a=x

trp-in-x=a

trp-in-x=x

(1) $\mathrm{trp}_e^{x\mapsto a=b}(i) = i$;

(2) $\mathrm{trp}_e^{x\mapsto a=x}(i) = e \cdot i$;

(3) $\mathrm{trp}_e^{x\mapsto x=b}(i) = i \cdot e^{-1}$;

(4) $\mathrm{trp}_e^{x\mapsto x=x}(i) = e \cdot i \cdot e^{-1}$ (also called *conjugation*). ⌟

There is also a dependent version of Lemma 2.14.3, which is again proved by induction on e.[41]

lem:trp-in-fx=Yxgx

Lemma 2.14.5. *Let $X, Y(x)$ be types and $f(x), g(x) : Y(x)$ for all $x : X$. Let $Z(x) :\equiv (f(x) = g(x))$, with the identification in $Y(x)$, for every $x : X$. Then for all x, x' in X, $e : x = x'$, and $i : f(x) = g(x)$ we have:*

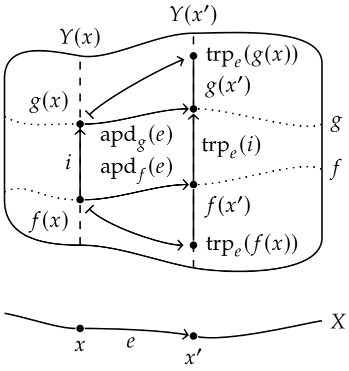
$$\mathrm{trp}_e^Z(i) = \mathrm{po}_e\left(\mathrm{apd}_g(e)\right) \cdot \mathrm{ap}_{\mathrm{trp}_e^Y}(i) \cdot \mathrm{po}_e\left(\mathrm{apd}_f(e)\right)^{-1}.$$

The following construction will be used later in the book.

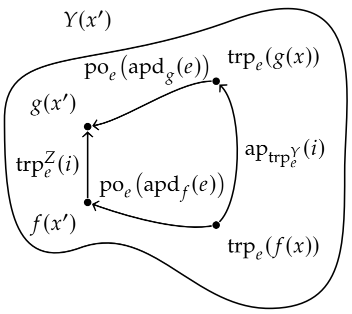
def:Dan's-lemma

Definition 2.14.6. Let $X, Y(x)$ be types and $f(x) : Y(x)$ for all $x : X$. Given elements $x, x' : X$ and a path $p : x = x'$, we define an equivalence $(f(x) =_e^Y f(x')) \simeq (f(x) = f(x))$. We do this by inducion on p, using Definition 2.7.1, thereby reducing to the case $(f(x) = f(x)) \simeq (f(x) = f(x))$, which we solve in the canonical way as before. ⌟

[41] We picture this in two stages. First, we show the fiberwise situation as follows:



Here, there's not room to show all that's going on in the fiber $Y(x')$, so we illustrate that as follows:


$Y(x')$
$\mathrm{po}_e(\mathrm{apd}_g(e))$
$\mathrm{trp}_e(g(x))$
$g(x')$
$\mathrm{trp}_e^Z(i)$
$\mathrm{ap}_{\mathrm{trp}_e^Y}(i)$
$\mathrm{po}_e(\mathrm{apd}_f(e))$
$f(x')$
$\mathrm{trp}_e(f(x))$

2.15 *Propositions, sets and groupoids*

sec:props-sets-grpds

Let P be a type. The property that P has at most one element may be expressed by saying that any two elements are equal. Hence it is encoded by $\prod_{a,b\,:\,P}(a = b)$. We shall call a type P with that property a *proposition*, and its elements will be called *proofs* of P. We will use them for doing logic in type theory. The reason for doing so is that the most relevant thing about a logical proposition is whether it has a proof or not. It is therefore reasonable to require for any type representing a logical proposition that all its members are equal.

Suppose P is a proposition. Then English phrases such as "P holds", "we know P", and "we have shown P", will all mean that we have an element of P. We will not use such phrases for types that are not propositions, nor will we discuss knowing P conditionally with a phrase such as "whether P". Similarly, if "Q" is the English phrase for a statement encoded by the proposition P, then the English phrases "Q holds", "we know Q", and "we have shown Q", will all mean that we have an element of P.

Let X be a type. If for any $x : X$ and any $y : X$ the identity type $x = y$ is a proposition, then we shall say that X is a *set*. The reason for doing so is that the most relevant thing about a set is which elements it has; distinct identifications of equal elements are not relevant.

Alternatively, we shall say that X is a 0-*type*.[42]

Let X be a type. If for any $x : X$ and any $y : X$ the identity type $x = y$ is a set, then we shall say that X is a *groupoid*, also called a 1-*type*.

The pattern continues. If for any $n : \mathbb{N}$, any $x : X$, and any $y : X$ the identity type $x = y$ is an n-*type*, then we shall say that X is an $(n+1)$-*type*. If X is an n-type, we also say that X is n-*truncated*.

We prove that every proposition is a set, from which it follows by induction that every n-type is an $(n+1)$-*type*.

lem:prop-is-set

Lemma 2.15.1. *Every type that is a proposition is also a set.*

Proof. Let X be a type and let $f : \prod_{a,b\,:\,X}(a = b)$. Let $a, b, c : X$ and let $P(x)$ be the type $a = x$ depending on $x : X$. Then $f(a,b) : P(b)$ and $f(a,c) : P(c)$. By path induction we prove for all $q : b = c$ that $q \cdot f(a,b) = f(a,c)$. For this it suffices to verify that $\mathrm{refl}_b \cdot f(a,b) = f(a,b)$, which follows immediately. So q is equal to $f(a,c) \cdot f(a,b)^{-1}$ which doesn't depend on q, so all such q are equal. Hence X is a set. □

A more interesting example of a set is Bool.

lem:isset-bool

Lemma 2.15.2. Bool *is a set.*

Proof. The following elegant, self-contained proof is due to Simon Huber. For proving $p = q$ for all $b, b' : \mathrm{Bool}$ and $p, q : b = b'$, it suffices (by induction on q) to show $p = \mathrm{refl}_b$ for all $b : \mathrm{Bool}$ and $p : b = b$. To this end, define by induction on $b, b' : \mathrm{Bool}$, a type $C(b, b', p)$ for all $p : b = b'$, by setting $C(\mathrm{yes}, \mathrm{yes}, p) :\equiv (p = \mathrm{refl}_{\mathrm{yes}})$, $C(\mathrm{no}, \mathrm{no}, p) :\equiv (p = \mathrm{refl}_{\mathrm{no}})$, and arbitrary in the other two cases. By induction on b one proves that $C(b, b, p) = (p = \mathrm{refl}_b)$ for all p. Hence it suffices to prove $C(b, b', p)$ for all $b, b' : \mathrm{Bool}$ and $p : b = b'$. By induction on p this reduces to $C(b, b, \mathrm{refl}_b)$, which is immediate by induction on $b : \mathrm{Bool}$. □

[42] Sets are thought to consist of points. Points are entities of dimension 0, which explains why the count starts here. One of the contributions of Vladimir Voevodsky is the extension of the hierarchy downwards, with the notion of proposition, including logic in the same hierarchy. Some authors therefore call propositions (-1)-*types*, and they call contractible types (-2)-*types*.

We now collect a number of useful results on propositions.

lem:prop-utils

Lemma 2.15.3. *Let A be a type, and let P and Q propositions. Let $R(a)$ be a proposition depending on $a : A$. Then we have:*

prop-utils-false-true

(1) False *and* True *are propositions;*

prop-utils-codom

(2) *$A \to P$ is a proposition;*

prop-utils-pi

(3) *$\prod_{a:A} R(a)$ is a proposition;*

prop-utils-times

(4) *$P \times Q$ is a proposition;*

prop-utils-sum

(5) *if A is a proposition, then $\sum_{a:A} R(a)$ is a proposition;*

prop-utils-eq

(6) *$P \simeq Q$ is a proposition;*

prop-utils-lem

(7) *$P \amalg (P \to \mathrm{False})$ is a proposition.*

Proof. (1): If $p, q : \mathrm{False}$, then $p = q$ holds vacuously. If $p, q : \mathrm{True}$, then $p = q$ is proved by double induction, which reduces the proof to observing that $\mathrm{refl}_{\mathrm{triv}} : \mathrm{triv} = \mathrm{triv}$.

(2): If $p, q : A \to P$, then $p = q$ is proved by first observing that p and q are functions which, by function extensionality, are equal if they have equal values $p(x) = q(x)$ in P for all x in A. This is actually the case since P is a proposition.

(3): If $p, q : \prod_{a:A} R(a)$ one can use the same argument as for $A \to P$ but now with *dependent* functions p, q.

(4): If $(p_1, q_1), (p_2, q_2) : P \times Q$, then $(p_1, q_1) = (p_2, q_2)$ is proved componentwise. Alternatively, we may regard this case as a special case of (5).

(5): Given $(a_1, r_1), (a_2, r_2) : \sum_a R(a)$, we must establish that $(a_1, r_1) = (a_2, r_2)$. Combining the map in Definition 2.10.1 with the identity in Definition 2.7.2 yields a map $\left(\sum_{u : a_1 = a_2} \mathrm{trp}^Y_u(r_1) = r_2\right) \to ((a_1, r_1) = (a_2, r_2))$, so it suffices to construct an element in the source of the map. Since A is a proposition, we may find $u : a_1 = a_2$. Since $R(a_2)$ is a proposition, we may find $v : \mathrm{trp}^Y_u(r_1) = r_2$. The pair (u, v) is what we wanted to find.

(6): Using Lemma 2.9.9, $P \simeq Q$ is equivalent to $(P \to Q) \times (Q \to P)$, which is a proposition by combining (2) and (4).

(7): If $p, q : P \amalg (P \to \mathrm{False})$, then we can distinguish four cases based on inl/inr, see Section 2.8. In two cases we have both P and $P \to \mathrm{False}$ and we are done. In the other two, either $p \equiv \mathrm{inl}_{p'}$ and $q \equiv \mathrm{inl}_{q'}$ with $p', q' : P$, or $p \equiv \mathrm{inr}_{p'}$ and $q \equiv \mathrm{inr}_{q'}$ with $p', q' : P \to \mathrm{False}$. In both these cases we are done since P and $P \to \mathrm{False}$ are propositions. □

Several remarks can be made here. First, the lemma supports the use of False and True as truth values, and the use of $\to, \prod, \times$ for implication, universal quantification, and conjunction, respectively. Since False is a proposition, it follows by (2) above that $\neg A$ as defined by $A \to \mathrm{False}$ is a proposition for any type A. As noted before, (2) is a special case of (3).

Notably absent in the lemma above are disjunction and existential quantification. This has a simple reason: $\mathrm{True} \amalg \mathrm{True}$ has two distinct elements $\mathrm{inl}_{\mathrm{triv}}$ and $\mathrm{inr}_{\mathrm{triv}}$, an is therefore *not* a proposition. Similarly, $\sum_{n:\mathbb{N}} \mathrm{True}$ has infinitely many distinct elements (n, triv) and is not a proposition. We will explain in Section 2.16 how to work with disjunction and existential quantification for propositions.

The lemma above has a generalization from propositions to n-types which we state without proving. (The proof goes by induction on n, with the lemma above serving as the base case where $n = -1$.)

lem:level-n-utils LEMMA 2.15.4. *Let A be a type, and let X and Y be n-types. Let $Z(a)$ be an n-type depending on $a : A$. Then we have*:

level-n-utils-codom (1) *$A \to X$ is an n-type;*

level-n-utils-pi (2) *$\prod_{a:A} Z(a)$ is an n-type;*

level-n-utils-times (3) *$X \times Y$ is an n-type.*

level-n-utils-sum (4) *if A is an n-type, then $\sum_{a:A} Z(a)$ is an n-type;*

We formalize the definitions from the start of this section.

def:isSet DEFINITION 2.15.5.

$$\begin{aligned} \mathrm{isProp}(P) &:\equiv \prod\nolimits_{p,q:P}(p = q) \\ \mathrm{isSet}(S) &:\equiv \prod\nolimits_{x,y:S} \mathrm{isProp}(x = y) \equiv \prod\nolimits_{x,y:S} \prod\nolimits_{p,q:(x=y)}(p = q) \\ \mathrm{isGrpd}(G) &:\equiv \prod\nolimits_{g,h:G} \mathrm{isSet}(g = h) \equiv \dots \end{aligned} \quad \lrcorner$$

lem:isX-is-prop LEMMA 2.15.6. *For any type A, the following types are propositions*:

(1) isContr(A);

(2) isProp(A);

(3) isSet(A);

(4) isGrpd(A);

(5) *the type that encodes whether A is an n-type, for $n \geq 0$.*

Consistent with that, we will use identifiers starting with "is" only for names of types that are propositions. Examples are isSet(A) and isGrpd(A), and also isEquiv(f).

Proof. Recall that isContr(A) is $\sum_{a:A} \prod_{y:A}(a = y)$. Let (a, f) and (b, g) be elements of the type isContr(A). By Definition 2.10.1, to give an element of $(a, f) = (b, g)$ it suffices to give an $e : a = b$ and an $e' : f =_e^{x \mapsto \prod_{y:A}(x=y)} g$. For e we can take $f(b)$; for e' it suffices by Definition 2.7.2 to give an $e'' : \mathrm{trp}_e f = g$. Clearly, A is a proposition and hence a set by Lemma 2.15.1. Hence the type of g is a proposition by Lemma 2.15.3(3), which gives us e''.

We leave the other cases as exercises. □

xca:isX-is-prop EXERCISE 2.15.7. Make sure you understand that isProp(P) is a proposition, using the same lemmas as for isContr(A). Show that isSet(S) and isGrpd(G) are propositions. ⌟

The following exercise shows that the inductive definition of n-types can indeed start with $n = -2$, where we have the contractible types.

xca:prop-contractible= EXERCISE 2.15.8. Given a type P, show that P is a proposition if and only if $p = q$ is contractible, for any $p, q : P$. ⌟

2.16 *Propositional truncation and logic*

sec:prop-trunc

As explained in Section 2.15, the type formers $\to, \prod, \times$ can be used for the logical operations of implication, universal quantification, and conjunction, respectively. Moreover, True and False can be used as truth values, and $\neg A :\equiv (A \to \mathrm{False})$ can be used for negation. We have also seen that $\amalg$ and Σ can lead to types that are not propositions, even though the constituents are propositions. This means we are still lacking $\vee$ and $\exists$ from the standard repertoire of logic, as well as the notion of *non-emptiness* of a type. In this section we explain how to implement those three notions.

To motivate the construction that follows, consider non-emptiness of a type T. In order to be in a position to encode the mathematical assertion expressed by the English phrase "T is non-empty", we will need a proposition P. The proposition P will have to be constructed somehow from T. Any element of T should somehow give rise to an element of P, but, since all elements of propositions are equal to each other, all elements of P arising from elements of T should somehow be made to equal each other. Finally, any proposition Q that is a consequence of having an element of T should also be a consequence of P.

We define now an operation called propositional truncation,[43] that enforces that all elements of a type become equal.

def:prop-trunc

Definition 2.16.1. Let T be a type. The *propositional truncation* of T is a type $\|T\|$ defined by the following constructors:

(1) an *element* constructor $|t| : \|T\|$ for all $t : T$;

(2) a constructor identifying $x = y$ for all $x, y : \|T\|$.

The (unnamed) identification constructor ensures that $\|T\|$ is a proposition. The induction principle states that, for any family of propositions $P(x)$ defined for each $x : \|T\|$, in order to prove $\prod_{x : \|T\|} P(x)$, it suffices to prove $\prod_{t : T} P(|t|)$. In other words, in order to define a function $f : \prod_{x : \|T\|} P(x)$, it suffices to give a function $g : \prod_{t : T} P(|t|)$. Computationally, we get $f(|t|) \equiv g(t)$ for all $t : T$. ⌟

In the non-dependent case we get that any function $g : T \to P$ yields a (unique) function $f : \|T\| \to P$ satisfying $f(|t|) \equiv g(t)$ for all $t : T$.[44] A useful consequence of this recursion principle is that, for any proposition P, precomposition with $|\_|$ is an equivalence

$$(\|T\| \to P) \quad \to \quad (T \to P).$$

We are now ready to define logical disjunction and existence, by

$$(P \vee Q) :\equiv \|P \amalg Q\| \quad \text{and} \quad (\exists_{x:T} P(x)) :\equiv \|\textstyle\sum_{x:T} P(x)\|.$$

For example, using Definition 2.16.1, from $x : \|T\|$ we may deduce $\exists_{t:T} x = |t|$.

We conclude this section with some important notions defined using truncation.

def:non-empty

Definition 2.16.2. Let A be a type. We call A *non-empty* if we have an element of $\|A\|$.[45] ⌟

[43] The name "truncation" is slightly misleading since it suggests leaving something out, whereas the correct intuition is one of adding identifications so everything becomes equal.

[44] Note that $f(|t|)$ respects equality since P is a proposition.

[45] This notion is also called *inhabited*, instead, in order to avoid confusion with the concept of A *not being empty*, which would be represented by the proposition $\neg(A = \emptyset)$, which is equivalent to $\neg\neg A$.

def:connected

Definition 2.16.3. Let A be a type. If $a : A$, then the subtype $A_{(a)} :\equiv \sum_{x:A} \|a = x\|$ is called the (*connected*) *component* of a in A. We say that elements x, y of A are *in the same component* if we have an element of $\|x = y\|$. The type A is called *connected*[46] if it is non-empty with all elements in the same component, that is, if

$$\mathrm{isConn}(A) :\equiv \|A\| \times \prod_{x,y:A} \|x = y\|. \qquad \lrcorner$$

Note that under this definition the empty type $\emptyset$ is not connected. One can view being connected as a weak form of being contractible – without direct access to a center and to identifications of elements.

xca:component-connected

Exercise 2.16.4. Show that the component of a in A is connected. Show that connected elements have the same *propositional* properties, that is, for any predicate $P : A \to \mathcal{U}$, $P(x)$ is equivalent to $P(y)$ for any connected $x, y : A$. ⌟

xca:prop-set-trivia

Exercise 2.16.5. Show that $\|P\| \simeq P$ for any proposition P. Show that any connected set is contractible. ⌟

xca:connected-trivia

Exercise 2.16.6. Let A be a connected type, and suppose that $a = a$ is a proposition for every $a : A$. Show that A is contractible. ⌟

In the following definition we introduce the adverb *merely*.

def:merely

Definition 2.16.7. If A is an English adjectival phrase encoded by a type T, then *merely* A is the English adjective encoded by the type $\|T\|$. ⌟

For example, a type is non-empty if and only if it *merely has an element*, and two elements of a connected type are *merely equal*.

On the other hand, if we want to emphasize that a phrase A refers to the untruncated type T, then we employ the adverb *purely* and write *purely* A.

2.17 *More on equivalences; surjections and injections*

sec:more-on-equivalences

In this section we collect a number of useful results on equivalences.

Another application of propositional truncation is the notion of image.

def:prop-image

Definition 2.17.1. Let A, B be types and let $f : A \to B$. We define the *image* of f as

$$\mathrm{im}(f) :\equiv \sum_{y:B} \exists_{x:A}(y = f\,x). \qquad \lrcorner$$

Note that $(\exists_{x:A}(y = f\,x)) \equiv \|f^{-1}(y)\|$, the propositional truncation of the fiber. For this reason, $\mathrm{im}(f)$ is called the *propositional* image. Later we will meet other notions of image, based on other truncation operations.

xca:unique-fact-image

Exercise 2.17.2. Show that the image of $f : A \to B$ induces a factorization $f = i \circ p$

$$\begin{tikzcd} A \arrow[rr, "f"] \arrow[dr, "p"'] & & B \\ & \mathrm{im}(f) \arrow[ur, "i"'] & \end{tikzcd}$$

where p is surjective and i is injective, and that each such factorization is equivalent to the image factorization. ⌟

[46] Below in Definition 2.22.4 we will define the *set of connected components* of a type.

xca:all-prop-image

Exercise 2.17.3. Let $f : A \to B$ for A and B types, and let $P(b)$ be a proposition depending on $b : B$. Show that $\prod_{z : \mathrm{im}(f)} P(\mathrm{fst}(z))$ if and only if $\prod_{a : A} P(f(a))$. ⌟

lem:ap-equivalence

Lemma 2.17.4. *If $f : A \to B$ is an equivalence, then so is each $\mathrm{ap}_f : (a = a') \to (f(a) = f(a'))$ for all $a, a' : A$.*[47]

Proof. Let $f^{-1} : B \to A$ be the inverse of f, then we have $\mathrm{ap}_{f^{-1}} : (b = b') \to (f^{-1}(b) = f^{-1}(b'))$ for all $b, b' : B$. Consider $\mathrm{ap}_{f^{-1}}$ for $b :\equiv f(a)$ and $b' :\equiv f(a')$. Then the codomain is $f^{-1}(f(a)) = f^{-1}(f(a'))$, and not $a = a'$. However, we have $h(x) : f^{-1}(f(x)) = x$ for every $x : A$, so the codomain of $\mathrm{ap}_{f^{-1}}$ is equivalent to the domain of ap_f.

We apply Lemma 2.9.9. We take the inverse of ap_f to map $q : f(a) = f(a')$ to $h(a') \cdot \mathrm{ap}_{f^{-1}}(q) \cdot h(a)^{-1} : a = a'$. We then have to prove $h(a') \cdot \mathrm{ap}_{f^{-1}}(\mathrm{ap}_f(p)) \cdot h(a)^{-1} = p$ for all $p : a = a'$, as well as $\mathrm{ap}_f(h(a')) \cdot \mathrm{ap}_{f^{-1}}(q) \cdot h(a)^{-1}) = q$ for all $q : f(a) = f(a')$.

The first round trip is easy by induction on p. The reader may also recognize a naturality square as in Definition 2.6.5 based on h. The second round trip also uses the family of identities $i(y) : f(f^{-1}(y)) = y$ for all $y : B$. This case is more involved and uses several applications of Definition 2.6.5. We refrain from giving all details.[48] □

The converse of Lemma 2.17.4 is not true: $f : \mathbb{1} \to \mathbb{2}$ sending 0 to 0 is not an equivalence. As a function between sets, f is an injection (one-to-one), but not a surjection. We need these important concepts for types in general. We define them as close as possible to their usual meaning in set theory: a function from A to B is surjective if the preimage of any $b : B$ is non-empty, and injective if such preimages contain at most one element.

def:surjection

Definition 2.17.5. A function $f : A \to B$ is a *surjection*, or is *surjective*, if for all $b : B$ there merely exists an $a : A$ such that $b = f(a)$, that is, $\|\sum_{a : A} b = f(a)\|$.[49] ⌟

lem:inj+surj

Lemma 2.17.6. *For all types A, B, a function $f : A \to B$ is an equivalence if and only if f is an injection and a surjection.*

Proof. If $f : A \to B$ is an equivalence, then all fibers are contractible, so f is both an injection and a surjection. Conversely, if f is both injective and surjective, we show that $f^{-1}(b)$ is contractible, for each $b : B$. Being contractible is a proposition, so by Definition 2.16.1 we can drop the truncation in $\|\sum_{a : A} b = f(a)\|$. Now apply injectivity.[50] □

cor:inj+connected

Corollary 2.17.7. *Let A, B be types such that A is non-empty and B is connected. Then any injection $f : A \to B$ is an equivalence.*

Proof. By Lemma 2.17.6 it suffices to show that f is surjective. This is a proposition, so by Definition 2.16.1 and $\|A\|$ we may assume $a : A$, so $f(a) : B$. By $\prod_{x,y : B} \|x = y\|$ we now get that all preimages under f are non-empty. □

lem:whenisbasespaceconnected

Lemma 2.17.8. *Let $f : X \to Y$ be a surjective map from a connected type X. Then Y is connected too.*

[47] TODO: Obsolete

[48] A short proof can be obtained from Lemma 2.17.9.

[49] A function $f : A \to B$ is a *split surjection* if for all $b : B$ there (purely) is an $a : A$ with $b = f(a)$. This is equivalent to saying we have a function $g : B \to A$ such that $f \circ g = \mathrm{id}_B$ (such a g is called a *section* of f).

[50] This argument applies generally: Any non-empty proposition is contractible.

Proof. For any map $f : X \to Y$ between arbitrary types, if $y, y' : Y$ and we are given $x, x' : X$, $p : y = f(x)$, $p' : y' = f(x')$ and $q : x = x'$, then we have an identity between y and y' given by the composite

$$y \overset{p}{=\!=} f(x) \overset{f(q)}{=\!=} f(x') \overset{p'^{-1}}{=\!=} y'.$$

Now the lemma follows by eliminating the propositional truncations in the assumptions, using that the conclusion is a proposition. □

lem:fib-vs-path

Lemma 2.17.9. *Let X and Y be types and let $f : X \to Y$ be a function. Let $x_i : X$, define $y_i :\equiv f(x_i)$ for $i = 0, 1$, and consider a path $q : y_0 = y_1$. Recall the map $\mathrm{ap}_f : (x_0 = x_1) \to (y_0 = y_1)$ and its fiber $(\mathrm{ap}_f)^{-1}(q) :\equiv \sum_{p : x_0 = x_1} q = \mathrm{ap}_f(p)$ over q. Then we have:*

$$((x_0, \mathrm{refl}_{y_0}) =_{f^{-1}(y_0)} (x_1, q)) \simeq (\mathrm{ap}_f)^{-1}(q)$$

Proof. We calculate, starting with Lemma 2.10.3,

$$\begin{aligned}
\big((x_0, \mathrm{refl}_{y_0}) = (x_1, q)\big) &= \sum_{p : x_0 = x_1} \mathrm{refl}_{y_0} \overset{y_0 = f(\_)}{\underset{p}{=\!=}} q \\
&\overset{\text{Definition 2.7.2}}{=} \sum_{p : x_0 = x_1} \mathrm{trp}_p^{y_0 = f(\_)}(\mathrm{refl}_{y_0}) = q \\
&\overset{\text{Lemma 2.14.3}}{=} \sum_{p : x_0 = x_1} \mathrm{ap}_f(p) \cdot \mathrm{refl}_{y_0} \cdot \mathrm{refl}_{y_0}^{-1} = q \\
&= \sum_{p : x_0 = x_1} q = \mathrm{ap}_f(p) \\
&\equiv (\mathrm{ap}_f)^{-1}(q)
\end{aligned}$$

□

[illegible]

Corollary 2.17.10. *Let X and Y be types and let $f : X \to Y$ be a function. Then we have:*

(1) *f is an injection if and only if each map $\mathrm{ap}_f : (x_0 = x_1) \to (f(x_0) = f(x_1))$ induced by f on identity types is an equivalence;*[51]

prop-fib-vs-path

(2) *All fibers of f are sets if and only if all fibers of each map induced by f on identity types are propositions;*

set-fib-vs-path

(3) *If X and Y are connected, then f is an equivalence if and only if each map induced by f on identity types is an equivalence.*

conn-fib-vs-path

Proof. (1) Follows from Lemma 2.17.9: if f is an injection, then $f^{-1}(y_0)$ is a proposition, and hence identities in $f^{-1}(y_0)$ are contractible. For the converse, use that $(x, p) =_{f^{-1}(y)} (x', p')$ is equivalent to $(x, \mathrm{refl}_{f(x)}) =_{f^{-1}(f(x))} (x', p' \cdot p^{-1})$, for all $(x, p), (x', p') : f^{-1}(y)$.

(2) Similar to the proof of (1).

(3) By (1) and Corollary 2.17.7. □

xca:sum-equivalences

Exercise 2.17.11. Let $A, B : \mathcal{U}$, $F : A \to \mathcal{U}$ and $G : B \to \mathcal{U}$, and $f : A \simeq B$ and $g : \prod_{a:A}(F(a) \simeq G(f(a)))$. Give an equivalence from $\sum_{a:A} F(a)$ to $\sum_{b:B} G(b)$. (An important special case is $F \equiv G \circ f$.) ⌟

[51] *Warning*: If X and Y are sets, then each ap_f is an equivalence if and only if we have the implication $(f(x_0) = f(x_1)) \to (x_0 = x_1)$, but this is in general not sufficient.

2.18 *Decidability, excluded middle and propositional resizing*

sec:decidability

Recall from Lemma 2.15.3(7) that $P \amalg (P \to \mathrm{False})$ is a proposition whenever P is a proposition.

def:decidability

Definition 2.18.1. A proposition P is called *decidable* if $P \amalg \neg P$ holds. ⌟

In traditional mathematics, it is usually assumed that every proposition is decidable. This is expressed by the following principle.

pri:lem

Principle 2.18.2 (Law of Excluded Middle). For every proposition P, the proposition $P \amalg (P \to \mathrm{False})$ holds. ⌟

(The "middle" ground excluded by this principle is the possibility that there is a proposition that it neither true nor false.)

Type theory is born in a constructivist tradition which aims at developing as much mathematics as possible without assuming the Law of Excluded Middle.[52] Following this idea, we will explicitly state whenever we are assuming the Law of Excluded Middle.

xca:lem-prop

Exercise 2.18.3. Show that the Law of Excluded Middle is equivalent to asserting that the map $(\mathrm{yes} = \_) : \mathrm{Bool} \to \mathrm{Prop}$ is an equivalence. ⌟

A useful consequence of the Law of Excluded Middle is the so called principle of "proof by contradiction": to prove a proposition P, assume its negation $P \to \mathrm{False}$ and derive a contradiction. Without the Law of Excluded Middle, this proves only the double negation of P, that is $(P \to \mathrm{False}) \to \mathrm{False}$. However, with the Law of Excluded Middle, one can derive P from the latter: indeed, according to the Law of Excluded Middle, either P or $P \to \mathrm{False}$ holds; but $P \to \mathrm{False}$ leads to a contradiction by hypothesis, making P hold necessarily.

xca:dne-lem

Exercise 2.18.4. Show that, conversely, LEM follows from the principle of *double-negation elimination*: For every proposition P, if $(P \to \mathrm{False}) \to \mathrm{False}$, then P holds. ⌟

Remark 2.18.5. We will later encounter a weaker version of the Law of Excluded Middle, called the Limited Principle of Omniscience (Principle 3.6.16), which is often enough.[53] ⌟

Sometimes we make use of the following, which is another consequence of the Law of Excluded Middle:

pri:prop-resizing

Principle 2.18.6 (Propositional Resizing). For any pair of nested universes $\mathcal{U} : \mathcal{U}'$, the inclusion $\mathrm{Prop}_{\mathcal{U}} \to \mathrm{Prop}_{\mathcal{U}'}$ is an equivalence. ⌟

xca:lem-prop-sizing

Exercise 2.18.7. Show that if the Law of Excluded Middle holds for all propositions, then propositional resizing holds. ⌟

2.19 *The replacement principle*

sec:replacement

In this section we fix a universe $\mathcal{U}$. We think of types $A : \mathcal{U}$ as *small* compared to arbitrary types, which are then *large* in comparison.[55] Often we run into types that are not in $\mathcal{U}$ (small) directly, but are nevertheless equivalent to types in $\mathcal{U}$.

def:ess-loc-small

Definition 2.19.1. We say that a type A is *essentially $\mathcal{U}$-small* if there (purely) is a type $X : \mathcal{U}$ and an equivalence $A \simeq X$. And A is *locally $\mathcal{U}$-small* if all its identity types are essentially $\mathcal{U}$-small. ⌟

[52] Besides any philosophical reasons, there are several pragmatic reasons for developing constructive mathematics. One is that proofs in constructive mathematics can be executed as programs, and another is that the results also hold in non-standard models, for instance a model where every type has a topological structure, and all constructions are continuous. See also Footnote 6.

[53] As the naming indicates, we can think of the Law of Excluded Middle itself as an omniscience principle, telling us for every proposition P, whether P is true or false. It was this interpretation of the Law of Excluded Middle that led Brouwer to reject it in his 1908 paper on *De onbetrouwbaarheid der logische principes*.[54]

[54] Mark Van Atten and Göran Sundholm. "L.E.J. Brouwer's 'Unreliability of the Logical Principles': A New Translation, with an Introduction". In: *History and Philosophy of Logic* 38.1 (2017), pp. 24–47. DOI: 10.1080/01445340.2016.1210986. arXiv: 1511.01113.

[55] The terminology *small/large* is also known from set theory, where classes are large collections, and sets are small collections.

Note that $\sum_{X:\mathcal{U}}(A \simeq X)$ is a proposition by the univalence axiom for $\mathcal{U}$. Of course, any $A:\mathcal{U}$ is essentially $\mathcal{U}$-small, and any essentially $\mathcal{U}$-small type is locally $\mathcal{U}$-small.

To show that a type is locally $\mathcal{U}$-small we have to give a reflexive relation $\mathrm{Eq}_A : A \to A \to \mathcal{U}$ that induces, by path induction, a family of equivalences $(x =_A y) \simeq \mathrm{Eq}_A\, x\, y$.

Exercise 2.19.2. Show that $\mathcal{U}$ is locally $\mathcal{U}$-small, and investigate the closure properties of essentially and locally $\mathcal{U}$-small types. (For instance, show that if $A:\mathcal{U}$ and $B(x)$ is a family of locally $\mathcal{U}$-small types parametrized by $x:A$, then $\prod_{x:A} B(x)$ is locally $\mathcal{U}$-small.) ⌟

Remark 2.19.3. Note that propositional resizing (Principle 2.18.6) equivalently says that any proposition is essentially $\mathcal{U}$-small, where we may take $\mathcal{U}$ to be the smallest universe $\mathcal{U}_0$. When we assume this, we get that any set is locally $\mathcal{U}_0$-small. ⌟

We will make use of the following (recall the definition of the image, Definition 2.17.1):

pri:replacement

Principle 2.19.4 (Replacement). For any map $f : A \to B$ from an essentially $\mathcal{U}$-small type A to a locally $\mathcal{U}$-small type B, the image $\mathrm{im}(f)$ is essentially $\mathcal{U}$-small. ⌟

This is reminiscent of the replacement principle of set theory which states that for a large (class-sized) function with domain a small set and codomain the class V of all small sets, the image is again a small set. This follows from our replacement principle, assuming propositional resizing, or the even stronger principle of the excluded middle.

The replacement principle can be proved using the join construction of the image, cf. Rijke[56], which uses as an assumption that the universes are closed under pushouts.[57]

xca:comp-loc-small-ess-small

Exercise 2.19.5. Show that the replacement principle implies that for any locally $\mathcal{U}$-small type A, and any element $a:A$, the connected component $A_{(a)}$ is essentially $\mathcal{U}$-small. ⌟

Another consequence is that the type of finite sets, which we'll define below in Definition 2.24.5, is essentially small.

2.20 *Predicates and subtypes*

sec:subtype

In this section, we consider the relationship between predicates on a type T and subtypes of T. The basic idea is that the predicate tells whether an element of T should belong to the subtype, and the predicate can be recovered from the subtype by asking whether an element of T is in it.

def:predicate

Definition 2.20.1. Let T be a type and let $P(t)$ be a family of types parametrized by an variable $t:T$, such that $P(t)$ is a proposition. Then we call P a *predicate* on T.[58] If $P(t)$ is a decidable proposition, then we say that P is a *decidable predicate* on T. ⌟

By Exercise 2.18.3, the decidable predicates P on T correspond uniquely to the characteristic functions $\chi_P : T \to \mathrm{Bool}$.

We now introduce the notion of *injection*, which will be key to saying what a *subtype* is.

[56] Egbert Rijke. *The join construction*. 2017. arXiv: 1701.07538.

[57] Pushouts are certain higher inductive types that suffice to construct all the higher inductive types that we need, but we don't actually need them in this book.

[58] Note that giving a predicate on T is equivalent to giving a map $Q : T \to \mathrm{Prop}_{\mathcal{U}}$ for a suitable universe $\mathcal{U}$, and we sometimes say that Q itself is the predicate.

def:injection

Definition 2.20.2. A function $f : A \to B$ is an *injection*, or is *injective*, if $f^{-1}(b)$ is a proposition for all $b : B$. The propery of being an injection is encoded by the type $\mathrm{isInj}(f) :\equiv \prod_{b:B} \mathrm{isProp}(\inf f(b))$. ⌟

lem:inj-ap

Lemma 2.20.3. *A function $f : A \to B$ is an injection if and only if each induced function $\mathrm{ap}_f : (a = a') \to (f(a) = f(a'))$ is an equivalence, for all $a, a' : A$.*

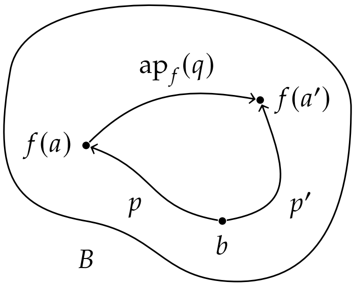
Proof. First we note that if $z, z' : f^{-1}(b)$ for some $b : B$, then there is an equivalence

{eq:inj-ap-equiv}

$$(z = z') \simeq \mathrm{ap}_f^{-1}(\mathrm{snd}\, z' \cdot \mathrm{snd}\, z^{-1}). \tag{2.20.1}$$

Indeed, we can construct this for $z \equiv (a, p)$ and $z' \equiv (a', p')$, where $a, a' : A$, $p : b = f(a)$ and $p' : b = f(a')$, as the composition

$$\begin{aligned}(z = z') &\equiv \big((a, p) = (a', p')\big)\\ &\simeq \sum_{q : a = a'} p \overset{b = f(\_)}{\underset{q}{=\!=}} p'\\ &\simeq \sum_{q : a = a'} \mathrm{ap}_f(q) \cdot p = p'\\ &\simeq \sum_{q : a = a'} p' \cdot p^{-1} = \mathrm{ap}_f(q)\\ &\equiv \mathrm{ap}_f^{-1}(p' \cdot p^{-1}).\end{aligned}$$



The second equivalence relies on Lemma 2.14.3.

It follows directly from (2.20.1) that if ap_f is an equivalence, then $f^{-1}(b)$ is a proposition, as all its identity types are contractible.

On the other hand, if we fix $a, a' : A$ and $p : f(a) = f(a')$, then (2.20.1) applied to $b :\equiv f(a)$, $z :\equiv (a, \mathrm{refl}_{f(a)})$ and $z' :\equiv (a', p)$, gives $\mathrm{ap}_f^{-1}(p) \simeq (z =_{f^{-1}(b)} z')$, which shows that if each $f^{-1}(b)$ is a proposition, then ap_f is an equivalence. □

xca:inj-sets

Exercise 2.20.4. Show that if A, B are sets, then a function $f : A \to B$ is injective if and only if $f(a) = f(a')$ implies $a = a'$ for all a, a'. ⌟

def:subtype

Definition 2.20.5. A *subtype* of a type T is a type S together with an injection $f : S \to T$. Selecting a universe $\mathcal{U}$ as a repository for such types S allows us to introduce the type of subtypes of T in $\mathcal{U}$ as follows.

$$\mathrm{Sub}_T^{\mathcal{U}} :\equiv \sum_{S : \mathcal{U}} \sum_{f : S \to T} \mathrm{isInj}(f).$$

We may choose to leave the choice of $\mathcal{U}$ ambiguous, in which case we will write Sub_T for $\mathrm{Sub}_T^{\mathcal{U}}$. ⌟

lem:subtype-eq-=

Lemma 2.20.6. *Let T be a type and P a predicate on T. Consider $\sum_{t:T} P(t)$ and the corresponding projection map $\mathrm{fst} : T_P :\equiv \left(\sum_{t:T} P(t)\right) \to T$. Then $\mathrm{ap}_{\mathrm{fst}} : ((x_1, p_1) = (x_2, p_2)) \to (x_1 = x_2)$ is an equivalence, for any elements (x_1, p_1) and (x_2, p_2) of T_P.*

Proof. We apply Lemma 2.9.9. Consider $q : x_1 = x_2$. By induction on q we get that each $p_1 =_q^P p_2$ is contractible, say with center c_q. We show that mapping q to $\overline{(q, c_q)}$ defines an inverse of $\mathrm{ap}_{\mathrm{fst}}$, applying Lemma 2.10.3 and the remarks after its proof. These give $\mathrm{ap}_{\mathrm{fst}}\, \overline{(q, c_q)} = q$ for all $q : x_1 = x_2$. Also, for any $r : (x_1, p_1) = (x_2, p_2)$, $r = \overline{(\mathrm{fst}(r), \mathrm{snd}(r))}$. The latter pair is equal to $\overline{(\mathrm{ap}_{\mathrm{fst}(r)}, c)}$ for any c in the contractible type $p_1 =_{\mathrm{fst}(r)}^P p_2$. □

Combined with Lemma 2.20.3, this gives that fst is an injection. Hence, given a predicate P on T, the *subtype* of T characterized by P is defined as $T_P :\equiv \sum_{t:T} P(t)$, together with the injection $\mathrm{fst} : T_P \to T$.

The above lemma has other important consequences.

cor:subtype-same-level

Corollary 2.20.7. *For each natural number n, if T is a n-type, then T_P is also a n-type.*

In particular, if T is a set, then T_P is again a set; we may denote this subset by $\{\, t : T \mid P(t) \,\}$.

rem:subtype-convention

Remark 2.20.8. Another important consequence of Lemma 2.20.6 is that we can afford not to distinguish carefully between elements (t, p) of the subtype T_P and elements t of type T for which the proposition $P(t)$ holds. We will hence often silently coerce from T_P to T via the first projection, and if $t : T$ is such that $P(t)$ holds, we'll write $t : T_P$ to mean any pair (t, p) where $p : P(t)$, since when $P(t)$ holds, the type $P(t)$ is contractible. ⌟

Given a set A and a function $\chi_B : A \to \mathrm{Bool}$, Lemma 2.15.2 yields that $\chi_B(a) = \mathrm{yes}$ is a proposition, and we can form the subset $\{\, a : A \mid \chi_B(a) = \mathrm{yes} \,\}$. However, not every subset as in Definition 2.20.5 can be given through a $\chi_B : A \to \mathrm{Bool}$. As proved in Section 2.12.1, any element of Bool is equal to yes or to no.

If $P : A \to \mathcal{U}$ is a decidable predicate, then we can define $\chi_P : A \to \mathrm{Bool}$ by induction (actually, only case distinction) on $p : P(a)$, setting $\chi_P(a) = \mathrm{yes}$ if $p \equiv \mathrm{inl}\_$ and $\chi_P(a) = \mathrm{no}$ if $p \equiv \mathrm{inr}\_$. We will often use a characteristic function $T \to \mathrm{Bool}$ to specify a decidable predicate on a type T.

xca:decidability

Exercise 2.20.9. Show that $f(t) = \mathrm{yes}$ is a decidable predicate on T, for any type T and function $f : T \to \mathrm{Bool}$. Show $(P \simeq \mathrm{True}) \amalg (P \simeq \mathrm{False})$ for every decidable proposition P. ⌟

We've seen how to make a subtype from a predicate. Conversely, from a subtype of T given by the injection $f : S \to T$, we can form a predicate $P_f : T \to \mathrm{Prop}$ defined by $P_f(x) :\equiv f^{-1}(x)$. We shall see in Lemma 2.25.4, that these operations form an equivalence between predicates on T and subtypes of T.

def:Prop-Set

Definition 2.20.10. The type of types that are propositions and the type of types that are sets are defined as:

$$\mathrm{Prop}_{\mathcal{U}} :\equiv \sum_{X:\mathcal{U}} \mathrm{isProp}(X) \quad \text{and} \quad \mathrm{Set}_{\mathcal{U}} :\equiv \sum_{X:\mathcal{U}} \mathrm{isSet}(X).$$

Both $\mathrm{Prop}_{\mathcal{U}}$ and $\mathrm{Set}_{\mathcal{U}}$ are subtypes of $\mathcal{U}$, and both are types in a universe one higher than $\mathcal{U}$. ⌟

When we don't care about the precise universe $\mathcal{U}$, we'll leave it out from the notation, and just write Prop and Set.

Following Remark 2.20.8, if we have a type A for which we know that it is a proposition or a set, we write also $A : \mathrm{Prop}$ or $A : \mathrm{Set}$, respectively.

def:decidable-set

Definition 2.20.11. A type A is called a *decidable set* if the equality relation $x =_A y$ is a decidable proposition for all $x, y : A$. ⌟

Note the slight subtlety of this definition together with Definition 2.18.1: Any proposition has a decidable equality relation (since each instance is contractible) and is thus a *decidable set*, even though it may not be a *decidable as a proposition*.

The way we phrased this definition, it builds in the condition that A is a set. The following celebrated and useful theorem states that this is unnecessary.

thm:hedberg

THEOREM 2.20.12 (Hedberg). *Any type A for which we have a function of type $\prod_{x,y:A}\big(x = y \amalg \neg(x = y)\big)$ is a decidable set.*

For a proof see Theorem 7.2.5 of the HoTT Book[59].

2.21 *Pointed types*

sec:pointedtypes

Sometimes we need to equip types with additional structure that cannot be expressed by a proposition such as $\mathrm{isProp}(X)$ and $\mathrm{isSet}(X)$ above. Therefore the following is *not* a subtype of $\mathcal{U}$.

def:pointedtypes

DEFINITION 2.21.1. A *pointed type* is a pair (A, a) where A is a type and a is an element of A. The *type of pointed types* is

$$\mathcal{U}_* :\equiv \sum_{A:\mathcal{U}} A.$$

Given a type A we let A_+ be the pointed type you get by adding a default element: $A_+ :\equiv (A \amalg \mathrm{True}, \mathrm{inr}_{\mathrm{triv}})$. Given a pointed type $X \equiv (A, a)$, the *underlying type* is $X_\div :\equiv A$, and the *base point* is $\mathrm{pt}_X :\equiv a$, so that $X \equiv (X_\div, \mathrm{pt}_X)$.

Let $X :\equiv (A, a)$ and $Y :\equiv (B, b)$ be pointed types. Define the map $\mathrm{ev}_a : (A \to B) \to B$ by $\mathrm{ev}_a(f) :\equiv f(a)$. Then the fiber of ev_a at b is the type $\mathrm{ev}_a^{-1} \equiv \sum_{f:A\to B}(b = f(a))$. The latter type is also called the type of *pointed functions* from X to Y and denoted by $X \to_* Y$. In the notation above

$$(X \to_* Y) \equiv \sum_{f:X_\div \to Y_\div} (\mathrm{pt}_Y = f(\mathrm{pt}_X)).$$

If Z is also a pointed type, and we have pointed functions $(f, f_0) : X \to_* Y$ and $(g, g_0) : Y \to_* Z$, then their composition $(g, g_0)(f, f_0) : X \to_* Z$ is defined as the pair $(gf, g(f_0)g_0)$. See the diagram below.

$$\begin{tikzcd}
 & \mathrm{pt}_Y \arrow[r, Rightarrow, "f_0"] \arrow[d, mapsto, "g"] & f(\mathrm{pt}_X) \arrow[d, mapsto, "g"] \\
\mathrm{pt}_Z \arrow[r, Rightarrow, "g_0"] & g(\mathrm{pt}_Y) \arrow[r, Rightarrow, "g(f_0)"] & g(f(\mathrm{pt}_X))
\end{tikzcd}$$

We may also use the notation $(g, g_0) \circ (f, f_0)$ for the composition. ⌟

def:pointedidentity

DEFINITION 2.21.2. If $X \equiv (A, a)$ is a pointed type, then we define the *pointed identity map* $\mathrm{id}_X : X \to_* X$ by setting $\mathrm{id}_X :\equiv (\mathrm{id}_A, \mathrm{refl}_a)$. ⌟

If X is a pointed type, then $X_\div$ is a type, but X itself is *not* a type. It is therefore unambiguous, and quite convenient, to write $x : X$ for $x : X_\div$, and $X \to \mathcal{U}$ for $X_\div \to \mathcal{U}$. We may also tacitly coerce $f : X \to_* Y$ to $f : X_\div \to Y_\div$.

xca:plusforgetadjoint

EXERCISE 2.21.3. If A is a type and B is a pointed type, prove that $A \to B_\div$ is equivalent to $A_+ \to_* B$. ⌟

xca:freemaps

EXERCISE 2.21.4. Let A be a pointed type and B a type. Show that $\sum_{b:B}(A \to_* (B, b))$ and $(A_\div \to B)$ are equivalent. ⌟

[59] Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*.

2.22 *Operations that produce sets*

sec:operations-on-sets
lem:Set-is-groupoid

LEMMA 2.22.1. *If X and Y are sets, then $X = Y$ is a set. In other words,* Set *is a groupoid.*

Proof. By univalence, $(X = Y) \simeq (X \simeq Y) \equiv \sum_{f : X \to Y} \mathrm{isEquiv}(f)$. Since X and Y are sets, so is $X \to Y$ by Lemma 2.15.4. Moreover, $\mathrm{isEquiv}(f)$ is a proposition by Lemma 2.15.6. It follows by Corollary 2.20.7 that $X = Y$ is a set. □

One may wonder whether $\mathbb{N}$ as defined in Section 2.12 is a set. The answer is yes, but it is harder to prove than one would think. In fact we have the following theorem.

thm:isset-inductive-types

THEOREM 2.22.2. *All inductive types in Section 2.12 are sets if all constituent types are sets.*

Proof. We only do the case of $\mathbb{N}$ and leave the other cases to the reader (cf. Exercise 2.22.3). The following proof is due to Simon Huber. We have to prove that $n = m$ is a proposition for all $n, m : \mathbb{N}$, i.e., $p = q$ for all $n, m : \mathbb{N}$ and $p, q : n = m$. By induction on q it suffices to prove $p = \mathrm{refl}_n$ for all $p : n = n$. Note that we can not simply prove this by induction on p. Instead we first prove an inversion principle for identifications in $\mathbb{N}$ as follows. We define a type $T(n, m, p)$ for $n, m : \mathbb{N}$ and $p : n = m$ by induction on n and m:

$$T(0, 0, p) :\equiv (p = \mathrm{refl}_0) \quad \text{and} \quad T(\mathrm{succ}(n), \mathrm{succ}(m), p) :\equiv \sum_{q : n = m} p = \mathrm{ap}_S\, q,$$

and for the other cases the choice does not matter, say $T(0, \mathrm{succ}(m), p) :\equiv T(\mathrm{succ}(n), 0, p) :\equiv \emptyset$. Next we prove $T(n, m, p)$ for all n, m, and p by induction on p, leaving us with $T(n, n, \mathrm{refl}_n)$ for all $n : \mathbb{N}$, which we in turn prove by distinguishing cases on n. Both the case for 0 and for $\mathrm{succ}(n)$ hold by reflexivity, where in the successor case we use refl_n for q and note that $\mathrm{ap}_S\, \mathrm{refl}_n \equiv \mathrm{refl}_{\mathrm{succ}(n)}$.

We can now prove $p = \mathrm{refl}_n$ for all $p : n = n$ by induction on n. In the base case this is simply $T(0, 0, p)$. And for the case $\mathrm{succ}(n)$ we get from $T(\mathrm{succ}(n), \mathrm{succ}(n), p)$ that $p = \mathrm{ap}_S\, q$ for some $q : n = n$. By induction hypothesis we have $e : q = \mathrm{refl}_n$ and thus also

$$p = \mathrm{ap}_S\, q = \mathrm{ap}_S\, \mathrm{refl}_n \equiv \mathrm{refl}_{S(n)}$$

using $\mathrm{ap}_{\mathrm{ap}_S}\, e$, concluding the proof. □

xca:set-sum

EXERCISE 2.22.3. Show that $X \amalg Y$ is a set if X and Y are sets. ⌟

Recall that propositional truncation is turning any type into a proposition by adding identifications of any two elements. Likewise, there is a operation turning any type into a set by adding (higher) identifications of any two identifications of any two elements. The latter operation is called set truncation. It is yet another example of a higher-inductive type.

def:set-truncation

DEFINITION 2.22.4. Let T be a type. The *set truncation* of T is a type $\|T\|_0$ defined by the following constructors:

(1) an *element* $|t|_0 : \|T\|_0$ for all $t : T$;

(2) a *identification* $p = q$ for all $x, y : \|T\|_0$ and $p, q : x = y$.

The (unnamed) second constructor ensures that $\|T\|_0$ is a set. The induction principle states that, for any family of sets $S(x)$ defined for each $x : \|T\|_0$, in order to define a function $f : \prod_{x : \|T\|_0} S(x)$, it suffices to give a function $g : \prod_{t : T} S(|t|_0)$. Computationally, we get $f(|t|_0) \equiv g(t)$ for all $t : T$. ⌟

In the non-dependent case we get that for any set S and any function $g : T \to S$ there is a (unique) function $f : \|T\|_0 \to S$ satisfying $f(|t|_0) \equiv g(t)$ for all $t : T$.[60] A consequence of this recursion principle is that, for any set S, precomposition with $|\_|_0$ is an equivalence

$$(\|T\|_0 \to S) \quad \to \quad (T \to S).$$

xca:sum-of-conn-components

Exercise 2.22.5. Let A be a type. Define for every element $z : \|A\|_0$ the connected component corresponding to z, $A_{(z)}$, a subtype of A, such that for $a : A$, you recover the notion from Definition 2.16.3: $A_{(|a|_0)} \equiv A_{(a)}$.[62]

Prove that the set truncation map $|\_|_0 : A \to \|A\|_0$ in this way exhibits A as the sum of its connected components, parametrized by $\|A\|_0$:

$$A \simeq \sum_{z : \|A\|_0} A_{(z)}. \quad ⌟$$

2.22.6 *Weakly constant maps*

The universal property of the propositional truncation, Definition 2.16.1, only applies directly to construct elements of *propositions* (that is, to prove them). Here we discuss how we can construct elements of *sets*.

Definition 2.22.7. A map $f : A \to B$ is *weakly constant* if $f(x) = f(x')$ for all $x, x' : A$. ⌟

This is in contrast to a *constant* map, which can be identified with one of the form $x \mapsto b$ for some $b : B$. Any constant map is indeed weakly constant. Note also that when B is a set, weak constancy of $f : A \to B$ is a proposition.

thm:wconstant-elim

Theorem 2.22.8. *If $f : A \to B$ is a weakly constant map, and B is a set, then there is an induced map $g : \|A\| \to B$ such that $g(|x|) \equiv f(x)$ for all $x : A$.*

Proof. Consider the image factorization $A \xrightarrow{p} \mathrm{im}(f) \xrightarrow{i} B$ of f, where $p(x) :\equiv (f(x), |(x, \mathrm{refl}_{f(x)})|)$ and $i(y, !) :\equiv y$.

The key point is that $\mathrm{im}(f)$ is a proposition: Let $(y_1, z_1), (y_2, z_2) : \mathrm{im}(f)$. Since B is a set, the type $y_1 =_B y_2$ is a proposition. Hence we may hypothesize (by induction on z_i) that we have $x_1, x_2 : A$ with $f(x_i) = y_i$ for $i = 1, 2$. By concatenation, we get $y_1 = f(x_1) = f(x_2) = y_2$ and hence $(y_1, z_1) = (y_2, z_2)$.

Thus, by the universal property of the truncation, we get $g' : \|A\| \to \mathrm{im}(f)$ such that $g'(|x|) \equiv p(x) \equiv (f(x), |(x, \mathrm{refl}_{f(x)})|)$. Composing with i we get $g :\equiv i \circ g' : \|A\| \to B$ with $g(|x|) :\equiv f(x)$, as desired. □

2.22.9 *Set quotients*

def:quotient-set

Definition 2.22.10. Given a set A and an equivalence relation[63] $R : A \to A \to \mathrm{Prop}$, we define the *quotient set*[64] A/R as the image of the map

[60] Lemma 7.3.12[61]gives an equivalence from $|t|_0 = |t'|_0$ to $\|t = t'\|$ for all $t, t : T$.

[61] Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*.

[62] *Hint*: Use a map $\|a = \_\| : A \to \mathrm{Prop}$ and the fact that the universe of propositions is a set.

[63] Recall that an *equivalence relation* is one that is (1) *reflexive*: $R(x, x)$, (2) *symmetric*: $R(x, y) \to R(y, x)$, and (3) *transitive*: $R(x, y) \to R(y, z) \to R(x, z)$.

[64] We may wonder about the universe level of A/R, assuming $A : \mathcal{U}$ and $R : A \to A \to \mathrm{Prop}_{\mathcal{U}}$. By the Replacement Principle 2.19.4, A/R is essentially $\mathcal{U}$-small, since $A \to \mathrm{Prop}_{\mathcal{U}}$ is locally $\mathcal{U}$-small. Alternatively, we could use Propositional Resizing Principle 2.18.6 to push the values of R into a lower universe.

$R : A \to (A \to \mathrm{Prop})$. For $a : A$ we define $[a] :\equiv (R(a), |(a, \mathrm{refl}_{R(a)})|)$ in A/R; $[a]$ is called the *equivalence class containing a*.[65] ⌟

Any element of the image of R is an equivalence class: a subset P of A for which there merely exists $a : A$ such that $P(x)$ holds if and only if $R(a, x)$ holds.

In the following proofs we frequently use Exercise 2.17.3.

lem:equiv-class-prop

Lemma 2.22.11. *For any equivalence class $P : A/R$ and $a : A$, $P = [a]$ if and only if $P(a)$ holds.*

Proof. Assume $P = [a]$. Then $P(x)$ is equivalent to $R(a, x)$ for all $x : A$. Take $x :\equiv a$ and use reflexivity $R(a, a)$ to conclude $P(a)$.

Conversely, assume $P(a)$, and let $x : A$ be given. To prove the proposition $P(x) \simeq R(a, x)$ we may assume that $P \equiv [b]$ for some $b : A$. Then $P(x) \equiv R(b, x)$, and we need to show $R(b, x) \simeq R(a, x)$. This follows from $P(a) \equiv R(b, a)$ using symmetry and transitivity. □

thm:quotient-property

Theorem 2.22.12. *We have $([x] = [x']) \simeq R(x, x')$ for all $x, x' : A$. Also, for any set B, a function $f : A \to B$ factors uniquely through the map $[\_] : A \to A/R$ if $f(x) = f(x')$ for all $x, x' : A$ with $R(x, x')$. Indeed we get a map $\bar{f} : A/R \to B$ with $\bar{f}([x]) \equiv f(x)$ for all $x : A$.*

Proof. For the first part we use Lemma 2.22.11 applied to $P_x :\equiv [x]$ and x'.

Now let B be a set and let $f : A \to B$ a function satisfying $f(x) = f(x')$ for all $x, x' : A$ with $R(x, x')$.

Uniqueness: If g, h are extensions of f through $[\_]$, then for any $z : A/R$, the type $g(z) = h(z)$ is a proposition since B is a set, so we may assume $z \equiv [x]$ for some $x : A$. Then $g([x]) = f(x) = h([x])$, as desired.

Existence: Let $z \equiv (P, !) : A/R$. To define the image of z in B, using the truth of the proposition $\exists_{x:A}(P = [x])$, it suffices by Theorem 2.22.8 to give a weakly constant map $\sum_{x:A}(P = [x]) \to B$, and $f \circ \mathrm{fst}$ does the trick.

Now we check the definitional equality: As an element of A/R, equivalence class $[x]$ is accompanied by the witness $|(x, \mathrm{refl}_{[x]})| : \exists_{y:A}([x] = [y])$. By Theorem 2.22.8, this is mapped, by definition, to $(f \circ \mathrm{fst})(x, \mathrm{refl}_{[x]}) \equiv f(x)$, as desired. □

We can use set quotients to give an alternative definition of the set truncation $\|A\|_0$ of a type A. Consider the relation $R : A \to A \to \mathrm{Prop}$ given by $R(x, y) :\equiv \|x = y\|$. This is easily seen to be an equivalence relation, using the groupoid structure of identity types. Hence we get a quotient set A/R that satisfies $(|x|_0 = |y|_0) \simeq \|x = y\|$, where we write $|\_|_0$ for the equivalence classes. Furthermore, Theorem 2.22.12 implies that A/R satisfies the recursion principle of Definition 2.22.4: If S is a set, and $g : A \to S$ is any function, then $g(x) = g(y)$ holds whenever $\|x = y\|$ by the elimination principle of the propositional truncation, and hence we get a function $f : A/R \to S$ satisfying $f(|x|_0) \equiv g(x)$ for all $x : A$, as desired.[66]

[65] We recall the convention to use $[a] \equiv (R(a), !)$ also to denote its first component, that is, to use $[a]$ and $R(a)$ interchangeably. The way in which $[a]$ contains a is by observing $R(a) : A \to \mathrm{Prop}$ and $(a, !) : \sum_{x:A} R(a, x)$, by $R(a, a)$.

[66] Expanding the definitions, this means that we can take the 0-truncation $\|A\|_0$ of $A : \mathcal{U}$ to be the $\mathcal{U}$-small image of the (-1)-truncated identity relation $A \to (A \to \mathrm{Prop}_{\mathcal{U}})$. Similarly, we can recursively construct the $(n+1)$-truncation by taking the $\mathcal{U}$-small image of the n-truncated identity relation $A \to (A \to \sum_{X:\mathcal{U}} \mathrm{is}n\mathrm{Type})$.

2.23 *More on natural numbers*

sec:more-on-ℕ

A useful function $\mathbb{N} \to \mathbb{N}$ is the *predecessor* pred defined by $\mathrm{pred}(0) :\equiv 0$ and $\mathrm{pred}(\mathrm{succ}(n)) :\equiv n$. Elementary properties of addition, multiplication and predecessor can be proved in type theory in the usual way. We freely use them, sometimes even in definitions, leaving most of the proofs/constructions to the reader.

def:orderonℕ

Definition 2.23.1. Let $n, m : \mathbb{N}$. We say that m is less than or equal to n, and write $m \leq n$, if there is a $k : \mathbb{N}$ such that $k + m = n$. Such a k is unique, and if it is not 0, we say that m is less than n, denoted by $m < n$. Both $m \leq n$ and $m < n$ are propositions for all $n, m : \mathbb{N}$. ⌟

xca:try-your-luck-ℕ

Exercise 2.23.2. Try your luck in type theory proving any of the following. The successor function satisfies $(\mathrm{succ}(n) = \mathrm{succ}(m)) \simeq (n = m)$. The functions $+$ and $\cdot$ are commutative and associative, $\cdot$ distributes over $+$. The relations $\leq$ and $<$ are transitive and preserved under $+$; $\leq$ also under $\cdot$. We have $(m \leq n) \simeq ((m < n) \amalg (m = n))$ (so $\leq$ is reflexive). Furthermore, $((m \leq n) \times (n \leq m)) \simeq (m = n)$, and $((m < n) \times (n < m)) \to \mathrm{False}$ (so $<$ is irreflexive). ⌟

We can prove the following lemma by double induction.

lem:dec-eq-order-ℕ

Lemma 2.23.3. *The relations $=$, $\leq$ and $<$ on $\mathbb{N}$ are decidable.*

By Hedberg's Theorem 2.20.12, we get an alternate proof that $\mathbb{N}$ is a set.

We will now prove an important property of $\mathbb{N}$, called the *least number principle for decidable, non-empty subsets of* $\mathbb{N}$. We give some more details of the proof, since they illustrate an aspect of type theory that has not been very prominent up to know, namely the close connection between proving and computing.

def:ℕwellordered

Construction 2.23.4. *Let $P(n)$ be a proposition for all natural numbers n. Define the type $P_{\min}(n)$ expressing that n is the smallest natural number such that $P(n)$:*

$$P_{\min}(n) :\equiv P(n) \times \prod_{m : \mathbb{N}} (P(m) \to n \leq m)$$

Then we seek a function

{eqn:min}

$$\min(P) : \prod_{n : \mathbb{N}} (P(n) \amalg \neg P(n)) \to \exists_{n : \mathbb{N}} P(n) \to \sum_{n : \mathbb{N}} P_{\min}(n), \tag{2.23.1}$$

computing a minimal witness for P from evidence that P is decidable and that a witness merely exists.

Implementation of Construction 2.23.4. First note that $P_{\min}(n)$ is a proposition, and that all n such that $P_{\min}(n)$ are equal. Therefore the type $\sum_{n : \mathbb{N}} P_{\min}(n)$ is also a proposition.

Given a function $d(n) : P(n) \amalg \neg P(n)$ deciding $P(n)$ for each $n : \mathbb{N}$, we define a function $\mu_P : \mathbb{N} \to \mathbb{N}$ which, given input n, searches for a $k < n$ such that $P(k)$. If such a k exists, μ_P returns the least such k, otherwise $\mu_P(n) = n$. This is a standard procedure that we will call *bounded search*. The function μ_P is defined by induction, setting $\mu_P(0) :\equiv 0$ and $\mu_P(\mathrm{succ}(n)) :\equiv \mu_P(n)$ if $\mu_P(n) < n$. Otherwise, we set $\mu_P(\mathrm{succ}(n)) :\equiv n$ if $P(n)$, and $\mu_P(\mathrm{succ}(n)) :\equiv \mathrm{succ}(n)$ otherwise, using $d(n)$ to decide, that is, by induction on $d(n) : P(n) \amalg \neg P(n)$. By design, μ_P 'remembers' where it has found the least k (if so). We are now done with the computational part and the rest is a correctness proof.

By induction on $n : \mathbb{N}$ and $d(n) : P(n) \amalg \neg P(n)$ we show

$$\mu_P(n) \leq n \quad \text{and} \quad \mu_P(n) < n \to P(\mu_P(n)).$$

The base case $n = 0$ is easy. For the induction step, review the computation of $\mu_P(\mathrm{succ}(n))$. If $\mu_P(\mathrm{succ}(n)) = \mu_P(n)$ since $\mu_P(n) < n$, then we are done by the induction hypothesis. Otherwise, either $\mu_P(\mathrm{succ}(n)) = n$ and $P(n)$, or $\mu_P(\mathrm{succ}(n)) = \mathrm{succ}(n)$. In both cases we are done.

Also by induction on $n : \mathbb{N}$ and $d(n) : P(n) \amalg \neg P(n)$ we show

$$P(m) \to \mu_P(n) \leq m, \text{ for all } m \text{ in } \mathbb{N}.$$

The base case $n = 0$ holds since $\mu_P(0) = 0$. For the induction step, assume $P(m) \to \mu_P(n) \leq m$ for all m (IH). Let $m : \mathbb{N}$ and assume $P(m)$. We have to prove $\mu_P(\mathrm{succ}(n)) \leq m$. If $\mu_P(\mathrm{succ}(n)) = \mu_P(n)$ we are done by IH. Otherwise we have $\mu_P(n) = n$ and $\mu_P(\mathrm{succ}(n)) = \mathrm{succ}(n)$ and $\neg P(n)$. Then $\mu_P(n) \leq m$ by IH, and $n \neq m$, so $\mu_P(\mathrm{succ}(n)) \leq m$.

By contraposition we get from the previous result

$$\mu_P(n) = n \to \neg P(m), \text{ for all } m < n.$$

Note that there may not be any n such that $P(n)$; the best we can do is to prove

$$P(n) \to P_{\min}(\mu_P(\mathrm{succ}(n)))$$

by combining previous results. Assume $P(n)$. Then $\mu_P(\mathrm{succ}(n)) \leq n < \mathrm{succ}(n)$, so that $P(\mu_P(\mathrm{succ}(n)))$. Moreover, $P(m) \to \mu_P(\mathrm{succ}(n)) \leq m$ for all m in $\mathbb{N}$. Hence $P_{\min}(\mu_P(\mathrm{succ}(n)))$.

Since $\sum_{n:\mathbb{N}} P_{\min}(n)$ is a proposition, we obtain the required function by the induction principle for propositional truncation, Definition 2.16.1:

$$\min(P) : \prod_{n:\mathbb{N}} (P(n) \amalg \neg P(n)) \to \Big\| \sum_{n:\mathbb{N}} P(n) \Big\| \to \sum_{n:\mathbb{N}} P_{\min}(n). \qquad \square$$

rem:computations-can-decide

Remark 2.23.5. In the interest of readability, we do not always make the use of witnesses of decidability in computations explicit. A typical example is the case distinction on $\mu_P(n) < n$ in Construction 2.23.4 above. This remark applies to all sets and decidable relations on them. We shall immediately put this convention to good use in the proof of a form of the so-called *Pigeonhole Principle* (PHP). ⌟

lem:PHP

Lemma 2.23.6. *For all $N : \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$ such that $f(n) < N$ for all $n < N + 1$, there exist $m < n < N + 1$ such that $f(n) = f(m)$.*

Proof. By induction on N. In the base case $N = 0$ there is nothing to do. For the induction case $N + 1$, assume the lemma proved for N (induction hypothesis, IH, for all f). Let f be such that $f(n) < N + 1$ for all $n < N + 2$. The idea of the proof is to search for an $n < N + 1$ such that $P(n) :\equiv (f(n) = N)$, by computing $\mu_P(N + 1)$ as in Construction 2.23.4. If $\mu_P(N + 1) = N + 1$, that is, $f(n) < N$ for all $n < N + 1$, then we are done by IH. Assume $\mu_P(N + 1) < N + 1$, so $f(\mu_P(N + 1)) = N$. If also $f(N + 1) = N$ then we are done. If $f(N + 1) < N$, then we define g by $g(n) = f(N + 1)$ if $f(n) = N$, and $g(n) = f(n)$ otherwise. Then IH applies to g, and we get $m < n < N + 1$ with $g(n) = g(m)$. If $f(n) = f(m)$ we are of course done. Otherwise, $f(n), f(m)$ cannot both be smaller than N, as $g(n) = g(m)$. In both remaining cases, $f(n) = g(n) = g(m) = f(N+1)$ and $f(N+1) = g(n) = g(m) = f(m)$, we are done. □

We can now rule out the existence of equivalences between finite sets of different size.

cor:Fin-n-injective

Corollary 2.23.7. *If $m < n$, then $(\sum_{k:\mathbb{N}} k < m) \neq (\sum_{k:\mathbb{N}} k < n)$.*

Another application of Construction 2.23.4 is a short proof of Euclidean division.

lem:euclid-div

Lemma 2.23.8. *For all $n, m : \mathbb{N}$ with $m > 0$ there exist unique $q, r : \mathbb{N}$ such that $r < m$ and $n = qm + r$.*

Proof. Define $P(k) :\equiv (n \leq km)$. Since $m > 0$ we have $P(n)$. Now set $k :\equiv \mu_P(n)$ as in Construction 2.23.4. If $n = km$ and we set $q :\equiv k$ and $r :\equiv 0$. If $n < km$, then $k > 0$ and we set $q :\equiv k - 1$. By minimality we have $qm < n < km$ and hence $n = qm + r$ for some $r < m$. □

2.24 *The type of finite types*

sec:typeFin

Recall from Section 2.12.1 the types False, True and Bool containing zero, one and two elements, respectively. We now define generally the type of n elements for any $n : \mathbb{N}$.

def:finiteset

Definition 2.24.1. For any type X define $\mathrm{succ}(X) :\equiv X \amalg \mathrm{True}$. Define inductively the type family $F(n)$, for each $n : \mathbb{N}$, by setting $F(0) :\equiv \emptyset$ and $F(\mathrm{succ}(n)) :\equiv \mathrm{succ}(F(n))$. Now abbreviate $\mathbb{n} :\equiv F(n)$. The type $\mathbb{n}$ is called the type with n elements, and we denote its elements by $0, 1, \ldots, n-1$ rather than by the corresponding expressions using inl and inr. ⌟

xca:finite-types

Exercise 2.24.2.

(1) Denote in full all elements of $\mathbb{0}$, $\mathbb{1}$, and $\mathbb{2}$.

(2) Show (using univalence) that $\mathbb{1} = \mathrm{True}$, $\mathbb{2} = \mathrm{Bool}$.

(3) Show (using univalence) that $\mathbb{n} = \sum_{k:\mathbb{N}} k < n$ for all $n : \mathbb{N}$.

(4) Show that $m = n$ if $\mathbb{m} = \mathbb{n}$. ⌟

def:is-finite

Definition 2.24.3. Given a type X, we define the proposition

$$\mathrm{isFinSet}(X) :\equiv \|\textstyle\sum_{n:\mathbb{N}} X = \mathbb{n}\| \equiv \exists_{n:\mathbb{N}}(X = \mathbb{n})$$

to express that X *is a finite set*.[67] ⌟

lem:maxonefinitetype

Lemma 2.24.4.

(1) *$\sum_{n:\mathbb{N}} \|X = \mathbb{n}\|$ is a proposition, for all types X.*

(2) $\sum_{X:\mathcal{U}} \sum_{n:\mathbb{N}} \|X = \mathbb{n}\| = \sum_{X:\mathcal{U}} \mathrm{isFinSet}(X)$.

Proof. (1) Assume $(n,p), (m,q) : \sum_{n:\mathbb{N}} \|X = \mathbb{n}\|$. Then we have $\|\mathbb{n} = \mathbb{m}\|$, so $\|n = m\|$ by Exercise 2.24.2. But $\mathbb{N}$ is a set by Theorem 2.22.2, so $\|n = m\| = (n = m)$. It follows that $(n,p) = (m,q)$.

(2) Follows from $\sum_{n:\mathbb{N}} \|X = \mathbb{n}\| = \|\sum_{n:\mathbb{N}} X = \mathbb{n}\|$, which is easily proved by giving functions in both directions and using the univalence axiom. □

[67] When moving beyond sets, there are two different ways in which a type can be finite: an *additive* way and a *multiplicative* way, but it would take us too far afield to define these notions here.

[68] Here it doesn't matter whether we say Set or $\mathcal{U}$, since any finite set is a set. Hence we also have $\mathrm{FinSet}_n \equiv \mathrm{Set}_{(\mathbb{n})} = \mathrm{FinSet}_{(\mathbb{n})} = \mathcal{U}_{(\mathbb{n})}$.

The above lemma remains true if X ranges over Set. If a set S is in the same component in Set[68] as $\mathbb{n}$ we say that S *has cardinality* n or that *the cardinality of* S *is* n.

def:groupoidFin

Definition 2.24.5. The *groupoid of finite sets* is defined by

$$\mathrm{FinSet} :\equiv \sum_{S\,:\,\mathrm{Set}} \mathrm{isFinSet}(S).$$

For $n : \mathbb{N}$, the *groupoid of sets of cardinality* n is defined by

$$\mathrm{FinSet}_n :\equiv \sum_{S\,:\,\mathrm{Set}} \|S = \mathbb{n}\|. \quad \lrcorner$$

Observe that $\mathrm{FinSet}_0 = \mathrm{FinSet}_1 = \mathbb{1}$ and $\mathrm{FinSet} = \sum_{n:\mathbb{N}} \mathrm{FinSet}_n$ by Lemma 2.24.4.

Note that being a finite set implies being a set, and hence $\mathrm{FinSet} = \sum_{X:\mathcal{U}} \mathrm{isFinSet}(X)$. Also, FinSet is the image of the map $F : \mathbb{N} \to \mathcal{U}$ from Definition 2.24.1, and is hence essentially $\mathcal{U}$-small (for any universe $\mathcal{U}$), by Principle 2.19.4.

2.25 *Type families and maps*

sec:typefam

There is a natural equivalence between maps into a type A and type families parametrized by A. The key idea is that the fibers of a map form a type family. We will elaborate this idea and some variations.

lem:fst-fiber(a)=B(a)

Lemma 2.25.1. *Let* $A : \mathcal{U}$ *and* $B : A \to \mathcal{U}$. *Recall the function* $\mathrm{fst} : (\sum_{a:A} B(a)) \to A$. *Then* $e_a : B(a) \to \mathrm{fst}^{-1}(a)$ *defined by* $e_a(b) :\equiv ((a,b), \mathrm{refl}_a)$ *is an equivalence, for all* $a : A$.

Proof. Note that $\mathrm{fst}(x,b) \equiv x$ and that $a = x$ does not depend on b. Hence $\mathrm{fst}^{-1}(a) \simeq \sum_{x:A}(B(x) \times (a = x))$ via rearranging brackets. Applying Corollary 2.9.11 leads indeed to the equivalence e_a. □

lem:sum-of-fibers

Lemma 2.25.2. *Let* $A, B : \mathcal{U}$ *and* $f : B \to A$. *Then* $e : B \to \sum_{a:A} f^{-1}(a)$ *defined by* $e(b) :\equiv (f(b), b, \mathrm{refl}_{f(b)})$ *is an equivalence.*

Proof. Define $e^{-1} : \sum_{a:A} f^{-1}(a) \to B$ by $e(a,b,p) :\equiv b$. Then $e^{-1}(e(b)) \equiv b$ for all $b : B$. Let $a : A$, $b : B$ and $p : f(b) = a$. Then $e(e^{-1}(a,b,p)) \equiv (f(b), b, \mathrm{refl}_{f(b)})$. We have to prove $(f(b), b, \mathrm{refl}_{f(b)}) = (a,b,p)$. We use p as identification of the first components, and refl_b as identification of the second components (whose type is constant). For the third component we use that the transport of $\mathrm{refl}_{f(b)}$ along p in the type family $(f(b) = \_)$ is indeed equal to p itself by Exercise 2.14.4(2). Now apply Lemma 2.9.9. □

If f above is an injection, then $\sum_{a:A} f^{-1}(a)$ is a subtype of A, and B is a n-type if A is a n-type by Corollary 2.20.7.

lem:typefamiliesandfibrations

Lemma 2.25.3. *Let* A *be a type. Then*

$$\mathrm{preim} \;:\; \sum_{B:\mathcal{U}} (B \to A) \quad\to\quad (A \to \mathcal{U})$$

given by $\mathrm{preim}(B,f)(a) :\equiv f^{-1}(a)$ *is an equivalence. An inverse equivalence is given by sending* $P : A \to \mathcal{U}$ *to* $(\sum_{a:A} P(a), \mathrm{fst})$.

Proof. We apply Lemma 2.9.9, and verify the two conditions. Let $P : A \to \mathcal{U}$. We have to prove that $P = \mathrm{preim}(\sum_{a:A} P(a), \mathrm{fst})$. By function extensionality it suffices to prove $\mathrm{preim}(\sum_{a:A} P(a), \mathrm{fst})(a) \equiv \mathrm{fst}^{-1}(a) = P(a)$. This follows directly from Lemma 2.25.1 and the univalence axiom.

Let $f : B \to A$. We have to prove that $(\sum_{a:A} f^{-1}(a), \mathrm{fst}) = (B, f)$. Using the univalence axiom, we get an identification $\bar{e} : \sum_{a:A} f^{-1}(a) = B$, where e is the equivalence from Lemma 2.25.2. Using Lemma 2.10.3, it remains to give an element of the type $\mathrm{fst} =^{(\_)\to A}_{\bar{e}} f$.

As an auxiliary step we note that for any $p : X = Y$ and $g : X \to A$, $h : Y \to A$, the type $g =^{(\_)\to A}_{p} h$ of paths over p is equal to the type $g = h \circ \tilde{p}$, since the two types are definitionally equal for $p \equiv \mathrm{refl}_X$. Applying this here means that we must give an element of $\mathrm{fst} = f \circ \tilde{\bar{e}}$. This in turn means that we must give an element of $\mathrm{fst} = f \circ e$, which follows by function extensionality from the definition of e in Lemma 2.25.2. □

Let A be a type and consider the subuniverse $\mathrm{Prop} \equiv \sum_{X:\mathcal{U}} \mathrm{isProp}(X)$ from Section 2.20. A function $P : A \to \mathrm{Prop}$ can be viewed as a family of propositions: $\mathrm{fst} \circ P : A \to \mathcal{U}$ is a type family, and $\mathrm{snd} \circ P : \prod_{a:A} \mathrm{isProp}(P(a))$ witnesses that each $\mathrm{fst}(P(a))$ is a proposition. The inverse equivalence in Lemma 2.25.3 sends $\mathrm{fst} \circ P$ to

$$\mathrm{fst} : \Big(\sum_{a:A} \mathrm{fst}(P(a))\Big) \to A.$$

All the fibers of this function are propositions by combining $\mathrm{snd} \circ P : \prod_{a:A} \mathrm{isProp}(P(a))$ with Lemma 2.25.1.

Conversely, for a function $f : B \to A$ with proof $g : \prod_{a:A} \mathrm{isProp}(f^{-1}(a))$ that all fibers of f are propositions, we can define $P_f : A \to \mathrm{Prop}$ by setting $P_f(a) :\equiv (f^{-1}(a), g(a))$.

The above argument can be refined for each of $\mathrm{Prop}, \mathrm{Set}, \mathcal{U}_*$ from Section 2.20, and one can prove the following analogues of Lemma 2.25.3.

lem:Prop-Set-pointed-families

lem:Set-families

Lemma 2.25.4. *Let A be a type. Then we have:*

(1) $(A \to \mathrm{Prop}) \simeq \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \mathrm{isProp}(f^{-1}(a))$;

(2) $(A \to \mathrm{Set}) \simeq \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \mathrm{isSet}(f^{-1}(a))$;

(3) $(A \to \mathcal{U}_*) \simeq \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} f^{-1}(a)$. *(Hard!)*

Since Prop is a set, we obtain the following corollary.

cor:Sub_T-is-set

Corollary 2.25.5. *Subtypes as in Definition 2.20.5 correspond to predicates and* Sub_T *is a set, for any type* T.

sec:stuff-struct-prop

2.26 *Higher structure: stuff, structure, and properties*

Recall from Lemma 2.25.2 that any map $f : B \to A$ can be described as "projecting away" its fibers, by using the equivalence e:

{eq:forget-fibers}

(2.26.1)
$$\begin{tikzcd} B \arrow[rr, "e", "\sim"'] \arrow[dr, "f"'] & & \sum_{a:A} f^{-1}(a) \arrow[dl, "\mathrm{fst}"] \\ & A & \end{tikzcd}$$

We say that f *forgets* these fibers. If A and B are groupoids, these fibers are themselves groupoids, but it can happen that they are sets, propositions, or even contractible. Accordingly, we say that:

The precise formalization of the intuitive notions of "stuff", "structure", and "properties" was worked out in terms of category theory in *UseNet* discussions between John Baez, Toby Bartels, and James Dolan on `sci.physics.research` in 1998. It was clear that the simplest description was in terms of homotopy types, and hence it's even simpler in type theory. See also Baez and Shulman[69] for further discussion.

[69] John C. Baez and Michael Shulman. "Lectures on n-categories and cohomology". In: *Towards higher categories.* Vol. 152. IMA Vol. Math. Appl. Springer, New York, 2010, pp. 1–68. DOI: `10.1007/978-1-4419-1524-5_1`. arXiv: `math/0608420`.

- f *forgets at most structure* if all the fibers are sets;
- f *forgets at most properties* if all the fibers are propositions;
- f *forgets nothing* if all the fibers are contractible.

Here, the structure and properties in question are *on* a or *of* a, respectively, as captured by the fibers at a, for each $a : A$. Of course, a map forgets properties if and only if it's an injection, and it forgets nothing if and only if it's an equivalence.

Going in the other direction, we say that:

- f *forgets at most* n*-structure* if all the fibers are n-truncated. If $n \geq 1$, this is therefore a kind of *higher structure*.[70]

Thus, an element of a groupoid is 1-structure (this is sometimes informally called *stuff*), while an element of a set is a structure, or 0-structure, while an proof of a proposition is a property, or (-1)-structure.

Looking at (2.26.1) another way, we see that to give an element of b of B lying over a given element $a : A$ amounts to specifying an element on $f^{-1}(a)$, so we say that the elements of B are elements of A *with extra* n*-structure*, if the fibers $f^{-1}(a)$ are n-truncated.

Refining the usual image and image factorization from Definition 2.17.1 and Exercise 2.17.2 we can factor $f : B \to A$ through first its 0-*image* and then its usual (-1)-image as follows:[71]

$$B = \sum_{a:A} f^{-1}(a) \to \sum_{a:A} \|f^{-1}(a)\|_0 \to \sum_{a:A} \|f^{-1}(a)\|_{-1} \to \sum_{a:A} \|f^{-1}(a)\|_{-2} = A.$$

Here, the first map *forgets pure higher structure*, the second map *forgets pure structure*, while the last forgets at most properties (this is the inclusion of the usual image). Of course, each of these maps may happen to forget nothing at all. Saying that the second map forgets *pure* structure indicates that not only are the fibers sets, they are *nonempty* sets, so the structure in question merely exists, at least. Note also that the fibers of the first map are connected, which indicates that what is forgotten at this step, if anything, is pure higher structure.

exa:stuff-struct-prop

Example 2.26.1. Let us look at some examples:

- The first projection fst : FinSet × FinSet → FinSet forgets 1-structure (stuff), namely the second set in the pair.
- The first projection fst : $\sum_{A:\mathrm{FinSet}} A \to \mathrm{FinSet}$ from the type of pointed finite sets to the type of finite sets forgets structure, namely the structure of a chosen point.
- The inclusion of the type of sets with cardinality n, FinSet_n, into the type of all finite sets, FinSet, forgets properties, namely the property "having cardinality n". ⌟

xca:stuff-struct-prop

xca:0Im-to-Im

Exercise 2.26.2. Analyze more examples of maps between groupoids in terms of "what is forgotten". ⌟

Exercise 2.26.3. Let $|\_|' : \|f^{-1}(a)\|_0 \to \|f^{-1}(a)\|$ be the map defined by the induction principle in Definition 2.22.4 from $|\_| : f^{-1}(a) \to \|f^{-1}(a)\|$. In the refined image factorization above, the map for the second arrow maps any pair (a, x) with $x : \|f^{-1}(a)\|_0$ to the pair $(a, |x|')$. Show that

[70] We're updating the terminology slightly: In the above references, n-structure is referred to as n-*stuff*, but nowadays the term *higher structure* is more common, so we have renamed n-stuff into n-*structure*.

[71] Using the general n-truncation, we can define the n-image in a similar way and prove that the n-image factorization is unique. Since the unit type $\mathbb{1}$ is the unique (-2)-type, we have $\|X\|_{-2} = \mathbb{1}$ for any type X.

for any $p : \|f^{-1}(a)\|$ the fiber of the latter map at (a, p) is equivalent to $\|f^{-1}(a)\|_0$. What is forgotten by this map, and what is remembered? ⌟

3
The universal symmetry: the circle

cha:circle

An effective principle in mathematics is that when you want to study a certain phenomenon you should search for a single type that captures this phenomenon. Here are two examples:[1]

it:one-out

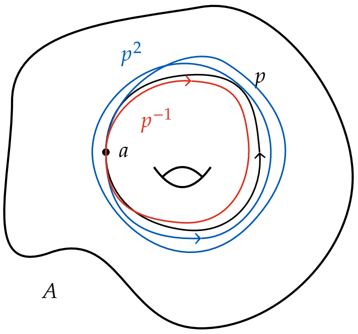
(1) The contractible type $\mathbb{1}$ has the property that given any type A a function $\mathbb{1} \to A$ provides exactly the same information as picking an element in A. For, an equivalence from A to $\mathbb{1} \to A$ is provided by the function $a \mapsto (x \mapsto a)$.

it:prop-in

(2) The type Prop of propositions has the property that given any type A a function $A \to \mathrm{Prop}$ provides exactly the same information as picking a subtype of A.

We are interested in symmetries, and so we should search for a type X which is so that given *any* type A the type of functions $X \to A$ (or $A \to X$, but that's not what we're going to do) picks out exactly the symmetries in A. We will soon see that there is such a type: the circle[2] which is built *exactly* so that this "universality with respect to symmetries" holds. It may be surprising to see how little it takes to define it; especially in hindsight when we eventually discover some of the many uses of the circle.

A symmetry in A is an identification $p : a =_A a$ for some $a : A$. Now, we can take any iteration of p (composing p with itself a number of times), and we can consider the inverse p^{-1} and *its* iterations. So, by giving one symmetry we at the same time give a lot of symmetries. For a particular p it may be that not all of the iterations are different, for instance it may be that $p^2 = p^0 :\equiv \mathrm{refl}_a$ (like in Exercise 2.13.3), or even more dramatic: if $p = \mathrm{refl}_a$, then *all* the iterates of p are equal. However, in general we must be prepared that all the powers of p (positive, 0 and negative) are distinct. Hence, the circle must have a distinct symmetry for every integer. We would have enjoyed defining the integers this way, but being that ideological would be somewhat inefficient. Hence we give a more hands-on approach defining the circle and the integers separately. Thereafter we prove that the type of symmetries in the circle is equivalent to the set of integers.



3.1 The circle and its universal property

sec:S1

Propositional truncation from Section 2.16 was the first *higher inductive type*, that is, an inductive type with constructors both for elements and for identifications, we introduced. The circle is another example of a higher inductive type, see Chapter 6 of the HoTT book[3] for more information.

[1] Notice that these have arrows pointing in different directions: In Item (1) we're mapping *out* of $\mathbb{1}$, while in Item (2) we're mapping *in* to Prop.

[2] We call this type the "circle" because it turns out to be the homotopy type of the topological circle $\{ (x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1 \}$. In the later chapters on geometry we'll return to "real" geometrical circles.

[3] Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*.

def:circle

Definition 3.1.1. The circle is a type $\mathrm{S}^1 : \mathcal{U}$ with an element (constructor) $\bullet : \mathrm{S}^1$ and an identification (constructor) $\circlearrowright : \bullet = \bullet$. For convenience and clarity the (higher) induction principle for S^1 is explained by first stating a recursion principle for S^1.

Let A be a type. In order to define a function $f : \mathrm{S}^1 \to A$, it suffices to give an element a of A together with an identification l of type $a = a$. The function f defined by this data satisfies $f(\bullet) \equiv a$ and the recursion principle provides an (unnamed) element of $\mathrm{ap}_f(\circlearrowright) = l$.

Let $A(x)$ be a type for every $x : \mathrm{S}^1$. The induction principle of S^1 states that, in order to define an element of $A(x)$ for every element x of S^1, it suffices to give an element a of $A(\bullet)$ together with an identification l of type $a =^A_{\circlearrowright} a$, cf. Fig. 3.1. The function f defined by this data satisfies $f(\bullet) \equiv a$ and the induction principle provides an element of $\mathrm{apd}_f(\circlearrowright) = l$. ⌟

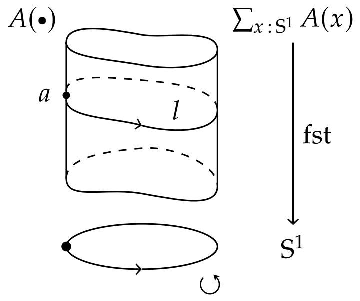


Figure 3.1: The induction principle of S^1.

fig:circle-induction

Giving a as above is referred to as 'the base case', and giving l as 'the loop case'. Given this input data to define a function f will often be abbreviated by writing $f(\bullet) :\equiv a$ and $f(\circlearrowright) := l$.

The following result states that any function from the circle exactly picks out an element and a symmetry of that element. This is a "universal property" of the circle.

lem:freeloopspace

Theorem 3.1.2. *For all types A, the evaluation function*

$$\mathrm{ev}_A : (\mathrm{S}^1 \to A) \to \sum_{a:A} (a =_A a) \text{ defined by } \mathrm{ev}_A(g) :\equiv (g(\bullet), g(\circlearrowright))$$

is an equivalence, with inverse ve_A defined by the recursion principle.

Proof. Fix $A : \mathcal{U}$. We apply Lemma 2.9.9. For all $a : A$ and $l : a = a$ we have $\mathrm{ev}(\mathrm{ve}(a, l)) = (a, l)$ by the recursion principle. It remains to prove $\mathrm{ve}(\mathrm{ev}(f)) = f$ for all $f : \mathrm{S}^1 \to A$. This will follow from the following more general result. Let $f, g : \mathrm{S}^1 \to A$, $p : f(\bullet) = g(\bullet)$, and $q : f(\circlearrowright) = p^{-1} \cdot g(\circlearrowright) \cdot p$. We show $f = g$, for which it suffices to prove by circle induction that $P(x) :\equiv (f(x) = g(x))$ for all $x : \mathrm{S}^1$. For the base case we take p. The loop case reduces to $\mathrm{trp}^P_{\circlearrowright}(p) = p$ by Definition 2.7.2. By Lemma 2.14.3 we have $\mathrm{trp}^P_{\circlearrowright}(p) = g(\circlearrowright) \cdot p \cdot f(\circlearrowright)^{-1}$. By q we have $g(\circlearrowright) = p \cdot f(\circlearrowright) \cdot p^{-1}$. Hence $\mathrm{trp}^P_{\circlearrowright}(p) = p$ by easy calculation. Using Lemma 2.10.3 we can phrase the result as: if $\mathrm{ev}(f) = \mathrm{ev}(g)$, then $f = g$.

Now we get $\mathrm{ve}(\mathrm{ev}(f)) = f$, as $\mathrm{ev}(\mathrm{ve}(\mathrm{ev}(f))) = (f(\bullet), f(\circlearrowright)) \equiv \mathrm{ev}(f)$ with $p :\equiv \mathrm{refl}_{f(\bullet)}$ and q coming from the induction principle. □

cor:circle-loopspace

Corollary 3.1.3. *For any $a : A$, the function $\mathrm{ev}^a_A : ((\mathrm{S}^1, \bullet) \to_* (A, a)) \to (a =_A a)$ sending (g, p) to $p^{-1} \cdot g(\circlearrowright) \cdot p$ is an equivalence.*

Proof.[4] It's easy to check commutativity of the diagram

$$\begin{tikzcd}
(\mathrm{S}^1 \to A) \arrow[rr, "g \mapsto (g(\bullet), g, \mathrm{refl}_{g(\bullet)})"] \arrow[dr, "\mathrm{ev}_A"'] & & \sum_{a:A} \big((\mathrm{S}^1, \bullet) \to_* (A, a)\big) \arrow[dl, "\mathrm{tot}(\mathrm{ev}^-_A)"] \\
& \sum_{a:A} (a =_A a), &
\end{tikzcd}$$

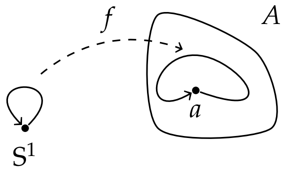
where the top map is an equivalence by Corollary 2.9.11, and the left map is an equivalence by Theorem 3.1.2. The result now follows from Lemma 2.9.16. □

[4] This can also be done directly: The inverse to ev^a_A sends $l : a =_A a$ to $(\mathrm{ve}_A(a, l), \mathrm{refl}_a)$. Try to verify this!

rem:dep-univ-prop-circle

Remark 3.1.4. By almost the same argument as for Theorem 3.1.2 one can obtain the dependent universal property of the circle. Given a type family $A : S^1 \to \mathcal{U}$, the evaluation function $(\prod_{x : S^1} A(x)) \to \sum_{a : A(\bullet)} (a =^{A}_{\circlearrowleft} a)$ is an equivalence. ⌟

Remark 3.1.5. A function $f : S^1 \to A$ is often called a *loop* in A, the picture being that f throws $\circlearrowleft : \bullet = \bullet$ as a lasso in the type A.



Using the above equivalence, so that $a =_A a$ is identified with the pointed functions from the circle, this allows for a very graphic interpretation of the symmetries of a in A: they are traced out by a function f from the circle and can be seen as loops in the type A starting and ending at a![5] ⌟

lem:circleisconnected

Lemma 3.1.6. *The circle is connected.*

Proof. We show $\|\bullet = z\|$ for all $z : S^1$ by circle induction as in Definition 3.1.1. For the base case we take $|\mathrm{refl}_\bullet|$. The loop case is immediate as $\|\bullet = \bullet\|$ is a proposition. □

In the proof above, the propositional truncation coming from the definition of connectedness is essential. If this truncation were removed we wouldn't know what to do in the induction step (actually, $\bullet = z$ for all $z : S^1$ contradicts the univalence axiom). This said, the family $R : S^1 \to \mathcal{U}$ with $R(z) :\equiv (\bullet = z)$ is extremely important for other purposes. We will call it in Definition 3.3.9 the "universal set bundle" of the circle and it is the key tool in proving that the set of integers and the symmetries in the circle can be identified. Recall that we use the phrase "symmetries *in* the circle" to refer to the elements of $\bullet =_{S^1} \bullet$,[6] whereas we use the phrase "symmetries *of* the circle" to refer to the elements of $S^1 =_{\mathcal{U}} S^1$. The latter type is equivalent to $S^1 \amalg S^1$, as follows from Exercise 3.9.6 and Exercise 3.9.7.

In order to proceed, we should properly define the set of integers and explore the concept of set bundles.

sec:integers

3.2 *The integers*

We define the type of integers in one of the many possible ways.[7]

def:zet

Definition 3.2.1. Let $\mathbb{Z}$ be the higher inductive type with the following three constructors:

(1) $\iota_+ : \mathbb{N} \to \mathbb{Z}$ for the nonnegative numbers, $0, 1, \ldots$

(2) $\iota_- : \mathbb{N}^- \to \mathbb{Z}$ for the nonpositive numbers, $-0, -1, \ldots$

(3) $\mathrm{zeq} : \iota_-(-0) = \iota_+(0)$.

Because we used the copy $\mathbb{N}^-$ for the nonpositive numbers from Example 2.12.6, we can leave out the constructor symbols $\iota_\pm$ when the type is clear from context. Thus we have $\ldots, -2, -1, -0, 0, 1, 2, \ldots : \mathbb{Z}$ and $\mathrm{zeq} : -0 =_{\mathbb{Z}} 0$.

The type $\mathbb{Z}$ comes with an induction principle: Let $T(z)$ be a family of types parametrized by $z : \mathbb{Z}$. In order to construct an element $f(z)$ of $T(z)$ for all $z : \mathbb{Z}$, it suffices to give functions g and h such that $g(n) : T(\iota_+(n))$ and $h(n) : T(\iota_-(m))$ for all $n : \mathbb{N}, m : \mathbb{N}^-$, together with $q : h(-0) =^{T}_{\mathrm{zeq}} g(0)$. Here g and h can be defined by induction on $n : \mathbb{N}, m : \mathbb{N}^-$.[8]

[5] This is of course how we have been picturing loops the whole time.

[6] Here we are using "the circle" to mean the *pointed* type $(S^1, \bullet)$. But it also turns out that the type $\bullet =_{S^1} \bullet$ is equivalent to the type $x =_{S^1} x$, for any $x : S^1$.

[7] Here are some of these alternatives:

- As the copy of $\mathbb{N}$ where $2n$ means n and $2n + 1$ means $-n - 1$, for $n : \mathbb{N}$.
- As the sum $\mathbb{N} \amalg \mathbb{N}$, where inl_n means $-n - 1$ and inr_n means n.
- As the sum $\mathbb{N} \amalg \mathbb{1} \amalg \mathbb{N}$, where from the left copy of $\mathbb{N}$ we get $-n - 1$, from the center $0 : \mathbb{1}$ we get 0, and from the right copy of $\mathbb{N}$ we get $n + 1$, for $n : \mathbb{N}$.
- As the quotient of $\mathbb{N} \times \mathbb{N}$ under the equivalence relation $(n, m) \sim (n', m')$ defined by $n + m' = n' + m$, where (n, m) represents $n - m$.
- As the subset of $\mathbb{N} \times \mathbb{N}$ consisting of those (n, m) with $n = 0 \vee m = 0$ (picking canonical representatives for the above equivalence relation).
- As the loops $\bullet =_{S^1} \bullet$ in the circle.

[8] Of course, giving h is the same as giving $h' : \prod_{n : \mathbb{N}} T(-n)$.

The resulting function $f : \prod_{z:\mathrm{Z}} T(z)$ satisfies $f(n) \equiv g(n)$ and $f(-n) \equiv h(-n)$ for $n : \mathbb{N}$, and there is an (unnamed) element of $\mathrm{apd}_f(\mathrm{zeq}) = q$. ⌟

Like the type $\mathbb{N}$, the type Z is a set with decidable equality and ordering relations.

One well-known self-equivalence is *negation*, $- : \mathrm{Z} \to \mathrm{Z}$, also called *complement*, inductively defined by setting $-\iota_+(n) :\equiv \iota_-(-n)$, $-\iota_-(m) :\equiv \iota_+(-m)$, $\mathrm{ap}_-(\mathrm{zeq}) := \mathrm{zeq}^{-1}$.[9] Negation is its own inverse.

The *successor* function $\mathrm{s} : \mathrm{Z} \to \mathrm{Z}$ is likewise defined inductively, setting $\mathrm{s}(n) :\equiv \mathrm{succ}(n)$, $\mathrm{s}(-0) :\equiv 1$, $\mathrm{s}(-\mathrm{succ}(n)) :\equiv -n$, and $\mathrm{ap}_{\mathrm{s}}(\mathrm{zeq}) := \mathrm{refl}_1$.

The successor function s is an equivalence. It is instructive to depict iterating s in both directions as a doubly infinite sequence containing all integers:

$$\begin{tikzcd}
 & & & -0 \arrow[dr, mapsto] \arrow[dd, equal, "\downarrow"'] & & & \\
\cdots \arrow[r, mapsto] & -2 \arrow[r, mapsto] & -1 \arrow[ur, mapsto] & & 1 \arrow[r, mapsto] & 2 \arrow[r, mapsto] & \cdots \\
 & & & 0 \arrow[ur, mapsto] & & &
\end{tikzcd}$$

The inverse s^{-1} of s is called the *predecessor* function. We recall the n-fold iteration s^n defined earlier; the n-fold iteration of s^{-1} will be denoted by s^{-n}. Since $\mathrm{s}^0 \equiv \mathrm{id} \equiv \mathrm{s}^{-0}$, this defines the iteration s^z for all $z : \mathrm{Z}$.[10]

Addition of integers is now defined by iteration: $z + y :\equiv \mathrm{s}^y(z)$. This extends + on the ι_+-image of $\mathbb{N}$, see Exercise 3.2.2. From addition and unary − one can define a binary *subtraction* function setting $z - y :\equiv z + (-y)$. Since addition and subtraction are mutually inverse, we may iterate addition to define *multiplication*: $zy :\equiv (w \mapsto w + z)^y(0)$.

xca:addition-on-Z-and-N

EXERCISE 3.2.2. Show that $\iota_+(n + m) = \iota_+(n) + \iota_+(m)$ and $\iota_+(nm) = \iota_+(n)\iota_+(m)$ for all $n, m : \mathbb{N}$. ⌟

The ordering relations < and ≤ on Z are easily defined and shown to extend those on $\mathbb{N}$.

Recall the induction principle for Z in Definition 3.2.1 above. Instead of defining g and h explicitly, we will often give $f(0)$ directly, and define g' and h' such that $g'(z) : T(z) \to T(z+1)$ for all $z : \mathrm{Z}$ with $z \geq 0$, and $h'(z) : T(z) \to T(z-1)$ for all $z : \mathrm{Z}$ with $z \leq 0$. The function f thus defined satisfies $f(-0) \equiv f(0)$, $f(z+1) \equiv g'(z, f(z))$ for all $z \geq 0$, and $f(z-1) \equiv h'(z, f(z))$ for all $z \leq 0$.

xca:commutative-add-Z

EXERCISE 3.2.3. Show that $x + y = y + x$ and $xy = yx$ for all $x, y : \mathrm{Z}$. ⌟

sec:covering

3.3 *Set bundles*

As mentioned earlier, it is possible to define the integers as the type $\bullet =_{S^1} \bullet$ of symmetries in the circle. Our investigation of $\bullet =_{S^1} \bullet$ will use the concept of set bundles. Since we are going to return to this concept several times, we take the time for a fuller treatment before we continue with proving the equivalence of $\bullet =_{S^1} \bullet$ and Z.

def:covering

DEFINITION 3.3.1. A *set bundle* over a type B is a map $f : A \to B$ such that for each $b : B$ the preimage $f^{-1}(b)$ is a set. We say that a set bundle $f : A \to B$ over B is

- *connected* if A is connected,

[9] Here we included the constructor symbols for clarity, but the definition allows us to use the negation symbol unadorned, because the following diagram commutes (even up to definitional equality):

$$\begin{tikzcd}
\mathbb{N} \arrow[r, shift left, "-"] \arrow[d, "\iota_+"'] & \mathbb{N}^- \arrow[l, shift left, "-"] \arrow[d, "\iota_-"] \\
\mathrm{Z} \arrow[r, shift left, "-"] & \mathrm{Z} \arrow[l, shift left, "-"]
\end{tikzcd}$$

[10] In the same way, we can define the iteration $f^z : X \to X$ for any *equivalence* $f : X \to X$.

- *universal* if A and all the identity types $a =_A a$ (for $a : A$) are connected,
- *finite* if all preimages are finite sets,
- *decidable* if all preimages are decidable sets.

If B is a pointed type, a *pointed* set bundle is a pointed map $f : A \to_* B$ such that, when forgetting the points, $f_\div : A_\div \to B_\div$ is a set bundle. Here we only require A to be a pointed type.[11] We do not require the preimages of $f_\div$ to be pointed types. ⌟

With a formula, given a type B, the type of set bundles over B is

$$\mathrm{SetBundle}(B) :\equiv \sum_{A:\mathcal{U}} \sum_{f:A\to B} \prod_{b:B} \mathrm{isSet}(f^{-1}(b)),$$

with variations according to the flavor.

Recall the equivalence in Lemma 2.25.4(2) between the type $B \to \mathrm{Set}$ of families of sets parametrized by elements of B, and the type of set bundles over B given above. We shall frequently use this equivalence, even without explicit mention.

lem:setbundle-is-groupoid

Lemma 3.3.2. *For any type B,* $\mathrm{SetBundle}(B)$ *is a groupoid.*

Proof. By Lemma 2.22.1 we have that Set is a groupoid, and hence $B \to \mathrm{Set}$ is a groupoid by Lemma 2.15.4(1). Moreover, by Corollary 2.20.7, all variations in Definition 3.3.1 defined by a predicate are groupoids as well. □

One notable exception to the above lemma is the type of *pointed* set bundles: a point is extra structure, not just a property.

We should notice that the notion of a set bundle is just one step up from the notion of an injection (a map such that all the preimages are propositions – following the logic, injections perhaps ought to be called "proposition bundles"). The formulation we give is not the only one and for some purposes a formulation based on $B \to \mathrm{Set}$ is more convenient.

xca:constant-cover

Exercise 3.3.3. Let A, B and C be types. Show:

(1) The (unique) map of type $A \to \mathbb{1}$ is a set bundle iff A is a set;

(2) For any $b : B$, the map $x \mapsto b$ from $\mathbb{1}$ to B is a set bundle iff $b = b$ is a set;

(3) If $f : A \to B$ and $g : B \to C$ are set bundles, then gf is a set bundle. ⌟

Figure 3.2 visualizes two examples of set bundles over the circle. Consider the picture on the left first. If we let b be the element on the circle marked at the bottom left hand side, then the preimage $f^{-1}(b)$ is marked by the the two dots in A straight above b, so that in this case each preimage contains two points (is *merely equal* to $\mathbb{2}$). However, A is not the constant family, like A' depicted on the right, since $A' = \sum_{z:\mathrm{S}^1} \mathbb{2} = \mathrm{S}^1 \times \mathbb{2} = \mathrm{S}^1 + \mathrm{S}^1$ is not connected. Obviously something way more fascinating is going on. (In fact the set bundle on the left is given by $\mathrm{ve}_{\mathcal{U}}(\mathrm{Bool}, \mathrm{twist})$, see Exercise 2.13.3 and Theorem 3.1.2.)

Remark 3.3.4. It *is* possible to misunderstand what a "connected set bundle" is: the other interpretation "all the preimages are connected" would simply give us an equivalence (since connected sets are contractible),

[11] If we forget the base point of B, and the pointedness of f, f_0, then these can be recovered uniquely, by setting $\mathrm{pt}_B :\equiv f(\mathrm{pt}_A)$ and $f_0 :\equiv \mathrm{refl}_{\mathrm{pt}_B} : \mathrm{pt}_B = f(\mathrm{pt}_A)$. Indeed, the forgetful map $\left(\sum_{b:B}(A,a) \to_* (B,b)\right) \to (A \to B)$ is an equivalence by Corollary 2.9.11.

fig:covering

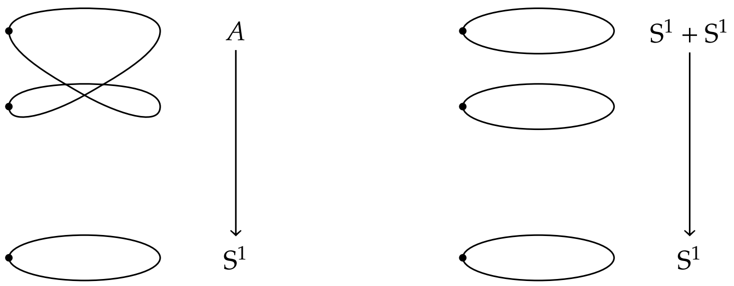


Figure 3.2: A visualization of two set bundles over the circle

and this is *not* what is intended. (Equivalences are set bundles, but not necessarily connected set bundles and connected set bundles are not neccesarily equivalences.)

Likewise for the other qualifications; for instance, in a "finite covering" $f : A \to B$, the type A is usually *not* a finite set.

We trust the reader to keep our definitions in mind and not the other interpretations. ⌟

Remark 3.3.5. Set bundles are closely related to a concept from topology called "covering spaces" (or any variant of this concept, including Galois theory) and from algebra as locally constant sheaves (of sets). Either way, the concept is useful because it singles out the (sub)symmetries. ⌟

In this chapter, we focus on set bundles over the circle.

thm:coveringsofS1perms

Theorem 3.3.6. *The evaluation function provides an equivalence*

$$\mathrm{ev}_{\mathrm{Set}} : (\mathrm{S}^1 \to \mathrm{Set}) \to \sum_{X : \mathrm{Set}} (X = X) \quad \textit{defined by } \mathrm{ev}_{\mathrm{Set}}(E) :\equiv (E(\bullet), E(\circlearrowleft)).$$

Consequently, we have a string of equivalences

$$\begin{aligned} \mathrm{SetBundle}(\mathrm{S}^1) \simeq (\mathrm{S}^1 \to \mathrm{Set}) \simeq \sum_{X : \mathrm{Set}} (X = X) \\ \simeq \sum_{X : \mathrm{Set}} (X \simeq X) \simeq \sum_{X : \mathcal{U}} \sum_{f : X \to X} \mathrm{isSet}(X) \times \mathrm{isEquiv}(f). \end{aligned}$$

Proof. The first part is the universal property of the circle, Theorem 3.1.2, applied to $A :\equiv \mathrm{Set}$. The equivalences then follow from Lemma 2.25.4(2) and the univalence axiom, together with minor manipulations. □

In slogan form: A set bundle over the circle is a set with a permutation of its elements. The fiber over $\bullet : \mathrm{S}^1$ gives the set, and transporting along $\circlearrowleft$ gives the permutation.

A particularly important example is the following:

def:RtoS1

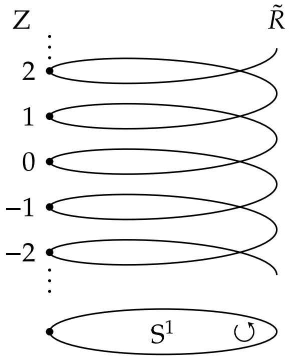
Definition 3.3.7. The set bundle $R : \mathrm{S}^1 \to \mathcal{U}$ corresponds to the integers with the successor operation. We have $R(\bullet) :\equiv \mathrm{Z}$ and $R(\circlearrowleft) := \bar{\mathrm{s}}$. (This is indeed a set bundle since S^1 is connected, so that $R(x)$ is a set for all $x : \mathrm{S}^1$. Abusing notation we also write $R : \mathrm{S}^1 \to \mathrm{Set}$.) Now define

$$\tilde{R} :\equiv \sum_{z : \mathrm{S}^1} R(z)$$

and let the first projection denoted by

$$\exp : \tilde{R} \to \mathrm{S}^1$$

be the *exponential set bundle of the circle*. ⌟



rem:expforreal

REMARK 3.3.8. The reason for the name “exponential” comes from the following visualization. If x is a real number, then the complex exponentiation $\mathrm{e}^{2\pi \mathrm{i}x} = \cos(2\pi x) + \mathrm{i}\sin(2\pi x)$ has absolute value 1 and so defines a continuous function from the real numbers to the unit circle. Choosing any point z on the unit circle, we see that the preimage of z under the exponential function is a shifted copy of the integers inside the reals.[12]

This connection between the integers and the unit circle is precisely captured in a form that we can take further by studying the set bundle $\exp : \tilde{R} \to S^1$. ⌟

We already defined a set bundle $f : A \to B$ to be universal if A is connected and all $a =_A a$ (for $a : A$) are connected. If moreover B is a pointed, connected groupoid we shall argue that we actually can speak of *the* universal set bundle.

Recall Corollary 2.17.10 stating that all the fibers of a map $f : A \to B$ are sets if and only if each

$$\mathrm{ap}_f : (a = a') \to (f(a) = f(a'))$$

is an injection. Assume $f : A \to B$ is a universal set bundle and B is a groupoid. We prove that A is contractible. Being contractible is a proposition, so we may assume we have an element a of A since A is connected. By Exercise 2.16.6 and 2.16.4 it suffices to prove that $a = a$ is contractible. By Exercise 2.16.5, using that $a = a$ is connected, it suffices to show that $a = a$ is a set. Using that ap_f is an injection, we can apply the remark after Lemma 2.25.2 and obtain that $a = a$ is a set since $f(a) = f(a)$ is a set, since B is a groupoid. This completes the proof that A is contractible.

Now assume (B, b_0) is a pointed connected groupoid and $f : A \to B$ a universal set bundle. Since A and $\sum_{b:B}(b_0 =_B b)$ are both contractible, and B is connected, we have $\|(A, f) = (\sum_{b:B}(b_0 =_B b), \mathrm{fst})\|$. Hence if (B, b_0) is a pointed connected groupoid, all universal set bundles are merely equal to a canonical one. Moreover, the type of universal set bundles is equivalent to $\mathbb{1} \to B$, and hence to B itself, so the type of *pointed* universal set bundles is contractible. This justifies the following definition.

def:universalcover

DEFINITION 3.3.9. Let (B, b_0) be a pointed connected groupoid. The *universal set bundle* of B is the set bundle of B given by the family of sets

$$\mathrm{P}_{b_0} : B \to \mathrm{Set}, \quad \mathrm{P}_{b_0}(b) :\equiv (b_0 =_B b),$$

or alternatively as the first projection from $\mathrm{P}_{b_0}B :\equiv \sum_{b:B}(b_0 =_B b)$ to B.

This is canonically pointed at $(b_0, \mathrm{refl}_{b_0}) : \sum_{b:B}(b_0 =_B b)$. ⌟

Note that, for a general pointed connected type (B, b_0), we have that $(b_0 =_B b)$ is a family of *sets* exactly when B is a groupoid. The type family $(b_0 =_B b)$ is also important if B is not a groupoid, but is then not a *set* bundle.[13]

REMARK 3.3.10. What’s so “universal” about this? The universal set bundle over the pointed connected groupoid (B, b_0) coincides with the constant function $\mathrm{cst}_{b_0} : \mathbb{1} \to B$ (with value b_0), and seems like an unnecessary complicated representation were it not for the manifold practical value of the formulation that we’ve given. In particular, we

[12] Again, we emphasize that we are here dealing with the *homotopy types* of the reals $\mathbb{R}$ and the unit circle, $\{(x, y) : \mathbb{R}^2 \mid x^2 + y^2 = 1\}$.

[13] Of course, the type $\sum_{b:B}(b_0 = b)$ is contractibe by Lemma 2.9.2, for any type B.

recognize the set of symmetries $b_0 =_B b_0$ as the preimage of b_0 under the first projection from $\mathrm{P}_{b_0}B$ to B; ultimately this will show that the study of symmetries coincides with the study of the universal set bundle.

The first instance of this comes already in the next section, where we show in Corollary 3.4.5 that the symmetries in the circle are given by the set of integers Z by showing that the universal set bundle and the exponential set bundle (Definition 3.3.7) of the circle coincide.

That said, one way to see that the constant function $\mathrm{cst}_{b_0} : \mathbb{1} \to B$ *does* deserve the label universal is the following. Given any function $f : A \to B$ and $(a_0, p) : f^{-1}(b_0)$, we get a function $\mathrm{cst}_{a_0} : \mathbb{1} \to A$, and $p : b_0 = f(a_0)$ gives rise to an element in $\mathrm{cst}_{b_0} =_{\mathbb{1}\to B} f \circ \mathrm{cst}_{a_0}$. In other words, any such f is "a factor of cst_{b_0}". Note, however, that this depends on $f^{-1}(b_0)$ being non-empty (classically, this is often demanded of a covering, which distinguishes it from our set bundles), and the factorization depends on the element (a_0, p) used.

Any $(a_0, p) : f^{-1}(b_0)$ gives rise to a commutative diagram:

$$\begin{tikzcd}
\mathbb{1} \arrow[rr, dashed, "\mathrm{cst}_{a_0}"] \arrow[dr, "\mathrm{cst}_{b_0}"'] & & A \arrow[dl, "f"] \\
& B &
\end{tikzcd}$$

The situation is even simpler for pointed maps: For any *pointed* map $f : A \to_* B$, with $(a_0, f_0) : f^{-1}(b_0)$, there is a *unique* pointed map $g : \mathbb{1} \to_* A$ (given by the base point of A), and this of course also gives the unique way to write f as a "pointed factor of cst_{b_0}". ⌟

We'll continue the general study of set bundles in Section 4.5 and indeed throughout the book. For now, we'll focus our attention on the circle and set bundles over it.

3.4 *The symmetries in the circle*

sec:symcirc

With the set Z of integers *defined* as in Section 3.2, we will now *prove* that Z is equivalent to the type $\bullet =_{\mathrm{S}^1} \bullet$, and that under this equivalence $0 : \mathrm{Z}$ corresponds to $\mathrm{refl}_\bullet : \bullet = \bullet$, and 1 to $\circlearrowleft$, and -1 to $\circlearrowleft^{-1}$. More generally, the successor $\mathrm{s} : \mathrm{Z} \to \mathrm{Z}$ corresponds to composition with $\circlearrowleft$, while the predecessor s^{-1} corresponds to composition with $\circlearrowleft^{-1}$.

It follows directly that *addition* of integers corresponds to *composition* of loops.

The first step is to prove that the exponential set bundle Definition 3.3.7 is equal to the universal set bundle in Definition 3.3.9, i.e., we prove that the family

$$R : \mathrm{S}^1 \to \mathcal{U}, \qquad R(\bullet) :\equiv \mathrm{Z},\ R(\circlearrowleft) := \bar{\mathrm{s}}$$

is equal to the family

$$\mathrm{P}_\bullet : \mathrm{S}^1 \to \mathcal{U}, \qquad \mathrm{P}_\bullet(z) :\equiv (\bullet = z).$$

What does it mean for the families $\mathrm{P}_\bullet$ and R to be equal? Type families are a special case of functions. Function extensionality reduces the question to pointwise equality of $\mathrm{P}_\bullet$ and R as functions. Using univalence, it suffices to give an equivalence from $\mathrm{P}_\bullet(z)$ to $R(z)$ for every $z : \mathrm{S}^1$, that is, recalling Definition 2.14.1, a (fiberwise) equivalence $f : \mathrm{P}_\bullet \to R$. We will use Lemma 2.9.9, so will also define $g : R \to \mathrm{P}_\bullet$.

We first recall from Section 2.14 how transport behaves in families of function types. Given a type A and two type families $P, Q : A \to \mathcal{U}$, transport along $p : a =_A a'$ of $h : P(a) \to Q(a)$ is $Q(p)\, h\, P(p)^{-1} : P(a') \to$

$Q(a')$. In a picture,

$$\begin{tikzcd} a \arrow[d, equal, "p"] & P(a) \arrow[r, "h"] \arrow[d, equal, "P(p)"] & Q(a) \arrow[d, equal, "Q(p)"] \\ a' & P(a') & Q(a'). \end{tikzcd}$$

If A is S^1, then the induction principle for the circle says that giving an $h(z) : P(z) \to Q(z)$ for all $z : \mathrm{S}^1$ is the same as specifying an $h(\bullet) : P(\bullet) \to Q(\bullet)$ and, using Definition 2.7.2 and the discussion above, an identity $h(\circlearrowleft) : Q(\circlearrowleft)\, h(\bullet)\, P(\circlearrowleft)^{-1} =_{P(\bullet)\to Q(\bullet)} h(\bullet)$, i.e., a witness that the composites in

$$\begin{tikzcd} P(\bullet) \arrow[r, "h(\bullet)"] \arrow[d, equal, "P(\circlearrowleft)"] & Q(\bullet) \arrow[d, equal, "Q(\circlearrowleft)"] \\ P(\bullet) \arrow[r, "h(\bullet)"] & Q(\bullet) \end{tikzcd}$$

are equal. If P, Q are families of sets, then the type of $h(\circlearrowleft)$ is a proposition.

We now define $f : \mathrm{P}_\bullet \to R$ and $g : R \to \mathrm{P}_\bullet$ that will turn out to give inverse equivalences between $\mathrm{P}_\bullet(z)$ and $R(z)$, for each $z : \mathrm{S}^1$.

def:fPtoR

Definition 3.4.1. The function $f : \prod_{z : \mathrm{S}^1} (\mathrm{P}_\bullet(z) \to R(z))$ is defined by transport: $f(z)(p) :\equiv \mathrm{trp}^R_p(0)$. ⌟

In Figure 3.3, the transport in the definition above has been visualised for $p = \circlearrowleft^n$, $n = -2, -1, 0, 1, 2$.

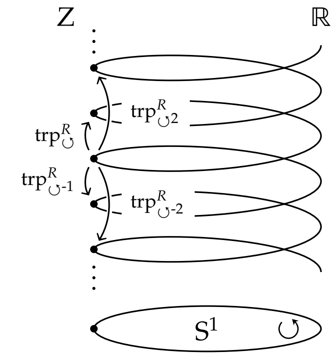

Z
ℝ
$\mathrm{trp}^R_{\circlearrowleft 2}$
$\mathrm{trp}^R_{\circlearrowleft}$
$\mathrm{trp}^R_{\circlearrowleft \text{-}1}$
$\mathrm{trp}^R_{\circlearrowleft \text{-}2}$
S^1
$\circlearrowleft$

Figure 3.3: Transport in the family R

fig:transportalongloop

lem:windingnumber

Lemma 3.4.2. *For f as in Definition 3.4.1 we have $f(\bullet)(\circlearrowleft^n) = n$ for all $n : \mathrm{Z}$.*

Proof. First consider positive $n : \mathbb{N}$ and apply induction. In the base case $n = 0$ we have $f(\bullet)(\circlearrowleft^0) \equiv f(\mathrm{refl}_\bullet) \equiv \mathrm{trp}^R_{\mathrm{refl}_\bullet}(0) \equiv 0$. For $n = \mathrm{s}(m)$ with $m : \mathbb{N}$ we have

$$\begin{aligned} f(\bullet)(\circlearrowleft^{\mathrm{s}(m)}) &\equiv \mathrm{trp}^R_{\circlearrowleft^{\mathrm{s}(m)}}(0) \\ &= \mathrm{trp}^R_{\circlearrowleft\circlearrowleft^m}(0) \\ &= \mathrm{trp}^R_{\circlearrowleft}(\mathrm{trp}^R_{\circlearrowleft^m}(0)) \\ &\equiv \mathrm{trp}^R_{\circlearrowleft}(f(\bullet)(\circlearrowleft^m)) \\ &= \mathrm{s}(f(\bullet)(\circlearrowleft^m)). \end{aligned}$$

The last step follows from $\bar{\mathrm{s}} = R(\circlearrowleft)$ and $\mathrm{s} = \mathrm{trp}^{\mathrm{id}_{\mathcal{U}}}_{\bar{\mathrm{s}}}$, see Principle 2.13.2, and hence $\mathrm{s} = \mathrm{trp}^{\mathrm{id}_{\mathcal{U}}}_{R(\circlearrowleft)} = \mathrm{trp}^{\mathrm{id}_{\mathcal{U}} R}_{\circlearrowleft} = \mathrm{trp}^R_{\circlearrowleft}$. This completes the induction step for positive n. For negative n the proof is similar. □

In the definition of the second map, take into account that $R(\bullet) \equiv \mathrm{Z}$ and $\mathrm{P}_\bullet(\bullet) \equiv (\bullet = \bullet)$.

def:gRtoP

Definition 3.4.3. The function $g : \prod_{z : \mathrm{S}^1} (R(z) \to \mathrm{P}_\bullet(z))$ is defined by circle induction:

$$g(\bullet) :\equiv (n \mapsto \circlearrowleft^n) : \mathrm{Z} \to (\bullet = \bullet)$$

and

$$g(\circlearrowleft) : \mathrm{P}_\bullet(\circlearrowleft)\, g(\bullet)\, R(\circlearrowleft)^{-1} =_{\mathrm{Z}\to(\bullet=\bullet)} g(\bullet).$$

So far we have only given the type of $g(\circlearrowleft)$. By definition, $R(\circlearrowleft)$ is s and $\mathrm{P}_\bullet(\circlearrowleft)$ is composition with $\circlearrowleft$. The element $g(\circlearrowleft)$ follows by a simple calculation: the proposition $\circlearrowleft \circlearrowleft^{n-1} = \circlearrowleft^n$ holds for all $n : \mathrm{Z}$. ⌟

In a picture, $g(\circlearrowleft)$ should prove that it does not matter what path you take around the square

$$\begin{tikzcd} \mathrm{Z} \arrow[r, "\circlearrowleft^-"] \arrow[d, equal, "\mathrm{s}"] & (\bullet = \bullet) \arrow[d, equal, "\circlearrowleft \cdot \_"] \\ \mathrm{Z} \arrow[r, "\circlearrowleft^-"] & (\bullet = \bullet). \end{tikzcd}$$

lem:univisexp

Theorem 3.4.4. *For every $z : \mathrm{S}^1$, the functions $f(z)$ defined in Definition 3.4.1 and $g(z)$ in Definition 3.4.3 are inverse equivalences between $\mathrm{P}_\bullet(z)$ and $R(z)$.*

Proof. We apply Lemma 2.9.9 and verify the two conditions. First, we need to give elements $H(z,p) : g(z)(f(z)(p)) = p$ for all $z : \mathrm{S}^1$ and $p : \mathrm{P}_\bullet(z) \equiv (\bullet = z)$. By induction on $p : \bullet = z$ it suffices to set $H(\bullet, \mathrm{refl}_\bullet) :\equiv \mathrm{refl}_{\mathrm{refl}_\bullet}$ since $g(\bullet)(f(\bullet)(\mathrm{refl}_\bullet)) \equiv g(\bullet)(0) \equiv \mathrm{refl}_\bullet$.

Secondly, we need to give elements $G(z)(n) : f(z)(g(z)(n)) = n$ for all $z : \mathrm{S}^1$ and $n : R(z)$. By circle induction it suffices to define $G(\bullet)$ and $G(\circlearrowleft)$, but since $\mathbb{Z}$ is a set the information for $G(\circlearrowleft)$ is redundant. Hence, we need to show that for all $n : \mathbb{Z}$ that $f(\bullet)(g(\bullet)(n)) \equiv f(\bullet)(\circlearrowleft^n)$ is equal to n. This follows from Lemma 3.4.2. □

cor:S1groupoid

Corollary 3.4.5. *The circle S^1 is a groupoid, and the function*

$$\circlearrowleft^- : \mathbb{Z} \to (\bullet =_{\mathrm{S}^1} \bullet)$$

sending n to $\circlearrowleft^n$ is an equivalence.

Proof. For any $z : \mathrm{S}^1$, the type $\mathrm{P}_\bullet(z) \equiv (\bullet =_{\mathrm{S}^1} z)$ is a set since $R(z)$ is a set and $\mathrm{P}_\bullet(z) \simeq R(z)$. Since the circle is connected and being a set is a proposition, it follows that $y =_{\mathrm{S}^1} z$ is a set, for any $y, z : \mathrm{S}^1$. Hence S^1 is a groupoid. By Definition 3.4.3, $\circlearrowleft^- \equiv g(\bullet)$ is an equivalence. □

def:windingnumber

Definition 3.4.6. The inverse function of $\circlearrowleft^-$ is called the *winding number* function $\mathrm{wdg} : (\bullet =_{\mathrm{S}^1} \bullet) \to \mathbb{Z}$. ⌟

The following lemma is a simple example of a technique later called *delooping*.

lem:S1-delooping

Lemma 3.4.7. *Let A be a connected type and $a : A$. Assume we have an equivalence $e : (\bullet =_{\mathrm{S}^1} \bullet) \to (a = a)$ of symmetries such that $e(\mathrm{refl}_\bullet) = \mathrm{refl}_a$ and $e(p \cdot q) = e(p) \cdot e(q)$, for all $p, q : (\bullet =_{\mathrm{S}^1} \bullet)$. Then $\check{e} : \mathrm{S}^1 \to A$ defined by circle recursion by setting $\check{e}(\bullet) :\equiv a$ and $\check{e}(\circlearrowleft) := e(\circlearrowleft)$ is an equivalence.*

Proof. We have $\mathrm{ap}_{\check{e}} = e$ since they produce equal values when applied to $\circlearrowleft^n$, for all $n : \mathbb{Z}$. Now use that A and S^1 are connected and apply Corollary 2.17.10(3). □

xca:general_winding

Exercise 3.4.8. Using circle induction, define for any point $x : \mathrm{S}^1$ of the circle an equivalence, $\mathrm{wdg}_x : (x =_{\mathrm{S}^1} x) \xrightarrow{\sim} \mathbb{Z}$, generalizing Definition 3.4.6. (You'll need commutativity of addition in $\mathbb{Z}$.) Conclude from Lemma 3.4.7 that we have equivalences $f_x : \mathrm{S}^1 \xrightarrow{\sim} \mathrm{S}^1$ with $f_x(\bullet) \equiv x$, for each $x : \mathrm{S}^1$.[14] ⌟

3.5 *A reinterpretation of the circle*

sec:S1isC

In this section we return to the equivalences in Theorem 3.3.6. We'll use these to get a different perspective on the circle, which highlights it as a type classifying very simple symmetries, namely sets with permutations. We have already seen one example in Definition 3.3.7, namely the set $\mathbb{Z}$ of integers together with the successor $\mathrm{s} : \mathbb{Z} \simeq \mathbb{Z}$, corresponding to the universal set bundle $\mathrm{P}_\bullet : \mathrm{S}^1 \to \mathrm{Set}$, which as a map is the constant function $\mathrm{cst}_\bullet : \mathbb{1} \to \mathrm{S}^1$.

[14] If we think of the circle as represented by the unit length complex numbers, then $f_x(y)$ corresponds to the usual product xy.

The importance of the latter example will become apparent when we eventually explain that *the circle is equivalent to the connected component of* (Z, s) *in the type* $\sum_{X:\mathcal{U}}(X \to X)$.[15]

The key of course is that the equivalences in Theorem 3.3.6 restrict to equivalences between their connected components, so to understand the components of $\mathrm{SetBundle}(\mathrm{S}^1)$ it suffices to understand the components of $\sum_{X:\mathcal{U}}(X \to X)$ at pairs (X, t), where X is a set with a permutation t.

We are particularly interested in understanding the symmetries in these components, so before we prove that the circle is equivalent to the component containing (Z, s), let us investigate the equalities in the type $\sum_{X:\mathcal{U}}(X \to X)$ a bit further.

Define the type family D by $D(X) :\equiv (X \to X)$ for all $X : \mathcal{U}$. Recall from Lemma 2.10.3 that, given $X, Y : \mathcal{U}$ and $t : X \to X$ and $u : Y \to Y$, the identity type $(X, t) = (Y, u)$ is equivalent to the type of pairs consisting of a $p : X = Y$ and an element of $t =^D_p u$. The latter type is equivalent to $\mathrm{trp}^D_p(t) = u$ by Definition 2.7.2. The transport is by conjugation, Lemma 2.14.2, so that the latter type is equivalent to $\tilde{p} \circ t \circ \tilde{p}^{-1} = u$. If $p \equiv \bar{e}$ for an equivalence $e : X \simeq Y$, this is equivalent to $e \circ t = u \circ e$, or $et = ue$ for short. In total, the identity type $(X, t) = (Y, u)$ is equivalent to

$$\sum_{e:X \simeq Y} et =_{X \to Y} ue.$$

This is a set whenever X and Y are; see Fig. 3.4 for an illustration.

In particular, the identity type $(\mathrm{Z}, \mathrm{s}) = (X, t)$ is equivalent to the set $\sum_{e:\mathrm{Z} \simeq X} e\,\mathrm{s} = te$, for any set X with a permutation t. Tautologically, then, any power s^n of s itself gives a symmetry $(\mathrm{s}^n, !) : (\mathrm{Z}, \mathrm{s}) = (\mathrm{Z}, \mathrm{s})$.

The following property jumps out at us when we contemplate Fig. 3.4.

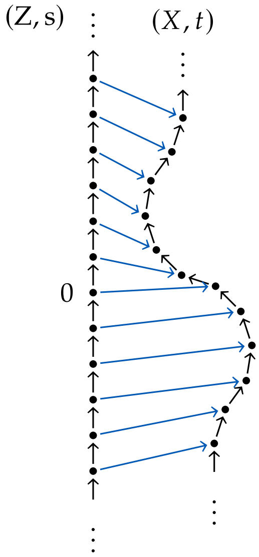


FIGURE 3.4: An identification of two infinite cycles. The equivalence $e : \mathrm{Z} \simeq X$ is marked in blue.

fig:Zs=Xt

lem:IdCisZet

LEMMA 3.5.1. *An element* $(e, !) : (\mathrm{Z}, \mathrm{s}) = (X, t)$, *with* (X, t) *in the component of* (Z, s), *is uniquely determined by the element* $e(0) : X$. *In other words, the function*

$$\mathrm{ev}_0 : \big((\mathrm{Z}, \mathrm{s}) = (X, t)\big) \to X \;\; \textit{defined by} \;\; \mathrm{ev}_0(e, !) :\equiv e(0)$$

is an equivalence.

Proof. We'll prove that every fiber of ev_0 is contractible. Given $x_0 : X$ we must determine a unique equivalence $e : \mathrm{Z} \to X$ such that $es = te$ and $e(0) = x_0$. Induction on $n : \mathrm{Z}$ (positive and negative n separately) shows that for such an e, we have $e(n) = t^n(x_0)$ for all $n : \mathrm{Z}$. It remains to prove that this is an equivalence. More precisely, it suffices to prove the proposition:

$$\prod_{x_0:X} \mathrm{isEquiv}(n \mapsto t^n(x_0))$$

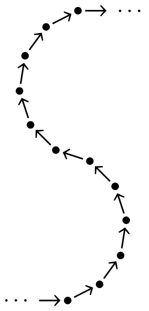
Since we are proving a proposition, and we are assuming (X, t) is in the component of (Z, s), it suffices to prove it for $(X, t) \equiv (\mathrm{Z}, \mathrm{s})$. However, for any $x_0 : \mathrm{Z}$, the map $n \mapsto \mathrm{s}^n(x_0) = n + x_0$ is an equivalence, with inverse $n \mapsto n - x_0$. □

In particular, the identity type $(\mathrm{Z}, \mathrm{s}) = (\mathrm{Z}, \mathrm{s})$ is equivalent to Z.

def:S1toC

DEFINITION 3.5.2. Let InfCyc be the component of $\sum_{X:\mathcal{U}}(X \to X)$ containing (Z, s). Elements of InfCyc are called *infinite cycles*.[16]

[15] The elements of this connected component can be thought of as *infinite cycles*: sets X with a successor function $t : X \to X$ such that (X, t) is merely equal to (Z, s). That is, (X, t) looks exactly like (Z, s), but we don't know which element of X is "zero":



[16] See also Definition 3.6.1 below for general cycles.

Define by circle induction

$$c : \mathrm{S}^1 \to \mathrm{InfCyc} \;\text{ setting }\; c(\bullet) :\equiv (\mathrm{Z}, \mathrm{s})$$

and $c(\circlearrowleft) : c(\bullet) = c(\bullet)$ given by the *predecessor* equivalence $\mathrm{s}^{-1} : \mathrm{Z} \to \mathrm{Z}$ and the trivial proof of the proposition $\mathrm{s}^{-1}\,\mathrm{s} = \mathrm{s}\,\mathrm{s}^{-1}$. ⌟

Note that, as usual, we leave out the propositional components of InfCyc (and other subtypes) from the notation.

Since it's such a crucial result, we are going to give two proofs that c from Definition 3.5.2 is an equivalence. Each proof illuminates a different aspect and gives methods that will be used later.

For the first, we return to the equivalences of Theorem 3.3.6. As we said above, these restrict to equivalences between the different components. In particular, $\mathrm{ev}_{\mathcal{U}} : (\mathrm{S}^1 \to \mathcal{U}) \to \sum_{X:\mathcal{U}} X = X$ maps the type family $\mathrm{P}_\bullet$ to the pair $(\bullet = \bullet, q \mapsto \circlearrowleft \cdot q)$, which can be identified with (Z, s) through Corollary 3.4.5. Hence, $\mathrm{ev}_{\mathcal{U}}$ restricts to an equivalence between the connected component of $\mathrm{P}_\bullet$ in $\mathrm{S}^1 \to \mathcal{U}$ and the connected component of (Z, s) in $\sum_{X:\mathcal{U}} X = X$. We claim that we get a commuting diagram

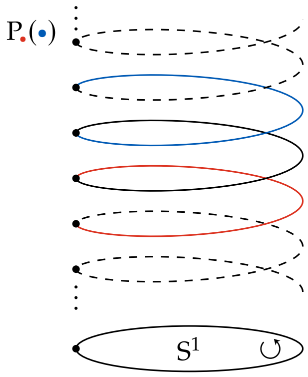


FIGURE 3.5: For the fiber of the universal set bundle, $\mathrm{P}_\bullet(\bullet) \equiv (\bullet = \bullet)$, we *increase* the winding number when we transport the endpoint (in blue) along $\circlearrowleft$, and we *decrease* it when we transport the starting point (in red) in the same way.

fig:plus-minus-one

(3.5.1)
$$\begin{tikzcd}
 & \mathrm{S}^1 \arrow[dl, "\mathrm{cst}_\_"'] \arrow[d, "\mathrm{P}_\_"] \arrow[dr, "c"] & \\
\mathrm{SetBundle}(\mathrm{S}^1)_{(\mathrm{cst}_\bullet)} \arrow[r, "\sim"] & (\mathrm{S}^1 \to \mathcal{U})_{(\mathrm{P}_\bullet)} \arrow[r, "\sim", "\mathrm{ev}_{\mathcal{U}}"'] & \mathrm{InfCyc},
\end{tikzcd}$$

{eq:setbundle-Sc-univ-comp}

where the left-most diagonal arrow maps $z : \mathrm{S}^1$ to the constant map $\mathrm{cst}_z : \mathbb{1} \to \mathrm{S}^1$. The left-hand triangle commutes, because the fiber $\sum_{\_:\mathbb{1}}(x = z)$ of cst_z at $x : \mathrm{S}^1$ is equivalent to $\mathrm{P}_z(x) \equiv (z = x)$. We prove that the right-hand triangle commutes by circle induction. That is, we show $\prod_{z:\mathrm{S}^1} c(z) = \mathrm{ev}_{\mathcal{U}}(\mathrm{P}_z)$. The case $z \equiv \bullet$ is exactly the equivalence $g(\bullet) \equiv \circlearrowleft^- : \mathrm{Z} \to \mathrm{P}_\bullet(\bullet)$ of Theorem 3.4.4 together with the fact that $\mathrm{trp}^{\mathrm{P}_\bullet}_{\circlearrowleft}$ corresponds to s. To finish, we observe that it doesn't matter which way you take in the diagram

$$\begin{tikzcd}
\mathrm{Z} \arrow[r, "\circlearrowleft^-"] \arrow[d, "\mathrm{s}^{-1}"] & (\bullet = \bullet) \arrow[d, "\_\cdot\circlearrowleft^{-1}"] \\
\mathrm{Z} \arrow[r, "\circlearrowleft^-"] & (\bullet = \bullet).
\end{tikzcd}$$

Note that to transport in the family $\mathrm{P}_\_(\bullet) \equiv (\_ = \bullet)$, we use Exercise 2.14.4(3), and *that* is why we picked the predecessor equivalence in Definition 3.5.2. This is also illustrated in Fig. 3.5.[17]

With (3.5.1) in hand, we see that c is an equivalence if and only if either of the two other downward maps are.[18] It is very direct to show that the map on the left is an equivalence. Indeed, the identity type $(\mathbb{1}, \mathrm{cst}_x) = (\mathbb{1}, \mathrm{cst}_y)$ is equivalent to pairs of an equivalence $e : \mathbb{1} \to \mathbb{1}$ and a commuting triangle

$$\begin{tikzcd}
\mathbb{1} \arrow[rr, "e"] \arrow[dr, "\mathrm{cst}_x"'] & & \mathbb{1} \arrow[dl, "\mathrm{cst}_y"] \\
 & \mathrm{S}^1. &
\end{tikzcd}$$

But since $\mathbb{1}$ is contractible, this just amounts to the equality $x = y$. Hence the map is an embedding, and we conclude by Corollary 2.17.10(3).

[17] Another option would have been to choose the opposite equivalence $\mathrm{Z} \simeq \mathrm{P}_\bullet(\bullet)$, sending n to $\circlearrowleft^{-n}$, in the base case. The point is: You can move the minus sign around, but it has to pop up somewhere.

[18] At this point we could conclude with an appeal to the type theoretic Yoneda lemma, which states that the map $X \to (X \to \mathcal{U})$, sending x to the family $y \mapsto x = y$, is an injection for any type X. Exercise: Prove this!

We now give the second, more direct, proof that c is an equivalence. For this we use the following lemma, which is of independent interest.

lem:conn-eq-f-ap-f-x

Lemma 3.5.3. *Let X and Y be connected types, x an element of X, and f a function from X to Y. Then f is an equivalence if and only if* $\mathrm{ap}_f : (x = x) \to (f(x) = f(x))$ *is an equivalence.*

Proof. Using Corollary 2.17.10(3) it suffices to show that each map induced by f on identity types is an equivalence if and only if the specific map $\mathrm{ap}_f : (x = x) \to (f(x) = f(x))$ is an equivalence. Being an equivalence is a proposition, so the result follows in two easy steps from X being connected, using Exercise 2.16.4. □

thm:S1bysymmetries

Theorem 3.5.4. *The function* $c : \mathrm{S}^1 \to \mathrm{InfCyc}$ *from Definition 3.5.2 is an equivalence.*

Proof. In view of Lemma 3.5.3 we only need to show that $\mathrm{ap}_c : (\bullet =_{\mathrm{S}^1} \bullet) \to ((\mathrm{Z}, \mathrm{s}) = (\mathrm{Z}, \mathrm{s}))$ is an equivalence. Note that both the domain and the co-domain of ap_c are equivalent to Z. Consider the following diagram in which we compose c with the equivalences from Corollary 3.4.5 and Lemma 3.5.1:

$$\begin{tikzcd}
\mathrm{Z} \arrow[r, "\circlearrowleft^{-}"] & (\bullet = \bullet) \arrow[r, "\mathrm{ap}_c"] & \big((\mathrm{Z}, \mathrm{s}) = (\mathrm{Z}, \mathrm{s})\big) \arrow[r, "\mathrm{ev}_0"] & \mathrm{Z}
\end{tikzcd}$$

For c to be an equivalence, it suffices to show that the composition is an equivalence from Z to itself. By definition, $\mathrm{ap}_c(\circlearrowleft)$ is the identification corresponding to s^{-1}, sending 0 to -1, and by induction on $n : \mathrm{Z}$ it follows that $\mathrm{ev}_0(\mathrm{ap}_c(\circlearrowleft^n)) = \mathrm{s}^{-n}(0) = -n$. And the map $n \mapsto -n$ is indeed an equivalence. □

3.6 *Connected set bundles over the circle*

sec:covS1

Let A be a type and $f : A \to \mathrm{S}^1$ a function. By Corollary 2.17.10(2), f is a set bundle over S^1 if and only if each map induced by f on identity types is injective. Assume that $f : A \to \mathrm{S}^1$ is a set bundle with A connected. Let (a_0, p) be an element of $f^{-1}(\bullet)$. By Exercise 2.16.4 the condition that *each* ap_f is injective can be relaxed to $\mathrm{ap}_f : (a_0 = a_0) \to (f(a_0) = f(a_0))$ being injective. Now look at the following subset:

{eq:subgroup-covering-1}

$$\{\, q : \bullet =_{\mathrm{S}^1} \bullet \mid {\mathrm{ap}_f}^{-1}(pqp^{-1}) \,\}. \tag{3.6.1}$$

Clearly, a classification of connected set bundles over the circle also classifies certain subsets of symmetries of $\bullet$, or equivalently, using Corollary 3.4.5, certain subsets of Z. Such subsets of $(\bullet =_{\mathrm{S}^1} \bullet)$ are closed under concatenation and inverses, since ap_f is compatible with these operations, see Lemma 2.6.2. Using language yet to be introduced, we actually "classify the subgroups of the integers".

Recall that set bundles over the circle are equivalent to sets with permutations. Which sets with permutations (X, t) correspond to connected set bundles? It is not so surprising that the answer has to do with whether any two points $x, x' : X$ can be connected by applying t some number of times.

Recall that the iteration t^n makes sense for all integers n since t is an equivalence.

def:Cyc

DEFINITION 3.6.1. Let Cyc be the subtype of $\sum_{X:\mathcal{U}}(X \to X)$ of those pairs (X,t) where X is a *nonempty* set with an *equivalence* t such that for any $x, x' : X$ there merely exists some $n : \mathrm{Z}$ with $x' = t^n(x)$. Elements of Cyc are called *cycles*.[19] ⌟

thm:cycset-connS1cover

THEOREM 3.6.2. *Under the equivalence of Theorem 3.3.6, connected set bundles of the circle correspond to cycles.*

Proof. Consider a set X with permutation t. The corresponding family of sets is $E :\equiv \mathrm{ve}_{\mathcal{U}}(X, \bar{t}) : \mathrm{S}^1 \to \mathcal{U}$, so the corresponding set bundle over the circle is the first projection, $\mathrm{fst} : A \to \mathrm{S}^1$, where we put $A :\equiv \sum_{z:\mathrm{S}^1} E(z)$. We need to show that A is connected if and only if X is nonempty and any two elements of X can be connected by t.

We show something a little more general, namely we give a bijection $g : \|A\|_0 \to X/\!\sim$, from the set of components of A to the quotient set of X by the equivalence relation $\sim$ defined by $(x \sim x') :\equiv \exists_{n:\mathrm{Z}}(x' = t^n(x))$.[21]

We define g using the universal property of set truncation (Definition 2.22.4), pair induction, and circle induction. To define $g_0 : \prod_{z:\mathrm{S}^1}(E(z) \to X/\!\sim)$, we need $g_0(\bullet) :\equiv [\_] : X \to X/\!\sim$ and $g_0(\circlearrowleft) : g_0(\bullet) =^{E(\_)\to X/\sim}_{\circlearrowleft} g_0(\bullet)$, equivalent to $g_0(\bullet) =_{X \to X/\sim} g_0(\bullet)t$. The latter we get by function extensionality and Theorem 2.22.12, since $x \sim t(x)$ for any $x : X$.

The inverse of g, $h : (X/\!\sim) \to \|A\|_0$, is defined as the extension of $h_0 : X \to \|A\|_0$ with $h_0(x) :\equiv |(\bullet, x)|_0$. We just need to check that $h_0(x) = h_0(x')$, or equivalently, $\|(\bullet, x) =_A (\bullet, x')\|$, whenever $x \sim x'$. Since this is a proposition, if $x' = t^n(x)$ with $n : \mathrm{Z}$, we may use induction on n (positive and negative) together with the paths, $\overline{(\circlearrowleft, \mathrm{refl}_{f(x)})} : (\bullet, x) = (\bullet, t(x))$, to conclude.

It's easy to check that g and h are mutually inverse. □

In Fig. 3.6 we see the set bundle corresponding to the set $\{1,2,3,4,5\}$ with the permutation $1 \mapsto 2 \mapsto 3 \mapsto 1, 4 \mapsto 5 \mapsto 4$. There are two components, showing that the permutation splits into two cycles.

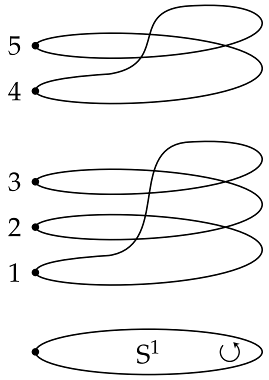


FIGURE 3.6: A set bundle with two components.

fig:two-comp-S1-cover

We already know one connected set bundle of the circle, namely the universal set bundle, which is also represented by the constant map $\mathrm{cst}_\bullet : \mathbb{1} \to \mathrm{S}^1$, and which we showed is equal to the exponential set bundle, which in turn corresponds to the infinite cycle (Z, s) consisting of the set of integers Z with the successor permutation.

We now introduce the remaining set bundles of the circle, first as functions to the circle, then as families of sets. Eventually we'll show – assuming a weak form of the Law of the Excluded Middle – that these (with the universal set bundle) are all the decidable connected set bundles over the circle.

def:mfoldS1cover

DEFINITION 3.6.3. For $m : \mathbb{N}$ positive, define the *degree m function* by circle induction

$$\delta_m : \mathrm{S}^1 \to \mathrm{S}^1, \ \text{ setting } \delta_m(\bullet) :\equiv \bullet \ \text{ and } \delta_m(\circlearrowleft) := \circlearrowleft^m. \qquad ⌟$$

On loops, the degree m function is the map $(\_)^m : (\bullet = \bullet) \to (\bullet = \bullet)$, which is indeed an injection for positive m, so δ_m is a set bundle corresponding to the subset of $(\bullet = \bullet)$ consisting of $\circlearrowleft^{mn} : \bullet = \bullet$ for all $n : \mathrm{Z}$.

As a subset of Z, this is simply all multiples of m.

Note that the degree 0 function would be constant, and hence not a set bundle since $(\_)^0 : (\bullet = \bullet) \to (\bullet = \bullet)$ is not injective.

[19] Our cycles are a special case of what is elsewhere called *cyclically ordered sets*, and they are closely related to the *cyclic sets* of Connes[20].

[20] Alain Connes. "Cohomologie cyclique et foncteurs Ext^n". In: *C. R. Acad. Sci. Paris Sér. I Math.* 296.23 (1983), pp. 953–958.

[21] Exercise: Check that this defines an equivalence relation, and that the bijection g proves the theorem.

Just as we in Section 3.4 gained a lot of insight into the universal set bundle, $\mathrm{cst}_\bullet : \mathbb{1} \to S^1$, by proving an equivalence with the exponential set bundle, in this section, we'll learn more about the degree m map, $\delta_m : S^1 \to S^1$, by constructing an equivalence with another concrete family.

Fix a positive number $m : \mathbb{N}$. Recall the finite set $\mathbb{m}$ from Definition 2.24.1 with elements denoted $0, 1, \dots, m-1$. Since $\mathbb{m} = \sum_{k:\mathbb{N}} k < m$ (Exercise 2.24.2), we may define a successor map $\mathrm{s} : \mathbb{m} \to \mathbb{m}$ by

$$\mathrm{s}(k) :\equiv \begin{cases} k+1 & \text{if } k < m-1, \\ 0 & \text{if } k = m-1. \end{cases}$$

Exercise 3.6.4. Show that $\mathrm{s} : \mathbb{m} \to \mathbb{m}$ is an equivalence by defining an explicit inverse. ⌟

Thus, $(\mathbb{m}, \mathrm{s})$ is another key example of a cycle. It is the standard finite m-element cycle, just as $(\mathbb{Z}, \mathrm{s})$ is the standard infinite cycle.

def:RmtoS1

Definition 3.6.5. Fix $m : \mathbb{N}$ positive. The set bundle $R_m : S^1 \to \mathrm{Set}$ corresponds to the standard m-cycle $(\mathbb{m}, \mathrm{s})$. We have $R_m(\bullet) :\equiv \mathbb{m}$ and $R_m(\circlearrowleft) := \bar{\mathrm{s}}$. We define

$$\tilde{R}_m :\equiv \sum_{z : S^1} R_m(z)$$

and let the first projection denoted by

$$\mathrm{pow}_m : \tilde{R}_m \to S^1$$

be the m^{th} *power bundle of the circle*. ⌟

Remark 3.6.6. The analogue of our degree m function is the m^{th} power of complex numbers restricted to the unit circle, mapping z to z^m if $|z| = 1$. If we parameterize the unit circle by the angle $\theta : \mathbb{R}$ (defined up to multiples of 2π), so $z = \mathrm{e}^{\theta \mathrm{i}}$, then $z^m = \mathrm{e}^{m\theta \mathrm{i}}$. Figure 3.7 illustrates the m^{th} power bundle as a family over the circle. Choosing any point z on the unit circle, we see that the preimage of z under the m^{th} power map is a shifted copy of the m different mth roots of unity inside the unit circle. ⌟

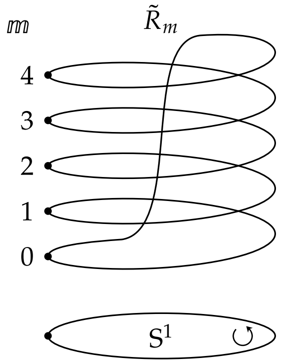


Figure 3.7: The m^{th} power bundle for $m = 5$.

To show that δ_m and pow_m are equal as bundles, it suffices to define an equivalence $\psi_m : \tilde{R}_m \to S^1$ and an element $\alpha_m : \delta_m \psi_m = \mathrm{pow}_m$, showing that the triangle below commutes.

$$\begin{tikzcd} \tilde{R}_m \arrow[rr, "\psi_m"] \arrow[dr, "\mathrm{pow}_m"'] & & S^1 \arrow[dl, "\delta_m"] \\ & S^1 & \end{tikzcd}$$

To see how to define ψ_m and α_m, we draw in Fig. 3.8 the type $\tilde{R}_m$ unrolled into a "clock", with marks $0, 1, \dots, m-1$ (the mark k is the element $(\bullet, k) : \tilde{R}_m$), and arcs following the successor permutation of $\mathbb{m}$. We denote these arcs by $a_k :\equiv \overline{(\circlearrowleft, \mathrm{refl}_{\mathrm{s}(k)})} : (\bullet, k) = (\bullet, \mathrm{s}(k))$. The m^{th} power map (which is just the first projection) sends each mark to $\bullet : S^1$ and each arc to $\circlearrowleft$.

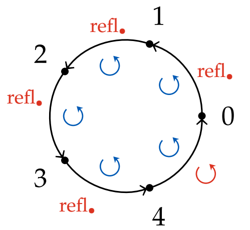


Figure 3.8: Unrolling $\tilde{R}_m$ as a "clock". (Here we're going around in a counterclockwise fashion as mathematicians are wont to do.)

fig:m-power-clock

This is indicated in blue on the inside of the clock. To define ψ_m, we must send all the marks to $\bullet : S^1$ and all arcs to $\mathrm{refl}_\bullet$, except one, which goes to $\circlearrowleft$. This is indicated in red on the outside of the clock.

con:psi-alpha-m

CONSTRUCTION 3.6.7. *For each positive integer m, there is an equivalence $\psi_m : \tilde{R}_m \to \mathrm{S}^1$ and an element $\alpha_m : \delta_m \psi_m = \mathrm{pow}_m$.*

Implementation of Construction 3.6.7. Since $\tilde{R}_m \equiv \sum_{z:\mathrm{S}^1} R_m(z)$, to define ψ_m we first split the argument into a pair (z, k). In a slight abuse of notation, we write $\psi_m : \prod_{z:\mathrm{S}^1}(R_m(z) \to \mathrm{S}^1)$ for the curried function as well. We define $\psi_m(z) : R_m(z) \to \mathrm{S}^1$ by circle induction on z. The base case is $\psi_m(\bullet) :\equiv \mathrm{cst}_\bullet : \mathbb{m} \to \mathrm{S}^1$, the constant function at $\bullet$. Since transport in a function type is by conjugation (Lemma 2.14.2), and the codomain type is constant, we need to give an identity $\psi_m(\circlearrowleft) : \psi_m(\bullet) =_{\mathbb{m}\to\mathrm{S}^1} \psi_m(\bullet)R_m(\circlearrowleft)$. We construct $\psi_m(\circlearrowleft)$ using function extensionality, by giving an element in $\mathbb{m} \to (\bullet = \bullet)$. Since ψ_m needs to send all arcs, except the last, in $\tilde{R}_m$ to reflexivity, we map k to $\mathrm{refl}_\bullet$ for $k < m - 1$, and we map $m - 1$ to $\circlearrowleft$.

The inverse of ψ_m maps $\bullet$ to $(\bullet, 0)$, i.e., the mark at 0, and $\circlearrowleft$ to $a_{m-1} \cdots a_0$, i.e., the product of all the arcs around the circle. We leave it as an exercise to prove that this really defines an inverse to ψ_m.

We likewise use function extensionality and pair and circle induction to define α, reducing the problem to giving (with a slight abuse of notation) $\alpha_m(\bullet, k) : \mathrm{pow}_m(\bullet, k) = \delta_m(\psi_m(\bullet, k))$ together with elements $\alpha_m(\circlearrowleft, k)$ witnessing that the two composites agree in the square

$$\begin{tikzcd}
\mathrm{pow}_m(\bullet, k) \arrow[r, equal, "\alpha_m(\bullet,k)"] \arrow[d, equal, "\mathrm{pow}_m(a_k)"'] & \delta_m(\psi_m(\bullet, k)) \arrow[d, equal, "\delta_m(\psi_m(a_k))"] \\
\mathrm{pow}_m(\bullet, \mathrm{s}(k)) \arrow[r, equal, "\alpha_m(\bullet,\mathrm{s}(k))"] & \delta_m(\psi_m(\bullet, \mathrm{s}(k))).
\end{tikzcd}$$

In Fig. 3.9 we show these m squares with the left and right hand sides simplified according to the definitions.

We see that we can pick $\alpha_m(\bullet, k) :\equiv \circlearrowleft^{-k}$, and then we can take for $\alpha_m(\circlearrowleft, k)$ the trivial proofs that $\mathrm{refl}_\bullet \circlearrowleft^{-k} = \circlearrowleft^{-(k+1)} \circlearrowleft$, for $k < m - 1$, and $\circlearrowleft^m \circlearrowleft^{-(m-1)} = \circlearrowleft^{-0} \circlearrowleft$, for $k = m - 1$. □

$$\begin{tikzcd}
\bullet \arrow[r, equal, "\alpha_m(\bullet,0)"] \arrow[d, equal, "\circlearrowleft"'] & \bullet \arrow[d, equal, "\mathrm{refl}_\bullet"] \\
\bullet \arrow[r, equal, "\alpha_m(\bullet,1)"] \arrow[d, equal, "\circlearrowleft"'] & \bullet \arrow[d, equal, "\mathrm{refl}_\bullet"] \\
\vdots \arrow[d, equal, "\circlearrowleft"'] & \vdots \arrow[d, equal, "\mathrm{refl}_\bullet"] \\
\bullet \arrow[r, equal, "\alpha_m(\bullet,m-1)"] \arrow[d, equal, "\circlearrowleft"'] & \bullet \arrow[d, equal, "\circlearrowleft^m"] \\
\bullet \arrow[r, equal, "\alpha_m(\bullet,0)"] & \bullet
\end{tikzcd}$$

FIGURE 3.9: The simplified types of the squares $\alpha_m(\circlearrowleft, k)$.

fig:psi-alpha-m-help

COROLLARY 3.6.8. *The degree m map $\delta_m : \mathrm{S}^1 \to \mathrm{S}^1$ is a connected set bundle for each positive integer m, and all the preimages $\delta_m^{-1}(z)$, $z : \mathrm{S}^1$, are m-element finite sets.*

We get an explicit equivalence $\mathbb{m} \simeq \delta_m^{-1}(\bullet)$ from ψ_m and α_m: send k to $(\bullet, \circlearrowleft^{-k})$, using the following exercise.

xca:preim-eq

EXERCISE 3.6.9. Let A, B, C be types and $f : A \to C$, $g : B \to C$ functions. Assume moreover we have an equivalence $e : A \to B$, a element of type $h : \prod_{x:A} f(x) = g(e(x))$, and an element $c : C$. Show that $(a, p) \mapsto (e(a), h(a)p)$ defines an equivalence $f^{-1}(c) \to g^{-1}(c)$. ⌟

Recall that our goal is to understand the *type* of connected set bundles over the circle. Since the type of set bundles is equivalent to $\mathrm{S}^1 \to \mathrm{Set}$, and Set is a groupoid (Lemma 2.22.1), Lemma 2.15.4(1) gives that the type of set bundles over the circle is a groupoid. We will pin this groupoid down by first analyzing the sets of identifications in it.

To do this, we generalize Lemma 3.5.1 to other kinds of cycles. However, since we're dealing with multiple components, it'll be useful to have a set labeling the components first.

def:subgroup-zet-of-cycle

DEFINITION 3.6.10. For any cycle (X, t), let $H_t :\equiv \{\, n : \mathrm{Z} \mid t^n = \mathrm{id} \,\} : \mathrm{Sub}_{\mathrm{Z}}$. ⌟

Thus, H_t is the subset of Z determined by the predicate $t^n = \mathrm{id}$ for $n : \mathrm{Z}$. Recall Corollary 2.25.5 implying that Sub_Z is a set.

lem:cycle-order-point-ap

LEMMA 3.6.11. *For any connected set bundle (A, f) with corresponding cycle (X, t) according to Theorem 3.6.2, if $x : X$, then $H_t = \{\, n : \mathrm{Z} \mid t^n(x) = x \,\}$, and for any $a : A$, we have that H_t also equals the image of the composite*

{eq:order-composite}

$$(a =_A a) \xrightarrow{\mathrm{ap}_f} (f(a) =_{\mathrm{S}^1} f(a)) \xrightarrow{\sim} \mathrm{Z}, \tag{3.6.2}$$

where the second map is the winding number function from Exercise 3.4.8.

Proof. We may suppose that the set bundle (A, f) over the circle has the form $(\sum_{z : \mathrm{S}^1} E(z), \mathrm{fst})$, where $E \equiv \mathrm{ve}_{\mathcal{U}}(X, \bar{t}) : \mathrm{S}^1 \to \mathcal{U}$ is the family corresponding to the cycle (X, t). To prove the proposition in the lemma quantifying over A, i.e., over $z : \mathrm{S}^1$ and $x : E(z)$, it suffices to consider the case $z \equiv \bullet$ and $x : X$, since the circle is connected.

For any point $x : X$, corresponding to the point $a :\equiv (\bullet, x) : A$, the type $(a =_A a)$ is equivalent to $\sum_{n : \mathrm{Z}} t^n(x) = x$ in such a way that the composite function (3.6.2) corresponds to the first projection. Hence the image of (3.6.2) is precisely $\{\, n : \mathrm{Z} \mid t^n(x) = x \,\}$.

It remains to show that $\{\, n : \mathrm{Z} \mid t^n(x) = x \,\} \subseteq H_t$ (the other inclusion being clear). So assume $t^n(x) = x$. Then if $x' : X$ is any other point, to prove the proposition $t^n(x') = x'$, we may assume we have $k : \mathrm{Z}$ with $x' = t^k(x)$. Then $t^n(x') = t^{n+k}(x) = t^k(x) = x'$, as desired. □

lem:IdCycle

LEMMA 3.6.12. *Let $(X, t) : \mathrm{Cyc}$ be a cycle with a chosen point $x_0 : X$. If (Y, u) is another cycle with $H_t =_{\mathrm{Sub}_\mathrm{Z}} H_u$, then it is in the same component as (X, t), and any equality $(e, !) : (X, t) = (Y, u)$ is uniquely determined by the element $e(x_0) : Y$. In other words, the function*

$$\mathrm{ev}_0 : \big((X, t) = (Y, u)\big) \to Y \;\; \textit{defined by} \;\; \mathrm{ev}_0(e, !) :\equiv e(x_0)$$

is an equivalence.

Proof. Assume $H_t = H_u$, i.e., $t^n(x) = x$ if and only if $u^n(y) = y$ for any $x : X$, $y : Y$ and $n : \mathrm{Z}$. Given $y_0 : Y$ we must determine a unique equivalence $e : X \to Y$ such that $et = ue$ and $e(x_0) = y_0$.

The key point is the following: For any $x : X$, there exists some $n : \mathrm{Z}$ with $x = t^n(x_0)$. For any such n, we must have

$$e(x) = e(t^n(x_0)) = u^n(e(x_0)) = u^n(y_0),$$

which shows uniqueness of e. For showing existence, we need to check that it doesn't matter which n we choose, if there are several. Technically, to use the proposition $\exists_{n : \mathrm{Z}}(x = t^n(x_0))$ to construct $e(x) : Y$, we consider instead the type $P_x :\equiv \sum_{y : Y} \prod_{n : \mathrm{Z}}(x = t^n(x_0) \to y = u^n(y_0))$, which we show to be a proposition. Note that P_x is a subtype of Y (the product part is a proposition since Y is a set), so we need to show that any two y, y' in P_x are equal. But this is clear, since there is *some* $n : \mathrm{Z}$ with $x = t^n(x_0)$, so $y = u^n(y_0) = y'$.

Now to prove the proposition P_x, we may assume we have $m : \mathrm{Z}$ such that $x = t^m(x_0)$. We let $y :\equiv u^m(y_0)$, and we need to show, for any $n : \mathrm{Z}$, that $x = t^n(x_0)$ implies $y = u^n(y_0)$. So now $t^m(x_0) = t^n(x_0)$ and we must show $u^m(y_0) = u^n(y_0)$. But this follows from our starting assumption, since the former is equivalent to $t^{m-n}(x_0) = x_0$ and the latter to $u^{m-n}(y_0) = y_0$.

It's easy to prove the proposition that this e is indeed an equivalence, so this is left to the reader. □

As a first consequence, we get the following for the type of loops at the standard m-cycles.

cor:id-m-cycle

Corollary 3.6.13. *Evaluation at* 0 *gives an equivalence* $\big((\mathbb{m}, \mathrm{s}) = (\mathbb{m}, \mathrm{s})\big) \simeq \mathbb{m}$ *under which reflexivity maps to* 0*, and composition with the equality* $\overline{(\mathrm{s}, !)} : ((\mathbb{m}, \mathrm{s}) = (\mathbb{m}, \mathrm{s}))$ *corresponds to the operation* $\mathrm{s} : \mathbb{m} \to \mathbb{m}$*.*

And as a second consequence, we get a more concrete description of the set of components of Cyc, and hence, by Theorem 3.6.2, of the type of connected set bundles of the circle.

cor:set-trunc-cyc

Corollary 3.6.14. *The map* $H : \mathrm{Cyc} \to \mathrm{Sub}_{\mathrm{Z}}$ *sending* (X, t) *to* H_t *induces an equivalence from* $\|\mathrm{Cyc}\|_0$ *onto the subset of* $\mathrm{Sub}_{\mathrm{Z}}$ *consisting of those subsets* $H \subseteq \mathrm{Z}$ *that contain* 0 *and are closed under addition and negation.*

By H being closed under addition and negation, we simply mean that if z, z' are in H, then so are $z + z'$ and $-z$.

Proof. The map H induces $g : \|\mathrm{Cyc}\|_0 \to \mathrm{Sub}_{\mathrm{Z}}$ by the universal property of set truncation, cf. Definition 2.22.4. From Lemma 3.6.12 we know that g is an injection, so it remains to prove that the image is as stated. It is clear that H_t, for a cycle (X, t), contains 0 and is closed under addition and negation. Conversely, suppose H contains 0 and is closed under addition and negation. Define the relation $\sim_H$ on Z by setting $z \sim_H z'$ if and only if the difference $z - z'$ is in H. This is an equivalence relation: it is reflexive since H contains 0, transitive since H is closed under addition, and symmetric since H is closed under negation. So let $X :\equiv \mathrm{Z}/{\sim_H}$, and define $t([z]) :\equiv [\mathrm{s}(z)]$ for $z : \mathrm{Z}$. This is well-defined, since $z \sim_H z'$ holds if and only if $\mathrm{s}(z) \sim_H \mathrm{s}(z')$. It is clear that (X, t) is a cycle with $H_t = H$. □

The components of Cyc will prop up many times from now on, so we make the following definitions to make it easier to talk about them.

def:Order

Definition 3.6.15. The type of *orders* is defined to be $\mathrm{Order} :\equiv \|\mathrm{Cyc}\|_0$. We say that the infinite cycle (Z, s) has *infinite order*, and the standard m-cycle $(\mathbb{m}, \mathrm{s})$ has *finite order* m, for positive $m : \mathbb{N}$.

We write $\mathrm{ord} :\equiv |\_|_0 : \mathrm{Cyc} \to \mathrm{Order}$ for the map from cycles to their orders, and we write $\mathrm{ord}(t) :\equiv \mathrm{ord}(X, t)$ for short.

We say that the order $d :\equiv \mathrm{ord}(X, t)$ *divides* the order $k :\equiv \mathrm{ord}(Y, u)$, written $d|k$, for cycles $(X, t), (Y, u)$, if $H_u \subseteq H_t$. ⌟

Note that we're still being cavalier with universe levels. Really, we should write $\mathrm{SetBundle}(\mathrm{S}^1)_{\mathcal{U}}$, $\mathrm{Cyc}_{\mathcal{U}}$, $\mathrm{Sub}^{\mathcal{U}}_{\mathrm{Z}}$, $\mathrm{Order}_{\mathcal{U}}$, etc., to indicate from which universe $\mathcal{U}$ we draw the types involved. We trust that the reader can fill these in if desired.

We have a canonical injection $\mathbb{N} \hookrightarrow \mathrm{Order}$, mapping 0 to the infinite order and each positive n to the finite order n. The orders in the image are called *principal*, and we don't make any notational distinction between a natural number d and the corresponding principal order. As a subset of Z, a principal order is simply $d\mathrm{Z}$, so we see that the divisibility relation on orders extends that on natural numbers.

The description in Corollary 3.6.14 is still not as concrete as we'd like. Is it true that any order is principal, in other words, that every cycle has either infinite order or finite order m for some positive $m : \mathbb{N}$? Any other textbook will tell you that the answer is yes, but the proof is unfortunately not constructive. It makes sense first to restrict to decidable set bundles/cycles.[22] Even so, we need one further non-constructive assumption, namely:

[22] This rules out certain pathological cycles, such as the subset $\{\, (\mathrm{e}^{2\pi\alpha \mathrm{i}})^n : \mathbb{C} \mid n : \mathrm{Z} \,\}$, with a suitable equivalence, e.g., incrementing the exponent. Here $\alpha : \mathbb{R}$ is an unknown real number, of which we don't know whether it is rational or not.

LPO

Principle 3.6.16 (Limited Principle of Omniscience). For any function $P : \mathbb{N} \to \mathbb{2}$, either there is a smallest number $n_0 : \mathbb{N}$ such that $P(n_0) = 1$, or P is a constant function with value 0. ⌟

The Limited Principle of Omniscience is weaker than the Law of Excluded Middle Principle 2.18.2, as we prove in the following lemma.[23]

Lemma 3.6.17. *The Law of Excluded Middle implies the Limited Principle of Omniscience.*

Proof. Let $P : \mathbb{N} \to \mathbb{2}$. By the Law of Excluded Middle, either P is constant 0, or there exists some $n : \mathbb{N}$ such that $P(n) = 1$. But in that case we may apply Construction 2.23.4 to conclude that there is a smallest $n_0 : \mathbb{N}$ such that $P(n_0) = 1$. □

xca:not-not-lpop

Exercise 3.6.18. Without using LEM or LPO, show that $(Q(P) \to \text{False}) \to \text{False}$ holds for every function $P : \mathbb{N} \to \mathbb{2}$, where $Q(P)$ is the proposition obtained by applying the Limited Principle of Omniscience to the function P. ⌟

As for the Law of Excluded Middle, we are free to assume the Limited Principle of Omniscience or not, and we will be explicit about where we will use it. The Limited Principle of Omniscience makes it possible to prove that the canonical map $\mathbb{N} \to \text{Order}^{\text{dec}}$ (the codomain being the subtype of Order given by decidable cycles), is an equivalence. We will elaborate this equivalence in the next paragraphs.

We already know from Corollary 3.6.14 that the map is an injection, and a cycle (X, t) has infinite order if and only if $H_t = \{0\}$,[24] and it has finite order m if and only if $H_t = m\mathrm{Z}$, for positive $m : \mathbb{N}$.

Fix now a decidable cycle (X, t), and consider the corresponding subset $H :\equiv H_t \equiv \{\, n : \mathrm{Z} \mid f^n = \text{id} \,\}$. This is a decidable subset, since $f^n = \text{id}$ is a proposition, and n is in H if and only if $f^n(x) = x$ for some/all $x : X$ (recall that X is non-empty).

Apply the Limited Principle of Omniscience (Principle 3.6.16) to the function $P : \mathbb{N} \to \mathbb{2}$ defined by $P(n) = 1$ if $n + 1$ is in H, and $P(n) = 0$ otherwise. If $P(n)$ is constant 0, then $H = \{0\}$, so (X, f) has infinite order. (As a set bundle, it is then merely equivalent to the universal set bundle.)

Otherwise, if n_0 is the smallest natural number with $m :\equiv n_0 + 1$ in H, then we claim $H = m\mathrm{Z}$, from which it follows that (X, t) has order m.

Clearly, $m\mathrm{Z} \subseteq H$, since if $t^m = \text{id}$, then so is $t^{nm} = \text{id}$. And if $t^q = \text{id}$, then by Euclidean division of integers, cf. Lemma 2.23.8, there exist $k : \mathbb{Z}$, $r : \mathbb{N}$ with $r < m$ so that $q = km + r$. Now, the number r is in H, since $t^r = t^{q-km} = \text{id}$, and is less than the minimal positive value m in H, and so we must conclude that $r = 0$. In other words, q is a multiple km, as desired.

We summarize these results in the following lemma.

lem:componentsofcoversofS1

Lemma 3.6.19. *Assuming the Limited Principle of Omniscience* (*Principle* 3.6.16), *the type of connected decidable set bundles over the circle is the sum of the component containing the universal set bundle and for each positive integer m, the component containing the m-fold set bundle.*

rem:flipthecircle

Remark 3.6.20. The reader may wonder how the "orientation reversing" map $r : \mathrm{S}^1 \to \mathrm{S}^1$ given by $r(\bullet) = \bullet$ and $r(\circlearrowleft) = \circlearrowleft^{-1}$ fits into the picture.[25] As connected decidable set bundles, we have $(\mathrm{S}^1, r) = (\mathrm{S}^1, \text{id})$, since r is

[23] It is also the case that the Limited Principle of Omniscience does not imply the Law of Excluded Middle, because a model that satisfies the Limited Principle of Omniscience but not the Law of Excluded Middle can be built using sheaves over the real line $\mathbb{R}$.

Nevertheless, the Limited Principle of Omniscience is not constructive, for otherwise we could simply decide the truth of every open problem in mathematics that can (equivalently) be expressed by a function $P : \mathbb{N} \to \mathbb{2}$ being constant with value 0. This type of argument was first given by Brouwer.

Here we give an example based on the famous Goldbach conjecture, which states that every even integer greater than 2 is the sum of two primes. Using that the latter two primes are necessarily smaller than the even integer itself, it is possible to (equivalently) express the truth of the Goldbach conjecture by a function $P : \mathbb{N} \to \mathbb{2}$ being constantly 0. Now assume we have a proof t of the Limited Principle of Omniscience in type theory, not using any axioms. Then $t(P)$ is an element of the sum type $L \amalg R$, where R expresses that the function P is constantly 0, and L implies the negation of R. By the computational properties of type theory one can compute the *canonical form* of $t(P)$, which is either inr_r for some element $r : R$, or inl_l for some element $l : L$. If $t(P) \equiv \text{inr}_r$ the Goldbach conjecture is true, and if $t(P) \equiv \text{inl}_l$ the Goldbach conjecture is false. Thus the Goldbach conjecture would be solved, and therefore it is unlikely that t exists. In the appendix [ref] we give a longer but decisive argument against the constructivity of the Limited Principle of Omniscience.

[24] This is why it's natural to associate to $0 : \mathbb{N}$ the infinite order.

[25] As an operation on infinite cycles, see Definition 3.5.2, $crc^{-1} : \text{InfCyc} \to \text{InfCyc}$ maps (X, t) to (X, t^{-1}), flipping the arrows.

an equivalence:

$$\begin{tikzcd} S^1 \arrow[rr, equal, "\bar{r}"] \arrow[rr, phantom, "\Longrightarrow"'] \arrow[dr, "r"'] & & S^1 \arrow[dl, "\mathrm{id}"] \\ & S^1 & \end{tikzcd}$$

This is a special case of the general case of an equivalence $e : A \to A'$ depicted in the diagram in the margin, implying $(A, fe, !) = (A', f, !)$. The point is that the degree m and degree $-m$ maps give the same *bundles* (by composing with r), while as *maps* they are different. ⌟

$$\begin{tikzcd} A \arrow[rr, equal, "\bar{e}"] \arrow[rr, phantom, "\Longrightarrow"'] \arrow[dr, "fe"'] & & A' \arrow[dl, "f"] \\ & C & \end{tikzcd}$$

3.7 *The m^{th} root: set bundles over the components of* Cyc

Recall the equivalence $c : S^1 \xrightarrow{\sim} \mathrm{Cyc}_0$ of Definition 3.5.2 between the circle and the type of infinite cycles. Here we set $\mathrm{Cyc}_0 :\equiv \mathrm{InfCyc}$.

In this section, we reinterpret the degree m function δ_m as a map of infinite cycles. In fact it makes sense as a map on all cycles, and we'll use it to begin the classification of the connected set bundles on the components Cyc_n, of Cyc, determined by the standard n-cycles, for positive integers n. That's why it's instructive to rephrase connected set bundles over S^1 in terms of cycles, even though they could just be transported along the identity $\bar{c} : S^1 = \mathrm{Cyc}_0$ corresponding to c.

Before we do the degree m maps, let's note that the universal set bundle over Cyc_0 is represented by the constant function $\mathrm{cst}_{\mathrm{pt}_0} : \mathbb{1} \to \mathrm{Cyc}_0$, sending the unique element of $\mathbb{1}$ to $\mathrm{pt}_0 :\equiv (\mathrm{Z}, \mathrm{s}) : \mathrm{Cyc}_0$, the standard infinite cycle.[26]

For the rest of this section, we fix some positive $m : \mathbb{N}$. We now give a description of the m-fold set bundle over the circle in terms of cycles.

We proceed as follows. First we present the answer, a set bundle we call $\rho_m : \mathrm{Cyc}_0 \to \mathrm{Cyc}_0$, and then we prove that $\delta_m : S^1 \to S^1$ and $\rho_m : \mathrm{Cyc}_0 \to \mathrm{Cyc}_0$ correspond to each other (and to $\mathrm{pow}_m : \tilde{R}_m \to S^1$) under the equivalence $c : S^1 \xrightarrow{\sim} \mathrm{Cyc}_0$.

What should we require of $\rho_m(X, t)$ for $(X, t) : \mathrm{Cyc}_0$? Well, $\delta_m : S^1 \to S^1$ sends $\bullet$ to $\bullet$ and $\circlearrowright$ to $\circlearrowright^m$; only the $\circlearrowright^k$ where k is a multiple of k is in the image of δ_m. So we have to find an infinite cycle (Y, u) with "u^m corresponding to t". We achieve this by "streching" X: Let Y be m copies of X and let u jump idly from one copy to another except every m^{th} time when u also is allowed to use t. This is illustrated in Fig. 3.10 with the shift by t being vertical and the movement from copy to copy going around a circle.

con:root

Construction 3.7.1. *For any type X and $t : X \to X$, we define the m^{th}* root

$$\sqrt[m]{t} : \mathbbm{m} \times X \to \mathbbm{m} \times X.$$

Implementation of Construction 3.7.1. We set

$$\sqrt[m]{t}(k, x) :\equiv \begin{cases} (k+1, x) & \text{for } k < m-1 \text{ and} \\ (0, t(x)) & \text{for } k = m-1. \end{cases}$$ □

Only one m^{th} of the time does $\sqrt[m]{t}$ use $t : X \to X$, the rest of the time it applies the successor in $\mathbbm{m}$. Indeed, iterating $\sqrt[m]{t}$ we get $(\sqrt[m]{t})^m(k, x) = (k, t(x))$; hence the term "$m^{\text{th}}$ root" is apt.

$$\sqrt[m]{t} : \mathbbm{m} \times X \to \mathbbm{m} \times X$$

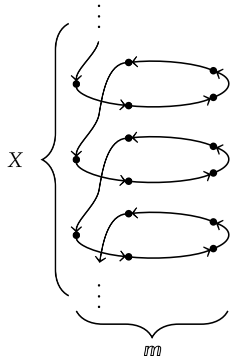


Figure 3.10: The m^{th} root $\sqrt[m]{t}$ of a function $t : X \to X$, here illustrated in the case $m = 5$.

fig:root

[26] In light of Lemma 3.5.1 we see that the fiber of this universal covering over $(X, t) : \mathrm{Cyc}_0$ is (equivalent to) X itself– that's certainly a universal set associated to the infinite cycle (X, t)!

Definition 3.7.2. The *formal m^{th} root function* is defined by:

$$\rho_m : \sum_{X:\mathcal{U}} (X \to X) \to \sum_{X:\mathcal{U}} (X \to X), \qquad \rho_m(X,t) :\equiv (\mathbb{m} \times X,\ \sqrt[m]{t}). \quad \lrcorner$$

We use ρ for "root" to denote this incarnation of the degree m function.

lem:root-pres-equiv

Lemma 3.7.3. *If $t : X \to X$ is an equivalence, then so is $\sqrt[m]{t} : \mathbb{m} \times X \to \mathbb{m} \times X$.*

Proof. On one hand, an element in $(\sqrt[m]{t})(\ell, y) = (0, x)$ consists of the assertion that $\ell = m - 1$ and an element in $t(y) = x$, so $(\sqrt[m]{t})^{-1}(0, x)$ is equivalent to $t^{-1}(x)$, which is contractible if t is an equivalence.

On the other, if $k : \mathbb{m}$ is not 0, then an element in $(\sqrt[m]{t})(\ell, y) = (k, x)$ consists of the assertion that $\ell + 1 = k$ and an element in $y = x$, and so $(\sqrt[m]{t})^{-1}(k, x)$ is equivalent to the contractible type $\sum_{y:X} y = x$. □

Of course, it's also quite easy to write down an inverse of $\sqrt[m]{t}$ given an inverse of t.

Lemma 3.7.4. *If (X, t) is a cycle, then so is $\rho_m(X, t)$.*

Proof. Clearly, $\mathbb{m} \times X$ is nonempty if X is. And we already know $\sqrt[m]{t}$ is an equivalence if t is.

Suppose $(k, x), (k', x') : \mathbb{m} \times X$. We need to show the proposition that there merely exists $n : \mathrm{Z}$ with $(k', x') = \left(\sqrt[m]{t}\right)^n (k, x)$. Let $n : \mathrm{Z}$ be such that $x' = t^n(x)$. Then $(\sqrt[m]{t})^{nm}(k, x) = (k, t^n(x)) = (k, x')$, so if $k = k'$ we're done. Assume $k < k'$. Then $(\sqrt[m]{t})^{k'-k}(k, x') = (k', x')$, so $(\sqrt[m]{t})^{nm+k'-k}(k, x) = (k', x')$, as desired. The case $k > k'$ is similar. □

The question now arises: how does ρ_m act on the components of Cyc, and what can we say about the preimages $\rho_m^{-1}(X, t)$ for an arbitrary cycle (X, t)?

The first part is easy, since the product of $\mathbb{m}$ with an n-element set is an mn-element set. We set $\mathrm{pt}_n :\equiv (\mathbb{n}, \mathrm{s}) : \mathrm{Cyc}_n$.

Lemma 3.7.5. *The degree m function restricts to give pointed maps*

$$\rho_m : \mathrm{Cyc}_n \to_* \mathrm{Cyc}_{mn} \quad \text{and} \quad \rho_m : \mathrm{Cyc}_0 \to_* \mathrm{Cyc}_0.$$

Proof. Note that the function $\varphi : (\mathbb{m} \times \mathrm{Z}) \to \mathrm{Z}$ given by $\varphi(k, r) = k + mr$ is an equivalence, with inverse given by Euclidean division by m. Moreover, we have $\varphi \sqrt[m]{\mathrm{s}} = \mathrm{s}\, \varphi$, since

In terms of iterated addition, we have $\varphi(k, r) = (z \mapsto z + m)^r(k)$.

$$\varphi\left(\sqrt[m]{\mathrm{s}}(k, r)\right) = k + 1 + mr = \mathrm{s}(\varphi(k, r)) \quad \text{for all } (k, r) : \mathbb{m} \times \mathrm{Z}.$$

This shows that φ gives an identification of infinite cycles $(\mathbb{m} \times \mathrm{Z},\ \sqrt[m]{\mathrm{s}}) = (\mathrm{Z}, \mathrm{s})$, and hence the m^{th} root construction maps the component Cyc_0 to itself.

Analogously, we can restrict φ to an equivalence $\mathbb{m} \times \mathbb{n} \xrightarrow{\sim} \sum_{k:\mathbb{N}}(k < mn)$, and get an identification of cycles $\rho_m(\mathrm{pt}_n) = \mathrm{pt}_{mn}$, showing that ρ_m maps the component Cyc_n to the component Cyc_{mn}. □

We now analyze how ρ_m acts on paths. Let $\overline{(\bar{e}, !)} : (X, t) = (X', t')$. Since ρ_m maps first components X to $\mathbb{m} \times X$, we get that the first projection of $\mathrm{ap}_{\rho_m} \overline{(\bar{e}, !)}$ is $\overline{\mathrm{id} \times e} : (\mathbb{m} \times X) = (\mathbb{m} \times X')$. We are particularly interested in the case of the loops, that is, $\overline{(\bar{e}, !)} : (X, t) = (X, t)$. We calculate $(\mathrm{id} \times e)(k, x) = (k, e(x))$, which by the property of the m^{th} root is equal to $(\sqrt[m]{e})^m(k, x)$. In particular, if we take $e :\equiv t^{-1}$, then we get $(\mathrm{id} \times t^{-1}) = (\sqrt[m]{t^{-1}})^m$, which means that $\mathrm{ap}_{\rho_m} \overline{(\overline{t^{-1}}, !)}$ is indeed the m^{th} power of a generating loop at the image cycle $\rho_m(X, t)$. In particular, this holds for the standard infinite cycle $(\mathrm{Z}, \mathrm{s}) : \mathrm{Cyc}_0$ and the standard n-cycle $(\mathbb{m}, \mathrm{s}) : \mathrm{Cyc}_n$.

Why does $\rho_m : \mathrm{Cyc}_0 \to \mathrm{Cyc}_0$ correspond to the m-fold set bundle we defined in Definition 3.6.3? This is encapsulated by the fact that under the equivalence $c : \mathrm{S}^1 \to C$, the two m-fold covers agree in the sense that the two functions given as composites in

$$\begin{tikzcd}
\mathrm{S}^1 \arrow[r, "c"] \arrow[d, "\delta_m"'] & \mathrm{Cyc}_0 \arrow[d, "\rho_m"] \\
\mathrm{S}^1 \arrow[r, "c"] & \mathrm{Cyc}_0
\end{tikzcd}$$

are equal; we need an element in

$$\rho_m c =_{\mathrm{S}^1 \to \mathrm{Cyc}_0} c\, \delta_m.$$

Under the equivalence

$$\mathrm{ev}_{\mathrm{Cyc}_0} : (\mathrm{S}^1 \to \mathrm{Cyc}_0) \xrightarrow{\simeq} \sum_{(X,t):\mathrm{Cyc}_0} \big((X,t) = (X,t)\big)$$

of Theorem 3.1.2, the composite $c\,\delta_m$ is given by $\big((\mathrm{Z}, \mathrm{s}), \mathrm{s}^{-m}\big)$ and the composite $\rho_m c$ is given by $\big((\mathbb{m} \times \mathrm{Z}, \sqrt[m]{\mathrm{s}}), \mathrm{id} \times \mathrm{s}^{-1}\big)$: we must produce an element in

$$\Big((\mathbb{m} \times \mathrm{Z}, \sqrt[m]{\mathrm{s}}), \mathrm{id} \times \mathrm{s}^{-1}\Big) = \big((\mathrm{Z}, \mathrm{s}), \mathrm{s}^{-m}\big).$$

Consider the equivalence $\varphi : (\mathbb{m} \times \mathrm{Z}) \to \mathrm{Z}$ with $\varphi(k, n) = k + mn$ discussed above. Transport of $\sqrt[m]{\mathrm{s}}$ along φ is exactly s. (I.e., $\varphi \sqrt[m]{\mathrm{s}} = \mathrm{s}\, \varphi$; note the way we formulate this so that we don't need to talk about the inverse of φ[27].) Likewise, transport of $\mathrm{id} \times \mathrm{s}^{-1}$ along φ is s^{-m}, so that φ lifts to an element in $\big((\mathbb{m} \times \mathrm{Z}, \sqrt[m]{\mathrm{s}}), \mathrm{id} \times \mathrm{s}^{-1}\big) = \big((\mathrm{Z}, \mathrm{s}), \mathrm{s}^{-m}\big)$.

xca:pointed-maps-circle

Exercise 3.7.6. Verify $\rho_m c =_{\mathrm{S}^1 \to_* \mathrm{Cyc}_0} c\, \delta_m$ in case all maps are taken to be pointed. ⌟

So we know that the fiber of ρ_m at an infinite cycle (X, t) is an m-element set. In fact, we can identify this set as $X/m :\equiv X/{\sim_m}$ where $\sim_m$ is the equivalence relation that identifies points that are a distance mr apart, for some $r : \mathrm{Z}$. Formally, let $x \sim_m x'$ if and only if $\exists_{r:\mathrm{Z}}(x' = t^{mr}(x))$. (Such an r is unique if it exists.) Indeed, the fiber is

$$\sum_{(Y,u):\mathrm{Cyc}_0} \big((X,t) = (\mathbb{m} \times Y, \sqrt[m]{u})\big).$$

The equivalence is obtained by sending an equivalence class Y of X/m to the corresponding infinite cycle (Y, u^m) together with the natural identification $(X, t) = (\mathbb{m} \times Y, \sqrt[m]{u^m})$. See Theorem 3.7.10 below for a careful proof of a more general statement.

The reader will no doubt have noticed that X/m is a *finite cycle*. We'll return to the significance of this below.

Our next step is to identify the fiber of ρ_m over a general cycle (X, t). Classically, the remaining cases are those of finite n-cycles, but it's illuminating to be a bit more general. Note that the equivalence relation $\sim_m$ defined above for an infinite cycle makes sense for all cycles.

lem:sum-cycle-point-contr

Lemma 3.7.7. *For any order $d : \mathrm{Order}$, the type $\sum_{(X,t):\mathrm{Cyc}_d} X$ is contractible, where Cyc_d denotes the component of Cyc consisting of cycles of order d.*

[27] Of course, the inverse of φ maps $z : \mathrm{Z}$ to the remainder and the integer quotient of z under Euclidean division by m, cf. Lemma 2.23.8.

Proof. This is relatively straight-forward from Lemma 3.6.12. The type in question is nonempty since all cycles have a nonempty underlying set, so it suffices to prove the type is a proposition. So let $(X,t),(X',t')$ be cycles of order o, and take $x:X$ and $x':X$. An identification $((X,t),x) = ((X',t'),x')$ is given by an equivalence of cycles $e:(X,t)=(X',t')$ with $e(x)=x'$. But evaluation at x induces an equivalence $\big((X,t)=(X',t')\big) \simeq X'$, so there exists a unique e with $e(x)=x'$. □

lem:m-root-id

LEMMA 3.7.8. *For any cycle (X,t), if $(\sqrt[m]{t})^n = \mathrm{id}$, then m divides n, i.e., $n = mq$ for some $q:\mathbb{Z}$, and $t^q = \mathrm{id}$. In other words, m divides the order of $\sqrt[m]{t}$.*

This follows simply by looking at the first component, where $\sqrt[m]{t}$ acts as the successor operation on $\mathbb{m}$.

We're almost ready to identify the fiber of ρ_m at a cycle (X,t). We know from Lemma 3.7.8 that the fiber is nonempty only if m divides the order of t. A key ingredient for the converse is the following.

lem:X-mod-m-chosen

LEMMA 3.7.9. *Let $(X,t):\mathrm{Cyc}$ be a cycle with a chosen point $x_0:X$ and with order divisible by m. Then the map $f:\mathbb{m}\to X/m$, $f(k):\equiv [t^k(x_0)]$ is an equivalence.*

Proof. Fix an equivalence class $V:X/m$ and consider its preimage under f, $f^{-1}(V) \equiv \sum_{k:\mathbb{m}}(V = [t^k(x_0)])$. The contractibility of this type is a proposition, so we may choose $x:X$ with $V=[x]$. Then $(V = [t^k(x_0)]) \simeq ([x]=[t^k(x_0)]) \simeq (x \sim_m t^k(x_0))$. So we need to show that $\sum_{k:\mathbb{m}}(x \sim_m t^k(x_0))$ is contractible. More simply, we need to show that there is a unique k with $x \sim_m t^k(x_0)$. Since (X,t) is a cycle, we may further choose $n:\mathbb{Z}$ with $x = t^n(x_0)$. By Euclidean division, write $n = qm+r$ with $q:\mathbb{Z}$, $r:\mathbb{m}$. Then $x = t^n(x_0) \sim_m t^r(x_0)$, so we have our center. Let $k:\mathbb{m}$ also satisfy $x \sim_m t^k(x_0)$. We need to show the proposition $k=r$. But $t^{r-k}(x_0)\sim_m x_0$, so we may take $q:\mathbb{Z}$ with $t^{qm+r-k}(x_0)=x_0$. Since m divides the order of t, this implies $r=k$, as desired. □

Now we have all the pieces needed to prove the main result.

thm:fiber-cdg

THEOREM 3.7.10. *For any cycle (X,t), the preimage $\rho_m^{-1}(X,t)$ is equivalent to $P\times X/m$, where $P :\equiv (m \mid \mathrm{ord}(t)) \equiv (H_t \subseteq m\mathbb{Z})$ expresses that m divides the order of t.*

Proof. We'll use Lemma 2.9.9, and we first define the function

$$g:\rho_m^{-1}(X,t)\to P\times X/m,$$

by mapping (Y,u) and an identification of cycles $e:(X,t)=(\mathbb{m}\times Y,\sqrt[m]{u})$ to the proof of P from Lemma 3.7.8 and the class $V_e :\equiv [e^{-1}(0,y)]:X/m$, for any $y:Y$. Note that this doesn't depend on y, so that Theorem 2.22.8 applies. As a subset of X, $V_e = \{\, x:X \mid \mathrm{fst}(e(x)) = 0\,\}$.

In the other direction, to define the function

$$h:P\times X/m\to\rho_m^{-1}(X,t),$$

fix an equivalence class $V:X/m$, and assume that m divides the order of t. Then we have, with a bit of abuse of notation, the cycle (V,t^m), where we also write V for the type of elements in X that lie in the class V, and t^m is the restriction of this power of t to V.[28] We also need an identification $(X,t) = (\mathbb{m}\times V,\sqrt[m]{t^m})$. This we define via a map

[28] If x lies in V, then so does $t^m(x)$.

$e : \mathbb{m} \times V \to X$, $e(k, x) :\equiv t^k(x)$. This is an equivalence as long as the orders match. So let $n : \mathbb{Z}$, and assume first that $t^n = \mathrm{id}$. Then P implies that we may write $n = qm$ for some $q : \mathbb{Z}$, so

$$(\sqrt[m]{t^m})^n = (\sqrt[m]{t^m})^{qm} = (\mathrm{id} \times t^m)^q = (\mathrm{id} \times t^{qm}) = \mathrm{id}.$$

Conversely, we know from Lemma 3.7.8 again, that if $(\sqrt[m]{t^m})^n = \mathrm{id}$, then we may write $n = qm$ for some $q : \mathbb{Z}$, and $(t^m)^q = \mathrm{id}$, which by Lemma 3.6.11 implies that $t^n = \mathrm{id}$.

Straight from these definitions, we see that $g \circ h = \mathrm{id}$. We leave to the reader to check that $h \circ g = \mathrm{id}$. □

3.8 *Getting our cycles in order*

sec:cycles-order

TODO: Exposition and figures

Exercise 3.8.1. Prove that if $(X, t), (Y, u)$ are cycles, $x_0 : X$, then the type of maps $f : (X, t) \to (Y, u)$ is equivalent to $P \times Y$, where $P :\equiv (\mathrm{ord}(u) \mid \mathrm{ord}(t)) \equiv (H_t \subseteq H_u)$. ⌟

Thus, an order p divides an order q if and only if there is a map of cycles from a cycle of order q to a cycle of order p.

Theorem 3.8.2. *The partially ordered set* $(\mathrm{Order}, |)$ *is a lattice with least element the finite order* 1 *and greatest element the infinite order, represented by the number* 0*, and meets and joins given by "gcd" and "lcm", respectively.*

subgroups of $\mathrm{C}_n = \mathrm{C}_k$ where $k|n$ connected set bundles of Cyc_n

3.8.3 *More TODO*

- Classify connected set bundles over Cyc_n.
- Universal property of Cyc_n among groupoids.
- Bijective proof of $mn = \mathrm{lcm}(m, n) \times \mathrm{gcd}(m, n)$ via the product of cycles. Chinese remainder stuff.
- Somehow sneak in totatives and automorphisms of cyclic groups?

3.8.4 *Universal property of* Cyc_n

This section is devoted to showing that maps out of Cyc_n into a groupoid A are equivalently given by the choice of a point together with a symmetry of order n: that is any map $\mathrm{Cyc}_n \to A$ is fully determined by a point a together with a symmetry $\sigma : a = a$ such that $\sigma^n = \mathrm{refl}_a$.[29]

Recall that Cyc_n contains the point $\mathrm{pt}_n :\equiv (\mathbb{m}, \mathrm{s})$. This point has a symmetry $\sigma_n :\equiv (\mathrm{s}^{-1}, !)$ whose second projection is a proof that $\mathrm{s}\,\mathrm{s}^{-1} = \mathrm{s}^{-1}\,\mathrm{s}$. Recall also from Corollary 3.6.13 that all elements of $\mathrm{pt}_n = \mathrm{pt}_n$ are of the form σ_n^i for $i = 0, \ldots, n-1$.

Given a groupoid A, and a map $f : \mathrm{Cyc}_n \to A$, one can consider $f(\mathrm{pt}_n)$ and $\mathrm{ap}_f(\sigma_n) : f(\mathrm{pt}_n) = f(\mathrm{pt}_n)$. The equation $\mathrm{refl}_{\mathrm{pt}_n} = \sigma_n^n$ in Cyc_n is mapping by f to a proof of $\mathrm{refl}_{f(\mathrm{pt}_n)} = \mathrm{ap}_f(\sigma_n)^n$. Hence, the following map is well-defined:

$$\mathrm{ev}_{n,A} : (\mathrm{Cyc}_n \to A) \to \sum_{a : A} \sum_{\sigma : a = a} \mathrm{refl}_a = \sigma^n, \quad f \mapsto (f(\mathrm{pt}_n), \mathrm{ap}_f(\sigma_n), !)$$

[29] Notice that this is a less general result than the universal property of the circle, or equivalently, the case $n = 0$, where we don't need to assume that A is a groupoid.

prop:ump-cycn-into-groupoids

Theorem 3.8.5. *For any groupoid A, the map $\mathrm{ev}_{n,A}$ is an equivalence.*

Proof. Let $a : A$ and $\sigma : a = a$ such that $\mathrm{refl}_a = \sigma^n$ holds. One wants to prove that the fiber

$$\sum_{f : \mathrm{Cyc}_n \to A} (a, \sigma, !) = \mathrm{ev}_{n,A}(f)$$

is contractible. Hence we first need to craft a function $f : \mathrm{Cyc}_n \to A$ such that there is $p : a = f(\mathrm{pt}_n)$ such that $\mathrm{ap}_f(\sigma_n) \cdot p = p \cdot \sigma$.

In order to do so, we will craft a function $f : \mathrm{Cyc}_n \to A$ together with a function $\hat{p}_x : \mathrm{pt}_n = x \to a = f(x)$ for each $x : \mathrm{Cyc}_n$ such that $\hat{p}_x(\_\sigma_n) = \hat{p}_x(\_)\sigma$. By setting $p \equiv \hat{p}_{\mathrm{pt}_n}(\mathrm{refl}_{\mathrm{pt}_n})$, we would have succeeded. Indeed, path induction on $\alpha : x = x'$ shows that $\hat{p}_{x'}(\alpha\_) = \mathrm{ap}_f(\alpha)\hat{p}_x(\_)$ on one hand, and the hyptohesis on $\hat{p}$ proves that $\hat{p}_{\mathrm{pt}_n}(\_)\sigma = \hat{p}_{\mathrm{pt}_n}(\_\sigma_n)$ on the other hand. This leads to the chain of equations:

$$p\sigma = \hat{p}_{\mathrm{pt}_n}(\mathrm{refl}_{\mathrm{pt}_n})\sigma = \hat{p}_{\mathrm{pt}_n}(\mathrm{refl}_{\mathrm{pt}_n}\sigma_n) = \hat{p}_{\mathrm{pt}_n}(\sigma_n\mathrm{refl}_{\mathrm{pt}_n}) = \mathrm{ap}_f(\sigma_n)\hat{p}_{\mathrm{pt}_n}(\mathrm{refl}_{\mathrm{pt}_n}) = \mathrm{ap}_f(\sigma_n)p$$

It remains to craft the promised f and $\hat{p}$. For each $x : \mathrm{Cyc}_n$, consider the type

$$T(x) :\equiv \sum_{b : A} \sum_{\pi : \mathrm{pt}_n = x \to a = b} \pi(\_\sigma_n) = \pi(\_)\sigma$$

One claims that $T(x)$ is contractible. To prove this proposition for x ranging over the connected type Cyc_n, it is enough to only prove it for $x \equiv \mathrm{pt}_n$. However, as $i \mapsto \sigma_n^i$ provides an equivalence $\mathbb{m} \to (\mathrm{pt}_n = \mathrm{pt}_n)$, one gets:

$$T(\mathrm{pt}_n) \simeq \sum_{b : A} \sum_{\pi : \mathbb{m} \to a = b} \pi(\_ + 1) = \pi(\_)\sigma$$

Now, note that $\pi : \mathbb{m} \to a = b$ such that $\pi(\_ + 1) = \pi(\_)\sigma$ is entirely determined by $\pi(0)$, as then $\pi(i) = \pi(0)\sigma^i$ for all $i : \mathbb{m}$. Moreover, an element q in $a = b$ defines a function $\pi_q : i \mapsto q\sigma^i$ which satisfies the equation $\pi_q(\_ + 1) = \pi_q(\_)q$. In other words, one has an equivalence:

$$\left(\sum_{\pi : \mathbb{m} \to a = b} \pi(\_ + 1) = \pi(\_)\sigma \right) \simeq (a = b), \quad (\pi, !) \mapsto \pi(0).$$

Hence, one can simplify further $T(\mathrm{pt}_n)$:

$$T(\mathrm{pt}_n) \simeq \sum_{b : A} a = b \simeq 1$$

One gets $f(x)$ by selecting a center of contraction for each $x : \mathrm{Cyc}_n$ and the function $\hat{p}_x$ is then defined as the first projection of the second component of this center of contraction.

Finally, we prove that the fiber $\mathrm{ev}_{n,A}^{-1}(a, \sigma, !)$ is a proposition. As we just proved that it is inhabited, we would have successfully shown that the fiber is contractible. Given two elements $(f, p, !)$ and $(f', p', !)$ of the fiber, one wants to find a path between the two, that is $\chi : \prod_{x : \mathrm{Cyc}_n} f(x) = f'(x)$ such that the following commutes:

$$\begin{tikzcd}
a \arrow[r, equals, "p"', "\rightarrow"] \arrow[d, equals, "p'", "\downarrow"'] & f(\mathrm{pt}_n) \arrow[dl, equals, "\chi(\mathrm{pt}_n)", "\swarrow"'] \\
f'(\mathrm{pt}_n) &
\end{tikzcd}$$

The construction of f is really an ad hoc version of the delooping of the abstract group morphism $\sigma_n^i \mapsto \sigma^i$. If we move this section forward, one can rewrite it as such.

Let us denote $U(x) :\equiv f(x) = f'(x)$ for $x : \mathrm{Cyc}_n$, and notice that these types are sets (as A is a groupoid). The element $\tau :\equiv p'p^{-1} : U(\mathrm{pt}_n)$ is peculiar in that $\mathrm{trp}^U_q(\tau) = \tau$ for all $q : \mathrm{pt}_n = \mathrm{pt}_n$. Indeed, we use once again that symmetries of pt_n in Cyc_n are of the form σ_n^i and we calculate:

$$\mathrm{trp}^U_{\sigma_n^i}(\tau) = \mathrm{ap}_{f'}(\sigma_n^i) \cdot \tau \cdot \mathrm{ap}_f(\sigma_n^i)^{-1} = p'\sigma^i p'^{-1} \cdot p'p^{-1}p\sigma^{-i}p^{-1} = p'p^{-1}$$

Now it is easy to prove that the following type is contractible:

$$V(x) :\equiv \sum_{\alpha : U(x)} \alpha = \mathrm{trp}^U_{\_}(\tau).$$

To do so, we use the connectedness of Cyc_n and verify the contractibility of $V(\mathrm{pt}_n)$ by pointing out that $V(\mathrm{pt}_n)$ is simply the singleton type of τ. Now χ is defined as the function mapping x to the center of contraction of $V(x)$. By definition, $\chi(\mathrm{pt}_n) = \tau$ as wanted. □

The construction of χ is really an ad hoc version of the following fact: for any G-set X, the type of fixed points of X is equivalent to the type of sections of $\sum_{z : BG} X(z) \to BG$. If we move this section forward, one can rewrite it as such.

3.9 *Old material yet to be integrated*

sec:decksS1

There are many other instances of the m^{th} root construction, Construction 3.7.1, which is of independent interest. We record the following for future reference.

def:Zetmodm

Definition 3.9.1. Let m be a positive integer. The element $\mathrm{Z}/m : \sum_{X:\mathrm{Set}}(X = X)$ has first projection $\mathbb{m} \times \mathbb{1}$ and second projection $\sqrt[m]{\mathrm{refl}_{\mathbb{1}}}$. ⌟

This realizes the cycle $0 \mapsto 1 \mapsto \cdots \mapsto m-1 \mapsto 0$ in $\mathbb{m}$, and so models "modular arithmetic".

The term "cyclic" in the chapter heading refers to the fact that we show that the symmetries of set bundles are determined by iterations of a single symmetry.

Remark 3.9.2. Since we are interested in the symmetries of particular set bundles it is good to spell out some details. By Lemma 2.10.3 the identity type $(A, f, !) = (A', f', !)$ of two set bundles over the type B is equivalent to

$$\sum_{p_A : A =_{\mathcal{U}} A'} (f \overset{X \mapsto (X \to B)}{\underset{p_A}{=\!=\!=}} f').$$

The latter type is by Definition 2.7.2 and Lemma 2.14.2 with $X \equiv \mathcal{U}$, $Y \equiv \mathrm{id}_{\mathcal{U}}$ and Z constant B, and the remark after Definition 2.13.1 that $\tilde{p}_A = \mathrm{trp}^{\mathrm{id}_{\mathcal{U}}}_{p_A}$, equivalent to

$$\sum_{p_A : A =_{\mathcal{U}} A'} (f =_{A \to B} f'\tilde{p}_A).$$

The situation can be depicted as

$$\begin{tikzcd}
A \arrow[rr, equal, "p_A"] \arrow[rr, phantom, "\longrightarrow"'] \arrow[dr, "f"'] & & A' \arrow[dl, "f'"] \\
& B. &
\end{tikzcd}$$

⌟

Recall that for any type X and element $x : X$, cst_x denotes the function $\mathbb{1} \to X$ defined by $\mathrm{cst}_x(*) :\equiv x$ on the unique element $* : \mathbb{1}$. In particular, $\mathrm{cst}_\bullet : \mathbb{1} \to \mathrm{S}^1$ denotes the universal set bundle of S^1.

thm:coveringsofS1

Theorem 3.9.3.

item:univ-cover-Sc-Z

(1) *There is an equivalence* $\mathrm{S}^1 \simeq \mathrm{SetBundle}(\mathrm{S}^1)_{(\mathrm{cst}_\bullet)}$ *mapping* $\bullet$ *to* $(\mathrm{cst}_\bullet, |\mathrm{refl}_{\mathrm{cst}_\bullet}|)$.

In particular, there is a symmetry

$$Q^1 : \mathrm{cst}_\bullet =_{\mathrm{SetBundle}(\mathrm{S}^1)} \mathrm{cst}_\bullet$$

such that every symmetry of $\mathrm{cst}_\bullet$ *(as a set bundle over* S^1*) can be identified with* $(Q^1)^k$ *for a unique* $k : \mathrm{Z}$.

item:setbundle-mcover

(2) *For a positive integer* m, *the set* $\delta_m = \delta_m$ *of symmetries of the* m*-fold set bundle of the circle is equivalent to the finite type* $\mathbb{m}$.

Furthermore, there is a symmetry

$$Q^1 : \delta_m =_{\mathrm{SetBundle}(\mathrm{S}^1)} \delta_m$$

so that every symmetry of δ_m *(as set bundle over* S^1*) can be identified with* $(Q^1)^i$ *for a uniquely determined* k *between* 0 *and* $m-1$. *In other words, following Definition 3.9.1,*

$$\mathrm{SetBundle}(\mathrm{S}^1)_{(\delta_m)} \simeq \left(\sum_{X:\mathrm{Set}} X = X\right)_{(\mathrm{Z}/m)}.$$

lessofgeneratorsofmodulararithmetic1

REMARK 3.9.4. The symmetries called Q^1 in Theorem 3.9.3 are not uniquely determined by the stated property. In the case of the universal set bundle there are two candidates, and for the m-fold set bundle there are as many as there are positive integers less than m that are relatively prime to m. This behavior has number theoretic consequences and origins and will be investigated further when we have the proper machinery to put it to good use. ⌟

Proof. Let us first prove (1). Through Lemma 3.4.7, it is enough to find an equivalence $(\bullet = \bullet) \simeq (\mathrm{cst}_\bullet =_{\mathrm{SetBundle}(\mathrm{S}^1)} \mathrm{cst}_\bullet)$ which preserves reflexivity and composition of paths. It is obtained as the composition

$$(\bullet =_{\mathrm{S}^1} \bullet) \simeq (\mathrm{cst}_\bullet =_{\mathbb{1}\to\mathrm{S}^1} \mathrm{cst}_\bullet) \simeq \left(\mathrm{cst}_\bullet =_{\mathrm{SetBundle}(\mathrm{S}^1)} \mathrm{cst}_\bullet\right)$$

where the first equivalence is given by induction on $\mathbb{1}$ and function extensionality, and the second one is mapping a path e to the path $(\mathrm{refl}_{\mathbb{1}}, e)$. Both equivalences clearly preserve reflexivity paths and composition of paths, hence so does their composition.

Let us move on to (2). First, let us emphasize that we are interested in the symmetries of δ_m *as a set bundle*, meaning we shall explore the loops $(\mathrm{S}^1, \delta_m) =_X (\mathrm{S}^1, \delta_m)$ in the type $\mathrm{SetBundle}(\mathrm{S}^1)$. Because $\mathrm{SetBundle}(\mathrm{S}^1)$ is a subtype of $\sum_{A:\mathcal{U}} A \to \mathrm{S}^1$, it is enough to determine the symmetries of (S^1, δ_m) in this later type (cf. Lemma 2.20.6). This is unfolded as:

$$D_m :\equiv \sum_{g:\mathrm{S}^1=\mathrm{S}^1} \delta_m =_{\mathrm{S}^1\to\mathrm{S}^1} \delta_m \tilde{g}$$

Recall the equivalence $c : \mathrm{S}^1 \to C$ of Theorem 3.5.4. Then the transport along $\bar{c}$ in the type family $X \mapsto (\mathrm{S}^1 = X)$ is an equivalence from $(\mathrm{S}^1 = \mathrm{S}^1)$ to $(\mathrm{S}^1 = C)$. Composing with univalence, we get an equivalence $(\mathrm{S}^1 = \mathrm{S}^1) \to (\mathrm{S}^1 \simeq C)$ defined as $g \mapsto c\tilde{g}$, with inverse provided by $t \mapsto \overline{c^{-1}}t$. Hence, by following Exercise 2.9.12 we get

$$D_m \simeq \sum_{t:\mathrm{S}^1\simeq C} \delta_m =_{\mathrm{S}^1\to\mathrm{S}^1} \delta_m c^{-1} t.$$

Then, denoting $\rho_m : C \to C$ for the m-fold cover of C,

$$\begin{aligned}(\delta_m =_{S^1 \to S^1} \delta_m c^{-1} t) &\simeq (c\delta_m =_{S^1 \to C} c\delta_m c^{-1} t)\\ &\simeq (\rho_m c =_{S^1 \to C} \rho_m t)\end{aligned}$$

where the first equivalence holds because c is an equivalence, and the second because $\rho_m c =_{S^1 \to C} c\delta_m$ has been proved in **??**. Then if we write

$$F_m :\equiv \sum_{t : S^1 \simeq C} (\rho_m c =_{S^1 \to C} \rho_m t),$$

one gets that $D_m \simeq F_m$ and we can now concentrate on proving that $F_m \simeq \mathbb{m}$.

Let us first sketch the proof in three steps:

STEP 1. We shall describe the elements of F_m as basically tuples (Y, g, q) with (Y, g) in the subtype C of $\sum_{X : \mathcal{U}}(X = X)$ and $q : \mathbb{m} \times \mathbb{Z} = \mathbb{m} \times Y$ such that q satisfies certain propositional equations, denoted $E(q)$ here.

STEP 2. We shall then give a characterization of these elements (Y, g, q) in the case where Y is $\mathbb{Z}$ and g is $\bar{s}$. This characterization will give q (viewed as an equivalence) the following form:

$$q_{i,n} : (j, z) \mapsto \sqrt[\mathbb{m}]{\bar{s}}^{\,j}(i, n + z)$$

In particular, it can be seen that $(\mathbb{Z}, \bar{s}, q_{i,n}) = (\mathbb{Z}, \bar{s}, q_{i,0})$ in F_m.

STEP 3. Finally we shall define a map $\psi : \mathbb{m} \to F_m$ properly as $i \mapsto (\mathbb{Z}, \bar{s}, q_{i,0})$ and prove that it is an equivalence. It means that given $(Y, g, !) : C$, one has to show

$$P(Y, g) :\equiv \prod_{q : \mathbb{m} \times \mathbb{Z} = \mathbb{m} \times Y} E(q) \to \mathrm{isContr}(\psi^{-1}(Y, g, q))$$

where $E(q)$ is as in step 1. The product is valued in propositions so $P(Y, g)$ itself is a proposition. By definition of C, (Y, g) lies in the same connected component as $(Z, \bar{s})$ in $\sum_{X : \mathcal{U}} X = X$, so using Exercise 2.16.4, $P(Y, g)$ holds as soon as $P(Z, \bar{s})$ holds. Otherwise put, it is enough to prove the contractibility of the fiber of ψ at elements of F_m of the form $(Z, \bar{s}, q)$ for which step 2 shows that q must be one of the $q_{i,n}$ for some (i, n). This i, together with the essentially unique proof that $(\mathbb{Z}, \bar{s}, q_{i,n}) = (\mathbb{Z}, \bar{s}, q_{i,0})$, is then easily seen to be a center of contraction for the fiber $\psi^{-1}(Z, \bar{s}, q)$.

STEP 1. An element of F_m is given by a map $t : S^1 \to C$ together with a term $! : \mathrm{isEquiv}(t)$ and a proof $Q : \rho_m c = \rho_m t$. Now such a t can be reduced through the universal property of S^1 to the data of $t(\bullet) :\equiv (Y, g, !)$ and $t(\circlearrowleft) := (p, !, !) : (Y, g, !) =_C (Y, g, !)$. Under identification through the universal property of S^1, $\rho_m c$ is given by

$$\begin{aligned}&\rho_m c(\bullet) :\equiv (\mathbb{m} \times \mathbb{Z},\ \sqrt[\mathbb{m}]{\bar{s}}, !)\\ &\rho_m c(\circlearrowleft) := (\mathrm{id} \times \bar{s}, !, !) : (\mathbb{m} \times \mathbb{Z},\ \sqrt[\mathbb{m}]{\bar{s}}, !) =_C (\mathbb{m} \times \mathbb{Z},\ \sqrt[\mathbb{m}]{\bar{s}}, !)\end{aligned}$$

and similarly $\rho_m t$ is given by

$$\begin{aligned}&\rho_m t(\bullet) :\equiv (\mathbb{m} \times Y,\ \sqrt[\mathbb{m}]{g}, !)\\ &\rho_m t(\circlearrowleft) := (\mathrm{id} \times p, !, !) : (\mathbb{m} \times Y,\ \sqrt[\mathbb{m}]{g}, !) =_C (\mathbb{m} \times Y,\ \sqrt[\mathbb{m}]{g}, !)\end{aligned}$$

By function extensionality and S^1-induction, Q becomes then a term $q : \mathbb{m} \times \mathrm{Z} = \mathbb{m} \times Y$ such that the two following propositions hold, whose product is denoted $E(q)$:

$$\sqrt[m]{g} \circ q = q \circ \sqrt[m]{\bar{s}} \quad \text{and} \quad q \circ (\mathrm{id} \times \bar{s}) = (\mathrm{id} \times p) \circ q.$$

Remark that repeated applications of the first equation imply that such a q is completely determined by $\tilde{q}(0,0) : \mathbb{m} \times Y$: indeed for all $j \in \mathbb{m}$ and $z \in \mathrm{Z}$

$$\tilde{q}(j,z) = \tilde{q}(\sqrt[m]{s}^{\,j+zm}(0,0)) = \sqrt[m]{g}^{\,j+zm}\tilde{q}(0,0)$$

Remark also for future references that the proposition $p = g$ holds: indeed,

$$\mathrm{id} \times p = q \circ (\mathrm{id} \times \bar{s}) \circ q^{-1} = (q \sqrt[m]{\bar{s}} q^{-1})^m = (\sqrt[m]{g})^m = \mathrm{id} \times g.$$

Step 2. In particular, when t is actually c, then Y is Z, and g and p are both $\bar{s}$. Define then for each pair $(i,n) \in \mathbb{m} \times \mathrm{Z}$ a function $q_{i,n} : \mathbb{m} \times \mathrm{Z} \to \mathbb{m} \times \mathrm{Z}$ as follows:

$$(j,z) \mapsto \sqrt[m]{s}^{\,j+zm}(i,n)$$

Such a $q_{i,n}$ is an equivalence as it admits $q_{m-i,-n-1}$ as pseudo inverse. Moreover direct computations show easily that the propositions $\sqrt[m]{s} q_{i,n} = q_{i,n} \sqrt[m]{s}$ and $q_{i,n} \circ (\mathrm{id} \times s) = (\mathrm{id} \times s) \circ q_{i,n}$ are satisfied. In other words, for each $(i,n) : \mathbb{m} \times \mathrm{Z}$, $(\mathrm{Z}, \bar{s}, !)$ together with $q_{i,n}$ yields, by the universal property of S^1, an element $(c, !, Q_{i,n})$ of F_m, and the analysis of step 1 ensures that they are the only ones.

Step 3. Let us then define $\psi : \mathbb{m} \to F_m$ by mapping i to $(c, !, Q_{i,0})$. The claim is that ψ is an equivalence. Recall that $\mathrm{S}^1 \simeq C$ is a subtype of $\mathrm{S}^1 \to C$, so that F_m is a subtype of

$$\overline{F_m} :\equiv \sum_{t : \mathrm{S}^1 \to C} \rho_m c =_{\mathrm{S}^1 \to C} \rho_m t$$

If one denotes ι for the canonical inclusion $F_m \hookrightarrow \overline{F_m}$, then the contractibility of the fiber of ψ at a point $(t, !, Q) : F_m$, is equivalent to the contractibility of the fiber of $\iota \circ \psi$ at $(t, Q) : \overline{F_m}$ (by Lemma 2.20.6). In other words, one need to provide, for every $t : \mathrm{S}^1 \to C$, an element of:

$$\prod_{Q : (\rho_m c = \rho_m t)} \mathrm{isContr}((\iota\psi)^{-1}(t, Q))$$

Taking advantage of the equivalence $\mathrm{ev}_C : (\mathrm{S}^1 \to C) \simeq \sum_{x : C}(x =_C x)$ once again, this is equivalent as to provide, for every $x : C$, an element of:

$$P(x) :\equiv \prod_{p : x =_C x} \left(\prod_{Q : (\rho_m c = \rho_m \mathrm{ve}_C(x,p))} \mathrm{isContr}((\iota\psi)^{-1}(\mathrm{ve}_C(x,p), Q)) \right)$$

Because isContr(_) is valued in propositions, then so is P. Through Exercise 2.16.4 and because C is connected, one needs to check P on only one point. We choose to do so on pt_C. Now, given $p : \mathrm{pt}_C = \mathrm{pt}_C$ and $Q : (\rho_m c = \rho_m \mathrm{ve}(\mathrm{pt}_c, p))$, step 1 proves that $(\bar{s}, !, !) = p$, so that in particular one has $\pi : c = \mathrm{ve}(\mathrm{pt}_C, p)$, and then step 2 ensures that $Q_{i,n} =_\pi Q$ for some $(i,n) : \mathbb{m} \times \mathrm{Z}$. Also notice that if one denotes $\sigma_n : c = c$ for the path induced by $(\bar{s}^n, !, !) : \mathrm{pt}_C = \mathrm{pt}_C$, then it holds that $Q_{i,0} =_{\sigma_n} Q_{i,n}$: indeed the transport along $\bar{s}^n$ (in the type family $X \mapsto (\mathbb{m} \times \mathrm{Z} = \mathbb{m} \times X)$) is just the postcomposition with $\mathrm{id} \times \bar{s}^n$, and

$$\begin{aligned}(\mathrm{id} \times s^n) q_{i,0} = \sqrt[m]{s}^{nm} q_{i,0} &= ((j,z) \mapsto \sqrt[m]{s}^{nm+j+zm}(i,0)) \\ &= ((j,z) \mapsto \sqrt[m]{s}^{j+zm}(i,n)) \\ &= q_{i,n}\end{aligned}$$

The compositions of the paths described above yield a path $\pi\sigma_n : c = \mathrm{ve}_C(\mathrm{pt}_C, p)$ and a path-over $Q_{i,0} =_{\pi\sigma_n} Q$, so that $(i, (\pi\sigma_n, !))$ is in the fiber of $\iota\psi$ at $\mathrm{ve}_C(\mathrm{pt}_C, p)$. We claim it is a center of contraction. Indeed, for any other $j : \mathbb{m}$ together with a path $\rho : c = \mathrm{ve}_C(\mathrm{pt}_C, p)$ such that $Q_{j,0} =_\rho Q$, one gets $Q_{i,0} =_{\rho^{-1}\pi\sigma_n} Q_{j,0}$. Lemma 3.5.1 show that $\rho^{-1}\pi\sigma_n = \sigma_k$ for some $k : \mathrm{Z}$ and the previous path-over between $Q_{i,0}$ and $Q_{j,0}$ then means that $(\mathrm{id} \times s^k) q_{i,0} = q_{j,0}$ which evaluated at $(0,0) : \mathbb{m} \times \mathrm{Z}$ gives the equations $i = j$ and $k = 0$. Hence $(j, (\rho, !)) = (i, (\pi\sigma_n, !))$, concluding thus the proof that $(i, (\pi\sigma_n, !))$ is a center of contraction for the fiber at play.

□

Remark 3.9.5. Regarding the symmetries of the m-fold set bundle of the circle, recall the picture we tried to evoke in Remark 3.6.6. How can I move my circle with m evenly spaced marked points (which we now call $0, 1, \ldots, m-1$ instead of $1, 2, \ldots 12$ since it fits better with future applications) without disturbing the projection down to the circle (sending all the marked points to 0). I can move the marked points, but a marked point has to be sent to a marked point (otherwise the projection down to the circle would be disturbed). Let's say that mark 0 is sent to mark 4. However, since we have to preserve the projection down to the circle, the arc from 0 to 1 must then be sent to the arc from 4 to 5. Continuing in this fashion, we see that we describe a certain rotation of the circle. Varying 4 between 0 and $m-1$ we see that there are exactly m different symmetries of the m-fold set bundle. Furthemore they are all rotations of the circle by an angle which is a multiple of $2\pi/m$. ⌟

Alternative proof of Theorem 3.9.3(2)

In this subsection we present yet another proof, one that is not using the type C. This proof uses some properties of S^1 that are interesting in their own right. We introduce them in the form of exercises.

xca:S1≃S1-components

Exercise 3.9.6. Let $-\mathrm{id}_{\mathrm{S}^1} : \mathrm{S}^1 \to \mathrm{S}^1$ be defined by $-\mathrm{id}_{\mathrm{S}^1}(\bullet) :\equiv \bullet$ and $-\mathrm{id}_{\mathrm{S}^1}(\circlearrowright) := \circlearrowright^{-1}$. Show $-\mathrm{id}_{\mathrm{S}^1} \neq \mathrm{id}_{\mathrm{S}^1}$. Prove the following proposition:

$$\prod_{t : \mathrm{S}^1 \simeq \mathrm{S}^1} \| \mathrm{id}_{\mathrm{S}^1} = t \| \amalg \| -\mathrm{id}_{\mathrm{S}^1} = t \|.$$ ⌟

xca:(S1->S1)_(f)-eqv-S1

Exercise 3.9.7. For any $f : \mathrm{S}^1 \to \mathrm{S}^1$, give an equivalence from S^1 to $(\mathrm{S}^1 \to \mathrm{S}^1)_{(f)}$, that is, from S^1 to the component of $\mathrm{S}^1 \to \mathrm{S}^1$ at f. Hint: use Lemma 3.4.7. ⌟

We note in passing that combining the above two exercises yields $(\mathrm{S}^1 = \mathrm{S}^1) \simeq (\mathrm{S}^1 \amalg \mathrm{S}^1)$.

Proof of Theorem 3.9.3(2). Let $m > 0$ and

$$D_m :\equiv \sum_{t : \mathrm{S}^1 \simeq \mathrm{S}^1} \delta_m =_{\mathrm{S}^1 \to \mathrm{S}^1} \delta_m t.$$

Define

$$f : \mathbb{m} \to D_m : k \mapsto (\mathrm{id}_{S^1}, \circlearrowleft^k) \quad \text{for all } k = 0, \dots, m-1.$$

The above definition of f is simplified in that we mean id_{S^1} as an equivalence. Moreover, the type $\delta_m =_{S^1 \to S^1} \delta_m \mathrm{id}_{S^1}$ is equivalent, by function extensionality and the universal property of the circle for the type family $T(x) :\equiv (\delta_m(x) = \delta_m(x))$, to the type $\sum_{p : \bullet = \bullet} (\circlearrowleft^m p = p \circlearrowleft^m)$. The latter type is equivalent to $\bullet = \bullet$ (use Exercise 3.2.3), and therefore we can give any element of $\delta_m =_{S^1 \to S^1} \delta_m \mathrm{id}_{S^1}$ as $\circlearrowleft^z$ for some $z : \mathrm{Z}$. Finally, we tacitly coerce any element $k : \mathbb{m}$ to $k : \mathrm{Z}$. We will frequently use such simplifications in the sequel.

Claim: the map f is an equivalence, that is, for all $t : S^1 \simeq S^1$ and $Q : \delta_m =_{S^1 \to S^1} \delta_m t$ we have

$$\mathrm{isContr}\Big(\sum_{k : \mathbb{m}} (t, Q) = f(k)\Big).$$

This claim is a proposition. In view of Exercise 3.9.6 it suffices to prove the claim for $t \equiv \mathrm{id}_{S^1}$ and for $t \equiv -\mathrm{id}_{S^1}$. The latter case can be discarded since $\delta_m \neq \delta_m(-\mathrm{id}_{S^1})$ (similar to $\mathrm{id}_{S^1} \neq -\mathrm{id}_{S^1}$ in Exercise 3.9.6). In the remaining paragraphs we deal with the case $t \equiv \mathrm{id}_{S^1}$. We will use the equivalence $w : (\bullet = \bullet) \to \mathrm{Z}$ that is the inverse of $\circlearrowleft^{-}$ from Corollary 3.4.5.

Consider a $Q : \delta_m =_{S^1 \to S^1} \delta_m$; then we have $Q(\bullet) : \bullet = \bullet$, and $Q(\circlearrowleft)$ proves a proposition. We have to propose a center of $\sum_{k : \mathbb{m}} (\mathrm{id}_{S^1}, Q) = f(k)$, and prove that it is a center of contraction.

For the center, we apply Euclidean Division, Lemma 2.23.8, albeit for integers. Write $w(Q(\bullet)) = ml + k$ with $k : \mathbb{m}$ and $l : \mathrm{Z}$, both unique. We take k as the first component of the center. For the second component of the center it suffices to give an element of $(\mathrm{id}_{S^1}, Q(\bullet)) = (\mathrm{id}_{S^1}, \circlearrowleft^k) \equiv f(k)$. We take $\circlearrowleft^{-l}$ as (simplified) element of $\mathrm{id}_{S^1} = \mathrm{id}_{S^1}$. The picture in Fig. 3.11 depicts transport in the family $R(t) :\equiv (\delta_m = \delta_m t)$ (top half) and specialize to the situation at hand (bottom half).

Clearly, the transport of $Q(\bullet)$ along $\circlearrowleft^{-l}$ is equal to $\circlearrowleft^k$ because of $w(Q(\bullet)) = ml + k$. This completes the construction of the center.

The last step is to show that the center is indeed a center of contraction. Let $p' : (\mathrm{id}_{S^1}, Q(\bullet)) = (\mathrm{id}_{S^1}, \circlearrowleft^{k'}) \equiv f(k')$ for $k' : \mathbb{m}$. Then $\mathrm{fst}(p') = \circlearrowleft^{-l'}$ for some $l' : \mathrm{Z}$. By exactly the same analysis as above we get $w(Q(\bullet)) = ml' + k'$. Since Euclidean Division is unique, we get $k' = k$ and $l' = l$. The type of $\mathrm{snd}(p')$ is a proposition. Hence the pair (k', p') is equal to the center. □

$$\begin{tikzcd}
t \arrow[d, equal, "p"] & \delta_m \arrow[r, equal, "X"] & \delta_m t \arrow[d, equal, "(\delta_m \_)(p)"] \\
t' & & \delta_m t'
\end{tikzcd}$$

$$\begin{tikzcd}
\mathrm{id}_{S^1} \arrow[d, equal, "\circlearrowleft^{-l}"] & \bullet \arrow[r, equal, "Q(\bullet)"] & \bullet \arrow[d, equal, "\circlearrowleft^{-ml}"] \\
\mathrm{id}_{S^1} & & \bullet
\end{tikzcd}$$

FIGURE 3.11: Transport in the type family R.

fig:transport-R

4
Groups

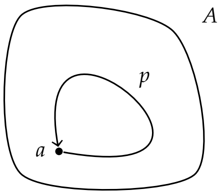
ch:groups

An identity type is not just any type: in the previous sections we have seen that the identity type $a =_A a$ reflects the "symmetries" of an element a in a type A.[1] Symmetries have special properties. For instance, you can rotate a square by 90°, and you can reverse that motion by rotating it by $-90°$. Symmetries can also be composed, and this composition respects certain rules that holds in all examples. One way to study the concept of "symmetries" would be to isolate the common rules for all our examples, and to show, conversely, that anything satisfying these rules actually *is* an example.

With inspiration of geometric and algebraic origins, it became clear to mathematicians at the end of the 19th century that the properties of such symmetries could be codified by saying that they form an abstract *group*. In Section 2.5 we saw that an identity type is "reflexive, symmetric and transitive" – and an abstract group is just a set with such operations satisfying corresponding rules.

We attack the issue more concretely: instead of focusing on the abstract properties, we bring the type exhibiting the symmetries to the fore. The axioms for an abstract group follow from the rules for identity types, without us needing to impose them. We will show that the two approaches give the same end result.

In this chapter we lay the foundations and provide some basic examples of groups.



4.0.1 *Brief overview of the chapter*

In Section 4.1 we give the formal definition of a group along with some basic examples. In Section 4.2 we spell out the details, expanding on the properties of the identity type of a group and comparing these properties with those of an abstract group. We then return in Section 4.2 to groups more generally, explaining how these map to each other through "homomorphisms" (which to us are simply pointed maps) and what this entails for the identity types: all the abstract group properties are preserved.

In most of our exposition we make the blanket assumption that the identity type in question is a set, but in Section 4.4 we briefly discuss ∞-groups where this assumption is dropped.

Classically, groups have appeared because they "act" on a set (or more general objects), that is to say, they collect some of the symmetries of the set. This is a point of view that we will return to many times and we give the basic theory in Section 4.5. This section should remind the reader of

[1] Since the symmetries $p : a =_A a$ are paths that start and end at the point $a : A$, we also call them *loops* at a.

the material in Chapter 3, where we dealt with the special case of the group of integers. More generally, connected set bundles now reappear in the guise of "transitive G-sets", laying the groundwork for our later discussion of the set of subgroups of a group.

Another important thing, which is discussed in Section 4.5, is the type of "G-torsors", which at first glance can appear frightening. However, a G-torsor corresponds to *a* universal set bundle, and the important step is to consider the *type* of these. This idea is used in Section 4.7 to build the equivalence between our definition of a group and the abstract version taught in most algebra classes. This is followed up for homomorphisms in Section 4.8 and for G-sets in Section 4.11.

With all this in place, the structure of the type of groups is in many aspects similar to the universe, in the sense that many of the constructions on the universe that we're accustomed to have analogues for groups, namely: functions are replaced by homomorphisms; products stay "the same," as we will see in Example 4.1.25 (and more generally, product types over sets "stay the same."); and the sum of two groups has a simple implementation as the sum of the underlying types with the base points identified, as defined more precisely in Definition 4.12.1. In the usual treatment this is a somewhat more difficult subject involving "words" taken from the two groups. This reappears in our setting when we show that homomorphisms from a sum to another group correspond to pairs of homomorphisms (just as for sums of types and functions between types).

[2]

4.1 *The type of groups*

sec::typegroup
looptype

Definition 4.1.1. Given a pointed type $X \equiv (A, a)$, we define its type of *loops* by $\Omega X :\equiv (a =_A a)$. ⌟

ex:base=base

Example 4.1.2. We defined the circle S^1 in Definition 3.1.1 by declaring that it has a point • and an identification ("symmetry") $\circlearrowleft : (\bullet = \bullet) \equiv \Omega\mathrm{S}^1$, and we proved in Corollary 3.4.5 that $\Omega\mathrm{S}^1$ is equivalent to the set Z (of integers), where $n \in \mathrm{Z}$ corresponds to the n-fold composition of $\circlearrowleft$ with itself (which works for both positive and negative n). We can think of this as describing the symmetries of •: we have one "generating symmetry" $\circlearrowleft$, and this can be applied any number of times, giving a symmetry for each number. Composition of loops here corresponds to addition of integers.

The circle is an efficient packaging of the "group" of integers, for the declaration of • and $\circlearrowleft$ not only gives the *set* Z of integers, but also the addition operation. ⌟

Example 4.1.3. Recall the finite set $\mathbb{2} : \mathrm{FinSet}_2$ from Definition 2.24.1, containing two elements. According to Exercise 2.13.3, the identity type $\mathbb{2} = \mathbb{2}$ has exactly two distinct elements, $\mathrm{refl}_{\mathbb{2}}$ and twist, and doing twist twice yields $\mathrm{xrefl}_{\mathbb{2}}$. We see that these are all the symmetries of a two point set you'd expect to have: you can let everything stay in place ($\mathrm{refl}_{\mathbb{2}}$); or you can swap the two elements (twist). If you swap twice, the results leaves everything in place. The pointed type FinSet_2 (of "finite sets with

[2] ((THEN SUBGROUPS TAKE CENTRAL STAGE, BUT I POSTPONE DISCUSSING THESE IN CASE THIS CHAPTER IS ALREADY OVERLY LONG AND WE WANT TO PUT THEM IN A SEPARATE CHAPTER))

two elements”), with $\mathbb{2}$ as the base point, is our embodiment of these symmetries, i.e., they are the elements of $\Omega(\mathrm{FinSet}_2, \mathbb{2})$.

Observe that (by the definition of S^1) there is an interesting function $S^1 \to \mathrm{FinSet}_2$, sending $\bullet : S^1$ to $\mathbb{2} : \mathrm{FinSet}_2$ and $\circlearrowleft$ to twist. ⌟

If we take the type of loops $\Omega(A, a) \equiv (a =_A a)$ for *some* type A and *some* element $a : A$ we get the notion of an ∞-group, cf. Section 4.4 below. However, in elementary texts it is customary to restrict the notion of a group to the case when $a =_A a$ is a *set*, as we will do, starting in Section 4.2. This makes some proofs easier, since if are we given two elements $g, h : a =_A a$, then the identity type $g = h$ is a proposition, i.e., g can be equal to h in at most one way. Hence questions relating to uniqueness of proofs of equality will never present a problem.

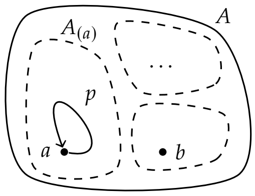
The examples of groups that Klein and Lie were interested in often had more structure on the set $\Omega(A, a)$, for instance a smooth structure. For such groups it makes sense to look at smooth maps from the real numbers to $\Omega(A, a)$, or to talk about a sequence of loops converging to some loop.[3] See Appendix A for a brief summary of the history of groups.

rem:heap-preview

Remark 4.1.4. The reader may wonder about the status of the identity type $a =_A a'$ where $a, a' : A$ are different elements. One problem is of course that if $p, q : a =_A a'$, there is no obvious way of composing p and q to get another element in $a =_A a'$, and another is that $a =_A a'$ does not have a distinguished element, such as $\mathrm{refl}_a : a =_A a$.[4] Given $f : a =_A a'$ we can use transport along f to compare $a =_A a'$ with $a =_A a$ (much as affine planes can be compared with the standard plane or a finite dimensional real vector space is isomorphic to some Euclidean space), but absent the existence and choice of such an f the identity types $a =_A a'$ and $a =_A a$ are different animals. We will return to this example after we have defined torsors. ⌟

rem:whypointedconngpoid

Remark 4.1.5. As a consequence of Lemma 2.20.6, the inclusion of the component $A_{(a)} :\equiv \sum_{x:A} \|a = x\|$ into A (i.e., the first projection) induces an equivalence of identity types from $(a, !) =_{A_{(a)}} (a, !)$ to $a =_A a$, and thus from $\Omega(A_{(a)}, (a, !))$ to $\Omega(A, a)$. This means that, when considering the loop type $\Omega(A, a)$, “only the elements $x : A$ with x merely equal to a are relevant”, and to avoid artificial extra components, we should consider only *connected* types A (cf. Definition 2.16.3).



Also, our preference for $\Omega(A, a)$ to be a *set* indicates that we should consider only the connected types A that are *groupoids*. ⌟

def:pt-conn-groupoid

Definition 4.1.6. The type of *pointed, connected groupoids* is the type

$$\mathcal{U}_*^{=1} :\equiv \sum_{A:\mathcal{U}} (A \times \mathrm{isConn}(A) \times \mathrm{isGrpd}(A)).$$ ⌟

xca:defgroup

Exercise 4.1.7. Show that a pointed type (A, a) is connected if and only if the type $\prod_{x:A} \|a =_A x\|$ has an element. Show that a connected pointed type X is a groupoid if and only if the type ΩX is a set. Conclude by showing that the type $\mathcal{U}_*^{=1}$ is equivalent to the type

$$\sum_{A:\mathcal{U}} \sum_{a:A} \Bigg(\Big(\prod_{x:A} \|a =_A x\| \Big) \times \mathrm{isSet}(\Omega(A, a)) \Bigg).$$ ⌟

The meaning of the superscript “= 1” can be explained as follows: We also define

$$\mathcal{U}^{\leq 1} :\equiv \mathrm{Groupoid} :\equiv \sum_{A:\mathcal{U}} \mathrm{isGrpd}(A)$$

to emphasize that groupoids are 1-types; the type of connected types is defined as follows.

$$\mathcal{U}^{>0} :\equiv \sum_{A:\mathcal{U}} \mathrm{isConn}(A)$$

Similar notations with a subscript “$*$” indicate pointed types.

[3] Such groups give rise to ∞-groups by converting smooth or continuous loops in A parametrized by real intervals, into identifications, as described already in Footnote 6 in Chapter 2. Then also smooth or continuous paths in $\Omega(A, a)$ turn into identifications of loops.

[4] The type $a =_A a'$ does have an interesting *ternary* composition, mapping p, q, r to $pq^{-1}r$. A set with this kind of operation is called a *heap*, and we’ll return to heaps in Section 6.3.

Remark 4.1.8. We shall refer to a pointed connected groupoid (A, a, p, q) simply by the pointed type $X :\equiv (A, a)$. There is no essential ambiguity in this, for the types $\mathrm{isConn}(A)$ and $\mathrm{isGrpd}(A)$ are propositions (Lemma 2.15.3 and Lemma 2.15.6), and so the witnesses p and q are unique.

We also write pt_X for the *base point* a of the pointed type X, as in Definition 2.21.1. $\lrcorner$

We are now ready to define the type of groups.

def:typegroup

Definition 4.1.9. The *type of groups* is a wrapped copy (see Section 2.12.5) of the type of pointed connected groupoids $\mathcal{U}_*^{=1}$,

$$\mathrm{Group} :\equiv \mathrm{Copy}_{\underline{\Omega}}(\mathcal{U}_*^{=1}),$$

with constructor $\underline{\Omega} : \mathcal{U}_*^{=1} \to \mathrm{Group}$. A *group* is an element of Group. $\lrcorner$

def:classifying-type

Definition 4.1.10. We write $\mathrm{B} : \mathrm{Group} \to \mathcal{U}_*^{=1}$ for the destructor associated with $\mathrm{Copy}_{\underline{\Omega}}(\mathcal{U}_*^{=1})$. For $G : \mathrm{Group}$, we call BG the *classifying type* of G.[5] Morever, the elements of BG will be referred to as the *shapes of* G, and we define the *designated shape of* G by setting $\mathrm{sh}_G :\equiv \mathrm{pt}_{BG}$, i.e., the designated shape of G is the base point of its classifying type. $\lrcorner$

def:group-symmetries

Definition 4.1.11. Let G be a group. We regard every group as a group of symmetries, and thus we refer to the elements of ΩBG as the *symmetries in* G; they are the symmetries of the designated shape sh_G of G. (Notice the careful distinction between the phrases "*symmetries in*" and "*symmetries of*".) We adopt the notation $\mathrm{U}G$ for the type ΩBG of symmetries in G; it is a set.[6] $\lrcorner$

rem:aut

Remark 4.1.12. We are emphasizing that the essential feature of a group is the symmetries of its designated shape. That is why we defined Group to be a copy of $\mathcal{U}_*^{=1}$, and not $\mathcal{U}_*^{=1}$ itself; the type $\mathrm{U}G$ is at least as important as BG – the copying forces us to use the notation BG, preventing a glib identification of G with its classifying type. As noted in Section 2.12.5, the constructor and destructor pair forms an equivalence $\mathrm{Group} \simeq \mathcal{U}_*^{=1}$. The type $\mathcal{U}_*^{=1}$ is a subtype of $\mathcal{U}_*$, so once you know that a pointed type X is a connected groupoid, you know also that X is the classifying type for a group, namely $G :\equiv \underline{\Omega}X$.

Note that the equivalence also entails that identifications $G = H$ of groups are equivalent to identifications $BG = BH$ of pointed types. $\lrcorner$

rem:BG-convention

Remark 4.1.13. To define a function $f : \prod_{G:\mathrm{Group}} T(G)$, where $T(G)$ is a type family parametrized by $G : \mathrm{Group}$, it suffices to consider the case $G \equiv \underline{\Omega}X$, where X is a pointed connected groupoid, namely the classifying type BG.[7] $\lrcorner$

Frequently we want to consider the symmetries $\Omega(A, a)$ of some element a in some groupoid A, so we introduce the following definition.

def:automorphism-group

Definition 4.1.14. [8] For a groupoid A with a specified point a, we define the *automorphism group* of $a : A$ by

$$\mathrm{Aut}_A(a) :\equiv \underline{\Omega}(A_{(a)}, (a, !)),$$

i.e., $\mathrm{Aut}_A(a)$ is the group with classifying type $\mathrm{BAut}_A(a) \equiv (A_{(a)}, (a, !))$, the connected component of A containing a, pointed at a. $\lrcorner$

[5] As a notational convention we always write the "B" so that it sits next to and matches the shape of its operand. You see immediately the typographical reason behind this convention: The italic letters B,G get along nicely, while the roman B would clash with its italic friend G if we wrote BG instead.

[6] Taking the symmetries in a group thus defines a map $\mathrm{U} : \mathrm{Group} \to \mathrm{Set}$, with $\underline{\Omega}X \mapsto \Omega X$.

Recall also the example of the negated natural numbers $\mathbb{N}^-$ from Section 2.12.5: Its elements are $-n$ for $n : \mathbb{N}$ to remind us how to think about them. And the same applies to Group: Its elements are $\underline{\Omega}X$ for $X : \mathcal{U}_*^{=1}$ to remind us how to think about them.

[7] If you are bothered by the convention to write the classifying type of G in *italic* like a variable, you can either think of BG as a locally defined variable denoting the classifying type that is defined whenever a variable G of type Group is introduced, or you can imagine that whenever such a G is introduced (with the goal of making a construction or proving a proposition), we silently apply the induction principle to reveal a wrapped variable $BG : \mathcal{U}_*^{=1}$.

[8] Previously, this allowed for A to not necessarily be a groupoid, as long as the component of A is. That seems like unnecessary generality?!

BID: this may have been done to refer to things like $\mathrm{Aut}_{\mathcal{U}}(7)$ – typically in situations where we don't want to introduce separate notation for the component. Don't know if we have instances of this still in place: tell if you find one rem:symmetriesofnonconnectedgroupoids

REMARK 4.1.15. For any $G \equiv \underline{\Omega}(A, a)$: Group, we have an identification $G = \mathrm{Aut}_A(a)$, because we have an identification of pointed types $(A_{(a)}, (a, !)) = (A, a)$, since A is connected.

In other words, for any $G \equiv \underline{\Omega}BG$, we have an identification $G = \mathrm{Aut}_{BG}(\mathrm{sh}_G)$, of G with the automorphism group of the designated shape $\mathrm{sh}_G : BG$. ⌟

4.1.16 *First examples*

sec:firstgroupexamples
excirclegroup

EXAMPLE 4.1.17. The circle S^1, which we defined in Definition 3.1.1, is a connected groupoid (Lemma 3.1.6, Corollary 3.4.5) and is pointed at $\bullet$. The identity type $\bullet =_{S^1} \bullet$ is equivalent to to the set of integers Z and composition corresponds to addition. This justifies our definition of the *group of integers* as

$$\mathbb{Z} :\equiv \mathrm{Aut}_{S^1}(\bullet).$$

It is noteworthy that along the way we gave several versions of the circle, each of which has its own merits, the version in Definition 3.5.2

$$C = (\sum_{X : \mathcal{U}} \sum_{f : X = X} \|(\mathrm{Z}, s) = (X, f)\|, (\mathrm{Z}, s))$$

being a very convenient one. ⌟

ex:groups

EXAMPLE 4.1.18. Apart from the circle, there are some important groups that come almost for free: namely the symmetries in the type of sets.

(1) Recall that the set $\mathbb{1}$ = True has the single element which we can call $*$. Then $\mathrm{Aut}_{\mathbb{1}}(*)$ is a group called the *trivial group*.

(2) If $n : \mathbb{N}$, then the *permutation group of n letters* is

$$\Sigma_n :\equiv \mathrm{Aut}_{\mathrm{FinSet}_n}(\mathbb{n}),$$

where FinSet_n is the groupoid of sets of cardinality n (cf. 2.24.1). The classifying type is thus $B\Sigma_n :\equiv (\mathrm{FinSet}_n, \mathbb{n})$. With our convention of Remark 4.1.15 we can tolerate $\mathrm{Aut}_{\mathrm{FinSet}}(\mathbb{n})$, $\mathrm{Aut}_{\mathrm{Set}}(\mathbb{n})$ or even $\mathrm{Aut}_{\mathcal{U}}(\mathbb{n})$ as synonyms for the group Σ_n (where FinSet and Set are the subtypes of $\mathcal{U}$ of finite sets and setsrespectively).

If the reader starts worrying about size issues, that is legitimate: see Remark 4.1.19.

(3) More generally, if S is a set, is there a pointed connected groupoid (A, a) so that $a =_A a$ models all the "permutations" $S =_{\mathrm{Set}} S$ of S? Again, the only thing wrong with the groupoid Set of set (apart from Set being large) is that Set is not connected. The *group of permutations of S* is defined to be

$$\Sigma_S :\equiv \mathrm{Aut}_{\mathrm{Set}}(S),$$

with classifying type $B\Sigma_S :\equiv (\mathrm{Set}_{(S)}, S)$.

⌟

rem:groupsarebig

REMARK 4.1.19. This remark is for those who worry about size issues – a theme we usually ignore in our exposition. If we start with a base universe $\mathcal{U}_0$, the groupoid FinSet_n of sets of cardinality n is the Σ-type $\sum_{A : \mathcal{U}_0} \|A = \mathbb{n}\|$ over $\mathcal{U}_0$ and so (without any modification) will lie in any bigger universe $\mathcal{U}_1$. In order to accommodate the finite permutation groups, the universe "$\mathcal{U}$" appearing as a subscript of the first Σ-type in Definition 4.1.6, appearing later in the definition of "group", needs to be at least as big as $\mathcal{U}_1$. If $\mathcal{U}$ is taken to be $\mathcal{U}_1$, then the type Group of groups will not be in $\mathcal{U}_1$, but in some bigger universe $\mathcal{U}_2$. If we then choose some group G : Group and look at *its* group of automorphisms, based on an identity type in Group, this will be an element of Group only if the universe $\mathcal{U}$ in the definition of Group is at least as big as $\mathcal{U}_2$, which is not the case. Our convention from Section 2.13 is that the universes form an ascending chain $\mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \dots$, corresponding to which there will an ascending chain of types of groups,

$$\mathrm{Group}_i :\equiv \mathrm{Copy}_{\underline{\Omega}}\big(\sum_{A\,:\,\mathcal{U}_i} (A \times \mathrm{isConn}(A) \times \mathrm{isGrpd}(A))\big),$$

and any group we encounter will be an element of Group_i for i large enough. These matters concerning universes are nontrivial and important, but in this text we have chosen to focus on other matters. ⌟

ex:cyclicgroups

EXAMPLE 4.1.20. In Theorem 3.9.3 we studied the symmetries of the "m-fold set bundle" of the circle for m a positive integer, and showed that there were m different such symmetries. Moreover we showed that these symmetries were the powers f^n (for $n = 0, 1, \dots, m-1$) of one (non-unique) symmetry f and that $f^{m+k} = f^k$ for any integer k. This very important phenomenon pops up in many situations, and is called the *cyclic group of order m*.

In other words, the cyclic group of order m is the (pointed) component of the type $\mathrm{SetBundle}(\mathrm{S}^1)$ of set bundles of the circle containing the m-fold set bundle. We analyzed this in Theorem 3.9.3 and found that this (pointed) component was equivalent to the connected groupoid

In this example we discover the cyclic group of order m as the group of symmetries of the m-fold set bundle of S^1.

Note that the cyclic group of order 1 is the trivial group, the cyclic group of order 2 is equivalent to the permutation group Σ_2: there are exactly one nontrivial symmetry f and f^2 is the identity. When $m > 2$ the cyclic group of order m is a group that does not appear elsewhere in our current list. In particular, the cyclic group of order m has only m different symmetries, whereas we will see that the group of permutations Σ_m has $m! = 1 \cdot 2 \cdot \dots \cdot m$ symmetries.

$$B\mathbb{Z}/m_{\div} :\equiv \big(\sum_{X\,:\,\mathrm{Set}} X = X\big)_{(\mathrm{Z}/m)}$$

pointed in $\mathrm{Z}/m = (\mathbb{m}, s_n)$, where

$$s_n : \mathbb{m} = \mathbb{m}$$

is (the identity corresponding to the equivalence given by) the cyclic permutation of $\mathbb{m}$, sending $m-1$ to 0 and, for $0 \le i < m-1$, sending $i : \mathbb{m}$ to $i+1$ (strictly speaking, Z/m when first presented in Definition 3.9.1 represented a symmetry of $\mathbb{m} \times \mathbb{1}$, but we trust we'll be forgiven for retaining the names of the symmetry under identification provided by univalence and the projection from $\mathbb{m} \times \mathbb{1}$ to $\mathbb{m}$). The role the successor function plays in C is played by the successor modulo m in $\mathbb{Z}/m$.

It is this modular behavior which inspires the "$\mathbb{Z}/m$" in the expression above: we are talking about the "integers modulo m". We make this our (first) definition of the cyclic group of order m: ⌟

def:Z/mgroup

DEFINITION 4.1.21. If m is a positive integer, the cyclic group of order m is the group

$$\mathbb{Z}/m :\equiv \mathrm{Aut}_{B\mathbb{Z}/m}(\mathrm{Z}/m).$$

⌟

$B\mathbb{Z}/m_{\div} :\equiv (\sum_{X\,:\,\mathrm{Set}} X = X)_{(\mathrm{Z}/m)}$,
$\mathrm{Z}/m = (\mathbb{m}, s_n)$.

ex:Cm

EXAMPLE 4.1.22. There are other (beside the symmetries of the m-fold set bundle and Definition 4.1.21) ways of obtaining the cyclic group of order m, which occasionally are more convenient. The prime other interpretation is as the group of "cyclic permutations of m points on a circle"; i.e., as the "set of mth roots of unity in the complex plane" equipped with complex multiplication.

We know the group $\Sigma_m :\equiv \mathrm{Aut}_{\mathrm{FinSet}_m}(\mathbb{m})$ of all permutations of the set $\mathbb{m}$ with m elements, but how do we select the "subgroup" of cyclic permutations? The key insight is provided by the function

$$\mathrm{cy}_m : S^1 \to \mathrm{FinSet}_m$$

with $\mathrm{cy}_m(\bullet) :\equiv \mathbb{m}$ and $\mathrm{cy}_m(\circlearrowleft) := s_m$, picking out exactly the cyclic permutation $s_m : \mathbb{m} = \mathbb{m}$ (and its iterates) among all permutations. Set truncation of Definition 2.22.4 provides us with a tool for capturing only the symmetries of FinSet_m hit by cy_m: the (in language to come) subgroup of the permutation group of cyclic permutations is the group

$$C_m :\equiv \mathrm{Aut}_{BC_m}(\mathrm{pt}_{C_m}),$$

where $(BC_m)_\div :\equiv \sum_{S : \mathrm{FinSet}_m} \|\mathrm{cy}_m^{-1}(S)\|_0$ and $\mathrm{pt}_{C_m} :\equiv (\mathbb{m}, |(\bullet, \mathrm{refl}_{\mathbb{m}})|_0))$. We must prove that BC_m is a pointed connected groupoid for it to earn the name "group", but since we will provide a pointed equivalence between $B\mathbb{Z}/m$ and BC_m, this will follow automatically.

More precisely, but using language not yet established: $\mathbb{Z}/m$ is equivalent to the "quotient group" (cf. Definition 5.4.7) of $\mathbb{Z}$ by the "kernel" (cf. Definition 5.2.2) of cy_m, whereas C_m is exactly the "image" (cf. **??**) of cy_m. Thanks to Lemma 5.2.12 these are equivalent groups. However, in our special case we can give a proof using only language we know.

Consider $t : \prod_{z : S^1}((\bullet = \bullet) \to (z = z))$ given by circle induction by sending $\bullet : S^1$ to the identity $t_\bullet :\equiv \mathrm{id}_{\bullet = \bullet}$ and $\circlearrowleft\ : \bullet = \bullet$ to $\mathrm{refl}_{\mathrm{id}_{\bullet=\bullet}}$. Since we'll be using it often, we let $\circlearrowleft_z\ : z = z$ be shorthand for $t_z(\circlearrowleft)$.

lem:CmisisotoZ/m

Lemma 4.1.23. *Given $S : \mathrm{FinSet}_m$, the function*

$$f_S : \sum_{z : S^1} (S = \mathrm{cy}_m(z)) \to \sum_{q : S = S} \|(S, q) = (\mathbb{m}, s_m)\|,$$
$$f_S(z, v) :\equiv (v^{-1}\,\mathrm{cy}_m(\circlearrowleft_z)v, !)$$

has connected fiber and defines an equivalence $f : BC_m \xrightarrow{\sim} B\mathbb{Z}/m$.

Proof. First and foremost, f_S is actually well defined in the sense that $(S, v^{-1}\,\mathrm{cy}_m(\circlearrowleft_z)v) = (\mathbb{m}, s_m)$ is inhabited. This is clear in view of the function

$$(\bullet = z) \to ((S, q) = (\mathbb{m}, s_m))$$

sending $p : \bullet = z$ to the identity represented by the picture

$$\begin{tikzcd}
\mathbb{m} \arrow[r, equal, "\mathrm{refl}", "\to"'] \arrow[ddd, equal, "s_m", "\downarrow"'] & \mathrm{cy}_m(\bullet) \arrow[r, equal, "\mathrm{cy}_m(p)", "\to"'] \arrow[ddd, equal, "\mathrm{cy}_m(\circlearrowleft)", "\downarrow"'] & \mathrm{cy}_m(z) \arrow[ddd, equal, "\mathrm{cy}_m(\circlearrowleft_z)", "\downarrow"'] & S \arrow[l, equal, "v"', "\leftarrow"] \arrow[d, equal, "v", "\downarrow"'] \\
 & & & \mathrm{cy}_m(z) \arrow[d, equal, "\mathrm{cy}_m(\circlearrowleft_z)", "\downarrow"'] \\
 & & & \mathrm{cy}_m(z) \\
\mathbb{m} \arrow[r, equal, "\mathrm{refl}", "\to"'] & \mathrm{cy}_m(\bullet) \arrow[r, equal, "\mathrm{cy}_m(p)", "\to"'] & \mathrm{cy}_m(z) & S, \arrow[l, equal, "v"', "\leftarrow"] \arrow[u, equal, "v", "\uparrow"']
\end{tikzcd}$$

where leftmost square commutes by definition of cy_m, the center square commutes by the definition as $\circlearrowleft_z$ by circle induction.

Note that once we have shown that f_S has connected fiber, we know that the set truncation $\|f_S\|_0$ is an equivalence;

$$\begin{tikzcd}
\sum_{z:S^1}(S = \mathrm{cy}_m(z)) \arrow[r, "f_S"] \arrow[d] & \sum_{q:S=S}\|(S,q) = (\mathbb{m}, s_m)\| \arrow[d, "\simeq"] \\
\|\sum_{z:S^1}(S = \mathrm{cy}_m(z))\|_0 \arrow[r, "\|f_S\|_0"] & \|\sum_{q:S=S}\|(S,q) = (\mathbb{m}, s_m)\|\|_0
\end{tikzcd}$$

and the fact that $\sum_{q:S=S}\|(S,q) = (\mathbb{m}, s_m)\|$ is a set gives us (upon applying $\sum_{S:\mathrm{FinSet}_m}$) the desired equivalence $BC_m \xrightarrow{\sim} B\mathbb{Z}_m$.

So to finish the proof we must show that f_S has connected fiber. Since S lies in the component of $\mathbb{m} \equiv \mathrm{cy}_m(\bullet)$ and since the proposition $\|(\mathrm{cy}_m(\bullet), q) = (\mathbb{m}, s_m)\|$ implies that q is identical to $\circlearrowleft^k$ for some k, it suffices to prove that for every k the fiber $f^{-1}_{\mathrm{cy}_m(\bullet)}(\mathrm{cy}_m(\circlearrowleft^k), !)$ – or written out:

$$\sum_{z:S^1}\ \sum_{v:\,\mathrm{cy}_m(\bullet)=\mathrm{cy}_m(z)} \mathrm{cy}_m(\circlearrowleft^k) = v^{-1}\,\mathrm{cy}_m(\circlearrowleft_z)v$$

– is connected.

Assume $(z, v, !) : f^{-1}_{\mathrm{cy}_m(\bullet)}(\circlearrowleft^k, !)$. Given $p : \bullet = z$ we get that the diagram

$$\begin{tikzcd}
\mathrm{cy}_m(\bullet) \arrow[r, equal, "\mathrm{cy}_m(p)"] \arrow[d, equal, "\mathrm{cy}_m(\circlearrowleft"] & \mathrm{cy}_m(z) \arrow[r, equal, "v"] & \mathrm{cy}_m(\bullet) \arrow[d, equal, "\mathrm{cy}_m(\circlearrowleft)"] \\
\mathrm{cy}_m(\bullet) \arrow[r, equal, "\mathrm{cy}_m(p)"] & \mathrm{cy}_m(z) \arrow[r, equal, "v"] & \mathrm{cy}_m(\bullet)
\end{tikzcd}$$

commutes; that is, $\mathrm{cy}_m(p)v : \mathrm{cy}_m(\bullet) = \mathrm{cy}_m(\bullet)$ commutes with $\mathrm{cy}_m(\circlearrowleft)$. Consequently, $\mathrm{cy}_m(p)v$ must be of the form $\mathrm{cy}_m(\circlearrowleft^k)$ for some $k = 0, 1, \dots, m-1$, or alternatively, we have an $\alpha : v = \mathrm{cy}_m(p^{-1} \circlearrowleft^k)$. Hence $(p^{-1} \circlearrowleft^k, \alpha) : (z, v, !) = (\bullet, \mathrm{refl})$. ⌟

⌟

ca:somedetailsonfinitegroupstocheck

Exercise 4.1.24. (1) Compare the definitions of Definition 2.24.1 and show that if $n : \mathbb{N}$, then $\Sigma_n = \Sigma_{\mathbb{n}}$ and (since $\mathrm{FinSet}_0 = \mathrm{FinSet}_1 = \mathbb{1}$) that $\Sigma_1 = \mathrm{Aut}_{\mathbb{1}}(\mathrm{triv})$.

(2) Prove that the set $\mathbb{n} =_{\mathrm{FinSet}_n} \mathbb{n}$ is finite of cardinality $n!$.

(3) Give an identification of the n-fold set bundle of S^1 of **??** with the first projection $\sum_{z:S^1}\mathrm{cy}_n^{-1}(z) \to S^1$ of Example 4.1.20. [9]

(4) Show that, given a type X, the type of functions $BC_n \to X$ is equivalent to the type

$$\sum_{f:S^1\to X}\ \prod_{z:S^1} f(z) = f(z^n)$$

of functions $f : S^1 \to X$ such that the two ways around

$$\begin{tikzcd}
S^1 \arrow[d, "(-)^n"'] \arrow[dr, "f"] & \\
S^1 \arrow[r, "f"] & X
\end{tikzcd}$$

agree. [10]

⌟

[9] Hint: for every $z : S^1$, $\mathrm{cy}_n(z) : \mathrm{FinSet}_n$ is a finite set of cardinality n. Decidability is not an issue, so you can appeal to our classification of the set bundles of the circle.

[10] Hint: define the function $F_1 : (BC_n \to X) \to (S^1 \to X)$ by precomposition: $F_1(g)(z) = g(\mathrm{cy}_n(z), !)$ and observe that since $\mathrm{cy}_n(z) = \mathrm{cy}_n(z^n)$ we have a function $F : (BC_n \to X) \to \sum_{f:S^1\to X}\prod_{z:S^1} f(z) = f(z^n)$.

ex:productofgroups

Example 4.1.25. If you have two groups G and H, their *product* $G \times H$ is given by taking the product of their classifying types:

$$G \times H :\equiv \underline{\Omega}(BG \times BH)$$

For instance, $\Sigma_2 \times \Sigma_2$ is called the *Klein four-group* or *Vierergruppe*, because it has four symmetries. ⌟

Note that $\mathrm{B}(G \times H) \equiv BG \times BH$ is pointed at $\mathrm{sh}_{G\times H} \equiv (\mathrm{sh}_G, \mathrm{sh}_H)$.

Remark 4.1.26. In Lemma 4.2.4 we will see that the identity type of a group satisfies a list of laws justifying the name "group" and we will later show that groups are uniquely characterized by these laws. ⌟

Some groups have the property that the order you perform the symmetries is immaterial. The prime example is the group of integers $\mathbb{Z} \equiv \mathrm{Aut}_{S^1}(\bullet)$ Any symmetry is of the form $\circlearrowleft^n$ for some integer n, and if $\circlearrowleft^m$, then $\circlearrowleft^n \circlearrowleft^m = \circlearrowleft^{n+m} = \circlearrowleft^{m+n} = \circlearrowleft^m \circlearrowleft^n$.

Such cases are important enough to have their own name:

def:abgp

Definition 4.1.27. A group G is *abelian* if all symmetries commute, in the sense that the proposition

$$\mathbf{isAb}(G) :\equiv \prod_{g,h\,:\,\mathrm{U}G} gh = hg$$

is true. In other words, the type of abelian groups is

$$\mathbf{Ab} :\equiv \sum_{G\,:\,\mathrm{Group}} \mathbf{isAb}(G).$$

⌟

exer:first examples

Exercise 4.1.28. Show that permutation group Σ_2 is abelian, but that Σ_3 is not. Show that if G and H are abelian groups, then so is their product $G \times H$. ⌟

We can envision g commuting with h by the picture

$$\begin{tikzcd}
a \arrow[r, equal, "g", "\to"'] \arrow[d, equal, "h"', "\downarrow"] & a \arrow[d, equal, "h"', "\downarrow"] \\
a \arrow[r, equal, "g", "\to"'] & a
\end{tikzcd}$$

and saying that going from (upper left hand corner) a to (lower right hand corner) a by either composition gives the same result.

rem:whatAREabeliangroups

Remark 4.1.29. Abelian groups have the amazing property that the classifying types are themselves identity types (of certain 2-types). This can be used to give a very important characterization of what it means to be abelian. We will return to this point in Section 4.10.

Alternatively, the reference to **isAb** in the definition of abelian groups is avoidable using the "one point union" of pointed types $X \vee Y$ of Definition 4.12.1 (it is the sum of X and Y where the base points are identified); Exercise 4.12.6 offers the alternative definition that a group G is abelian if and only if the "fold" map $BG \vee BG \to BG$ (both summands are mapped by the identity) factors over the inclusion $BG \vee BG \to BG \times BG$. ⌟

$$\begin{tikzcd}
BG \vee BG \arrow[r, "\mathrm{fold}"] \arrow[d, "\mathrm{inclusion}"'] & BG \\
BG \times BG \arrow[ur, dotted]
\end{tikzcd}$$

Exercise 4.1.30. Let $\mathrm{Aut}_A(a) : \mathrm{Group}$ and let b be an arbitrary element of A. Prove that the groups $\mathrm{Aut}_A(a)$ and $\mathrm{Aut}_A(b)$ are identical, in the sense that $\|\mathrm{Aut}_A(a) = \mathrm{Aut}_A(b)\|$ is true. Similarly for ∞-groups when you get that far. ⌟

rem:monoidandabsgplarge

Remark 4.1.31. In Definition 4.1.9 the first $\sum$ in the definition of the type Group ranges over the entire universe $\mathcal{U}$. Hence, Group does not belong to $\mathcal{U}$, but rather to the next universe as discussed briefly in Section 2.13. This tendency that the “type of all the types we are interested in” is a “large type” is a regular feature of the theory and since it will not cause any trouble for us, we will not be consistent in pointing it out. ⌟

xca:typegroupisgroupoid

Exercise 4.1.32. Given two groups G and H. Prove that $G = H$ is a set.[11] Prove that the type of groups is a groupoid. This means that, given a group G, the component of Group, containing (and pointed at) G, is again a group, which we will call the *group* $\mathrm{Aut}(G)$ *of automorphisms* of G. ⌟

4.2 *Abstract groups*

sec:identity-type-as-abstract

Studying the identity type leads one to the definition of what an abstract group should be. We fix a type A and an element $a : A$ for the rest of the section, and we focus on the identity type $a = a$. We make the following observations about its elements and operations on them.

(1) There is an element $\mathrm{refl}_a : a = a$. (See page 11, item (E2).) We set $e :\equiv \mathrm{refl}_a$ as notation for the time being.

(2) For $g : a = a$, the inverse $g^{-1} : a = a$ was defined in Definition 2.5.1. Because it was defined by path induction, this inverse operation satisfies $e^{-1} \equiv e$.

(3) For $g, h : a = a$, the product $h \cdot g : a = a$ was defined in Definition 2.5.2. Because it was defined by path induction, this product operation satisfies $e \cdot g \equiv g$.

For any elements $g, g_1, g_2, g_3 : a = a$, we consider the following 4 equations.

it:right-unit
(1) *the right unit law*: $g = g \cdot e$

it:left-unit
(2) *the left unit law*: $g = e \cdot g$

it:associativity
(3) *the associativity law*: $g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$

it:inverse
(4) *the law of inverses*: $g \cdot g^{-1} = e$

In Exercise 2.5.3, the reader has constructed explicit elements of these equations. If $a = a$ were a set, then the equations above would all be propositions, and then, in line with the convention adopted in Section 2.15, we could simply say that Exercise 2.5.3 establishes that the equations hold. That motivates the following definition, in which we introduce a new set S to play the role of $a = a$. We introduce a new element $e : S$ to play the role of refl_a, a new multiplication operation, and a new inverse operation. The original type A and its element a play no further role.

def:abstractgroup

Definition 4.2.1. An *abstract group* consists of the following data:

struc:under-set

(1) a type S.

Moreover, the type S should be a set. It is called the *underlying set*.

[11] We might tone down exercises like “prove that Group is a groupoid”, even though we will want to use these results. They take the geometry/fun out of the exposition.

(2) an element $e : S$, called the *unit* or the *neutral element*. struc:unit

(3) a function $S \to S \to S$, called *multiplication*, taking two elements $g_1, g_2 : S$ to their *product*, denoted by $g_1 \cdot g_2 : S$. struc:mult-op

Moreover, the following equations should hold, for all $g, g_1, g_2, g_3 : S$.

(a) $g \cdot e = g$ and $e \cdot g = g$ (the *unit laws*); axiom:unit-laws

(b) $g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$ (the *associativity law*); axiom:ass-law

(4) a function $S \to S$, the *inverse operation*, taking an element $g : S$ to its *inverse* g^{-1}. struc:inv-op

Moreover, the following equation should hold, for all $g : S$.

(a) $g \cdot g^{-1} = e$ (the *law of inverses*). ⌟ axiom:inv-law

Remark 4.2.2. Strictly speaking, the proofs of the various equations are part of the data defining an abstract group, too. But, since the equations are propositions, the proofs are unique, and by the convention introduced in Remark 2.20.8, we can afford to omit those items when speaking informally. Moreover, one needn't wonder whether one gets a different group if the equations are given different proofs, because the equations cannot be given different proofs. ⌟

Remark 4.2.3. A *monoid* is a collection of data consisting only of (1), (2), and (3) from the list in Definition 4.2.1. In other words, the existence of inverses is not assumed. For this reason, the propertry that S is a set, the unit laws, and the associativity law are, together, known as the *monoid laws*. ⌟

Taking into account the introductory comments we have made above, we may state the following lemma.

lem:idtypesgiveabstractgroups Lemma 4.2.4. *If G is a group, then* $\mathrm{U}G$, *together with* $e :\equiv \mathrm{refl}_{\mathrm{sh}_G}$, $g^{-1} :\equiv \mathrm{symm}_{\mathrm{sh}_G,\mathrm{sh}_G}\, g$ *and* $h \cdot g :\equiv \mathrm{trans}_{\mathrm{sh}_G,\mathrm{sh}_G,\mathrm{sh}_G}(g)(h)$, *define an abstract group. We will let* $\mathrm{abs}(G)$ *denote that abstract group.*

Proof. The type $\mathrm{U}G$ is a set, because $\mathrm{B}G$ is a groupoid. Exercise 2.5.3 shows that all the relevant equations hold, as required. □

def:abstrG Definition 4.2.5. Given a group G, the abstract group $\mathrm{abs}(G)$ of Lemma 4.2.4 is called the *abstract group associated to G*. ⌟

rem:inverses-as-property Remark 4.2.6. Instead of including the inverse operation as part (4) of the structure (including with the property (4) (a)), some authors assume the existence of inverses by positing the following property.

axiom:mere-inverse (5) for all $g : S$ there exists an element $h : S$ such that $e = g \cdot h$.

We will now compare (5) to (4). Property (5) contains the phrase "there exists", and thus its translation into type theory uses the quantifier $\exists$, as defined in Section 2.16. Under this translation, property (5) does not immediately allow us to speak of "the inverse of g". However, the following lemma shows that we can define an inverse operation as in (4) from a witness of (5) – its proof goes by using the properties (3) (a) and (3) (b) to prove that inverses are unique. As a consequence, we *can* speak "the inverse of g". ⌟

lem:group-inv-operation

Lemma 4.2.7. *Given a set S together with e and $\cdot$ as in Definition 4.2.1 satisfying the unit laws, the associativity law, and property (5), there is a "inverse" function $S \to S$ having property (4) (a) of Definition 4.2.1.*

Proof. Consider the function $\mu : S \to (S \to S)$ defined as $g \mapsto (h \mapsto g \cdot h)$. Let $g : S$. We claim that the fiber $\mu(g)^{-1}(e)$ is contractible. Contractibility is a proposition, hence to prove it from (5), one can as well assume the actual existence of h such that $g \cdot h = e$. Then $(h, !)$ is an element of the fiber $\mu(g)^{-1}(e)$. We will now prove that it is a center of contraction. For any other element $(h', !)$, we want to prove $(h, !) = (h', !)$, which is equivalent to the equation $h = h'$. In order to prove the latter, we show that h is also an inverse on the left of g, meaning that $h \cdot g = e$. This equation is also a proposition, so we can assume from (5) that we have an element $k : S$ such that $h \cdot k = e$. Multiplying that equation by g on the left, one obtains

$$k = e \cdot k = (g \cdot h) \cdot k = g \cdot (h \cdot k) = g \cdot e = g,$$

from which we see that $h \cdot g = e$. Now it follows that

$$h = h \cdot e = h \cdot (g \cdot h') = (h \cdot g) \cdot h' = e \cdot h' = h',$$

as required. Hence $\mu(g)^{-1}(e)$ is contractible, and we may define g^{-1} to be the center of the contraction, for any $g : S$. The function $g \mapsto g^{-1}$ satisfies the law of inverses (4) (a), as required. □

Note that the proof above also shows the other *law of inverses*: for all $g : S$ we have $g^{-1} \cdot g = e$.

rem:typemonoidabstrgp

Remark 4.2.8. We may encode the type of abstract groups as follows. We let S denote the underlying set, $e : S$ denote the unit, $\mu : S \to S \to S$ denote the multiplication operation $g \mapsto h \mapsto g \cdot h$, and $\iota : S \to S$ denote the inverse operation $g \mapsto g^{-1}$. Using that notation, we introduce names for the relevant propositions.

$$\begin{aligned}
\mathrm{UnitLaws}(S, e, \mu) &:\equiv \prod_{g:S} (\mu(g)(e) = g) \times (\mu(e)(g) = g) \\
\mathrm{AssocLaw}(S, \mu) &:\equiv \prod_{g_1, g_2, g_3 : S} \mu(g_1)(\mu(g_2)(g_3)) = \mu(\mu(g_1)(g_2))(g_3) \\
\mathrm{MonoidLaws}(S, e, \mu) &:\equiv \mathrm{isSet}(S) \times \mathrm{UnitLaws}(S, e, \mu) \times \mathrm{AssocLaw}(S, \mu) \\
\mathrm{InverseLaw}(S, e, \mu, \iota) &:\equiv \prod_{g:S} (\mu(g)(\iota(g)) = e) \\
\mathrm{GroupLaws}(S, e, \mu, \iota) &:\equiv \mathrm{MonoidLaws}(S, e, \mu) \times \mathrm{InverseLaw}(S, e, \mu, \iota)
\end{aligned}$$

Now we define the type of abstract groups in terms of those propositions.

$$\mathrm{Group}^{\mathrm{abs}} :\equiv \sum_{S:\mathcal{U}} \sum_{e:S} \sum_{\mu : S \to S \to S} \sum_{\iota : S \to S} \mathrm{GroupLaws}(S, e, \mu, \iota)$$

Thus, following the convention introduced in Remark 2.8.1, an abstract group G will be a quintuple of the form $G \equiv (S, e, \mu, \iota, !)$. For brevity, we will usually omit the proof of the properties from the display, since it's unique, and write an abstract group as though it were a quadruple $G \equiv (S, e, \mu, \iota)$. ⌟

Remark 4.2.9. That the concept of an abstract group synthesizes the idea of symmetries will be justified in Section 4.7 where we prove that the function $\mathrm{abs} : \mathrm{Group} \to \mathrm{Group}^{\mathrm{abs}}$, whose definition can be inferred from the proof of Lemma 4.2.4, is an equivalence. ⌟

Remark 4.2.10. If $\mathcal{G} = (S, e, \mu, \iota)$ and $\mathcal{G}' = (S', e', \mu', \iota')$ are abstract groups, an element of the identity type $\mathcal{G} = \mathcal{G}'$ consists of quite a lot of information, provided we interpret it by repeated application of Lemma 2.10.3. First and foremost, we need an identity $p : S = S'$ of sets, but from there on the information is a proof of a conjunction of propositions (this is more interesting for ∞-groups). An analysis shows that this conjunction can be shortened to the equations $e' = p(e)$ and $\mu'(p(s), p(t)) = p(\mu(s,t))$. A convenient way of obtaining an identity p that preserves the equations is to apply univalence to an equivalence $f : S \simeq S'$ that preserves the equations. Such a function will be called an abstract *isomorphism*. ⌟

xca:conj

Exercise 4.2.11. Let $\mathcal{G} = (S, e, \mu, \iota)$ be an abstract group and let $g : S$. If $s : S$, let $c^g(s) :\equiv g \cdot s \cdot g^{-1}$. Show that the resulting function $c^g : S \to S$ preserves the group structure (for instance $g \cdot (s \cdot s') \cdot g^{-1} = (g \cdot s \cdot g^{-1}) \cdot (g \cdot s \cdot g^{-1})$) and is an equivalence. The resulting identity $c^g : \mathcal{G} = \mathcal{G}$ is called *conjugation* by g. ⌟

Remark 4.2.12. Without the demand that the underlying type of an abstract group or monoid is a set, life would be more complicated. For instance, for the case when g is e, the unit laws (3) (a) of Definition 4.2.1 would provide *two* (potentially different) proofs that $e \cdot e = e$, and we would have to separately assume that they agree. This problem vanishes in the setup we adopt below for ∞-groups. ⌟

Exercise 4.2.13. For an element g in an abstract group, prove that $e = g^{-1} \cdot g$ and $g = (g^{-1})^{-1}$. In other words (for the machines among us), given an abstract group (S, e, μ, ι), give an element in the proposition

$$\prod_{g:S} (e = \mu(\iota(g))(g)) \times (g = \iota(\iota(g))).$$

⌟

xca:typemonoidisgroupoid

Exercise 4.2.14. Prove that the types of monoids and abstract groups are groupoids. ⌟

xca:cheapgroup

Exercise 4.2.15. There is a leaner way of characterizing what an abstract group is: define a *sheargroup* to be a set S together with an element $e : S$, a function $S \times S \to S$ sending $(a, b) : S \times S$ to $a * b : S$ and the following propositions where we use the shorthand $\bar{a} :\equiv a * e$:

(1) $e * a = a$,

(2) $a * a = e$ and

(3) $c * (b * a) = \overline{(c * \bar{b})} * a$,

for all $a, b, c : S$. Show that the type of abstract groups is equivalent to the type of sheargroups.

Hint: setting $a \cdot b :\equiv \bar{b} * a$ gives you an abstract group from a sheargroup and conversely, letting $a * b = b \cdot a^{-1}$ takes you back. On your way you may need at some point to show that $\bar{\bar{a}} = a$: setting $c = \bar{a}$ and $b = a$ in the third formula will do the trick (after you have established that $\bar{e} = e$). This exercise may be good to look back to in the many instances where the inverse inserted when "multiplying from the right by a" is forced by transport considerations. ⌟

Exercise 4.2.16. Another and even leaner way to define abstract groups, highlighting how we can do away with both the inverse and the unit: a *Furstenberg group* is a non-empty set S together with a function

$$S \times S \xrightarrow{(s,t)\mapsto s\circ t} S$$

with the property that

(1) for all $a, b, c : S$ we have that $(a \circ c) \circ (b \circ c) = a \circ b$

(2) far all $a, c : S$ there is a $b : S$ such that $a \circ b = c$.

Show that the type of Furstenberg groups is equivalent to the type of groups. ⌟

Named after Hillel Furstenberg who at the age of 20 published a paper in the Proceedings of the American Mathematical Society doing this exercise.

Hint: show that the function $a \mapsto a \circ a$ is constant, with value, say, e. Then show that S together with the "unit" e, "multiplication" $a \cdot b :\equiv a \circ (e \circ b)$ and "inverse" $b^{-1} :\equiv e \circ b$ is an abstract group

4.3 *Homomorphisms*

sec:homomorphisms
homom-idens

Remark 4.3.1. Let G and H be groups, and suppose we have a function $f : \mathrm{B}G \to \mathrm{B}H$. Suppose also, for simplicity, that $\mathrm{pt}_{\mathrm{B}H} \equiv f(\mathrm{pt}_{\mathrm{B}G})$, so the pair $(f, \mathrm{refl}_{\mathrm{pt}_{\mathrm{B}H}})$ is a pointed function of type $\mathrm{B}G \to_* \mathrm{B}H$. Applying Definition 2.6.1 yields a function $F :\equiv \mathrm{ap}_f : \mathrm{U}G \to \mathrm{U}H$, which satisfies the following identities.

$$\begin{aligned} F(\mathrm{refl}_{\mathrm{pt}_{\mathrm{B}G}}) &= \mathrm{refl}_{\mathrm{pt}_{\mathrm{B}H}} \\ F(g^{-1}) &= (F(g))^{-1} && \text{for any } g : \mathrm{U}G \\ F(g' \cdot g) &= F(g') \cdot F(g) && \text{for any } g, g' : \mathrm{U}G \end{aligned}$$

The first one is true by definition, the second can be proved by induction on g, and the third one follows from Lemma 2.6.2. These three identities assert that the function ap_f *preserves*, in a certain sense, the operations provided by Lemma 4.2.4 that make the abstract group abs(G) from UG and the abstract group abs(H) from UH. In the traditional study of abstract groups, these three identities play an important role and entitle one to call the function F a *homomorphism of abstract groups*. ⌟

A slight generalization of the discussion above will be to suppose that we have an identification of type $\mathrm{pt}_{\mathrm{B}H} = f(\mathrm{pt}_{\mathrm{B}G})$ not necessarily given by reflexivity. Indeed, that works out, thereby motivating the following definition.

def:grouphomomorphism

Definition 4.3.2. The type of *group homomorphisms* from G : Group to H : Group is defined to be

$$\mathrm{Hom}(G, H) :\equiv \mathrm{Copy}_{\underline{\Omega}}(BG \to_* BH),$$

i.e., it is a wrapped copy of the type of pointed maps of classifying spaces with constructor $\underline{\Omega} : (BG \to_* BH) \to \mathrm{Hom}(G, H)$. We again write $\mathrm{B} : \mathrm{Hom}(G, H) \to (BG \to_* BH)$ for the destructor, and we call Bf *classifying map* of the homomorphism. ⌟

Sometimes we write $f : G \to_{\mathrm{Group}} H$ instead of f : Hom(G, H) to emphasize that f is a map of groups. When it is clear from context that a homomorphism is intended, we even write $f : G \to H$.

We would like to understand the effect of a homomorphism f : Hom(G, H) on the underlying symmetries UG, UH. Since the underlying symmetries are obtained by taking loops, we should first study how pointed maps affect loops:

def:loops-map

CONSTRUCTION 4.3.3. *Given pointed types X and Y and a pointed function $f : X \to_* Y$ (as defined in Definition 2.21.1), we construct a function $\Omega f : \Omega X \to \Omega Y$.*

Implementation of Construction 4.3.3. We write $X \equiv (A, a)$, $Y \equiv (B, b)$, and $f \equiv (p, e)$, where $p : X \to Y$ and $e : b = p(a)$. By induction on e we are reduced to the case where $b \equiv p(a)$, so we may define $\Omega f :\equiv \mathrm{ap}_p$ (see Definition 2.6.1). □

def:USym-hom

DEFINITION 4.3.4. Given groups G and H and a homomorphism $f : \mathrm{Hom}(G, H)$, we define a function $\mathrm{U}f : \mathrm{U}G \to \mathrm{U}H$ by setting $\mathrm{U}f :\equiv \Omega \mathrm{B}f$. In other words, the homomorphism $\underline{\Omega}Bf$ induces ΩBf as the map on underlying symmetries. ⌟

LEMMA 4.3.5. *Given groups G and H and a homomorphism $f : \mathrm{Hom}(G, H)$, the function $\mathrm{U}f : \mathrm{U}X \to \mathrm{U}Y$ defined above satisfies the following identities.*

{gp-homo-unit}
{gp-homo-comp}
{gp-homo-inv}

$$(\mathrm{U}f)(\mathrm{refl}_{\mathrm{pt}_{\mathrm{B}G}}) = \mathrm{refl}_{\mathrm{pt}_{\mathrm{B}H}} \tag{4.3.1}$$

$$(\mathrm{U}f)(g^{-1}) = ((\mathrm{U}f)(g))^{-1} \qquad \text{for any } g : \mathrm{U}G \tag{4.3.2}$$

$$(\mathrm{U}f)(g' \cdot g) = (\mathrm{U}f)(g') \cdot (\mathrm{U}f)(g) \qquad \text{for any } g, g' : \mathrm{U}G \tag{4.3.3}$$

Proof. We write $f \equiv (p, e)$, where $p : \mathrm{B}G \to \mathrm{B}H$ and $e : \mathrm{pt}_{\mathrm{B}H} = p(\mathrm{pt}_{\mathrm{B}G})$. By induction on e, we reduce to the case where $\mathrm{pt}_{\mathrm{B}H} \equiv p(\mathrm{pt}_{\mathrm{B}G}$. We finish by applying Remark 4.3.1. □

def:groupisomorphism

DEFINITION 4.3.6. A homomorphism $f : G \to_{\mathrm{Group}} H$ is an *isomorphism* if its classifying map Bf is an equivalence. ⌟

def:identity-group-homomorphism

DEFINITION 4.3.7. If G is a group, then we use Definition 2.21.2 to define the *identity homomorphism* $\mathrm{id}_G : G \to_{\mathrm{Group}} G$ by setting $\mathrm{id}_G :\equiv \underline{\Omega}(\mathrm{id}_{\mathrm{B}G})$. It is an isomorphism. ⌟

def:group-homomorphism-composition

DEFINITION 4.3.8. If G, G', and G'' are groups, and $f : G \to_{\mathrm{Group}} G'$ and $f' : G' \to_{\mathrm{Group}} G''$ are homomorphisms, then we use the definition of composition of pointed functions in Definition 2.21.1 to define the *composite homomorphism* $f' \circ f : G \to_{\mathrm{Group}} G''$ by setting $f' \circ f :\equiv \underline{\Omega}(\mathrm{B}f' \circ \mathrm{B}f)$. ⌟

Recall from Section 2.21, that when there is little danger of confusion, we may drop the subscript "÷" when talking about the unpointed structure.

rem:Bf-convention

REMARK 4.3.9. To construct a function $\varphi : \prod_{f : \mathrm{Hom}(G,H)} T(f)$, where $T(f)$ is a family of types parametrized by $f : \mathrm{Hom}(G, H)$, it suffices to consider the case $f \equiv \underline{\Omega}Bf$.[12]

Identifications of homomorphisms $f =_{\mathrm{Hom}(G,H)} h$ are equivalent to identifications of pointed maps $Bf =_{BG \to_* BH} Bh$; the latter are (by Lemma 2.10.3) given by pairs of an identification of (unpointed) maps $H : Bf_\div =_{(BG_\div \to BH_\div)} Bh_\div$ with an identification $K : H(\mathrm{sh}_G) Bf_0 =_{(\mathrm{sh}_H = Bh(\mathrm{sh}_G))} Bh_0$. ⌟

ex:groups-morphisms

EXAMPLE 4.3.10.

(1) Consider two sets S and T. Recall from Example 4.1.18 that $\mathrm{Set}_{(S)} :\equiv \sum_{X : \mathrm{Set}} \|S = X\|$ is the component of the groupoid Set containing S, and when pointed at S represents the permutation group Σ_S. The map $\_ \amalg T : \mathrm{Set}_{(S)} \to \mathrm{Set}_{(S \amalg T)}$ sending X to $X \amalg T$ induces a group

[12] We use the same notational convention regarding "B" applied to homomorphisms as we do for groups.

homomorphism $\Sigma_S \to \Sigma_{S \amalg T}$, pointed by the path $\mathrm{refl}_{S\amalg T} : S \amalg T = (\_ \amalg T)(S)$. Thought of as symmetries, this says that if you have a symmetry of S, then we get a symmetry on $S \amalg T$ (which doesn't do anything to T).

Likewise, we get a map $\_ \times T : \mathrm{Set}_{(S)} \to \mathrm{Set}_{(S\times T)}$ sending X to $X \times T$ induces a group homomorphism $\Sigma_S \to \Sigma_{S\times T}$, pointed by the path $\mathrm{refl}_{S\times T} : S \times T = (\_ \times T)(S)$.

In particular, we get homomorphisms $\Sigma_m \to \Sigma_{m+n}$ and $\Sigma_m \to \Sigma_{mn}$. [13]

(2) Let G be a group. Since there is a unique map from BG to $\mathbb{1}$ (obviously pointed by the reflexivity path of the unique element of $\mathbb{1}$), we get a unique homomorphism from G to the trivial group. Likewise, there is a unique morphism from the trivial group to G, sending the unique element of $\mathbb{1}$ to sh_G, and pointed by $\mathrm{refl}_{\mathrm{sh}_G}$.

(3) If G and H are groups, the projections $BG \leftarrow BG \times BH \to BH$ and inclusions $BG \to BG \times BH \leftarrow BH$ (e.g., the inclusion $BG \to BG \times BH$ is given by $z \mapsto (z, \mathrm{sh}_H)$) give rise to group homomorphisms between $G \times H$ and G and H.

the "projections" $G \leftarrow G \times H \to H$ and "inclusions" $G \to G \times H \leftarrow H$ are homomorphisms

(4) In Lemma 4.1.23 we gave an example of an isomorphism, namely one from the cyclic group C_m to $\mathbb{Z}/m$.

⌟

Remark 4.3.11. In the examples above, we insisted on writing the path pointing a group homomorphism, even when this path was a reflexivity path. We now adopt the convention that there is no need to specify the path in this case. Thus, given a map $f : A \to B$ between connected groupoids and $a : A$, the group homomorphism $\mathrm{Aut}_A(a) \to \mathrm{Aut}_B(f(a))$ defined by $(f, \mathrm{refl}_{f(a)})$ will simply be referred to as f.

However, it is important to understand that different homomorphisms can have the same underlying unpointed function. Consider, for example, the group Σ_3, whose classifying space is $\mathrm{B}\Sigma_3 :\equiv (\mathrm{FinSet}_3, \mathbb{3})$, and the path $\tau : \mathrm{U}\Sigma_3$ that is defined (through univalence) as

$$0 \mapsto 1, \quad 1 \mapsto 0, \quad 2 \mapsto 2$$

Then the function $\mathrm{id} : \mathrm{FinSet}_3 \to \mathrm{FinSet}_3$ gives rise to two elements of $\mathrm{Hom}(\Sigma_3, \Sigma_3)$: the first one is $(\mathrm{id}, \mathrm{refl}_3)$, which is simply denoted id_{Σ_3}; the second one is (id, τ), that we will denote $\tilde{\tau}$ temporarily. Let us prove $\mathrm{id}_{\Sigma_3} \neq \tilde{\tau}$, that is suppose a path $\mathrm{id}_{\Sigma_3} = \tilde{\tau}$ and derive a contradiction. Such a path is the data of a path $p : \mathrm{id} = \mathrm{id}$ such that $\mathrm{refl}_3 =^T_p \tau$ where the type family $T : (\mathrm{FinSet}_3 \to \mathrm{FinSet}_3) \to \mathcal{U}$ is given by $f \mapsto (\mathrm{sh}_{\Sigma_3} = f(\mathrm{sh}_{\Sigma_3}))$. It is easy to determine the transport in that type family T and we find that $(\mathrm{refl}_3 =^T_p \tau) \simeq (p(\mathrm{sh}_{\Sigma_3}) = \tau)$. Now, by induction on $q : \mathrm{id} = f$ for $f : \mathrm{FinSet}_3 \to \mathrm{FinSet}_3$, one shows that

$$\mathrm{ap}_f : (\mathrm{U}\Sigma_3) \to (f(\mathrm{sh}_{\Sigma_3}) = f(\mathrm{sh}_{\Sigma_3}))$$

is equal to $q(\mathrm{sh}_{\Sigma_3}) \cdot \_ \cdot q(\mathrm{sh}_{\Sigma_3})^{-1}$. In particular, when $f \equiv \mathrm{id}$ and $q \equiv p$, it means that conjugating by $p(\mathrm{sh}_{\Sigma_3})$ is trivial. But by hypothesis $p(\mathrm{sh}_{\Sigma_3}) = \tau$, so it means that τ commutes with every other element of $\mathrm{U}\Sigma_3$. One

[13] find a good description of the sign $\Sigma_n \to \Sigma_2$

can check that it actually fails to be the case for the element θ defined by

$$0 \mapsto 0, \quad 1 \mapsto 2, \quad 2 \mapsto 1.$$

Indeed, $\theta\tau(0) = 2$ while $\tau\theta(0) = 1$. (See also Exercise 4.1.28.) ⌟

def:loops-compose CONSTRUCTION 4.3.12. *For pointed types X, Y, Z and pointed maps $f : X \to_* Y$ and $g : Y \to_* Z$, we get an identification of type*

$$\Omega(g \circ f) =_{(\Omega X \to \Omega Z)} \Omega(g) \circ \Omega(f).$$

Implementation of Construction 4.3.12. Let x denote the base point of X. By induction on f_0 and on g_0, we reduce to the case where $f_0 \equiv \mathrm{refl}_{f(x)}$ and $g_0 \equiv \mathrm{refl}_{g(f(x))}$, and it suffices to identify $\mathrm{ap}_{g\circ f}$ with $\mathrm{ap}_g \circ \mathrm{ap}_f$. By Principle 2.9.17, it suffices to identify $\mathrm{ap}_{g\circ f}(p)$ with $\mathrm{ap}_g(\mathrm{ap}_f(p))$ for each $p : \Omega X$. For that purpose, it suffices to prove the stronger statement that $\mathrm{ap}_{g\circ f}(p) = \mathrm{ap}_g(\mathrm{ap}_f(p))$ for any $x' : X$ and any $p : x = x'$. By induction on p, it suffices to prove that $\mathrm{ap}_{g\circ f}(\mathrm{refl}_x) = \mathrm{ap}_g(\mathrm{ap}_f(\mathrm{refl}_x))$, and that can be done by observing that both sides are equal, by definition, to $\mathrm{refl}_{g(f(x))}$. □

cor:USym-compose COROLLARY 4.3.13. *For composable group homomorphisms φ : $\mathrm{Hom}(G, H)$, ψ : $\mathrm{Hom}(H, K)$, we get an identification $\mathrm{U}(\psi \circ \varphi) = \mathrm{U}\psi \circ \mathrm{U}\varphi$.*

EXAMPLE 4.3.14. We will later show that if G and H are groups, then $\mathrm{Hom}(G, H)$ is equivalent to the *set* of "abstract group homomorphisms" from $\mathrm{abs}(G)$ to $\mathrm{abs}(H)$ (cf. Lemma 4.8.1), but it is instructive to prove that $\mathrm{Hom}(G, H)$, or equivalently

$$(BG \to_* BH) :\equiv \sum_{F : BG_\div \to BH_\div} (\mathrm{sh}_H = F(\mathrm{sh}_G)),$$

is a set directly from the definition. Recall our notation: a homomorphism f : $\mathrm{Hom}(G, H)$ is recorded as the pair

$$Bf \equiv (Bf_\div, p_f) : \sum_{F : BG_\div \to BH_\div} (\mathrm{sh}_H = F(\mathrm{sh}_G)).$$

Let us spell out the data needed to give an identity between two group homomorphisms f, f' : $\mathrm{Hom}(G, H)$. We clearly must have a

$$J : Bf_\div = Bf'_\div,$$

which by function extensionality (Principle 2.9.17) is equivalently given by its image given by the element (with the same name) in the Π-type

$$J : \prod_{z : BG_\div} Bf_\div(z) = Bf'_\div(z).$$

$$\begin{tikzcd}
 & \mathrm{sh}_H \arrow[dl, equal, "p_f"', "\swarrow"] \arrow[dr, equal, "p_{f'}", "\searrow"'] & \\
Bf_\div(\mathrm{sh}_G) \arrow[rr, equal, "J(\mathrm{sh}_G)", "\rightarrow"'] & & Bf'_\div(\mathrm{sh}_G).
\end{tikzcd}$$

Induction on $\gamma : \mathrm{sh}_G = z$ gives

$$\begin{tikzcd}
Bf_\div(\mathrm{sh}_G) \arrow[r, equal, "J(\mathrm{sh}_G)", "\rightarrow"'] \arrow[d, equal, "Bf_\div(\gamma)"', "\downarrow"] & Bf'_\div(\mathrm{sh}_G) \arrow[d, equal, "Bf'_\div(\gamma)"', "\downarrow"] \\
Bf_\div(z) \arrow[r, equal, "J(z)", "\rightarrow"'] & Bf'_\div(z).
\end{tikzcd}$$

Together these two commutative diagrams (and the fact that we're in a set) show that $(J, !)$ is unique.

Transport along J of $p_f : \mathrm{sh}_H = Bf_\div(\mathrm{sh}_G)$ shows that in addition we must have a path $! : J(\mathrm{sh}_G) \cdot p_f = p_{f'}$ in the type $\mathrm{sh}_H = Bf'_\div(\mathrm{sh}_G)$. In other words, we have an equivalence

$$(f = f') \simeq \sum_{J : Bf_\div = Bf'_\div} J(\mathrm{sh}_G) p_f = p_{f'},$$

and our goal is to prove that any two elements are identical. Take two elements $(J, !), (K, !) : \sum_{J : Bf_\div = Bf'_\div} J(\mathrm{sh}_G) p_f = p_{f'}$. We must prove that $(J, !) = (K, !)$ has an element. Indeed, this type is equivalent to $J = K$, which is in turn equivalent to $\prod_{z:BG_\div} J(z) = K(z)$, and because $Bf_\div(z) = Bf'_\div(z)$ is a set for every z ($BH_\div$ being a groupoid), the type $J(z) = K(z)$ is a proposition. Hence, with the propositional goal $(J, !) = (K, !)$, one can now use connectedness of $BG_\div$, and only check the equality on the point sh_G. By definition,

$$J(\mathrm{sh}_G) = p_{f'} p_f^{-1} = K(\mathrm{sh}_G).$$

This concludes the proof that $f = f'$ is a proposition, or in other words that $\mathrm{Hom}(G, H)$ is a set.

⌟

The following example expresses that $\mathbb{Z}$ is a "free group with one generator".

ex:Zinitial

Example 4.3.15. Chapter 3 was all about the circle S^1 and its role as a "universal symmetry" and how it related to the integers. In our current language, $\mathbb{Z} :\equiv \mathrm{Aut}_{S^1}(\bullet)$ and large parts of the universality is found in the following observation. If G is a group then the evaluation equivalence $\mathrm{ev}_{BG_\div} : (S^1 \to BG_\div) \xrightarrow{\sim} \sum_{y:BG_\div}(y = y)$ of Theorem 3.1.2 yields an equivalence of sets

$$\mathrm{ev}_{BG} : \big((S^1, \bullet) \to_* BG\big) \xrightarrow{\sim} (\mathrm{U}G) : (f, f_0) \mapsto f_0^{-1} f(\circlearrowleft) f_0.$$

The domain of this equivalence ev_{BG} is nothing but the definition of $\mathrm{Hom}(\mathbb{Z}, G)$. Hence, ev_{BG} provides a way to identify $\mathrm{Hom}(\mathbb{Z}, G)$ with the abstract group $\mathrm{U}G$. Like in Theorem 3.1.2, the inverse of ev_{BG} is denoted ve_{BG} and satisfies $\mathrm{ve}_{BG}(g)(\bullet) \equiv \mathrm{sh}_G$, $\mathrm{ve}_{BG}(g)(\circlearrowleft) = g$. Moreover, $\mathrm{ve}_{BG}(g)$ is pointed by $\mathrm{refl}_{\mathrm{sh}_G}$. ⌟

The following lemma states "the naturality of ev_{BG} in the previous example". (DISCUSS AMONG US)

lem:Znatural

Lemma 4.3.16. *Let G and H be groups and $f : \mathrm{Hom}(G, H)$. Then the following diagram commutes,*

$$\begin{tikzcd}
\mathrm{Hom}(\mathbb{Z}, G) \arrow[r, "\mathrm{ev}", "\simeq"'] \arrow[d, "f \circ \_"'] & \mathrm{U}G \arrow[d, "\mathrm{U}f"] \\
\mathrm{Hom}(\mathbb{Z}, H) \arrow[r, "\mathrm{ev}", "\simeq"'] & \mathrm{U}H,
\end{tikzcd}$$

where the horizontal maps evaluate the map on underlying symmetries at the loop $\circlearrowleft : \mathrm{U}\mathbb{Z}$.

Proof. Let $k : \mathrm{Hom}(\mathbb{Z}, G)$, giving $\mathrm{U}k : \mathrm{U}\mathbb{Z} \to \mathrm{U}G$. Going across horizontally and then down, k is mapped first to $\mathrm{U}k(\circlearrowleft)$, and then to $\mathrm{U}f(\mathrm{U}k(\circlearrowleft))$. Going the other way takes k to $\mathrm{U}(f \circ k)(\circlearrowleft)$, which is equal to $\mathrm{U}f(\mathrm{U}k(\circlearrowleft))$ by Corollary 4.3.13. □

xca:BGtotype

Exercise 4.3.17. Let G be a group and A a groupoid. Use the definitions and Exercise 2.21.4 to show that the types

(1) $BG_\div \to A$,

(2) $\sum_{a:A} \sum_{f:BG_\div \to A} (a = f(\mathrm{sh}_G))$,

(3) $\sum_{a:A} (BG \to_* (A, a))$ and

(4) $\sum_{a:A} \mathrm{Hom}(G, \mathrm{Aut}_A(a))$

are all equivalent. ⌟

The definition of group homomorphisms in Definition 4.3.2 should be contrasted with the usual – and somewhat more cumbersome – notion of a group homomorphism $f : \mathcal{G} \to \mathcal{H}$ of abstract groups where we must specify that in addition to preserving the neutral element "$f(e_G) = e_H$" it must preserve multiplication: "$f(g) \cdot_{\mathcal{H}} f(g') = f(g \cdot_{\mathcal{G}} g')$ (where we have set the name of the abstract group as a subscript to e and $\cdot$). In our setup this is simply true, as we record in the following remark.

def:grouphomomaxioms

Remark 4.3.18. Let G and H be groups and assume given a group homomorphism $f : G \to H$. We now define something that we will later call an "abstract group homomorphism $\mathrm{abs}(f) : \mathrm{abs}(G) \to \mathrm{abs}(H)$", i.e., a function of sets from $\mathrm{U}G$ to $\mathrm{U}H$ "preserving" the abstract group structure, cf. Definition 4.2.5 for the definition of $\mathrm{abs}(G)$ and Definition 4.3.19 for a condensation of what the discussion below end up with concluding that "preserves" means.

Recall that such an f : $\mathrm{Hom}(G, H)$ is recorded as a pair

$$(Bf_\div, p_f) : \sum_{F : BG_\div \to BH_\div} (\mathrm{sh}_H = F(\mathrm{sh}_G)),$$

and recall the function

$$\mathrm{U}f : \mathrm{U}G \to \mathrm{U}H$$

defined in Definition 4.3.4.

We take the time to develop from first principles the properties that $\mathrm{U}f$ satisfies. One proves easily (by induction) that $\mathrm{trp}_{p_f^{-1}} = p_f^{-1} \cdot \_ \cdot p_f$. It means that the element $\mathrm{U}f(g)$ is the "up, over and down" identity of sh_H depicted in the following diagram:

$$\begin{tikzcd}
Bf_\div(\mathrm{sh}_G) \arrow[r, equal, "\mathrm{ap}_{Bf_\div}(g)"] & Bf_\div(\mathrm{sh}_G) \\
\mathrm{sh}_H \arrow[u, equal, "p_f"] \arrow[r, dotted, "\mathrm{U}f(g)"] & \mathrm{sh}_H \arrow[u, equal, "p_f"'].
\end{tikzcd}$$

With the shorthand

$$e_G :\equiv \mathrm{refl}_{\mathrm{sh}_G} : \mathrm{U}G$$

and writing (to remind us where things happen)

$$g \cdot_G g' : \mathrm{U}G$$

for the composite $g\,g'$ of g and g' (note that we use functional notation, so that composition is "first g' and then g" as in the picture $\mathrm{sh}_G \xrightarrow{g'} \mathrm{sh}_G \xrightarrow{g} \mathrm{sh}_G$) and likewise with a subscript H, we have the following statements.

(1) the proposition $\mathrm{U}f(e_G) = e_H$ holds by Eq. (4.3.1)

(2) the proposition $\mathrm{U}f(g \cdot_G g') = \mathrm{U}f(g) \cdot_H \mathrm{U}f(g')$ holds by Eq. (4.3.2)

⌟

def:abstrisfunctor

Definition 4.3.19. If $\mathcal{G} :\equiv (S, e_{\mathcal{G}}, \cdot_{\mathcal{G}}, \iota_{\mathcal{G}})$ and $\mathcal{H} :\equiv (T, e_{\mathcal{H}}, \cdot_{\mathcal{H}}, \iota_{\mathcal{H}})$ are two abstract groups, then the set of homomorphisms from $\mathcal{G}$ to $\mathcal{H}$ is

$$\mathrm{Hom}^{\mathrm{abs}}(\mathcal{G}, \mathcal{H}) :\equiv \sum_{f : S \to T} (e_{\mathcal{H}} =_T f(e_{\mathcal{G}})) \times \prod_{s, s' : S} f(s \cdot_{\mathcal{G}} s') =_T f(s) \cdot_{\mathcal{H}} f(s').$$

For $s : S$ both $(e_{\mathcal{H}} = f(e_{\mathcal{G}}))$ and $f(s \cdot_{\mathcal{G}} s') =_T f(s) \cdot_{\mathcal{H}} f(s')$ are propositions; hence a homomorphism of abstract groups is uniquely determined by its underlying function of sets, and unless there is danger of confusion we may write f instead of $(f, !)$.

If G and H are groups, the function

$$\mathrm{abs}\colon \mathrm{Hom}(G,H) \to \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$$

is defined as the function $f \mapsto \mathrm{abs}(f) :\equiv (\mathrm{U}f, !)$ made explicit in Remark 4.3.18. ⌟

$\mathrm{U}f :\equiv \mathrm{trp}_{p_f^{-1}}\, \mathrm{ap}_{Bf_\div} : \mathrm{U}G = \mathrm{U}H$, and so

$$\begin{tikzcd} Bf_\div(\mathrm{sh}_G) \arrow[r, equal, "\mathrm{ap}_{Bf_\div}(g)", "\to"'] & Bf_\div(\mathrm{sh}_G) \\ \mathrm{sh}_H \arrow[u, equal, "p_f", "\uparrow"'] \arrow[r, equal, "\mathrm{U}f(g)", "\to"'] & \mathrm{sh}_H . \arrow[u, equal, "p_f", "\uparrow"'] \end{tikzcd}$$

commutes.

Exercise 4.3.20. Note that the inverses play no rôle in the definition of a homomorphism of abstract groups. Prove that if $(f, !)\colon \mathrm{Hom}^{\mathrm{abs}}(\mathcal{G}, \mathcal{H})$ then the proposition $f(g^{-1}) = (f(g))^{-1}$ for all $g : \mathcal{G}$, so that we don't have to require it separately. ⌟

ex:conjhomo

Example 4.3.21. Let $\mathcal{G} = (S, e, \mu, \iota)$ be an abstract group and let $g : S$. In Exercise 4.2.11 we defined $c^g : S \to S$ by setting $c^g(s) :\equiv g \cdot s \cdot g^{-1}$ for $s : S$ and asked you to show that it "preserves the group structure", i.e., represents a homomorphism

$$c^g : \mathrm{Hom}^{\mathrm{abs}}(\mathcal{G}, \mathcal{G})$$

called *conjugation* by g. Actually, we asked more: namely that conjugation represents an identity (for which we used the same symbol) $c^g : \mathcal{G} = \mathcal{G}$.

If $\mathcal{H}$ is some other abstract group, transport along c^g gives an identity $c^g_* : \mathrm{Hom}(\mathcal{H}, \mathcal{G}) = \mathrm{Hom}(\mathcal{H}, \mathcal{G})$ which should be viewed as "postcomposing with conjugation". (Naturally, similar considerations goes for elements in $\mathcal{H}$, giving rise to "precomposition with conjugation".) ⌟

Exercise 4.3.22. Prove that composition of the functions on the underlying sets gives a composition of homomorphisms of abstract groups.

Prove that if $f_0 : \mathrm{Hom}(G_0, G_1)$ and $f_1 : \mathrm{Hom}(G_1, G_2)$ then

$$\mathrm{abs}(f_1 f_0) = \mathrm{abs}(f_1)\mathrm{abs}(f_0)$$

and that $\mathrm{abs}(\mathrm{id}_G) = \mathrm{id}_{\mathrm{abs}(G)}$. ⌟

4.4 ∞-*groups*

sec:inftygps

Disregarding the set-condition we get the simpler notion of ∞-groups:

def:inftygps

Definition 4.4.1. The type of ∞-groups is

$$\infty\mathrm{Group} :\equiv \mathrm{Copy}(\mathcal{U}_*^{>0}), \quad \text{where} \quad \mathcal{U}_*^{>0} :\equiv \sum_{A:\mathcal{U}} \sum_{a:A} \prod_{x:A} \|x =_A a\|,$$

is the type of pointed, connected types.

We again call the constructor $\underline{\Omega} : \mathcal{U}_*^{>0} \to \infty\mathrm{Group}$ and the destructor $\mathrm{B} : \infty\mathrm{Group} \to \mathcal{U}_*^{>0}$. ⌟

rem:pointedtypes

Remark 4.4.2. Just as "group" is a synonym for "pointed, connected groupoid" (wrapped with $\underline{\Omega}$), "∞-group" is a synonym for "pointed, connected type" (wrapped with $\underline{\Omega}$). As for pointed, connected groupoids, we suppress the propositional information from the notation, and write (A, a) instead of $(A, a, !)$ for an pointed, connected type. ⌟

def:classifyingspace

Definition 4.4.3. If $G \equiv \underline{\Omega}BG : \infty\mathrm{Group}$, then the underlying pointed type $BG : \mathcal{U}_*$ is still called the *classifying type* and $\mathrm{sh}_G :\equiv \mathrm{pt}_{BG}$ is the *base point*. ⌟

Definition 4.4.4. A homomorphism of ∞-groups is a pointed function of classifying types, i.e., given two ∞-groups G and H,we define

$$\mathrm{Hom}(G, H) :\equiv \mathrm{Copy}(BG \to_* BH).$$

If $f \equiv \underline{\Omega} Bf : \mathrm{Hom}(G, H)$, then we call $Bf : BG \to_* BH$ the *classifying map*. ⌟

4.5 *G-sets*

sec:gsets

One of the goals of Section 4.7 is to prove that the tyoes of groups and abstract groups are equivalent. In doing that, we are invited to explore how abstract groups should be thought of as symmetries and introduce the notion of a G-set. However, this takes a pleasant detour where we have to explore the most important feature of groups: they *act* on things (giving rise to symmetries)!

Before we handle the more complex case of abstract groups, let us see what this looks like for groups.

DEFINITION 4.5.1. For G a group, a *G-set* is a function

$$X : BG \to \mathrm{Set},$$

and $X(\mathrm{sh}_G)$ is referred to as the *underlying set*. If $p : x = y$ in BG, then the transport function $X(x) \to X(y)$ induced by $X(p) :\equiv \mathrm{ap}_X(p) : X(x) = X(y)$ is also denoted by $X(p)$. We denote $X(p)(a)$ by $p \cdot_X a$. The operation $\cdot_X$ is called the *group action* of X. When X is clear from the context we may leave out the subscript X.[14] In particular, if $g : UG$, then $X(g)$ is a permutation of the underlying set of X.

The type of G-sets is

$$G\text{-Set} :\equiv (BG \to \mathrm{Set}).$$

⌟

REMARK 4.5.2. The reader will notice that the type of G-sets is equivalent to the type of set bundles over BG. The reason we have allowed ourselves two names is that our focus is different: for a G-set $X : BG \to \mathrm{Set}$ we focus on the sets $X(z)$, whereas when talking about set bundles the first projection $\sum_{z:BG} X(z) \to BG$ takes center stage. Each focus has its advantages. ⌟

def:principaltorsor

EXAMPLE 4.5.3. If G is a group, then

$$\mathrm{Pr}_G : BG \to \mathrm{Set}, \qquad \mathrm{Pr}_G(z) :\equiv \mathrm{P}_{\mathrm{sh}_G}(z) :\equiv (\mathrm{sh}_G = z)$$

is a G-set called the *principal G-torsor*. We've seen this family before in the guise of the (preimages of the) "universal set bundle" of Definition 3.3.9!

There is nothing sacred about starting the equality $\mathrm{sh}_G = z$ in sh_G. Let

$$\mathrm{P}\_ : BG \to G\text{-Set}$$

denote the map sending $y : BG$ to the G-set P_y. Applying $\mathrm{P}\_$ to a path $q : y = y'$ induces an equivalence from P_y to $\mathrm{P}_{y'}$ that sends $p : y = z$ to $pq^{-1} : y' = z$. As a matter of fact, Theorem 4.6.6 will identify BG with the type of G-torsors via the map P_, simply denoted as P, using the full transport structure of the identity type $\mathrm{P}_y(z) :\equiv (y = z)$. ⌟

def:adjointrep

EXAMPLE 4.5.4. If G is a group (or ∞-group), then

$$\mathrm{Ad}_G : BG \to \mathcal{U}, \qquad \mathrm{Ad}_G(z) :\equiv (z = z)$$

Much of what follows will work equally well for ∞-groups; if G is an infinity group, a *G-type* is a function $X : BG \to \mathcal{U}$.

The term "G-torsor" will reappear several times and will mean nothing but a G-type in the component of Pr_G – a "twisted" version of Pr_G.

The name "adjoint" comes from how transport works in this case; if $p : y = z$, then $\mathrm{Ad}_G(p) : (y = y) = (z = z)$ is given by conjugation:

$$\mathrm{Ad}_G(p)(q) = pqp^{-1} : z = z,$$

the picture

$$\begin{tikzcd}
y \arrow[r, equal, "p", "\rightarrow"'] \arrow[d, equal, "q"', "\downarrow"] & z \arrow[d, equal, "\downarrow"', "\mathrm{Ad}_G(p)(q)"] \\
y \arrow[r, equal, "p", "\rightarrow"'] & z
\end{tikzcd}$$

is a mnemonic device illustrating that it couldn't have been different, and should be contrasted with the picture for $\mathrm{Pr}_G(p) : (\mathrm{sh}_G = y) = (\mathrm{sh}_G = z)$:

$$\begin{tikzcd}
\mathrm{sh}_G \arrow[r, equal, "\mathrm{refl}_{\mathrm{sh}_G}", "\rightarrow"'] \arrow[d, equal, "q"', "\downarrow"] & \mathrm{sh}_G \arrow[d, equal, "\downarrow"', "\mathrm{Pr}_G(p)(q)"] \\
y \arrow[r, equal, "p", "\rightarrow"'] & z.
\end{tikzcd}$$

[14] Note that in this case $\cdot : (x = y) \to X(x) \to X(y)$. See Example 4.5.3 for a special case where $\cdot_X$ is indeed path composition.

is a G-set (or G-type) called the *adjoint G-set* (*or G-type*). Notice that by the induction principle for the circle,

$$\sum_{z\,:\,BG} \mathrm{Ad}_G(z) = \sum_{z\,:\,BG} (z = z)$$

is equivalent to the type of (unpointed!) maps $\mathrm{S}^1 \to BG$, known in other contexts as the *free loop space* of BG, an apt name given that it is the type of "all symmetries in BG." The first projection $\sum_{z\,:\,BG} \mathrm{Ad}_G(z) \to BG$ correspond to the function $(\mathrm{S}^1 \to BG) \to BG$ given by evaluating at $\bullet$. ⌟

ex:HomHGasGset

Example 4.5.5. Recall that a homomorphism $f : \mathrm{Hom}(H, G)$ consists of an unpointed map $F : BH_\div \to BG_\div$ together with a $p_f : \mathrm{sh}_G = F(\mathrm{sh}_H)$, so if, for $x : BH$ and $y : BG$, we define

$$\mathrm{Hom}(H,G)(x,y) :\equiv \sum_{F\,:\,BH_\div \to BG_\div} (y = F(x))$$

we see that $\mathrm{Hom}(H, G)$ may be considered to be a $H \times G$-set

$$\mathrm{Hom}(H,G) : BH \times BG \to \mathrm{Set}.$$

We will be particularly interested in the restriction to G, giving a G-set for which we recycle the notation:

$$\mathrm{Hom}(H,G)(y) :\equiv \mathrm{Hom}(H,G)(\mathrm{sh}_H, y) :\equiv \sum_{F\,:\,BH_\div \to BG_\div} (y = F(\mathrm{sh}_H)).$$

⌟

xca:HomZGvsAdG

Exercise 4.5.6. Provide an identification between the G-sets Ad_G and $\mathrm{Hom}(\mathbb{Z}, G)$ of Example 4.5.4 and Example 4.5.5. ⌟

Hint: This is similar to Example 4.3.15: identify $\mathrm{Hom}(\mathbb{Z}, G)(y)$ with $\sum_{z\,:\,BG} \sum_{p\,:\,z=z} (y = z)$ and consider the map to $y = y$ sending (z, p, q) to $q^{-1} p\, q$.

def:trivGset

Example 4.5.7. If G is a group and X is a set, then

$$\mathrm{triv}_G X(z) :\equiv X$$

is a G-set. Examples of this sort (regardless of X) are called *trivial G-sets*. ⌟

remark:GsetsareGsets

Remark 4.5.8. A G-set X is often presented by focusing on the underlying set $X(\mathrm{sh}_G)$ and providing it with a structure relating it to G determining the entire function $X : BG \to \mathrm{Set}$.

More precisely, since BG is connected, a G-set $X : BG \to \mathrm{Set}$ factors over the component $\mathrm{Set}_{(X(\mathrm{sh}_G))} :\equiv \sum_{Y\,:\,\mathrm{Set}} \lVert X(\mathrm{sh}_G) = Y \rVert$ which contains the point $X(\mathrm{sh}_G)$. Since $B\Sigma_{X(\mathrm{sh}_G)} :\equiv (\mathrm{Set}_{(X(\mathrm{sh}_G))}, X(\mathrm{sh}_G))$ the G-set X can, without loss of information, be considered as a homomorphism from G to the permutation group $\Sigma_{X(\mathrm{sh}_G)}$ of $X(\mathrm{sh}_G)$, that is, a pointed map

$$BG \to_* \mathrm{B}\Sigma_{X(\mathrm{sh}_G)}.$$

Conversely, if X is any set *and* we have a homomorphism from G to Σ_X, i.e., a pointed map $(f, p) : BG \to_* \mathrm{B}\Sigma_X)$, then the composite

$$BG \xrightarrow{f} \mathrm{Set}_{(X)} \xrightarrow{\mathrm{fst}} \mathrm{Set}$$

is a G-set, and the value at sh_G is identified with X.

The reasoning in the previous two paragraphs yields the following equivalence:

$$G\text{-Set} \simeq \sum_{X\,:\,\mathrm{Set}} BG \to_* \mathrm{B}\Sigma_X.$$

⌟

We must be careful not to focus too much on the underlying set. For instance, even though the underlying set of both Ad_G and Pr_G is $\mathrm{U}G$, in general Ad_G and Pr_G are very different G-sets. To drive this point home, compare the illustrations of transport along a $p : \mathrm{U}G$ for the two:

$$\begin{tikzcd}
\mathrm{sh}_G \arrow[r, equal, "p", "\rightarrow"'] \arrow[d, equal, "q"', "\downarrow"] & \mathrm{sh}_G \arrow[d, equal, "\downarrow"', "\mathrm{Ad}_G(p)(q)"] \\
\mathrm{sh}_G \arrow[r, equal, "p", "\rightarrow"'] & \mathrm{sh}_G
\end{tikzcd}$$

$$\begin{tikzcd}
\mathrm{sh}_G \arrow[r, equal, "\mathrm{refl}_{\mathrm{sh}_G}", "\rightarrow"'] \arrow[d, equal, "q"', "\downarrow"] & \mathrm{sh}_G \arrow[d, equal, "\downarrow"', "\mathrm{Pr}_G(p)(q)"] \\
\mathrm{sh}_G \arrow[r, equal, "p", "\rightarrow"'] & \mathrm{sh}_G .
\end{tikzcd}$$

A third G-set with underlying set $\mathrm{U}G$ is $\mathrm{triv}_G(\mathrm{U}G)$.

Exercise 4.5.9. Show: if X is a type family with parameter type BG and $X(\mathrm{sh}_G)$ is a set, then X is a G-set. ⌟

Exercise 4.5.10. Prove that a group G is abelian group if and only if the G-sets Ad_G and $\mathrm{triv}_G(\mathrm{U}G)$ are identical. ⌟

Transitive G-sets

sec:transitiveGsets

We end the section with some observations regarding so-called transitive G-sets which will be valuable when we move on to discussing subgroups. Classically, a abs(G)-set (a notion *we* have yet not defined) $\mathcal{X}$ is said to be *transitive* if there exists a $b : \mathcal{X}$ such that for all $a : \mathcal{X}$ there is a $g : \mathcal{X}$ with $a = g \cdot b$. In our world this translates to

def:transitiveGset

Definition 4.5.11. A G-set $X : BG \to \mathrm{Set}$ is *transitive* if the proposition

$$\mathrm{isTrans}(X) :\equiv \prod_{y:BG} \Big\| \sum_{b:X(y)} \prod_{a:X(y)} \sum_{g:y=y} a = g \cdot b \Big\|$$

holds. ⌟

Remark 4.5.12. Note that the mention of y is redundant in the definition: by connectedness (cf. Exercise 2.16.4) it is enough to demand

$$\Big\| \sum_{b:X(\mathrm{sh}_G)} \prod_{a:X(\mathrm{sh}_G)} \sum_{g:\mathrm{U}G} a = g \cdot b \Big\|.$$

In other words, X is transitive if and only if there merely is a $b : X(\mathrm{sh}_G)$ such that the map $- \cdot b : \mathrm{U}G \to X(\mathrm{sh}_G)$ is surjective. ⌟

lem:conistrans

Lemma 4.5.13. *A G-set is transitive if and only if the associated set bundle is connected.*

Proof. Consider a G-set $X : BG \to \mathrm{Set}$ and the associated set bundle $f : \tilde{X} \to BG$ where $\tilde{X} :\equiv \sum_{y:BG} X(y)$ and f is the first projection. Now, $\tilde{X}$ is connected if and only if there *merely* exists a $y : BG$ and a $b : X(y)$ such that for all $z : BG$ and $a : X(z)$ there is a $g : y = z$ such that $a = g \cdot b$. Since BG is connected, this is equivalent to asserting that there merely is is a $b : X(\mathrm{sh}_G)$ such that for all $a : X(\mathrm{sh}_G)$ there is a $g : \mathrm{U}G$ such that $a = g \cdot b$. □

Recall that for type families $X, X' : T \to \mathcal{U}$, and $f : \prod_{y:T} X(y) \to X'(y)$, we write $f_y : X(y) \to X'(y)$ (instead of the more correct $f(y)$) for its evaluation at $y : T$.

lem:evisinjwhentransitive

Lemma 4.5.14. *Let $X : BG \to \mathrm{Set}$ be a transitive G-set, $y : BG$ and $b : X(y)$. Then the evaluation map*

$$\mathrm{ev} : (X = X) \to X(y), \qquad \mathrm{ev}(f) :\equiv f_y(b)$$

is injective.

Proof. In view of function extensionality, our claim is that the evaluation map $\mathrm{ev} : \prod_{x:BG}(X(x) = X(x)) \to X(y)$ given by the same formula is injective; that is all fs with the same value $f_y(b)$ are identical.

For $a : X(y)$, consider an $f : X = X$ with $f_y(b) = a$. Let $z : BG$ and $c : X(z)$. For any $g : y = z$ such that $g \cdot b = c$ we have $f_z(c) = f_z(g \cdot b) = g \cdot f_y(b) = g \cdot a$: the value does not depend on f. Since we try to prove a proposition we are done. □

4.6 *The classifying type is the type of torsors*

This section can be seen as a motivation for the use of torsors. In Section 4.7 we'll use this concept to prove that the type of groups and the type of abstract groups are equivalent by classifying abstract groups in terms of their pointed connected groupoid of torsors. To see how this might work it is good to start with the case of a (concrete) group G. In the end we want the torsors of $\mathrm{abs}(G)$ to be equivalent to BG, so to get the right definition we should first explore what the torsors of G look like and prove Theorem 4.6.6 showing that BG is equivalent to the type of G-torsors.

sec:torsors

Definition 4.6.1. Given a group G, the type of *G-torsors* is

$$\mathrm{Torsor}_G :\equiv \sum_{X\,:\,G\text{-Set}} \|\mathrm{Pr}_G = X\|,$$

where Pr_G is the principal G-torsor of Example 4.5.3. ⌟

This works equally well with ∞-groups: G-torsors are in that case G-types in the component of the principal torsor. There is no conflict with the case when the ∞-group G is actually a group since then any G-type in the component of the principal G-torsor will be a G-set.

Remark 4.6.2. For G a group, the type of G-torsors is just another name for the component of the type of set bundles of BG containing the universal set bundle.

Observe that for a group G, Torsor_G is a connected groupoid (admittedly in a higher universe) and so – by specifying the base point Pr_G – it represents a group! Guess which one! ⌟

For $z : BG$, recall the definition of $\mathrm{P}_z : BG \to \mathrm{Set}$ as the G-set with $\mathrm{P}_z(y) :\equiv (z = y)$ (so that in particular $\mathrm{Pr}_G :\equiv \mathrm{P}_{\mathrm{sh}_G}$). Note that P_z is a G-torsor.

def:BG2TorsG

Definition 4.6.3. Let

$$\mathrm{P} : BG \to_* (\mathrm{Torsor}_G, \mathrm{Pr}_G)$$

be the pointed map given by sending $z : BG$ to P_z and by the identification $\mathrm{refl}_{\mathrm{P}_{\mathrm{sh}_G}} : \mathrm{P}_{\mathrm{sh}_G} = \mathrm{Pr}_G$. ⌟

If G is not clear from the context, we may choose to write P^G instead of P.

ex:pathsptransport

Example 4.6.4. For $y, z : BG$ we make the induced map

$$\mathrm{P} : (y = z) \to (\mathrm{P}_y = \mathrm{P}_z),$$

or rather its composite with the equivalence to $\prod_{x:BG} \mathrm{P}_y(x) = \mathrm{P}_z(x)$, explicit. For $q : y = z$, the transport $\mathrm{P}_q : \prod_{x:BG} \mathrm{P}_y(x) = \mathrm{P}_z(x)$ is obtained by sending $p : \mathrm{P}_y(x) :\equiv (y = x)$ to

$$\mathrm{P}_q(p) :\equiv pq^{-1} : \mathrm{P}_z(x) :\equiv (z = x).$$

⌟

In a picture,

$$\begin{tikzcd}
y \arrow[r, equal, "q", "\rightarrow"'] \arrow[d, equal, "p", "\downarrow"'] & z \arrow[d, equal, "\mathrm{P}_q(p)", "\downarrow"'] \\
x \arrow[r, equal, "\mathrm{refl}_x", "\rightarrow"'] & x.
\end{tikzcd}$$

lem:pathsptransportiseq

Lemma 4.6.5. *For $y, z : BG$ the induced map (i.e., transport) of identity types*

$$\mathrm{P} : (y = z) \to (\mathrm{P}_y = \mathrm{P}_z)$$

is an equivalence.

Proof. We craft an inverse $Q : (\mathrm{P}_y = \mathrm{P}_z) \to (y = z)$ for P. Given an identity $f : \mathrm{P}_y = \mathrm{P}_z$, the map $f_y : (y = y) \to (z = y)$ maps the reflexivity path refl_y to a path $f_y(\mathrm{refl}_y) : z = y$, and we define

$$Q(f) :\equiv f_y(\mathrm{refl}_y)^{-1}$$

First, Q is an inverse on the right for P as $\mathrm{P}_{Q(f)}$ is equal to the map $p \mapsto pf_y(\mathrm{refl}_y)$, and by induction on $p : y = x$, $pf_y(\mathrm{refl}_y) = f_x(p)$ (indeed this is true for $p \equiv \mathrm{refl}_y$). This means that $\mathrm{P}_{Q(f)} = f$. Next, we show that Q is an inverse on the left for P: indeed for any $q : y = z$ $\mathrm{P}_q(\mathrm{refl}_y)^{-1} = (\mathrm{refl}_y q^{-1})^{-1} = q$; in other words $Q(\mathrm{P}_q) = q$. □

lem:BGbytorsor

Theorem 4.6.6. *If G is a group (or ∞-group), then the function*

$$\mathrm{P} \colon BG \to (\mathrm{Torsor}_G, \mathrm{Pr}_G), \qquad z \mapsto \mathrm{P}_z :\equiv (x \mapsto (z =_{BG} x))$$

is an equivalence. Univalence then provides us with an identity

$$\bar{\mathrm{P}} \colon G = (\mathrm{Torsor}_G, \mathrm{Pr}_G)$$

of groups (or ∞-groups).

Proof. Since both Torsor_G and BG are connected, it suffices by Corollary 2.17.10 to show that each $\mathrm{ap}_\mathrm{P} : (y = z) \to (\mathrm{P}_y = \mathrm{P}_z)$ is an equivalence. One prove first that ap_P is indeed equal to the function

$$\mathrm{P} \colon (y = z) \to (\mathrm{P}_y = \mathrm{P}_z)$$

made explicit in Example 4.6.4. It is done by induction on $q : y = z$, as indeed

$$\mathrm{ap}_\mathrm{P}(\mathrm{refl}_y) = \mathrm{refl}_{\mathrm{P}_y} = (p \mapsto p\,\mathrm{refl}_y^{-1}).$$

Then Lemma 4.6.5 states exactly that ap_P is an equivalence. □

sec:homotor

4.6.7 *Homomorphisms and torsors*

In view of the equivalence P^G between BG and $(\mathrm{Torsor}_G, \mathrm{Pr}_G)$ of Theorem 4.6.6 one might ask what a group homomorphism $f : \mathrm{Hom}(G, H)$ translates to on the level of torsors. Off-hand, the answer is $(\mathrm{P}^H)Bf(\mathrm{P}^G)^{-1}$, but we can be more concrete than that. We do know that for $x : BG$ the G-torsor P^G_x should be sent to $\mathrm{P}^H_{Bf(x)}$, but how do we express this for an arbitrary G-torsor?

def:restrictandinduce

Definition 4.6.8. Let $f : \mathrm{Hom}(G, H)$ be a group homomorphism. If $Y : BH \to \mathrm{Set}$ is an H-set then the *restriction* f^*Y of Y to G is the G-set given by precomposition

$$f^*Y :\equiv Y\,f : BG \to \mathrm{Set}.$$

If $X : BG \to \mathrm{Set}$ is a G-set and $y : BH$ define the *induced H-type* $f_* BH \to \mathcal{U}$ by

$$f_*X(y) :\equiv \sum_{x : BG} (Bfx = y) \times X(x).$$

⌟

Note that the induced H-type may or may not be an H-set. As an example, consider the homomorphism $\mathrm{cy}_2 : \mathrm{Hom}(\mathbb{Z}, \Sigma_2)$ discussed above, given by sending $\bullet : S^1$ to $\mathbb{2} : \mathrm{FinSet}_2$ and $\circlearrowleft$ to the twist.

If we consider cy_2 also as a $\mathbb{Z}$-set (by including FinSet_2 in the type of all sets), the induced Σ_2-set $\Sigma_2 \times_\mathbb{Z} \mathrm{cy}_2 : \mathrm{FinSet}_2 \to \mathrm{Set}$ is given by

$$y \mapsto \Sigma_{z : S^1}(\mathrm{cy}_2(z) = y) \times \mathrm{cy}_2(z),$$

and it is instructive to see that the symmetry of $(\bullet, \mathrm{refl}_\mathbb{2}, 0)$ induced by $\circlearrowleft^2$ is not identical to refl, and so the induced Σ_2-type in question is *not* a set.

This situation is common in algebra and is often referred to by saying that some construction is not "exact".

For X being the principal G-torsor Pr_G, the contraction of $\sum_{x : BG}(\mathrm{sh}_G = x)$ induces an equivalence

$$\eta_y : f_*\mathrm{Pr}_G(y) = \sum_{x : BG} (Bf(x) = y) \times (\mathrm{sh}_G = x) \simeq (Bf(x) = y) \equiv \mathrm{Pr}^H_{Bf(x)}(y).$$

The resulting identity $\bar{\eta} : f_*\mathrm{Pr}_G = \mathrm{Pr}^H_{Bf(x)}$ shows that for every G-torsor X the H-type f_*X is an H-torsor.

Summing up:

lem:inducedtorsor

Lemma 4.6.9. *Let $f : \mathrm{Hom}(G,H)$ be a group homomorphism. If X is a G-torsor, then the induced H-type f_*X is an H-torsor and so we get an induced map*

$$f_* : \mathrm{Torsor}_G \to \mathrm{Torsor}_H.$$

The identity $\bar{\eta} : f_*\mathrm{P}^G_x = \mathrm{P}^H_{Bf(x)}$ shows that

$$\begin{tikzcd} BG \arrow[r, "Bf"] \arrow[d, "\mathrm{P}^G"] & BH \arrow[d, "\mathrm{P}^H"] \\ \mathrm{Torsor}_G \arrow[r, "f_*"] & \mathrm{Torsor}_H \end{tikzcd}$$

commutes.

nducedGsetfromabstracthomomorphisms

Remark 4.6.10. Notice that our construction of the induced G-set works equally well for a homomorphism $\phi : \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$: if $X : BG \to \mathrm{Set}$ is a G-set, then we define the H-set $\phi_*X : BH \to \mathrm{Set}$ by

$$\phi_*X(y) :\equiv (\mathrm{sh}_H = y) \times_{\mathrm{U}G} X(\mathrm{sh}_G)$$

to be the set quotient of $(\mathrm{sh}_H = y) \times X(\mathrm{sh}_G)$ by the relation $(p,x) \sim (p\,\phi(q)^{-1}, X(q)x)$ for all $q : \mathrm{sh}_G = pt_G$. Just as above, for X the principal G-torsor we get an identity $\eta_\phi : \phi_*\mathrm{Pr}_G = \mathrm{Pr}_H$ which, when evaluated at $y : BH$, corresponds under univalence to the equivalence

$$(\mathrm{sh}_H = y) \times_{\mathrm{U}G} \mathrm{U}G \to (\mathrm{sh}_H = y)$$

sending $[p,q] : (\mathrm{sh}_H = y) \times_{\mathrm{U}G} \mathrm{U}G$ to $p\,\phi(q) : (\mathrm{sh}_H = y)$. ⌟

4.7 *Groups; concrete vs. abstract*

sec:Gsetforabstract

We use Theorem 4.6.6 as our inspiration for trying to construct a group from an abstract group. That is, by totally analogy, we define the torsors for an abstract group, and it will then be a relative simple matter to show that the processes of

(1) forming the abstract group of a group and

(2) taking the group represented by the torsors of an abstract group

are inverse to each other.

Note that we have not considered an "abstract" counterpart of the concept of ∞-group, so all we do in this section is set-based.

def:abstrGtorsors

Definition 4.7.1. If $\mathcal{G} = (S, e, \mu, \iota)$ is an abstract group, a *$\mathcal{G}$-set* is a set $\mathcal{X}$ together with a homomorphism $\mathcal{G} \to \mathrm{abs}(\Sigma_\mathcal{X})$ from $\mathcal{G}$ to the (abstract) permutation group of $\mathcal{X}$:

$$Set^{\mathrm{abs}}_\mathcal{G} :\equiv \sum_{\mathcal{X} : \mathrm{Set}} \mathrm{Hom}_{\mathrm{abs}}(\mathcal{G}, \mathrm{abs}(\Sigma_\mathcal{X})).$$

The *principal $\mathcal{G}$-torsor* $\mathrm{Pr}^{\mathrm{abs}}_\mathcal{G}$ is the $\mathcal{G}$-set consisting of the underlying set S together with the homomorphism $\mathcal{G} \to \mathrm{abs}(\Sigma_S)$ with underlying function of sets $S \mapsto (S = S)$ given by sending $g : S$ to $\mathrm{ua}(s \mapsto s \cdot g^{-1})$.

The type of *$\mathcal{G}$-torsors* is

$$\mathrm{Torsor}^{\mathrm{abs}}_\mathcal{G} :\equiv \sum_{S : \mathrm{Set}^{\mathrm{abs}}_\mathcal{G}} \|\mathrm{Pr}^{\mathrm{abs}}_\mathcal{G} = S\|.$$

⌟

Example 4.7.2. If G is a group we can unravel the definition and see that an abs(G)-set consists of

(1) a set S,

(2) a function $f : \mathrm{U}G \to (S = S)$

(3) such that $f(e_G) = \mathrm{refl}_S$ and for all $p, q : \mathrm{U}G$ we have that $f(p\,q) = f(p)\,f(q)$.

⌟

To help reading the coming proofs we introduce some notation that is redundant, but may aid the memory in cluttered situations: Let x, y, z be elements in some type, then

$$\mathrm{preinv} : (y = x) \to ((y = z) = (x = z)), \quad \mathrm{preinv}(q)(p) :\equiv \mathrm{P}_q p :\equiv p q^{-1}$$
$$\mathrm{post} : (y = z) \to ((x = y) = (x = z)), \quad \mathrm{post}(p)(q) :\equiv \mathrm{post}_p q :\equiv pq$$

We recognize preinv from Lemma 4.6.5 as the induced map of identity types $\mathrm{P} : (y = z) \to (\mathrm{P}_y = \mathrm{P}_z)$ evaluated at x, while post-composition post is transport in the family P_x.

ex:qG

Example 4.7.3. If G is a group, then for any $x : BG$ the principal G-torsor *evaluated at* x, i.e., the set $\mathrm{Pr}_G x :\equiv (\mathrm{sh}_G = x)$, has a natural structure of an abs(G)-set by means of

$$\mathrm{preinv} : \mathrm{U}G \to ((\mathrm{sh}_G = x) = (\mathrm{sh}_G = x))$$

and the fact that $\mathrm{preinv}(e_G) :\equiv \mathrm{refl}_{\mathrm{sh}_G = x}$ and that for $p, q : \mathrm{U}G$ we have that

$$\mathrm{preinv}(p\,q) = \mathrm{preinv}(p)\mathrm{preinv}(q).$$

i.e., if $r : \mathrm{sh}_G = x$ we have that $\mathrm{preinv}(p\,q)(r) = r(p\,q)^{-1} = r\,q^{-1}p^{-1} = \mathrm{preinv}(p)\mathrm{preinv}(q)(r)$ – demonstrating why we chose preinv: without the inverse this would have gone badly wrong.

That this abs(G)-set is an abs(G)-torsor then follows since BG is connected (any $\mathrm{sh}_G = x$ will serve as a proof of $(\mathrm{sh}_G = x, \mathrm{preinv}, !) = \mathrm{Pr}^{\mathrm{abs}}_{\mathrm{abs}(G)}$).

Though it sounded like we made a choice ending up with preinv; we really didn't – it is precisely what happens when you abstract the homomorphism $G \to \Sigma_{\mathrm{Pr}_G(x)}$: you get the function of identity types

$$\mathrm{U}G \to (\mathrm{Pr}_G(x) = \mathrm{Pr}_G(x))$$

which by the very definition of transport for Pr_G is preinv. ⌟

Definition 4.7.4. If $\mathcal{G}$ is an abstract group, then the *concrete group* $\mathrm{concr}(\mathcal{G})$ *associated with* $\mathcal{G}$ is the group given by the pointed connected groupoid $(\mathrm{Torsor}^{\mathrm{abs}}_{\mathcal{G}}, \mathrm{Pr}_{\mathcal{G}})$. ⌟

We give the construction of Example 4.7.3 a short name since it will occur in important places.

Definition 4.7.5. Let G be a group. The group homomorphism

$$q_G : \mathrm{Hom}(G, \mathrm{concr}(\mathrm{abs}(G)))$$

is defined in terms of the pointed map by the same name

$$q_G : BG \to_* (\mathrm{Torsor}^{\mathrm{abs}}_{\mathrm{abs}(G)}, \mathrm{Pr}_{\mathrm{abs}(G)}), \quad q_G(z) = (\mathrm{Pr}_G(z), \mathrm{preinv}, !).$$

⌟

lem:Groupsareidentitytypes

Lemma 4.7.6. *For all groups G, the pointed function $q_G : G \to_* \mathrm{concr}(\mathrm{abs}(G))$ is a equivalence.*

Proof. To prove that q_G is an equivalence it is, by Corollary 2.17.10, enough to show that if $x, y : BG$ then the induced map

$$q_G : (x =_{BG} y) \to (q_G(x) = q_G(y))$$

is an equivalence. Now, $q_G(x) = q_G(y)$ is equivalent to the set

$$((\mathrm{sh}_G = x), \mathrm{preinv}) =_{\mathrm{abs}(G)\text{-set}} ((\mathrm{sh}_G = y), \mathrm{preinv})$$

which in turn is equivalent to

$$\sum_{f : (\mathrm{sh}_G = x) = (\mathrm{sh}_G = y)} f\mathrm{preinv} = \mathrm{preinv} f$$

($f\mathrm{preinv} = \mathrm{preinv}f$ is shorthand for $\prod_{q : \mathrm{sh}_G = x} \prod_{p : \mathrm{sh}_G = p} f(pq^{-1}) = f(p)q^{-1}$ and the rest of the data is redundant at the level of symmetries) and under these identities q_G is given by

$$(\mathrm{post}, !) : (x = y) \to \sum_{f : (\mathrm{sh}_G = x) = (\mathrm{sh}_G = y)} f\mathrm{preinv} = \mathrm{preinv} f.$$

Given an element $(f, !) : \sum_{f : (\mathrm{sh}_G = x) = (\mathrm{sh}_G = y)} f\mathrm{preinv} = \mathrm{preinv}f$, the preimage $(\mathrm{post}, !)^{-1}(f, !)$ is equivalent to the set $\sum_{r : x = y}(f = \mathrm{post}_r)$. But if $(r, !), (s, !) : \sum_{r : x = y}(f = \mathrm{post}_r)$, then for all $p : \mathrm{sh}_G = x$ we get that $r\,p = f(p) = s\,p$, that is $r = s$, so that the preimage is in fact a proposition. To show that the preimage is contractible, it is enough to construct a function $(\mathrm{sh}_G = x) \to \sum_{r : x = y}(f = \mathrm{post}_r)$, and sending p to $f(p)p^{-1}$ will do. □

ex:abstrconcrG

Example 4.7.7. Let $\mathcal{G} = (S, e, \mu, \iota)$ be an abstract group. Then the underlying set of abs(concr($\mathcal{G}$)) is $\mathrm{Pr}^{\mathrm{abs}}_{\mathcal{G}} =_{\mathrm{Torsor}^{\mathrm{abs}}_{\mathcal{G}}} \mathrm{Pr}^{\mathrm{abs}}_{\mathcal{G}}$. Unraveling the definitions we see that this set is equivalent to

$$\sum_{p : S = S} \prod_{q, s : S} (p(s\,q^{-1}) = p(s)\,q^{-1}).$$

Setting $s :\equiv e$ and renaming $t :\equiv q^{-1}$ in the last equation, we see that $p(t) = p(e)t$; that is p is simply multiplication with an element $p(e) : S$. in other words, the function

$$r_{\mathcal{G}} : S \to \sum_{p : S = S} \prod_{q, s : S} (p(s\,q^{-1}) = p(s)\,q^{-1}), \qquad r_{\mathcal{G}}(u) :\equiv (u \cdot, !)$$

is an equivalence of sets, which we by univalence is converted into an identity. The abstract group structure of abs(concr($\mathcal{G}$)) is given by it being the symmetries of $\mathrm{Pr}^{\mathrm{abs}}_{\mathcal{G}}$; translated to $\sum_{p : S = S} \prod_{q, s : S}(p(s\,q^{-1}) = p(s)\,q^{-1})$ this corresponds via the first projection to the symmetries of S. This means that we need to know that if $u, v : S$ and consider the two symmetries $u \cdot, v\, \cdot : S = S$, then their composite (the operation on the symmetry on S) is equal to $(u \cdot v) \cdot : S = S$ (the abstract group operation), but this is true by associativity $(u \cdot (v \cdot s) = (u \cdot v) \cdot s)$. That $r_{\mathcal{G}}$ also sends $e : S$ to refl_S is clear. Hence our identity $r_{\mathcal{G}}$ underlies an identity of abstract groups

$$r_{\mathcal{G}} : \mathcal{G} =_{\mathrm{Group}^{\mathrm{abs}}} \mathrm{abs}(\mathrm{concr}(\mathcal{G})).$$

⌟

This shows that every abstract group encodes the symmetries of something essentially unique. Summing up the information we get lem:Groupsareidentitytypes

Theorem 4.7.8. *Let $\mathcal{G}$ be an abstract group. Then*

$$\mathrm{abs} : \mathrm{Group} \to \mathrm{Group}^{\mathrm{abs}}$$

is an equivalence.

sec:homabsisconcr

4.8 *Homomorphisms; abstract vs. concrete*

Now that we know that the type of groups is identical to the type of abstract groups, it is natural to ask if the notion of group homomorphisms also coincide.

They do, and we provide two independent and somewhat different arguments. Translating from group homomorphisms to abstract group homomorphisms is easy: if G and H are groups, then we defined

$$\mathrm{abs} \colon \mathrm{Hom}(G, H) \to \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$$

in Definition 4.3.19 as the function which takes a homomorphism, aka a pointed map $f = (Bf_\div, p_f) : BG \to_* BH$ to the induced map of identity types

$$\mathrm{U}f :\equiv \mathrm{ad}_{p_f}\, \mathrm{ap}_{Bf_\div} : \mathrm{U}G \to \mathrm{U}H$$

together with the proofs that this is an abstract group homomorphism from $\mathrm{abs}(G)$ to $\mathrm{abs}(H)$, c.f Remark 4.3.18.

Going back is somewhat more involved, and it is here we consider two approaches. The first is a compact argument showing directly how to reconstruct a pointed map $Bf : BG \to_* BH$ from an abstract group homomorphism from $\mathrm{abs}(G)$ to $\mathrm{abs}(H)$, the second translates back and forth via our equivalence between abstract and concrete groups.

The statement we are after is

lem:homomabstrconcr

Lemma 4.8.1. *If G and H are groups, then*

$$\mathrm{abs} \colon \mathrm{Hom}(G, H) \to \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$$

is an equivalence.

and the next two subsections offer two proofs.

sec:thierrysdelooping

"Delooping" a group homomorphism

We now explore the first approach.

Proof. Suppose we are given an abstract group homomorphism

$$f \colon \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$$

and we explain how to build a map $g : BG_\div \to BH_\div$ with a path $p : \mathrm{sh}_H = g(\mathrm{sh}_G)$ such that $pf(\omega) = g(\omega)p$ for all $\omega : \mathrm{sh}_G = \mathrm{sh}_G$ (so that g is a "delooping" of f, that is, $f = \mathrm{abs}(g)$).

To get an idea of our strategy, let us assume the problem solved. The map $g : BG_\div \to BH_\div$ will then send any path $\alpha : \mathrm{sh}_G = x$ to a path $g(\alpha) : g(\mathrm{sh}_G) = g(x)$ and so we get a family of paths $p(\alpha) :\equiv g(\alpha)p$ in $\mathrm{sh}_H = g(x)$ such that

$$p(\alpha\omega) = g(\alpha)g(\omega)p = g(\alpha)pf(\omega) = p(\alpha)f(\omega)$$

for all $\omega : \mathrm{sh}_G = \mathrm{sh}_G$ and $\alpha : \mathrm{sh}_G = x$.

We will thus have displayed a map $\mathrm{deloop} : \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H)) \to \mathrm{Hom}(G, H)$ with $\mathrm{abs}\, \mathrm{deloop} = \mathrm{refl}$. We leave it to the reader to prove that $\mathrm{deloop}\, \mathrm{abs} = \mathrm{refl}$.

This suggests to introduce the following family

$$C(x) \;:=\; \sum_{y\,:\,BH_\div}\;\sum_{p\,:\,(\mathrm{sh}_G=x)\to(\mathrm{sh}_H=y)}\;\prod_{\omega\,:\,\mathrm{sh}_G=\mathrm{sh}_G}\;\prod_{\alpha\,:\,\mathrm{sh}_G=x} p(\alpha\omega)=p(\alpha)f(\omega)$$

An element of $C(x)$ has three components, the last component being a proposition since $BH_\div$ is a groupoid.

The type $C(\mathrm{sh}_G)$ has a simpler description. An element of $C(\mathrm{sh}_G)$ is a pair y,p such that $p(\alpha\omega)=p(\alpha)f(\omega)$ for α and ω in $\mathrm{sh}_G=\mathrm{sh}_G$. Since f is an abstract group homomorphism, this condition can be simplified to $p(\omega)=p(1_{\mathrm{sh}_G})f(\omega)$, and the map p is completely determined by $p(1_{\mathrm{sh}_G})$. Thus $C(\mathrm{sh}_G)$ is equal to $\sum_{y\,:\,BH_\div}\mathrm{sh}_H=y$ and is contractible.

It follows that we have

$$\prod_{x\,:\,BG_\div}(\mathrm{sh}_G=x)\to\mathrm{isContr}\;\;C(x)$$

and so, since isContr $C(x)$ is a proposition

$$\prod_{x\,:\,BG_\div}\|a=x\|\to\mathrm{isContr}\;\;C(x)$$

Since $BG_\div$ is connected, we have $\prod_{x\,:\,BG_\div}$ isContr $C(x)$ and so, in particular, we have an element of $\prod_{x\,:\,BG_\div}C(x)$.

We get in this way a map $g:BG_\div\to BH_\div$ together with a map $p:(a=x)\to(\mathrm{sh}_H=g(x))$ such that $p(\alpha\omega)=p(\alpha)f(\omega)$ for all α in $\mathrm{sh}_G=x$ and ω in $\mathrm{sh}_G=\mathrm{sh}_G$. We have, for $\alpha:\mathrm{sh}_G=x$

$$\prod_{x'\,:\,BG_\div}\;\prod_{\lambda\,:\,x=x'}p(\lambda\alpha)=g(\lambda)p(\alpha)$$

since this holds for $\lambda=1_x$. In particular, $p(\omega)=g(\omega)p(1_{\mathrm{sh}_G})$.

We also have $p(\omega)=p(1_{\mathrm{sh}_G})f(\omega)$, hence $p(1_{\mathrm{sh}_G})g(\alpha)=f(\alpha)p(1_{\mathrm{sh}_G})$ for all $\alpha:\mathrm{sh}_G=\mathrm{sh}_G$ and we have found a delooping of f.

□

The concrete vs. abstract homomorphisms via torsors.

sec:absconctorsor

The second approach to Lemma 4.8.1 is as follows:

Proof. The equivalence of $\mathrm{P}^G:BG\xrightarrow{\sim}(\mathrm{Torsor}_G,\mathrm{Pr}_G)$ of Theorem 4.6.6 gives an equivalence

$$\mathrm{P}\colon\mathrm{Hom}(G,H)\xrightarrow{\sim}((\mathrm{Torsor}_G,\mathrm{Pr}_G)\to_*(\mathrm{Torsor}_H,\mathrm{Pr}_H))$$

Consider the map

$$A:((\mathrm{Torsor}_G,\mathrm{Pr}_G))\to_*(\mathrm{Torsor}_H,\mathrm{Pr}_H)\to\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G),\mathrm{abs}(H))$$

given by letting $A(f,p)$ be the composite

$$\begin{tikzcd}
\mathrm{U}G \arrow[d, equal, "\downarrow"', "\mathrm{P}^G"] & & \mathrm{U}H \arrow[d, equal, "\downarrow"', "\mathrm{P}^H"] \\
(\mathrm{Pr}_G=\mathrm{Pr}_G) \arrow[r, "f"] & (f\mathrm{Pr}_G=f\mathrm{Pr}_G) \arrow[r, equal, "q\mapsto p^{-1}qp", "\rightarrow"'] & (\mathrm{Pr}_H=\mathrm{Pr}_H)
\end{tikzcd}$$

(together with the proof that this is an abstract group homomorphism). We are done if we show that A is an equivalence.

The reason to complicate abs this way is that it gets easier to write out the inverse function.

If $(\phi, !): \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$ and $X: BG \to \mathrm{Set}$ is a G-torsor, recall the induced H-torsor $\phi_* X$ from Remark 4.6.10 and the identity $\eta_\phi : \phi_* \mathrm{Pr}_G = \mathrm{Pr}_H$. Let

$$B: \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H)) \to ((\mathrm{Torsor}_G, \mathrm{Pr}_G) \to_* (\mathrm{Torsor}_H, \mathrm{Pr}_H)$$

be given by $B(\phi, !) = (\phi_* X, \eta_\phi)$

We show that A and B are inverse equivalences. Given an abstract group homomorphism $(\phi, !): \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(H))$, then $AB(\phi, !)$ has as underlying set map

$$\begin{tikzcd}
\mathrm{U}G \arrow[d, equal, "\mathrm{P}^G", "\downarrow"'] & & \mathrm{U}H \arrow[d, equal, "\mathrm{P}^H", "\downarrow"'] \\
(\mathrm{Pr}_G = \mathrm{Pr}_G) \arrow[r, "\phi_*"] & (\phi_*\mathrm{Pr}_G = \phi_*\mathrm{Pr}_G) \arrow[r, equal, "q\mapsto\eta_\phi^{-1} q\eta_\phi", "\rightarrow"'] & (\mathrm{Pr}_H = \mathrm{Pr}_H),
\end{tikzcd}$$

and if we start with a $g: \mathrm{U}G$, then P^G sends it to $\mathrm{P}^G_g \equiv \mathrm{preinv}(g)$. Furthermore, $\phi_*\mathrm{preinv}(g)$ is $[\mathrm{id}, \mathrm{preinv}(g)]$ which is sent to $\mathrm{preinv}(\phi(g))$ in $\mathrm{Pr}_H = \mathrm{Pr}_H$ which corresponds to $\phi(g): \mathrm{U}H$ under P^H. In other words, $AB(\phi, !) = (\phi, !)$. The composite BA is similar. □

4.9 *Monomorphisms and epimorphisms*

sec:monoepi

In set theory we say that a function $f: B \to C$ of sets is an injection if for all $b, b': B$ we have that $f(b) = f(b')$ implies that $b = b'$. This conforms with our definitions. Furthermore, since giving a term $b: B$ is equivalent to giving a (necessarily constant) function $c_b: \mathbb{1} \to B$, we could alternatively say that a function $f: B \to C$ is an injection if and only if for any two $g, h: \mathbb{1} \to B$ such that $fg = fh$ we have that $g = h$. In fact, by function extensionality we can replace $\mathbb{1}$ by any set A (two functions are identical if and only if they have identical values at every point).

Similarly, a function $f: B \to C$ is surjective if for all $c: C$ the preimage $f^{-1}(c) = \sum_{b:B} c = f(b)$ is non-empty. A smart way to say this is to say that the first projection from $\sum_{c:C} \| f^{-1}(c) \|$ to C is an equivalence. Since B is always equivalent to $\sum_{c:C} f^{-1}(c)$, we see that for a surjection $f: B \to C$ and family of propositions $P: C \to \mathrm{Prop}$, the propositions $\prod_{c:C} P(c)$ and $\prod_{b:B} Pf(b)$ are equivalent. In particular, if $g, h: C \to D$ are two functions into a set D the proposition $\prod_{c:C}(g(c) = h(c))$ is equivalent to $\prod_{b:B}(gf(b) = hf(c))$.

From this we condense the following characterizations of injections and surjections of sets which will prove to generalize nicely to other contexts.

lem:injofsetsaremono

Lemma 4.9.1. *Let $f: B \to C$ be a function between sets.*

(1) *the function is an injection of and only if for any set A and functions $g, h: A \to B$,*

$$\begin{tikzcd}
A \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & B \arrow[r, "f"] & C ,
\end{tikzcd}$$

then $fg = fh: A \to C$ implies $g = h$

(2) *the function is an injection of and only if for any set D and functions $g, h : C \to D$,*

$$\begin{tikzcd} B \arrow[r, "f"] & C \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & D \end{tikzcd},$$

then $gf = hf : A \to C$ implies $g = h$.

By Lemma 4.9.1 there is a pleasing reformulation which highlights that injections/surjections of sets are characterized by injections of sets of functions: a function of sets $f : B \to C$ is

(1) an injection if and only if for any set A postcomposition by f given an injection from $A \to B$ to $A \to C$

(2) a surjection if and only if for any set D precomposition by f gives an injection from $B \to D$ to $B \to D$.

This observation about sets translates fruitfully to other contexts and in particular to groups. To make it clear that we talk about group homomorphisms (and not about the underlying unpointed functions of connected groupoids) we resort to standard categorical notation.

def:monomorphism
def:epimorphism

Definition 4.9.2. Given groups G, H, a homomorphism $f : \mathrm{Hom}(G, H)$ is called a

(1) *monomorphism* if for any group F, postcomposition by f is an injection from $\mathrm{Hom}(F, G)$ to $\mathrm{Hom}(F, H)$, and an

(2) *epimorphism* if for any group I, precomposition by f is an injection from $\mathrm{Hom}(H, I)$ to $\mathrm{Hom}(G, I)$.

The corresponding families of propositions are called

$$\mathrm{isMono}, \mathrm{isEpi} : \mathrm{Hom}(G, H) \to \mathrm{Prop}.$$

⌟

We've seen that for any group G, the underlying set $\mathrm{U}G :\equiv (\mathrm{sh}_G = \mathrm{sh}_G)$ of $\mathrm{abs}(G)$ is equivalent to the set of homomorphisms $\mathrm{Hom}(\mathbb{Z}, G)$ which in turn is equivalent to the set of abstract homomorphisms $\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(\mathbb{Z}), \mathrm{abs}(G))$ and that abstraction preserves composition. Hence, if $f : \mathrm{Hom}(G, H)$ is a group homomorphism, then saying that $\mathrm{U}f$ is an injection is equivalent to saying that postcomposition by f is an injection $\mathrm{Hom}(\mathbb{Z}, G) \to \mathrm{Hom}(\mathbb{Z}, H)$. In this observation, the integers $\mathbb{Z}$ plays no more of a rôle than $\mathbb{1}$ does in Lemma 4.9.1; we can let the source vary over any group F:

$$\begin{tikzcd} \mathrm{U}G \arrow[r, "\mathrm{U}f"] \arrow[d, "\simeq"] & \mathrm{U}H \arrow[d, "\simeq"] \\ \mathrm{Hom}(\mathbb{Z}, G) \arrow[r, "f_*"] \arrow[d, "\mathrm{abs}"', "\simeq"] & \mathrm{Hom}(\mathbb{Z}, H) \arrow[d, "\mathrm{abs}"', "\simeq"] \\ \mathrm{Hom}^{\mathrm{abs}}(\mathbb{Z}, \mathrm{abs}(G)) \arrow[r, "\mathrm{abs} f_*"] & \mathrm{Hom}^{\mathrm{abs}}(\mathbb{Z}, \mathrm{abs}(H)) \end{tikzcd}$$

commutes (we've written $\mathbb{Z}$ also for $\mathrm{abs}(\mathbb{Z})$ since otherwise it wouldn't fit.

lem:eq-mono-cover
lem:eq-epi-comm

Lemma 4.9.3. *Let G and H be groups and $f : \mathrm{Hom}(G, H)$ a homomorphism. The following propositions are equivalent*:

it:mono

(1) *f is a monomorphism;*

it:injection

(2) *$\mathrm{U}f : \mathrm{U}G \to \mathrm{U}H$ is an injection;*

it:cover

(3) *$Bf_{\div} : BG_{\div} \to BH_{\div}$ is a set bundle.*

Similarly, the following propositions are equivalent:

it:epi

(1′) *f is an epimorphism;*

it:surjection

(2′) *$\mathrm{U}f : \mathrm{U}G \to \mathrm{U}H$ is a surjection.*

it:connfib

(3′) *$Bf_\div : BG_\div \to BH_\div$ has connected fibers.*

Proof. We only do the monomorphism case; the epimorphism case is very similar. We have already seen that condition (1) implies condition (2) (let F be $\mathbb{Z}$). Conversely, suppose that (2) holds and F is a group. Consider the commutative diagram

$$\begin{tikzcd}
\mathrm{Hom}(F,G) \arrow[r] \arrow[d] & \mathrm{Hom}(F,H) \arrow[d] \\
(\mathrm{Hom}(\mathbb{Z},F) \to \mathrm{Hom}(\mathbb{Z},G)) \arrow[r] & (\mathrm{Hom}(\mathbb{Z},F) \to \mathrm{Hom}(\mathbb{Z},H)),
\end{tikzcd}$$

where the vertical maps are the injections from the sets of (abstract) homomorphism to the sets of functions of underlying sets and the horizontal maps are postcomposition with f. Since the bottom function is by assumption is an injection, so is the upper one. [15]

The equivalence of (3) and (2) follows immediately from Corollary 2.17.10(2), using that BG is connected and f is pointed and the equivalence between $\mathrm{Hom}(G,H)$ and $BG \to_* BH$. □

4.9.4 *Any symmetry is a symmetry in* Set

sec:groupssubperm

The correspondence between groups and abstract groups allows for a cute proof of what is often stated as "any group is a permutation group", which in our parlance translates to "any symmetry is a symmetry in Set".

Recall the principal G-torsor $\mathrm{Pr}_G : BG \to \mathrm{Set}$. Since $\mathrm{Pr}_G(\mathrm{sh}_G) :\equiv \mathrm{U}G$ this defines a pointed function $\rho_G : BG \to_* \mathrm{B}\Sigma_{\mathrm{U}G} :\equiv (\mathrm{Set}_{(\mathrm{U}G)}, \mathrm{U}G)$, i.e., a homomorphism from G to the permutation group

$$\rho_G : \mathrm{Hom}(G, \Sigma_{\mathrm{U}G}).$$

the letter ρ commemorates the word "regular"

lem:allgpsarepermutationgps

Lemma 4.9.5. *Let G be a group. Then $\rho_G : \mathrm{Hom}(G, \Sigma_{\mathrm{U}G})$ is a monomorphism.*

By Lemma 4.9.3 "ρ_G is a monomorphism" means that the induced map ρ_G^{abs} from the symmetries of sh_G in $BG_\div$ to the symmetries of $\mathrm{U}G$ in Set is an injection, i.e., "any symmetry is a symmetry in Set".

Proof. In view of Lemma 4.9.3 we need to show that $\rho_G : BG \to \mathrm{Set}_{(\mathrm{U}G)}$ is a set bundle. Under the identity

$$\bar{\mathrm{P}} : BG = (\mathrm{Torsor}_G, \mathrm{Pr}_G) \equiv (\mathrm{Set}_{(\mathrm{U}G)}, \mathrm{U}G)$$

of Theorem 4.6.6, ρ_G translates to the evaluation map

$$\mathrm{ev}_{\mathrm{sh}_G} : (BG \to \mathrm{Set})_{(\mathrm{Pr}_G)} \to \mathrm{Set}_{(\mathrm{U}G)}, \qquad \mathrm{ev}_{\mathrm{sh}_G}(E) = E(\mathrm{sh}_G).$$

We must show that the preimages $\mathrm{ev}_{\mathrm{sh}_G}^{-1}(X)$ for $X : \Sigma_{\mathrm{U}G}$ are sets. This fiber is equivalent to $\sum_{E : (BG \to \mathrm{Set})_{(\mathrm{Pr}_G)}} (X = E(\mathrm{sh}_G))$ which is a set precisely when $\sum_{E : BG \to \mathrm{Set}} (X = E(\mathrm{sh}_G))$ is a set. We must then show that if $(E,p),(F,q) : \sum_{E : BG \to \mathrm{Set}} (X = E(\mathrm{sh}_G))$, then the type $(E,p) = (F,q)$, which is equivalent to

$$\prod_{x : BG} \sum_{\phi(x) : E(x) = F(x)} \phi(\mathrm{sh}_G) = qp^{-1},$$

is a proposition. If $\phi, \psi : ((E,p) = (F,q))$, we must show that $\phi = \psi$ and since that for $x : BG$ both $E(x)$ and $F(x)$ are sets, it is enough to show that the proposition $\phi(x) = \psi(x)$ is not empty. Let $f : (\mathrm{sh}_G = x) \to$

Note that if $r : \mathrm{sh}_G = x$, then $\phi(x) = F(r)qp^{-1}E(r)^{-1}$, in a picture

$$\begin{tikzcd}
 & E(pt_G) \arrow[r, equal, "E(r)", "\rightarrow"'] \arrow[dd, equal, "\phi(pt_G)"', "\downarrow"] & E(x)\ , \arrow[dd, equal, "\phi(x)"', "\downarrow"] \\
X \arrow[ur, equal, "p", "\nearrow"'] \arrow[dr, equal, "q"', "\searrow"] & & \\
 & F(\mathrm{sh}_G) \arrow[r, equal, "F(r)", "\rightarrow"'] & F(x)
\end{tikzcd}$$

and so $\phi(x)$ (which by nature is independent of such an $r : \mathrm{sh}_G = x$) is uniquely determined by (E,p) and (F,q). The text says this formally.

[15] Alternatively: and $g, h : \mathrm{Hom}(F,G)$. Then $fg = fh$ implies that for all $p : \mathrm{Hom}(\mathbb{Z},F)$ we have by associativity that $f(gp) = (fg)p = (fh)p = f(hp)$, and so, by assumption, that $gp = hp$. Again, by function extensionality (of functions $\mathrm{Hom}(\mathbb{Z},F) \to \mathrm{Hom}(\mathbb{Z},G)$), this is exactly saying that $\mathrm{U}g$ is identical to $\mathrm{U}h$.

$(\phi(x) = \psi(x))$ be given by letting $f(r)$ be the composite of the identities $\phi(x) = F(r)qp^{-1}E(r)^{-1} = \psi(x)$ given above. Since BG is connected, and $\phi(x) = \psi(x)$ is a proposition, one can consider that $\mathrm{sh}_G = x$ is not empty, and we are done. □

4.10 *Abelian groups*

sec:abelian-groups

Recall that given a pointed type X, we coerce it silently to its underlying unpointed type $X_\div$ whenever this coercion can be inferred from context. For example, given a group G, the type $BG \simeq BG$ can not possibly mean anything but $BG_\div \simeq BG_\div$ as the operator "$\simeq$" acts on bare types. To refer to the type of pointed equivalences (that is the pointed functions whose underlying functions are equivalences), we shall use the notation $BG \simeq_* BG$.

4.10.1 *Center of a group*

sec:center-group

Definition 4.10.2. Let G be a group. The *center* of G, denoted $\mathrm{Z}(G)$, is the group $\mathrm{Aut}_{(BG_\div = BG_\div)}(\mathrm{id}_{BG_\div})$. ⌟

We work transparently through the equivalence

$$(BG_\div = BG_\div) \simeq (BG \simeq BG)$$

so that $\mathrm{id}_{BG_\div}$ is freely used in place of $\mathrm{refl}_{BG_\div}$ when convenient.

There is a natural map $\mathrm{ev}_{\mathrm{sh}_G} : (BG_\div = BG_\div) \to BG_\div$ defined by $\mathrm{ev}_{\mathrm{sh}_G}(\varphi) :\equiv \varphi(\mathrm{sh}_G)$. In particular, $\mathrm{ev}_{\mathrm{sh}_G}(\mathrm{id}_{BG_\div}) \equiv \mathrm{sh}_G$. It makes the restriction of this map to the connected component of $\mathrm{id}_{BG_\div}$ a pointed map, hence it defines a homomorphism

$$\mathrm{z}_G : \mathrm{Hom}(\mathrm{Z}(G), G).$$

We will now justifiy the name *center* for $\mathrm{Z}(G)$, and connect it to the notion of center for abstract groups in ordinary mathematics. The homomorphism z_G induces a homomorphism of abstract groups from $\mathrm{abs}(\mathrm{Z}(G))$ to $\mathrm{abs}(G)$. By induction on $p : \mathrm{id}_{BG_\div} = \varphi$ for $\varphi : BG_\div = BG_\div$, one proves that $\mathrm{ap}_{\mathrm{Bz}_G}(p) = p(\mathrm{sh}_G)$: indeed, this is true when $p \equiv \mathrm{refl}_{\mathrm{id}_{BG_\div}}$. One proves furthermore, again by induction on $p : \mathrm{id}_{BG_\div} = \varphi$, that $\mathrm{ap}_\varphi = (q \mapsto p(\mathrm{sh}_G)^{-1}qp(\mathrm{sh}_G))$. In particular, when $\varphi \equiv \mathrm{id}_{BG_\div}$, it shows that for every $p : \mathrm{id}_{BG_\div} = \mathrm{id}_{BG_\div}$, the following proposition holds:

$$\prod_{g : \mathrm{U}G} p(\mathrm{sh}_G)g = gp(\mathrm{sh}_G)$$

In other words, $\mathrm{abs}(\mathrm{z}_G)$ maps elements of $\mathrm{abs}(Z(G))$ to elements of $\mathrm{abs}(G)$ that commute with every other elements. (The set of these elements is usually called the center of the group $\mathrm{abs}(G)$ in ordinary group theory.)

lemma:center-is-subgroup

Lemma 4.10.3. *The map* Bz_G *is a set bundle over* BG*.*

Proof. One wants to prove the proposition $\mathrm{isSet}((\mathrm{Bz}_G)^{-1}(x))$ for each $x : BG$. By connectedness of BG, one can show the proposition only at $x \equiv \mathrm{sh}_G$. However,

$$(\mathrm{Bz}_G)^{-1}(\mathrm{sh}_G) \simeq \sum_{\varphi : \mathrm{B}\,\mathrm{Z}(G)} \mathrm{sh}_G = \varphi(\mathrm{sh}_G)$$

Recall that $\mathrm{B}\,\mathrm{Z}(G)$ is the connected component of $\mathrm{id}_{BG_\div}$ in $BG_\div = BG_\div$. In particular, if (φ, p) and (ψ, q) are two elements of the type on the right hand-side above, their identity type $(\varphi, p) = (\psi, q)$ is equivalent to their identity type in $BG_{\div} = BG_{\div}$, or in other words:

$$((\varphi, p) = (\psi, q)) \simeq \sum_{\pi : \varphi = \psi} \pi(\mathrm{sh}_G)p = q.$$

We shall prove that this type is a proposition, and it goes as follows:

(1) for $\pi : \varphi = \psi$, the type $\pi(\mathrm{sh}_G)p = q$ is a proposition; hence for two such elements $(\pi, !)$ and $(\pi', !)$, one has $(\pi, !) = (\pi', !)$ equivalent to $\pi = \pi'$,

(2) for all $x : BG$, $\varphi(x) = \psi(x)$ is a set, hence for $\pi, \pi' : \varphi = \psi$, the type $\pi = \pi'$ is a proposition,

(3) connectedness of BG proves then that $\pi = \pi'$ is equivalent to $\pi(\mathrm{sh}_G) = \pi'(\mathrm{sh}_G)$,

(4) finally the propositional condition on π and π' allows us to conclude as $\pi(\mathrm{sh}_G) = qp^{-1} = \pi'(\mathrm{sh}_G)$.

□

lemma:center-inc-inj-on-paths

Corollary 4.10.4. *The induced map* $\mathrm{abs}(\mathrm{z}_G) : \mathrm{abs}(\mathrm{Z}(G)) \to \mathrm{abs}(G)$ *is injective.*

The following result explains how every element of the "abstract center" of G is picked out by $\mathrm{abs}(\mathrm{z}_G)$.

lemma:center-inc-surj-on-paths

Lemma 4.10.5. *Let* $g : \mathrm{U}G$ *and suppose that* $gh = hg$ *for every* $h : \mathrm{U}G$. *The fiber* $(\mathrm{ap}_{\mathrm{Bz}_G})^{-1}(g)$ *contains an element.*

Proof. One must construct an element $\hat{g} : \mathrm{id}_{BG_{\div}} = \mathrm{id}_{BG_{\div}}$ such that $g = \hat{g}(\mathrm{sh}_G)$. We shall use function extensionality and produce an element $\hat{g}(x) : x = x$ for all $x : BG$ instead. Note that $x = x$ is a set, and that connectedness of BG is not directly applicable here. We will use a technique that has already proven useful in many situations in the book, along the lines of the following sketch:

(1) for a given $x : BG$, if such a $\hat{g}(x) : x = x$ existed, it would produce an element of the type $T(\hat{g}(x))$ for a carefully chosen type family T,

(2) aim to prove $\mathrm{isContr}(\sum_{u : x = x} T(u))$ for any $x : BG$,

(3) this is a proposition, so connectedness of BG can be applied and only $\mathrm{isContr}(\sum_{u : \mathrm{U}G} T(u))$ needs to be proven,

(4) hopefully, $\sum_{u : \mathrm{U}G} T(u)$ reduces to an obvious singleton type.

Here, for any $x : BG$, we define the type family $T : (x = x) \to \mathcal{U}$ by

$$T(q) :\equiv \prod_{p : \mathrm{sh}_G = x} (pg = qp).$$

And we claim that $\sum_{q : x = x} T(q)$ is contractible for any $x : BG$. Because this is a proposition, one only need to check that it holds on one point of the connected type BG, say $x \equiv \mathrm{sh}_G$. Now,

$$\begin{aligned}\sum_{q\,:\,\mathrm{U}G} T(q) &\equiv \sum_{q\,:\,\mathrm{U}G}\prod_{p\,:\,\mathrm{U}G}(pg = qp)\\ &\simeq \sum_{q\,:\,\mathrm{U}G}\prod_{p\,:\,\mathrm{U}G}(g = q)\\ &\simeq \sum_{q\,:\,\mathrm{U}G} \mathrm{U}G \to (g = q)\\ &\simeq \sum_{q\,:\,\mathrm{U}G}(g = q)\\ &\simeq 1\end{aligned}$$

The first equivalence is using that g commutes with every other element $p : \mathrm{U}G$, so that $pgp^{-1} = g$. The second equivalence acknowledges the fact that $(g = q)$ does not depend on p anymore. The third equivalence makes two reductions at the same time: first it recognizes a proposition in the type $g = q$ and then uses that the propositional truncation of $\mathrm{U}G$ is indeed inhabited (by $|\mathrm{refl}_{\mathrm{sh}_G}|$).

We have just shown that for all $x : BG$, the type $\sum_{q\,:\,x=x} T(q)$ is contractible. We define now $\hat{g}(x) : x = x$ as the chosen center of contraction of that type. In particular, in the previous proof that $\sum_{q\,:\,\mathrm{U}G} T(q)$ is connected, we chose g as center of contraction, so that $\hat{g}(\mathrm{sh}_G) = g$ as wanted. □

Together, Corollary 4.10.4 and Lemma 4.10.5 show that $\mathrm{abs}(\mathrm{z}_G)$ establishes an equivalence

$$\mathrm{U\,Z}(G) \simeq \sum_{g\,:\,\mathrm{U}G}\prod_{h\,:\,\mathrm{U}G} gh = hg \tag{4.10.1}$$

In yet other words, $\mathrm{B\,Z}(G) :\equiv (BG_{\div} = BG_{\div})_{(\mathrm{id}_{BG_{\div}})}$ is (equivalent to) the classifying type of a group whose abstract group is the "abstract center" of $\mathrm{abs}(G)$.

The following lemma is then immediate:

def:abelian-groups

Lemma 4.10.6. *A group G is* abelian *if and only if z_G is an isomorphism of groups.*

Remark 4.10.7. In the style of this book, we could have used Lemma 4.10.6 directly as the definition of abelian groups. However, the definition of z_G would have been to intricate to give properly as early as Definition 4.1.27. ⌟

4.10.8 *Universal cover and simple connectedness*

sec:univ-cover-simple

Let us say that a pointed type (A, a) is *simply connected* when both A and $a = a$ are connected types.

Definition 4.10.9. Let A be a type and $a : A$ an element. The *universal cover* of A at a is the type

$$A^0_{(a)} :\equiv \sum_{x\,:\,A} \|a = x\|_0.$$

⌟

The definition of the universal cover is reminiscent of the notion of connected component: instead of selecting elements that are merely equal to a fixed element a, the universal cover selects elements together with mere witnesses of the equality with a.

When needed, we will consider $A^0_{(a)}$ as a pointed type, with distinguished point $(a, |\mathrm{refl}_a|_0)$. Note that when A is a groupoid, then the set truncation is redundant and the universal cover of A at a is then the singleton at a. In particular, groupoids have contractible universal covers.

The identity types in $A^0_{(a)}$ can be understood easily once we introduce the following function for elements $x, y, z : A$:

$$\_\cdot\_ : \|y = z\|_0 \times \|x = y\|_0 \to \|x = z\|_0.$$

It is defined as follows: given $\chi : \|y = z\|_0$, we want to define $\chi \cdot \_$ in the set $\|x = y\|_0 \to \|x = z\|_0$, hence we can suppose $\chi \equiv |q|_0$ for some $q : y = z$; now given $\pi : \|x = y\|_0$, one want to define $|q|_0 \cdot \pi$ in the set $\|x = z\|_0$, hence one can suppose $\pi \equiv |p|_0$ for some $p : x = y$; finally, we define

$$|q|_0 \cdot |p|_0 :\equiv |q \cdot p|_0.$$

Then one proves, by induction on $p : x = y$, that $\mathrm{trp}_p^{\|a=\_\|_0}$ is equal to the function $\alpha \mapsto |p|_0 \cdot \alpha$. In particular, one gets an equivalence for the type of path between two points (x, α) and (y, β) of the universal cover $A^0_{(a)}$:

{eq:id-types-universal-cover}

$$((x, \alpha) = (y, \beta)) \simeq \sum_{p : x = y} |p|_0 \cdot \alpha = \beta. \tag{4.10.2}$$

This description allows us to prove the following lemma.

ma:universal-cover-simply-connected

Lemma 4.10.10. *Let A be a type and $a : A$ an element. The universal cover $A^0_{(a)}$ is simply connected.*

Proof. First, we prove that $A^0_{(a)}$ is connected. It has a point $(a, |\mathrm{refl}_a|_0)$ and, for every $(x, \alpha) : A^0_{(a)}$, one wants $\|(a, |\mathrm{refl}_a|_0) = (x, \alpha)\|$. This is proposition, hence a set, so that one can suppose $\alpha = |p|_0$ for a path $p : a = x$. Now, the proposition $|p|_0 \cdot |\mathrm{refl}_a|_0 = |p|_0$ holds. So one can use Eq. (4.10.2) to produce a path $(a, |\mathrm{refl}_a|_0) = (x, \alpha)$.

Next, we prove that $(a, |\mathrm{refl}_a|_0) = (a, |\mathrm{refl}_a|_0)$ is connected. One uses again Eq. (4.10.2) to compute:

$$\begin{aligned} ((a, |\mathrm{refl}_a|_0) = (a, |\mathrm{refl}_a|_0)) &\simeq \sum_{p : a = a} \big(|p|_0 = |\mathrm{refl}_a|_0\big) \\ &\simeq \sum_{p : a = a} (\|p = \mathrm{refl}_a\|) \end{aligned}$$

In other words, $(a, |\mathrm{refl}_a|_0) = (a, |\mathrm{refl}_a|_0)$ is equivalent to the connected component of refl_a in $a = a$. In particular, it is connected. □

sec:abel-groups-simply

4.10.11 *Abelian groups and simply connected 2-types*

We will now give an alternative characterization of the type of abelian groups, more in line with the geometrical intuition we are trying to build in this chapter. Recall that a type A is called a *2-truncated type*, or *2-type* for short, when every identity type $x = y$ is a groupoid for $x, y : A$.

Theorem 4.10.12. *The type* AbGroup *of abelian groups is equivalent to the type of pointed simply connected 2-types.*

Proof. Define the map $\mathrm{B}^2 : \mathrm{AbGroup} \to \mathcal{U}_*$ by $\mathrm{B}^2 G :\equiv \mathcal{U}^0_{(BG_\div)}$.[16] Proving that $\mathrm{B}^2 G$ is a 2-type is equivalent to proving the proposition $\mathrm{isSet}(p = q)$ for all $p, q : x = y$ and all $x, y : \mathrm{B}^2 G$. One can then use connectedness of $\mathrm{B}^2 G$ and restrict to only show that $p = q$ is a set for all path $p, q : (BG_\div, |\mathrm{id}_{BG_\div}|_0) = (\mathrm{B}G_\div, |\mathrm{id}_{BG_\div}|_0)$. As part of the definition of the group G, the type $BG_\div$ is a 1-type, hence $BG_\div = BG_\div$ is also a 1-type through univalence. Moreover, $\mathrm{trp}_p\,(|\mathrm{id}_{BG_\div}|_0) = |\mathrm{id}_{BG_\div}|_0$ and $\mathrm{trp}_q\,(|\mathrm{id}_{BG_\div}|_0) = |\mathrm{id}_{BG_\div}|_0$ both are propositions, as they are identity types in the set $\|BG_\div = BG_\div\|_0$. Hence $\mathrm{isSet}(p = q)$ holds.

So one gets a map, denoted again B^2 abusively,

$$\mathrm{B}^2 : \mathrm{AbGroup} \to \mathcal{U}_*^{=2}$$

[16] This is slightly misleading: If G is an abelian group in universe $\mathcal{U}$, then this definition makes $\mathrm{B}^2 G$ a pointed type in a successor universe, which is not what we want. The solution is to note that $\mathrm{B}^2 G$ is a locally $\mathcal{U}$-small type, which as a connected type is the image of the base point map $\mathrm{pt} : \mathbb{1} \to \mathrm{B}^2 G$, so it's an essentially $\mathcal{U}$-small type by the Replacement Principle 2.19.4. So really, $\mathrm{B}^2 G$ should be the $\mathcal{U}$-small type equivalent to $\mathcal{U}^0_{(BG_\div)}$.

where the codomain $\mathcal{U}_*^{=2}$ is the type of pointed simply connected 2-types, that is

$$\mathcal{U}_*^{=2} :\equiv \sum_{(A,a):\mathcal{U}_*} \big(\mathrm{isConn}(A) \times \mathrm{isConn}(a = a) \times \mathrm{isGrpd}(a = a)\big)$$

We shall now provide an inverse for this map. Given a pointed simply connected 2-type (A, a), one can construct a group, denoted $\mathrm{Aut}^2(A, a)$, with classifying type:

$$\mathrm{BAut}^2(A, a) :\equiv (a = a, \mathrm{refl}_a).$$

Indeed, this pointed type is connected because (A, a) is simply connected, and it is a 1-type because A is a 2-type. Moreover, $\mathrm{Aut}^2(A, a)$ is abelian. To see it, let us use the bare definition of abelian groups (cf. Definition 4.1.27). We shall then prove that for all elements $g, h : \mathrm{refl}_a = \mathrm{refl}_a$, the proposition $gh = hg$ holds. This property holds in even more generality and is usually called "Eckmann-Hilton's argument". It goes as follows: for $x, y, z : A$, for $p, q : x = y$ and $r, s : y = z$ and for $g : p = q$ and $h : r = s$, one prove

{eq:horizontal-comp}

$$\mathrm{ap}_{\_\cdot q}(h) \cdot \mathrm{ap}_{r\cdot\_}(g) = \mathrm{ap}_{s\cdot\_}(g) \cdot \mathrm{ap}_{\_\cdot p}(h). \tag{4.10.3}$$

This equality takes place in $r \cdot p = s \cdot q$ and is better represented by the diagram in Fig. 4.1. One prove such a result by induction on h. Indeed,

fig:horizontal-comp

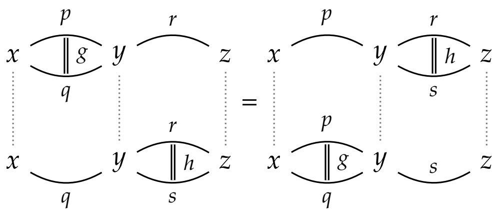


Figure 4.1: Visual representation of Eq. (4.10.3). The vertical dotted lines denotes composition.

when $h \equiv \mathrm{refl}_r$, then both sides of the equation reduces through path algebra to $\mathrm{ap}_{r\cdot\_}(g)$. Now we are interested in this result when x, y, z are all definitionally a, and p, q, r, s are all definitionally refl_a. In that case, one has that $\mathrm{ap}_{\mathrm{refl}_a\cdot\_}$ and $\mathrm{ap}_{\_\cdot\mathrm{refl}_a}$ both act trivially, and the equation becomes: $h \cdot g = g \cdot h$.

One still has to prove that the function Aut^2 is an inverse for B^2. Given an abelian group G, the proof of Lemma 4.10.10 gives an equivalence between $\mathrm{BAut}^2(\mathrm{B}^2G)$ and the connected component of $\mathrm{id}_{BG_\div}$ in $BG_\div = BG_\div$. By definition, this is the classifying type of $\mathrm{Z}(G)$. Being abelian, G is isomorphic to its center (Lemma 4.10.6), and so it yields an element of $\mathrm{Aut}^2(\mathrm{B}^2G) =_{\mathrm{Group}} G$. Conversely, take a pointed simply connected 2-type (A, a). One wants to prove $\mathrm{B}^2(\mathrm{Aut}^2(A, a)) \simeq_* (A, a)$. One should first notice that, because $\mathrm{Aut}^2(A, a)$ is an abelian group,

If $X \simeq_* Y$ denote the type of pointed equivalences between pointed types $X, Y : \mathcal{U}_*$, then the univalence axiom implies that there is an equivalence

$$(X = Y) \simeq (X \simeq_* Y).$$

{eq:loopspace-A-abelian}

$$\mathrm{BAut}^2\left(\mathrm{B}^2(\mathrm{Aut}^2(A, a))\right) \simeq ((a = a) = (a = a))_{(\mathrm{refl}_{a=a})} \simeq (a = a, \mathrm{refl}_a). \tag{4.10.4}$$

This equivalence maps a path

$$(p, !) : (a = a, |\mathrm{refl}_{a=a}|_0) = (a = a, |\mathrm{refl}_{a=a}|_0)$$

to the evaluation $p(\mathrm{refl}_a) : a = a$.

We will now define a pointed map $\Phi : (A, a) \to_* \mathrm{B}^2(\mathrm{Aut}^2(A, a))$, and prove subsequently that this is an equivalence. Let $T : A \to \mathcal{U}$ be the type family define by

$$T(a') :\equiv \sum_{\alpha : \|(a=a)\simeq(a=a')\|_0} \prod_{p : a=a'} \alpha = |p \cdot \_|_0$$

We claim that $T(a')$ is contractible for all $a' : A$. By connectedness of A, it is equivalent to showing that $T(a)$ is contractible. However

$$\begin{aligned} T(a) &\equiv \sum_{\alpha : \|(a=a)\simeq(a=a)\|_0} \prod_{p : a=a} \alpha = |p \cdot \_|_0 \\ &\simeq \sum_{\alpha : \|(a=a)\simeq(a=a)\|_0} \alpha = |\mathrm{id}_{a=a}|_0 \\ &\simeq 1 \end{aligned}$$

Let then $\Phi(a')$ be the element $(a = a', \kappa_{a'}) : \mathcal{U}^0_{(a=a)}$ where $\kappa_{a'}$ is the first projection of the center of contraction of $T(a')$. In particular, following the chain of equivalences above, $\Phi(a)$ is defined as $(a = a, |\mathrm{refl}_{a=a}|_0)$, hence $\Phi(a)$ is trivially pointed by a reflexivity path. To verify that Φ, thus defined, is an equivalence, one can use connectedness of $\mathrm{B}^2(\mathrm{Aut}^2(A, a))$ and only check that $\Phi^{-1}(a = a, |\mathrm{refl}_{a=a}|_0)$ is contractible. However,

$$\Phi^{-1}(a = a, |\mathrm{refl}_{a=a}|_0) \simeq \sum_{a' : A} \sum_{\varphi : (a=a)\simeq(a=a')} |\varphi|_0 = \kappa_{a'}.$$

For an element $a' : A$ together with $\varphi : (a = a) \simeq (a = a')$ such that the proposition $|\varphi|_0 = \kappa_{a'}$ holds, a path between $(a, \mathrm{id}_{a=a}, !)$ and $(a', \varphi, !)$ consists of a path $p : a = a'$ and a path $q : (x \mapsto px) = \varphi$. We have a good candidate for p, namely $p :\equiv \varphi(\mathrm{refl}_a) : a = a'$. However we don't have quite q yet. Consider, for any $a' : A$, the function

$$\mathrm{ev}^{a'}_{\mathrm{refl}_a} : ((a = a, |\mathrm{refl}_{a=a}|_0) = (a = a', \kappa_{a'})) \to (a = a')$$

defined as $(\psi, !) \mapsto \psi(\mathrm{refl}_a)$. Note that $\mathrm{ev}^{a}_{\mathrm{refl}_a}$ is precisely the equivalence $\mathrm{BAut}^2(\mathrm{B}^2\mathrm{Aut}^2(A, a))_{\div} \simeq (a = a)$ described in Eq. (4.10.4). Hence, by connectedness of A, one gets that the proposition $\mathrm{isEquiv}(\mathrm{ev}^{a'}_{\mathrm{refl}_a})$ holds for all $a' : A$. In particular, because the propositions $|\varphi|_0 = \kappa_{a'}$ and $|p \cdot \_|_0 = \kappa_{a'}$ holds, one gets elements $(\varphi, !)$ and $(x \mapsto px, !)$ in the domain of $\mathrm{ev}^{a'}_{\mathrm{refl}_a}$. Their image $\mathrm{ev}^{a'}_{\mathrm{refl}_a}(\varphi, !)$ and $\mathrm{ev}^{a'}_{\mathrm{refl}_a}(x \mapsto px, !)$ are both equal to p, which provides a path $(x \mapsto px, !) = (\varphi, !)$ in the domain. The first component is the path $q : (x \mapsto px) = \varphi$ that we wanted. □

4.11 *G-sets vs* abs(G)*-sets*

sec:Gsetsabstrconcr

Given a group G it should by now come as no surprise that the type of G-sets is equivalent to the type of abs(G)-sets. According to Lemma 4.8.1

$$\mathrm{abs} : \mathrm{Hom}(G, \Sigma_{\mathcal{X}}) \to \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(\Sigma_{\mathcal{X}}))$$

is an equivalence, where the group $\Sigma_{\mathcal{X}}$ (as a pointed connected groupoid) is the component of the groupoid Set, pointed at $\mathcal{X}$. The component information is moot since we're talking about pointed maps from BG

Recall from Definition 4.7.1 that the type of abs(G)-set is

$$Set^{\mathrm{abs}}_{\mathrm{abs}(G)} :\equiv \sum_{\mathcal{X} : \mathrm{Set}} \mathrm{Hom}_{\mathrm{abs}}(\mathrm{abs}(G), \mathrm{abs}(\Sigma_{\mathcal{X}})).$$

and we see that $\mathrm{Hom}(G,\Sigma_{\mathcal{X}})$ is equivalent to $\sum_{F:BG_\div\to\mathrm{Set}}(\mathcal{X}=F(\mathrm{sh}_G))$. Finally,

$$\mathrm{pr}:\sum_{\mathcal{X}}\sum_{F:BG_\div\to\mathrm{Set}}(\mathcal{X}=F(\mathrm{sh}_G))\xrightarrow{\sim}(BG_\div\to\mathrm{Set}),\quad \mathrm{pr}(\mathcal{X},F,p):\equiv F$$

is an equivalence (since $\sum_{\mathcal{X}}(\mathcal{X}=F(\mathrm{sh}_G))$ is contractible). Backtracking these equivalences we see that we have established

lem:actionsconcreteandabstract

Lemma 4.11.1. *Let G be a group. Then the map*

$$\mathrm{ev}_{\mathrm{sh}_G}:G\text{-Set}\to\mathrm{Set}^{\mathrm{abs}}_{\mathrm{abs}(G)},\qquad \mathrm{ev}_{\mathrm{sh}_G}(X):\equiv(X(\mathrm{sh}_G),a_X)$$

is an equivalence, where the homomorphism $a_X:\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(G),\mathrm{abs}(\Sigma_{X(\mathrm{sh}_G)}))$ is given by transport $X:\mathrm{U}G:\equiv(\mathrm{sh}_G=\mathrm{sh}_G)\to(X(\mathrm{sh}_G)=X(\mathrm{sh}_G))$.

If X is a G-set, $g:\mathrm{U}G$ and $x:X(\mathrm{sh}_G)$, we seek forgiveness for writing $g\cdot x:X(\mathrm{sh}_G)$ instead of $\mathrm{cast}(a_X(g))(x)$.[17]

ex:abstrandconj

Example 4.11.2. Let H and G be groups. Recall that the set of homomorphisms from H to G is a G-set in a natural way:

$$\mathrm{Hom}(H,G):BG\to\mathrm{Set},\quad \mathrm{Hom}(H,G)(y):\equiv\sum_{F:BH_\div\to BG_\div}(y=F(\mathrm{sh}_H)).$$

What abstract $\mathrm{abs}(G)$-set does this correspond to? In particular, under the equivalence $\mathrm{abs}:\mathrm{Hom}(H,G)\to\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(H),\mathrm{abs}(G))$, what is the corresponding action of $\mathrm{abs}(G)$ on the abstract homomorphisms?

The answer is that $g:\mathrm{U}G$ acts on $\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(H),\mathrm{abs}(G))$ by postcomposing with conjugation c^g by g as defined in Example 4.3.21.

Let us spell this out in some detail: If $(F,p):\mathrm{Hom}(H,G)(\mathrm{sh}_G):\equiv\sum_{F:BH_\div\to BG_\div}(\mathrm{sh}_G=F(\mathrm{sh}_H))$ and $g:\mathrm{U}G$, then $g\cdot(F,p):\equiv(F,p\,g^{-1})$. If we show that the action of g sends $\mathrm{abs}(F,p)$ to $c^g\circ\mathrm{abs}(F,p)$ we are done.

Recall that $\mathrm{abs}(F,p)$ consists of the composite

$$\mathrm{U}H\xrightarrow{F^=}(F(\mathrm{sh}_H)=F(\mathrm{sh}_G))\xrightarrow{t\mapsto p^{-1}t\,p}\mathrm{U}G\ ,$$

(i.e., $\mathrm{abs}(F,p)$ applied to $q:\mathrm{U}H$ is $p^{-1}F^=(q)\,p$) together with the proof that this is an abstract group homomorphism. We see that $\mathrm{abs}(F,p\,g^{-1})$ is given by conjugation: $q\mapsto(p\,g^{-1})^{-1}F^=(q)\,(p\,g^{-1})=g\,(p^{-1}F^=(q)\,p)\,g^{-1}$, or in other words $c^g\circ\mathrm{abs}(F,p)$. ⌟

For reference we list the conclusion of this example as a lemma:

lem:abstrandconj

Lemma 4.11.3. *If H and G are groups, then the equivalence of Lemma 4.11.1 sends the G-set $\mathrm{Hom}(H,G)$ to the $\mathrm{abs}(G)$-set $\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(H),\mathrm{abs}(G))$ with action given by postcomposing with conjugation by elements of $\mathrm{abs}(G)$.*

If $f:\mathrm{Hom}(G,G')$ is a homomorphism, then precomposition with $Bf:BG\to BG'$ defines a map

$$f^*:(G'\text{-Set})\to(G\text{-Set}).$$

We will have the occasion to use the following result which essentially says that if $f:\mathrm{Hom}(G,G')$ is an epimorphism, then f^* embeds the type of G'-sets as some of the components of the type of G-sets.

lem:epifullyfaithful

Lemma 4.11.4. *Let G and G' be groups and let $f:\mathrm{Hom}(G,G')$ be an epimorphism. Then the map $f^*:(G'\text{-Set})\to(G\text{-Set})$ (induced by precomposition with $Bf:BG\to BG'$) is "fully faithful" in the sense that if X,Y are G'-sets, then*

$$f^*:(X=Y)\to(f^*X=f^*Y)$$

is an equivalence.

Recall that Lemma 4.9.3 told us that f is an epimorphism precisely when $\mathrm{U}f$ is a surjection.

[17] and I ask forgiveness for strongly disliking the use of "cast" as a name for some tacitly understood map!

Proof. Evaluation at sh_G yields an injective map

$$\mathrm{ev}_{\mathrm{sh}_G} : (f^*X = f^*Y) \to (X(f(\mathrm{sh}_G) = Y(f(\mathrm{sh}_G)))$$

and the composite

$$\mathrm{ev}_{\mathrm{sh}_G} f^* = \mathrm{ev}_{f(\mathrm{sh}_G)} : (X = Y) \to (X(f(\mathrm{sh}_G) = Y(f(\mathrm{sh}_G)))$$

is the likewise injective, so $f^* : (X = Y) \to (f^*X = f^*Y)$ is injective.

For surjectivity, let $F' : f^*X = f^*Y$ and write, for typographical convenience, $a : X(f(\mathrm{sh}_G) = Y(f(\mathrm{sh}_G))$ for $\mathrm{ev}_{\mathrm{sh}_G} F' :\equiv F'_{\mathrm{sh}_G}$. By the equivalence between G-sets and abs(G)-sets, F' is uniquely pinned down by a and the requirement that for all $g' = f(g)$ with $g : UG$ the diagram

$$\begin{tikzcd}
X(f(\mathrm{sh}_G)) \arrow[r, equal, "X(g')"] \arrow[d, equal, "a"'] & X(f(\mathrm{sh}_G)) \arrow[d, equal, "a"'] \\
Y(f(\mathrm{sh}_G)) \arrow[r, equal, "Y(g')"] & Y(f(\mathrm{sh}_G))
\end{tikzcd}$$

commutes. Likewise, (using transport along the identity $p_f : \mathrm{sh}_{G'} = f(\mathrm{sh}_G)$) an $F : X = Y$ in the preimage of a is pinned down by the commutativity of the same diagram, but with $g' : f(\mathrm{sh}_G) = f(\mathrm{sh}_G)$ arbitrary (an a priori more severe requirement, again reflecting injectivity). However, when $f : UG \to UG'$ is surjective these requirements coincide, showing that f^* is an equivalence.

□

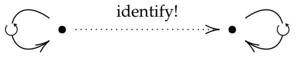
4.12 *Sums of groups*

sec:coprod

We have seen how the group of integers $\mathbb{Z} = (S^1, \bullet)$ synthesizes the notion of one symmetry with no relations: every symmetry in the circle is of the form $\circlearrowleft^n$ for some unique n. Also, given any group $G = \mathrm{Aut}_A(a)$, the set $a = a$ of symmetries of a corresponds to the set of homomorphisms $\mathbb{Z} \to G$, i.e., to pointed functions $(S^1, \bullet) \to_* (A, a)$ by evaluation at $\circlearrowleft$. What happens if we want to study more than one symmetry at the time?

For instance, is there a group $\mathbb{Z} \vee \mathbb{Z}$ so that for any group $G = \mathrm{Aut}_A(a)$ a homomorphism $\mathbb{Z} \vee \mathbb{Z} \to G$ corresponds to two symmetries of a? At the very least, $\mathbb{Z} \vee \mathbb{Z}$ itself would have to have two symmetries and these two can't have any relation, since in a general group $G = \mathrm{Aut}_A(a)$ there is a priori no telling what the relation between the symmetries of a might be. Now, *one* symmetry is given by a pointed function $(S^1, \bullet) \to_* (A, a)$ and so a *pair* of symmetries is given by a function $f : S^1 + S^1 \to A$ with the property that f sends each of the base points of the circles to a. But $S^1 + S^1$ is not connected, and so not a group. To fix this we take the clue from the requirement that both the base points were to be sent to a common base point and *define* $S^1 \vee S^1$ to be what we get from $S^1 + S^1$ when we *insert an identity* between the two basepoints.

$S^1 \vee S^1$ if formed from $S^1 + S^1$ by inserting an identity



The amazing thing is that this works – an enormous simplification of the classical construction of the "free products" or "amalgamated sum" of groups. We need to show that the "wedge" $S^1 \vee S^1$ is indeed a group, and this proof simultaneously unpacks the classical description.

We start by giving a definition of the wedge construction which is important for pointed types in general and then prove that the wedge of two groups is a group whose symmetries are arbitrary "words" in the original symmetries.

def:wedge

Definition 4.12.1. Let (A_1, a_1) and (A_2, a_2) be pointed types. The *wedge* is the pointed type $(A_1 \vee A_2, a_{12})$ given as a higher inductive type by

(1) functions $i_1 : A_1 \to A_1 \vee A_2$ and $i_2 : A_2 \to A_1 \vee A_2$

(2) an identity $g : i_1 a_1 = i_2 a_2$.

We point this type at $a_{12} :\equiv i_1 a_1$. The function

$$i_2^g : (a_2 =_{A_2} a_2) \to (a_{12} =_{A_1 \vee A_2} a_{12})$$

is defined by $i_2^g(p) :\equiv g^{-1} i_2(p) g$, whereas (for notational consistency only) we set $i_1^g :\equiv i_1 : (a_1 =_{A_1} a_1) \to (a_{12} =_{A_1 \vee A_2} a_{12})$. Simplifying by writing $i : A_1 + A_2 \to A_1 \vee A_2$ for the function given by i_1 and i_2 (with basepoints systematically left out of the notation), the induction principle is

$$\prod_{C:(A_1 \vee A_2)\to\mathcal{U}} \; \sum_{s:\prod_{a:A_1+A_2} Ci(a)} ((s(a_1) = C(g^{-1})s(a_2)) \to \prod_{x:(A_1 \vee A_2)} C(x)).$$

⌟

Unraveling the induction principle we see that if B is a pointed type, then a pointed function $f : A_1 \vee A_2 \to_* B$ is given by providing pointed functions $f_1 : A_1 \to_* B$ and $f_2 : A_2 \to_* B$ – the identity $f_1(a_1) = f_2(a_2)$ which seems to be missing is provided by the requirement of the functions being pointed. For the record

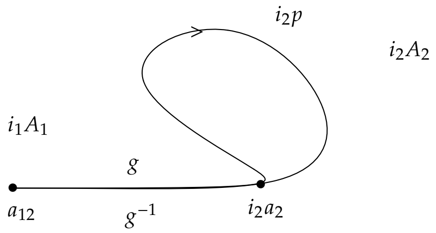
lem:univvee

Lemma 4.12.2. *If B is a pointed type, then the function*

$$i^* : (A_1 \vee A_2 \to_* B) \to (A_1 \to_* B) \times (A_2 \to_* B), \qquad i^*(f) = (f i_1, f i_2)$$

is an equivalence.

To the right you see a picture of $i_2^g(p)$: it is the symmetry of the base point $a_{12} :\equiv i_1 a_1$ you get by *first* moving to $i_2 a_2$ with g, *then* travel around with p ($i_2 p$, really) and finally go home to the basepoint with the inverse of g.



def:sumofgroup

Definition 4.12.3. If $G_1 = \mathrm{Aut}_{A_1}(a_1)$ and $G_2 = \mathrm{Aut}_{A_2}(a_2)$ are groups, then their *sum* is defined as

$$G_1 \vee G_2 :\equiv \mathrm{Aut}_{A_1 \vee A_2}(a_{12}).$$

The homomorphisms $i_1 : G_1 \to G_1 \vee G_2$ and $i_2 : G_2 \to G_1 \vee G_2$ induced from the structure maps $i_1 : A_1 \to A_1 \vee A_2$ and $i_2 : A_2 \to A_1 \vee A_2$ are also referred to as *structure maps*. ⌟

lem:sumofgroupsISsum

Lemma 4.12.4. *If G_1, G_2 and G are groups, then the function*

$$\mathrm{Hom}(G_1 \vee G_2, G) \to \mathrm{Hom}(G_1, G) \times \mathrm{Hom}(G_2, G)$$

given by restriction along the structure maps is an equivalence.

Proof. This is a special case of Lemma 4.12.2. □

Specializing further, we return to our initial motivation and see that mapping out of a wedge of two circles *exactly* captures the information of two independent symmetries:

The idea is that an identity in $a_{12} = x$ can be factored into a string of identities, each lying solely in A_1 or in A_2. We define a family of sets consisting of exactly such strings of identities – it is a set since A_1 and A_2 are groupoids – and prove that it is equivalent to the family $P(x) :\equiv (a_{12} =_{A_1 \vee A_2} x)$ which consequently must be a family of sets. We need to be able to determine whether a symmetry is reflexivity or not, but once we know that, the symmetries of the base point in the wedge are then given by "words $p_0 p_1 \ldots p_n$" where the p_j alternate between being symmetries in the first or the second group, and none of the p_j for positive j are allowed to be reflexivity. Note that there order of the p_js is not negotiable: if I shuffle them I get a new symmetry.

cor:ZplusZuniv

Corollary 4.12.5. *If G is a group, then the functions*

$$\mathrm{Hom}(\mathbb{Z} \vee \mathbb{Z}, G) \to \mathrm{Hom}(\mathbb{Z}, G) \times \mathrm{Hom}(\mathbb{Z}, G) \simeq \mathrm{U}G \times \mathrm{U}G$$

is an equivalence.

xca:whatAREabeliangroups

Exercise 4.12.6. This leads to the following characterization of abelian groups formulated purely in terms of pointed connected groupoids (no reference to the identity types). A group G is abelian if and only if the canonical map

$$+ : G \vee G \to G$$

(given via Lemma 4.12.4 by $\mathrm{id}_G : G \to G$) extends over the inclusion

$$i : G \vee G \to G \times G$$

(given by the inclusions $\mathrm{in}_1, \mathrm{in}_2 : G \to G \times G$).[18]

$$\begin{tikzcd} G \vee G \arrow[r, "+"] \arrow[d, "\text{inclusion}"'] & G \\ G \times G \arrow[ur, dotted] & \end{tikzcd}$$

As a cute aside, one can see that the required map $BG \times BG \to_* BG$ actually doesn't need to be pointed: factoring $+ : BG \vee BG \to_* BG$ over $i : BG \vee BG \to_* BG \times BG$ – even in an unpointed way – kills all "commutators" $ghg^{-1}h^{-1} : \mathrm{U}(G \vee G)$. ⌟

We end the section by proving that wedges of decidable groups are decidable groups and that they can be given the classical description in terms of words.

lem:wedgeofgpoidisgpoid

Lemma 4.12.7. *Let $G_1 :\equiv \mathrm{Aut}_{A_1}(a_1)$ and $G_2 :\equiv \mathrm{Aut}_{A_2}(a_2)$ be decidable groups, then the wedge sum $G_1 \vee G_2 :\equiv \mathrm{Aut}_{A_1 \vee A_2}(a_{12})$ is a decidable group.*

Let C_1 be the set of strings $(p_0, n, p_1, \dots, p_n)$ with $n : \mathbb{N}$ and, for $0 \le j \le n$

- *$p_j : \mathrm{U}G_1$ for even j*
- *$p_j : \mathrm{U}G_2$ for odd j and*
- *p_j is not reflexivity for j positive*

(*the last requirement makes sense and is a proposition since our groups are decidable*).

Then the function given by composition in $\mathrm{U}G_{12} :\equiv (a_{12} = a_{12})$

$$\beta : C_1 \to \mathrm{U}G_{12}, \qquad \beta(p_0, n, p_1, \dots p_n) :\equiv i_1^g p_0 i_2^g p_1 i_1^g p_2 \dots i_?^g p_n$$

(*where $i_?^g p_n$ is $i_1^g p_n$ or $i_2^g p_n$ according to whether n is even or odd*) *is an equivalence.*

Proof. That the wedge is connected follows by transitivity of identifications, if necessary passing through the identification $g : i_1 a_1 = i_2 a_2$ in the wedge.

We must prove that the wedge is a groupoid, i.e., that all identity types are sets, which we do by giving an explicit description of the universal set bundle.

We use the notation of Definition 4.12.1 freely, and for ease of notation, let $a_{2k+i} :\equiv a_i$ and $i_{2k+i}^g :\equiv i_i^g$ for $i = 1, 2$, $k : \mathbb{N}$. Define families of sets

$$C_i : A_i \to \mathrm{Set}, \qquad i = 1, 2$$

by

$$C_i(x) :\equiv (a_i =_{A_i} x) \times \sum_{n : \mathbb{N}} \prod_{1 \le k \le n} \sum_{p_k : a_{i+k} = a_{i+k}} (p_k \neq \mathrm{refl}_{a_{i+k}})$$

[18] I haven't written out a formalization myself

when $x : A_i$. Note that $p_k \neq \mathrm{refl}_{a_{i+k}}$ is a proposition; we leave it out when naming elements. Hence, an element in $C_1(a)$ is a tuple $(p_0, n, p_1, \ldots, p_n)$ where $p_0 : a_1 =_{A_1} a$, $p_1 : a_2 =_{A_2} a_2$, $p_2 : a_1 =_{A_1} a_1$, and so on – alternating between symmetries of a_1 and a_2, and where p_0 is the only identity allowed to be refl. Define $C_{12} : C_1(a_1) \to C_2(a_2)$ by

$$C_{12}(p_0, n, p_1 \ldots, p_n) = \begin{cases} (\mathrm{refl}_{a_2} 0,) & \text{if } p_0 = \mathrm{refl}_{a_1}, n = 0, \\ (p_1, n-1, p_2 \ldots, p_n) & \text{if } p_0 = \mathrm{refl}_{a_1}, n \neq 0, \\ (\mathrm{refl}_{a_2}, n+1, p_0, \ldots, p_n) & \text{if } p_0 \neq \mathrm{refl}_{a_1}. \end{cases}$$

It is perhaps instructive to see a table of the values $C_{12}(p_0, n, p_1, \ldots, p_n)$ for $n < 3$:

	$(p_0, 0)$	$(p_0, 1, p_1)$	$(p_0, 2, p_1, p_2)$
$p_0 = \mathrm{refl}_{a_1}$	$(\mathrm{refl}_{a_2}, 0)$	$(p_1, 0)$	$(p_1, 1, p_2)$
$p_0 \neq \mathrm{refl}_{a_1}$	$(\mathrm{refl}_{a_2}, 1, p_0)$	$(\mathrm{refl}_{a_2}, 2, p_0, p_1)$	$(\mathrm{refl}_{a_2}, 3, p_0, p_1, p_2)$

Since C_{12} is an equivalence, the triple (C_1, C_2, C_{12}) defines a family

$$C : A_1 \vee A_2 \to \mathrm{Set}.$$

In particular, $C(a_{12}) :\equiv C_1(a_1)$. For $x : A_1$ we let $i_1^C : C_1(x) \to C(i_1(x))$ be the induced equivalence, and likewise for i_2^C. We will show that C is equivalent to $P :\equiv \mathrm{P}_{a_{12}}$, where $P(x) :\equiv (a_{12} = x)$, and so that the identity types in the wedge are equal to the sets provided by C.

One direction is by transport in C; more precisely,

$$\alpha : \prod_{x : A_1 \vee A_2} (P(x) \to C(x))$$

is given by transport with $\alpha(a_{12})(\mathrm{refl}_{a_{12}}) :\equiv (\mathrm{refl}_{a_1}, 0) : C(a_{12})$. The other way,

$$\beta : \prod_{x : A_1 \vee A_2} (C(x) \to P(x))$$

is given by composing identities, using the glue g to make their ends meet:

$$\beta(i_1 a)(p_0, n, p_1, \ldots, p_n) :\equiv i_1(p_0) i_2^g(p_1) i_3^g(p_2) \ldots i_{n+1}^g(p_n)$$

(here the definition $\ldots i_3^g :\equiv i_1^g :\equiv i_1$ proves handy since we don't need to distinguish the odd and even cases) and likewise

$$\beta(i_2 a)(p_0, n, p_1, \ldots, p_n) :\equiv i_2(p_0) g\, i_1^g(p_1) i_2^g(p_2) \ldots i_n^g(p_n)$$

and compatibility with the glue C_{12} is clear since the composite $\mathrm{refl}_x p$ is equal to p.

For notational convenience, we hide the x in $\alpha(x)(p)$ and $\beta(x)(p)$ from now on.

That $\beta\alpha(p) = p$ follows by path induction: it is enough to prove it for $x = a_{12}$ and $p :\equiv \mathrm{refl}_{a_{12}}$:

$$\beta\alpha(\mathrm{refl}_{a_{12}}) = \beta(\mathrm{refl}_{a_1}, 0) = i_1^g \mathrm{refl}_{a_1} = \mathrm{refl}_{a_{12}}.$$

That $\alpha\beta(p_0, n, p_1 \ldots, p_n) = (p_0, n, p_1, \ldots, p_n)$ follows by induction on n and p_0. For $n = 0$ it is enough to consider $x = a_{12}$ and $p_0 = \mathrm{refl}_{a_1}$, and then $\alpha\beta(\mathrm{refl}_{a_1}, 0) :\equiv \alpha(\mathrm{refl}_{a_{12}}) :\equiv (\mathrm{refl}_{a_1}, 0)$. In general, (for $n > 0$)

$$\begin{aligned} \alpha\beta(p_0, n, p_1 \ldots, p_n) &= \mathrm{trp}^C_{i_1(p_0) i_2^g(p_1) i_1^g(p_2) \ldots i_{n+1}^g(p_n)}(\mathrm{refl}_{a_1, 0}) \\ &= \mathrm{trp}^C_{i_1(p_0)} \ldots \mathrm{trp}^C_{i_{n+1}^g(p_n)}(\mathrm{refl}_{a_1, 0}). \end{aligned}$$

The induction step is as follows: let $0 < k \leq n$, then

$$\begin{aligned}
&\mathrm{trp}^{C}_{i^{g}_{k} p_{k\text{-}1}}\ i^{C}_{k-1}(p_k, n-k-1, p_{k+1}, \ldots, p_n) \\
=\,&\mathrm{trp}^{C}_{i^{g}_{k} p_{k\text{-}1}}\ i^{C}_{k}(\mathrm{refl}_{a_{k-1}}, n-k, p_k, \ldots, p_n) \\
=&i^{C}_{k}\ \mathrm{trp}^{C_k}_{p_{k\text{-}1}}(\mathrm{refl}_{a_{k-1}}, n-k, p_k, \ldots, p_n) \\
=&(p_{k-1}, n-k, p_k, \ldots, p_n).
\end{aligned}$$

□

5
Subgroups

ch:subgroups

sec:subgroups

5.1 Subgroups

In our discussion of the group $\mathbb{Z} = \mathrm{Aut}_{S^1}(\bullet)$ of integers in Chapter 3 we discovered that some of the symmetries of $\bullet$ were picked out by the n-fold covering (for some particular natural number n). On the level of the set $\bullet = \bullet$, the symmetries picked out are all the iterates (positive or negative or even zero-fold) of $\circlearrowleft^n$. The important thing is that we can compose or invert any of iterates of $\circlearrowleft^n$ and get new symmetries of the same sort (because of distributivity $nm_1 + nm_2 = n(m_1 + m_2)$). So, while we do not get all symmetries of $\bullet$ (unless $n = 1$), we get what we'd like to call a subgroup of the group of integers.

The other extreme of the idea of a subgroup was exposed in Section 4.9.4 in the form of the slogan "any symmetry is a symmetry in Set". By this we meant that, if $G = \mathrm{Aut}_A(a)$ is a group, we produced a monomorphism $\rho_G : \mathrm{Hom}(G, \mathrm{Aut}_{\mathrm{U}G}(\mathrm{Set}))$, i.e., any symmetry of a is uniquely given by a symmetry ("permutation") of the set $\mathrm{U}G :\equiv (a = a)$.

For yet another example, consider the cyclic group C_6 of order 6; perhaps visualized as the rotational symmetries of a regular hexagon, i.e., the rotations by $2\pi \cdot m/6$, where $m = 0, 1, 2, 3, 4, 5$. The symmetries of the regular triangle (rotations by $2\pi \cdot m/3$, where $m = 0, 1, 2$) can also be viewed as symmetries of the hexagon. Thus there is a subgroup of C_6 which, as a group, is isomorphic to C_3. There are other subgroups of C_6, but in this example they are accounted for simply by the various factorizations of the number 6.

For other groups the subgroups form more involved structures and reveal much about the nature of the object whose symmetries we study. There are several ways to pin down the subgroups and so capture this information. If A is a groupoid, singling out a group of subsymmetries of $a : A$ should be a way of picking out just some of the symmetries of a in A in a way so that we can compose subsymmetries compatibly. To make a long story short; we want a group H and a homomorphism $i : \mathrm{Hom}(H, G)$ so that $\mathrm{U}i : \mathrm{U}H \to \mathrm{U}G$ is injective.[1] We have a name for such a setup: i is a *monomorphism* as laid out in different intermpretations in Lemma 4.9.3.

5.1.1 Subgroups as monomorphisms

The proposition $\mathrm{isMono}(i)$ is equivalent to saying that $\mathrm{U}i : \mathrm{U}H \to \mathrm{U}G$ is an injection (all preimages of $\mathrm{U}i$ are propositions), and also to saying that $Bi : BH \to BG$ is a set bundle, and (since BG is connected), to $\mathrm{isSet}((Bi)^{-1}(\mathrm{sh}_G))$.

[1] in classical set theory it is an important difference between saying that a function is the inclusion of a subset (which is what one classically wants) and saying that it is an injection. We'll address this in a moment, so rest assured that all is well as you read on.

def:typeofmono

Definition 5.1.2. If G is a group, the *type of monomorphisms into G* is

$$\mathrm{Mono}_G :\equiv \sum_{H:\mathrm{Group}} \sum_{i:\,\mathrm{Hom}(H,G)} \mathrm{isMono}(i).$$

A monomorphism $(H, i, !)$ is

(1) *trivial* if BH is contractible (or, equivalently, if $\mathrm{U}H$ is contractible),

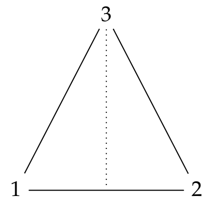
(2) *proper* if Bi is not an equivalence (or, equivalently, if $\mathrm{U}i$ is not an equivalence).

⌟

ex:sigma2inSigma3

Example 5.1.3. Consider the homomorphism $i : \Sigma_2 \to \Sigma_3$ of permutation groups corresponding to $(? + \mathbb{1}, \mathrm{refl}_3) : \mathrm{B}\Sigma_2 \to_* \mathrm{B}\Sigma_3$ (sending A to $A + \mathbb{1}$). This is a monomorphism since $\mathrm{U}i : \mathrm{U}\Sigma_2 \to \mathrm{U}\Sigma_3$ is an injection. ⌟

That $i : \Sigma_2 \to \Sigma_3$ is a monomorphism can visualized as follows: if Σ_3 represent all symmetries of an equilateral triangle in the plane (with vertices 1, 2, 3), then i is represented by the inclusion of the symmetries leaving 3 fixed; i.e., reflection through the line marked with dots in the picture.



ex:prodinclismono

Example 5.1.4. If G and G' are groups, then the first inclusion $i_1 : \mathrm{Hom}(G, G \times G')$ is a monomorphism because $\mathrm{U}i_1 : \mathrm{U}G \to \mathrm{U}(G \times G')$ is an injection.

More generally, if $i : \mathrm{Hom}(H, G)$ is a homomorphism for which there (merely) exists a homomorphism $f : \mathrm{Hom}(G, H)$ such that $\mathrm{id}_H = fi$, then i is a monomorphism. ⌟

We will be interested in knowing when two monomorphisms into G are identical.

lem:setofsubgroups

Lemma 5.1.5. *Let G be a group and $(H, i_H, !), (H', i_{H'}, !) : \mathrm{Mono}_G$ be two monomorphisms into G. The identity type $(H, i_H, !) = (H', i_{H'}, !)$ is equivalent to*

$$\sum_{f:\,\mathrm{Hom}(H,H')} \mathrm{isEquiv}(\mathrm{U}f) \times (i_{H'} = i_H f)$$

and is a proposition. In particular, the type Mono_G *of monomorphisms into G is a set.*

$$\begin{tikzcd} H \arrow[rr, "f", "\simeq"'] \arrow[dr, "i_H"'] & & H' \arrow[dl, "i_{H'}"] \\ & G & \end{tikzcd}$$

Proof. By Lemma 2.10.3 an identity between $(H, i_H, !)$ and $(H', i_{H'}, !)$ is uniquely given by an identity $p : H' =_{\mathrm{Group}} H$ such that $i_{H'} = i_H\, p$ (a proposition since $\mathrm{Hom}(H', G)$ is a set). The description of the identity type follows since by univalence and **??**, the identity type $H = H'$ is equivalent to the set

$$\sum_{f:\,\mathrm{Hom}(H,H')} \mathrm{isEquiv}(\mathrm{U}f).$$

Now, $i_{H'} = i_H f$ is equivalent to $\mathrm{U}i_{H'} = \mathrm{U}i_H \mathrm{U}f$, and since $\mathrm{U}i_H$ is an injection of sets there is at most one such function $\mathrm{U}f$; translating back we see that there is at most one f, making $\sum_f \mathrm{isEquiv}(\mathrm{U}f) \times (i_{H'} = i_H f)$ a proposition. □

If you're familiar with the set-theoretic flavor of things, you know that it is important to distinguish between subgroups and injective group homomorphisms. Our use of monomorphisms can be defended because two monomorphisms into G are identical exactly if they differ by precomposition by an identitification. In set-theoretic language this corresponds to saying that a subgroup is an injective abstract homomorphism *modulo* the relation forcing that precomposing with an isomorphism yields identical subgroups. Classical set-theory offers the luxury of having a preferred representative in every equivalence class: namely the image of the injection, type theory does not. We only know that the type Mono is a set.

5.1.6 *Subgroups through G-sets*

For many purposes it is useful to define "subgroups" slightly differently. A monomorphism into G is given by a pointed connected groupoid $BH = (BH_\div, \mathrm{pt}_H)$, a function $F : BH_\div \to BG_\div$ whose fibers are sets (a set bundle) and an identification $p_f : \mathrm{sh}_G = F(\mathrm{sh}_H)$. There is really no need to specify that $BH_\div$ is a groupoid: if $F : T \to BG$ is a set bundle, then T is automatically a groupoid.

On the other hand, the type of set bundles over BG is equivalent to the type of G-sets: if $X : BG \to \mathrm{Set}$ is a G-set, then the set bundle is given by the first projection $\tilde{X} \to BG$ where $\tilde{X} :\equiv \sum_{y:BG} X(y)$ and the inverse is obtained by considering the fibers of a set bundle. Furthermore, we saw in Lemma 4.5.13 that $\tilde{X}$ being connected is equivalent to the condition $\mathrm{isTrans}(X)$ of Definition 4.5.11 claiming that the G-set X is transitive.

Hence, the type (set, really) Mono_G of monomorphisms into G is equivalent to the type of pointed connected set bundles over BG, which again is equivalent to the type Sub_G of transitive G-sets $X : BG \to \mathrm{Set}$ together with a point in $X(\mathrm{sh}_G)$.

Definition 5.1.7. Let G be a group then the set of *subgroups of* G is

$$\mathrm{Sub}_G :\equiv \sum_{X : BG \to \mathrm{Set}} X(\mathrm{sh}_G) \times \mathrm{isTrans}(X).$$

The preferred equivalence with the set of monomorphisms into G is given by the function

$$E : \mathrm{Mono}_G \to \mathrm{Sub}_G \qquad (H, i, !) \mapsto E(H, i, !) :\equiv ((Bi)^{-1}, (\mathrm{sh}_H, p_i), !),$$

where the monomorphism $i : \mathrm{Hom}(H, G)$ is – as always – given by the pointed map $(Bi_\div, p_i) : (BH_\div, \mathrm{sh}_H) \to_* (BG_\div, \mathrm{sh}_G)$; and where $(Bi)^{-1} : BG \to \mathrm{Set}$ is the preimage G-set (since i is a monomorphism) and finally $(\mathrm{sh}_H, p_i) : (Bi)^{-1}(y) :\equiv \sum_{x:BH}(y = Bi(x))$. ⌟

The inverse equivalence to E is given by sending $(X, \mathrm{pt}, !)$ to the monomorphism associated with the first projection $\sum_{z:BG} X(z) \to BG$.

Example 5.1.8. The monomorphism of Σ_2 into Σ_3 of Example 5.1.3 can be displayed as a subgroup of Σ_3 through

$$X : \mathrm{FinSet}_3 \to \mathrm{Set}$$

given by $A \mapsto \sum_{B:\mathrm{FinSet}2}(A = B + \mathbb{1})$ together with a proof that $\sum_{B:\mathrm{FinSet}2}(A = B + \mathbb{1})$ is a set [the identity type $(B, p) = (B', p')$ is equivalent to $\sum_{q:B=B'}(q + \mathbb{1})p = p'$, so $q + \mathbb{1} = p'p^{-1}$ which pins down q uniquely – check!] ⌟

Which of the equivalent sets Mono_G and Sub_G is allowed to be called "the set of subgroups of G" is, of course, a choice. It could easily have been the other way around and we informally refer to elements in either sets as "subgroups" and use the given equivalence E as needed.

An argument for our choice can be that the identity type in Sub_G seems more transparent than the one in Mono_G ("more things are equal" in Mono_G?), just as $A \to \mathrm{Prop}$ gives more the intuition of picking out a subset by means of a characteristic function than what you get when considering the equivalent type of injections into A.

Exercise 5.1.9. Given a group G we defined in Section 4.9.4 a monomorphism from G to the permutation group $\mathrm{Aut}_{\mathcal{U}G}(\mathrm{Set})$. Write out the corresponding subgroup of $\mathrm{Aut}_{\mathcal{U}G}(\mathrm{Set})$. ⌟

ex:prodinclisGset

Example 5.1.10. We saw in Example 5.1.4 that the first inclusion $i_1 : G \to G \times G'$ is a monomorphism. The corresponding $G \times G'$-set is the composite of the first projection $\mathrm{proj}_1 : BG_\div \times BG'_\div \to BG_\div$ followed by the principal G-torsor $\mathrm{Pr}_G : BG \to \mathrm{Set}$.

More generally, if $i : \mathrm{Hom}(H, G)$ and $f : \mathrm{Hom}(G, H)$, and $fi = \mathrm{id}_H$, then $(H, i, !) : \mathrm{Mono}_G$, corresponding to the subgroup with G-set given by the composite of Bf with the princial H-torsor Pr_H. ⌟

Translating the concepts in Definition 5.1.2 through the equivalence E we say that a subgroup $(X, \mathrm{pt}, !) : \mathrm{Sub}_G$ is

(1) *trivial* if X is identical to the principal G-torsor.

(2) *proper* if $X(\mathrm{sh}_G)$ is not contractible.

rem:notationsubgroup

Remark 5.1.11. A note on classical notation is in order. If $(X, \mathrm{pt}, !)$ is a subgroup corresponding to a monomorphism $(H, i, !)$ into a group G, tradition would permit us to relax the burden of notation and we could write "a subgroup $i : H \subseteq G$", or, if we didn't need the name of $i : \mathrm{Hom}(H, G)$, simply "a subgroup $H \subseteq G$" or "a subgroup H of G". ⌟

5.1.12 *The geometry of subgroups: some small examples*

smallsubgpex

[2]

As a teaser, and in order to get a geometric feel for the subgroups and their intricate interplay, it can be useful to have some fairly manageable examples to stare at. Some of the main tools for analyzing the geometry of subgroups are collected in Section 8.0.1 on finite groups, and we hope the reader will be intrigued by our mysterious claims and go on to study Section 8.0.1. That said, the examples we'll present are possible to muddle through by hand without any fancy machinery, but brute force is generally not an option and even for the present examples it is not something you want to show publicly.

When presenting the subgroups of a group G, three types are especially revealing: the set of subgroups $\mathrm{Sub}_G(\mathrm{sh}_G)$, the *groupoid of subgroups* $\mathrm{Sub}(G) :\equiv \sum_{y:BG} \mathrm{Sub}_G(y)$ and what we for now call the "set of normal subgroups" $\prod_{y:BG} \mathrm{Sub}_G(y)$. Our local use of "normal subgroup" is equivalent to the official definition to come.

The first projection $\mathrm{Sub}(G) \to BG$ is referred to as the *set bundle of subgroups*.

[3]

5.2 *Images, kernels and cokernels*

subsec:ker

The set of subgroups of a group G encodes much information about G, partially because homomorphisms between G and other groups give rise to subgroups.

In Example 4.1.22 we studied a homomorphism from $\mathbb{Z}$ to Σ_m defined via the pointed map $cy_m : S^1 \to_* \mathrm{B}\Sigma_m$ given by sending $\bullet$ to $\mathbb{m}$ and $\circlearrowleft$ to the cyclic permutation $s_m : \mathrm{U}\Sigma_m \equiv (\mathbb{m} = \mathbb{m})$, singling out the iterates of s_m among all permutations. From this we defined the group C_m through a quite general process which we define in this section, namely by taking the *image* of cy_m.

We also noted that the resulting pointed map from S^1 to $\mathrm{B}C_m$ was intimately tied up with the m-fold set bundle $-^m : S^1 \to_* S^1$ – picking out exactly the iterates of $\circlearrowleft^m$ – which in our current language corresponds to a monomorphism $i_m : \mathrm{Hom}(\mathbb{Z}, \mathbb{Z})$. This process is also a special case of something, namely the *kernel*.

The construction of $\mathbb{Z}/m$ and the isomorphism between $\mathbb{Z}/m$ and C_m is also a consequence of what we do in this section, but we'll have to wait till [4] to say this elegantly.

In our setup with a group homomorphism $f : \mathrm{Hom}(G, G')$ being given by a pointed function $Bf : BG \to_* BG'$, the above mentioned kernel, cokernel and image are just different aspects of the preimages

$$(Bf)^{-1}(z) :\equiv \sum_{x:BG} (z = Bf(x))$$

for $z : BG'$. Note that all these preimages are groupoids.

The kernel will correspond to a preferred component of the preimage of $\mathrm{sh}_{G'}$, the cokernel will be the $(G'$-$)$set of components and for the image we will choose the monomorphism into G' corresponding to the cokernel. This point of view makes it clear that the image will be a subgroup of G',

For those familiar with the classical notion, the following summary may guide the intuition.

If $\phi : \mathrm{Hom}^{\mathrm{abs}}(\mathcal{G}, \mathcal{G}')$ is an abstract group homomorphism, the preimage $\phi^{-1}(e_G)$ is an abstract subgroup which is classically called the kernel of ϕ.

On the other hand, the cokernel is the quotient set of $\mathcal{G}'$ by the relation that if $g : \mathcal{G}$ and $g' : \mathcal{G}'$, then $g' \sim g' \cdot \phi(g)$.

[2] this subsection is not touched: it needs attention

[3] Write out and fix the concrete examples (cyclic groups and Σ_3) commented out

[4] findref

the kernel will be a subgroup of G, whereas there is no particular reason for the cokernel to be more than a (G'-) set.

Even though the cokernel is in general just a G'-set, we will see in Definition 5.4.7 that in certain situations it gives rise to a group called the quotient group.

5.2.1 *Kernels and cokernels*

sec:kerandcoker
def:kernel

Definition 5.2.2. We define a function

$$\ker : \mathrm{Hom}(G, G') \to \mathrm{Mono}_G$$

which we call the *kernel*. If $f : \mathrm{Hom}(G, G')$ is a homomorphism we must specify the ingredients in $\ker f :\equiv (\mathrm{Ker}\, f, \mathrm{in}_{\ker f}, !) : \mathrm{Mono}_G$. Consider the element

$$\mathrm{sh}_{\mathrm{Ker}\, f} :\equiv (\mathrm{sh}_G, p_f) : (Bf)^{-1}(\mathrm{sh}_{G'})$$

(where $p_f : \mathrm{sh}_{G'} = Bf(\mathrm{sh}_G)$ is the part of Bf claiming it is a pointed map). Define the *kernel group* (or most often just the kernel) of f to be the group $\mathrm{Ker}\, f$ defined by the pointed component of $\mathrm{sh}_{\mathrm{Ker}\, f}$ in $(Bf)^{-1}(\mathrm{sh}_{G'})$:

$$\mathrm{Ker}\, f :\equiv \mathrm{Aut}_{(Bf)^{-1}(\mathrm{sh}_{G'})}(\mathrm{sh}_{\mathrm{Ker}\, f}).$$

There is an inherent abiguity in our notation: is the kernel of f a group or a monomorphism into G? This is common usage and is only resolved by a typecheck.

The first projection $B\ker f \to BG$ is a set bundle (the preimages are equivalent to the sets $\sum_{p : \mathrm{sh}_{G'} = Bf(z)} \|\mathrm{sh}_{\mathrm{Ker}\, f} = (z, p)\|$) giving a monomorphism $\mathrm{in}_{\ker f}$ of $\ker f$ into G; together defining $\ker f = (\mathrm{Ker}\, f, \mathrm{in}_{\ker f}, !) : \mathrm{Mono}_G$. ⌟

Written out, the classifying type of the kernel, $B\ker f_{\div}$, is

$$\sum_{z : BG} \sum_{p : \mathrm{sh}_{G'} = f(z)} \|\mathrm{sh}_{\mathrm{Ker}\, f} = (z, p)\|.$$

def:cokernel

Definition 5.2.3. Let $f : \mathrm{Hom}(G, G')$ be a homomorphism. The *cokernel* of f is the G'-set

$$\mathrm{coker}\, f : BG' \to \mathrm{Set}, \qquad \mathrm{coker}\, f(z) :\equiv \|(Bf)^{-1}(z)\|_0;$$

defining a function of sets

$$\mathrm{coker} : \mathrm{Hom}(G, G') \to G'\text{-Set}.$$

If $f : \mathrm{Hom}(G, G')$ is clear from the context and displays G as a subgroup of G', we often write G'/G for the cokernel of f. ⌟

THIS WILL BE CHANGED A subgroup is said to be *normal* if it is the kernel of a surjective homomorphism. clarify the relation between the surjective homomorphism and the subgroup

The associated $\mathrm{abs}(G')$-set $\mathrm{coker}\, f(\mathrm{sh}_{G'})$ is also referred to as the cokernel of f.

Exercise 5.2.4. Show that the cokernel is transitive. ⌟

The monomorphisms into G' associated with the cokernel is the “image” of the next section.

Exercise 5.2.5. Given a homomorphism $f : \mathrm{Hom}(G, G')$, prove that

(1) f is a monomorphism if and only if the kernel is trivial

(2) f is an epimorphims if and only if the cokernel is contractible.

(3) if $h : \mathrm{Hom}(L, G)$ is a homomorphism such that $fh : \mathrm{Hom}(H, G')$ factors over the trivial group 1, then there is a unique $k : \mathrm{Hom}(H, \mathrm{Ker}\, f)$ such that $h = \mathrm{in}_{\ker f} k$.

⌟

Hint: consider the corresponding property of the preimage of Bf.

$$\begin{tikzcd}
L \arrow[dr, dotted, "k"] \arrow[drr, "h"] \arrow[ddr] & & \\
& \mathrm{Ker}\, f \arrow[r, "\mathrm{in}_{\ker f}"'] \arrow[d] & G \arrow[d, "f"] \\
& 1 \arrow[r] & G'.
\end{tikzcd}$$

The kernel, cokernel and image constructions satisfy a lot of important relations which we will review in a moment, but in our setup many of them are just complicated ways of interpreting the following fact about preimages lem:fibersofcomposites

Lemma 5.2.6. *Consider pointed functions* $(f_1, p_1) : (X_0, x_0) \to_* (X_1, x_1)$ *and* $(f_2, p_2) : (X_1, x_1) \to_* (X_2, x_2)$ *and the resulting functions*

$$F_1 : f_1^{-1}(x_1) \to (f_2 f_1)^{-1}(x_2), \qquad F_1(x, p) :\equiv (x, p_2 f_2 p),$$

$$F_2 : (f_2 f_1)^{-1}(x_2) \to f_2^{-1}(x_2), \qquad F_2(x, q) :\equiv (f_1 x, q)$$

$$H : F_2^{-1}(x_1, p_2) \to f_1^{-1}(x_1), \qquad H(x, q, (\overline{p, r})) :\equiv (x, p))$$

Then

(1) *H is an equivalence with inverse*

$$H^{-1}(x, q) :\equiv ((x, p_2 f_2 q), (\overline{q, \mathrm{refl}_{p_2 f_2 q}})),$$

(2) *the composite* $F_1 H$ *is definitionally equal to the first projection* $\mathrm{fst} : F_2^{-1}(x_1, p_2) \to (f_2 f_1)^{-1}(x_2)$.

Hence, through univalence, H provides an identification

$$\bar{H} : (F_2^{-1}(x_1, p_2), \mathrm{fst}) = (f_1^{-1}(x_1), F_1)$$

in the type $\sum_{X : \mathcal{U}} (X \to (f_2 f_1)^{-1}(x_2))$ *of function with codomain* $(f_2 f_1)^{-1}(x_2)$.

(here the function

$$((x_1, p_2) = (f_1 x, q)) \xrightarrow{(\overline{p,r}) \mapsto p} (x_1 = f_1(x))$$

is the "first projection" explained in the discussion of the interpretation of pairs following Definition 2.10.1)

$$\begin{tikzcd}
F_2^{-1}(x_1, p_2) \arrow[r, "H", "\simeq"'] \arrow[d, "\mathrm{fst}"'] & f_1^{-1}(x_1) \arrow[d, "\mathrm{fst}"] \arrow[dl, "F_1"'] & \\
(f_2 f_1)^{-1}(x_2) \arrow[r, "\mathrm{fst}"] \arrow[d, "F_2"] & X_0 \arrow[r, "f_2 f_1"] \arrow[d, "f_1"] & X_2 \arrow[d, equal] \\
f_2^{-1}(x_2) \arrow[r, "\mathrm{fst}"] & X_1 \arrow[r, "f_2"] & X_2.
\end{tikzcd}$$

Proof. That H is an equivalence is seen by noting that $F_2^{-1}(x_1, p_2)$ is equivalent to $\sum_{x : X_0} \sum_{q : x_2 = f_2 f_1 x} \sum_{p : x_1 = f_1 x} q = p_2 f_2 p$ and that $\sum_{q : x_2 = f_2 f_1 x} q = p_2 f_2 p$ is contractible. □

From the universal property of the preimage it furthermore follows that F is the unique map such that $\mathrm{fst}\, F =_{f_1^1(x_1) \to X_0} \mathrm{fst}$ and H^{-1} is similarly unique wrt. $\mathrm{fst}\, H^{-1} = F$.

cor:cokermaps

Corollary 5.2.7. *Consider homomorphisms* $f_1 : \mathrm{Hom}(G_0, G_1)$ *and* $f_2 : \mathrm{Hom}(G_1, G_2)$. *There is a unique monomorphisms* F_1 *from* $\ker f_1$ *to* $\ker(f_2 f_1)$ *and a unique homomorphism* F_2 *from* $\ker(f_2 f_1)$ *to* $\ker f_2$ *such that* $\mathrm{in}_{\ker f_1} = \mathrm{in}_{\ker f_2 f_1} F_1$ *and* $f_1 \mathrm{in}_{\ker f_2 f_1} = \mathrm{in}_{\ker f_2} F_2$. *Furthermore,*

$$\begin{tikzcd}
\mathrm{Ker}\, f_1 \arrow[r, equal] \arrow[d, "F_1"] & \mathrm{Ker}\, f_1 \arrow[d, "\mathrm{in}_{\ker f_1}"] & \\
\mathrm{Ker}\, f_2 f_1 \arrow[r, "\mathrm{in}_{\ker f_2 f_1}"] \arrow[d, "F_2"] & G_0 \arrow[r, "f_2 f_1"] \arrow[d, "f_1"] & G_2 \arrow[d, equal] \\
\mathrm{Ker}\, f_2 \arrow[r, "\mathrm{in}_{\ker f_2}"] & G_1 \arrow[r, "f_2"] & G_2
\end{tikzcd}$$

$$F_1 =_{\mathrm{Mono}_{G_1}} \mathrm{in}_{\ker F_2}$$

and

$$(\mathrm{coker}\, f_1)\, B\mathrm{in}_{\ker f_2} =_{B\,\mathrm{Ker}\, f_2 \to \mathrm{Set}} \mathrm{coker}(F_2).$$

Consequequently,

(1) *if* f_2 *is a monomorphism then* $F_1 : \mathrm{Ker}\, f_1 \to \mathrm{Ker}\, f_2 f_1$ *is an isomorphism and*

(2) *if* f_1 *is a monomorphism then* $F_2 : \mathrm{Ker}\, f_2 f_1 \to \mathrm{Ker}\, f_2$ *is an isomorphism.*

[5] *Likewise, the set truncation of the maps* F_1 *and* F_2 *constructed in Lemma 5.2.6 give maps of families* $F_1' : \mathrm{coker}\, f_1 \to_{BG_1 \to \mathrm{Set}} \mathrm{coker}\, f_2 f_1\, B f_2$ *and* $F_2' : \mathrm{coker}\, f_2 f_1 \to_{BG_2 \to \mathrm{Set}} \mathrm{coker}\, f_2$ *such that*

(1) *if* f_2 *is an epimorphism then* $F_1' : \mathrm{coker}\, f_1 \to_{BG_2 \to \mathrm{Set}} \mathrm{coker}\, f_2 f_1\, B f_2$ *is an equivalence and*

(2) *if* f_1 *is an epimorphism then* $F_2' : \mathrm{coker}\, f_2 f_1 \to_{BG_2 \to \mathrm{Set}} \mathrm{coker}\, f_2$ *is an equivalence.*

[5] If $f, g : A \to \mathrm{Set}$ are two A-sets, then $f \to g$ is defined to be the set

$$\prod_{a : A} (f(a) \to g(a))$$

and we say that $\phi : f \to g$ is an equivalence if $\prod_{a : A} \mathrm{isEquiv}\, \phi(a)$.

Exercise 5.2.8. Let $f : \mathrm{Hom}(G, G')$. Then the subgroup $E(\ker f) : \mathrm{Sub}_G$ associated with the kernel is given by a G-set equivalent to the one sending $x : BG$ to

$$\sum_{p\,:\,\mathrm{sh}_{G'}=Bf(x)} \|\sum_{\beta\,:\,\mathrm{sh}_G=x} p = \mathrm{U}f(\beta)p_f\|.$$

If f is an epimorphism this is furthermore equivalent to

$$x \mapsto (\mathrm{sh}_{G'} = Bf(x)).$$

⌟

5.2.9 *The image*

sec:image

For a function $f : A \to B$ of sets (or, more generally, of types) the notion of the "image" gives us a factorization through a surjection followed by an injection: noting that $a \mapsto (f(a), !)$ is a surjection from A to the "image" $\sum_{b\,:\,B} \|f^{-1}(b)\|$, from which we have an injection (first projection) to B. For homomorphism $f : \mathrm{Hom}(G, G')$ of groups we similarly have a factorization through an epimorphism followed by a monomorphism. The image is now represented by $\sum_{zBG'} \|(Bf)^{-1}(z)\|_0$.

In other words, the image is nothing but the subgroup $E(\mathrm{coker}\, f) : \mathrm{Sub}_{G'}$ associated with the cokernel. There is quite a lot to say about this construction and we summarize the most important features.

The formula for the image in group theory is the same as the one for sets, except that the propositional truncation we have for the set factorization is replaced by the set truncation present in our formulation of the cokernel $\mathrm{coker}(f) :\equiv \|(Bf)^{-1}(z)\|_0$.

con:im

Construction 5.2.10. *We define a function*

$$\mathrm{im} : \mathrm{Hom}(G, G') \to \mathrm{Mono}_G$$

called the image *and a function*

$$\mathrm{pr}^{\mathrm{im}} : \mathrm{Hom}(G, G') \to \mathrm{Epi}_G$$

called the projection to the image, *so that if* $f : \mathrm{Hom}(G, G')$ *is a homomorphism, then* $\mathrm{im}\, f :\equiv (\mathrm{Im}\, f, \mathrm{in}_{\mathrm{im}\, f}, !) : \mathrm{Mono}_G$ *and* $\mathrm{pr}^{\mathrm{im}} f :\equiv (\mathrm{Im}\, f, \mathrm{pr}_{\mathrm{Im}\, f}, !) : \mathrm{Epi}_G$ *with common first projection the* image group (*or most often just the image*) $\mathrm{Im}\, f$, *and so that we have a definitional equality in* $\mathrm{Hom}(G, G')$

$$f \equiv \mathrm{in}_{\mathrm{Im}\, f}\mathrm{pr}_{\mathrm{Im}\, f},$$

referred to as the factorization of f through its image.

$$\begin{tikzcd} G \arrow[rr, "f"] \arrow[dr, "\mathrm{pr}_{\mathrm{Im}\, f}"'] & & G' \\ & \mathrm{Im}\, f. \arrow[ur, "\mathrm{in}_{\mathrm{Im}\, f}"'] & \end{tikzcd}$$

Given two homomorphisms $f_1 : \mathrm{Hom}(G_0, G_1)$ *and* $f_2 : \mathrm{Hom}(G_1, G_2)$, *there is a unique homomorphism* $u : \mathrm{Hom}(\mathrm{Im}\, f_2f_1, \mathrm{Im}\, f_2)$ *satisfying*

(1) $\mathrm{in}_{\mathrm{im}\, f_2f_1} = \mathrm{in}_{\mathrm{im}\, f_2}u$ *and*

(2) $\mathrm{pr}_{\mathrm{im}\, f_2}f_1 = u\, prj_{\mathrm{im}\, f_2f_1}$.

$$\begin{tikzcd} G_0 \arrow[r, "\mathrm{pr}_{\mathrm{im}\, f_2f_1}"] \arrow[d, "f_1"'] & \mathrm{Im}\, f_2f_1 \arrow[r, "\mathrm{in}_{\mathrm{im}\, f_2f_1}"] \arrow[d, "u"'] & G_2 \arrow[d, equal] \\ G_1 \arrow[r, "\mathrm{pr}_{\mathrm{im}\, f_2}"'] & \mathrm{Im}\, f_2 \arrow[r, "\mathrm{in}_{\mathrm{im}\, f_2}"'] & G_2 \end{tikzcd}$$

The homomorphism u is always a monomorphism and is an isomorphism if and only if f_1 *is an epimorphism.*

The factorization of $f : \mathrm{Hom}(G, G')$ *through its image is unique in the sense that if given two homomorphisms* $f_1 : \mathrm{Hom}(G_0, G_1)$ *and* $f_2 : \mathrm{Hom}(G_1, G_2)$ *such that* f_1 *is an epimorphism and* f_2 *in a monomorphism, then there is a unique isomorphism* $t : \mathrm{Hom}(\mathrm{Im}\, f_2f_1, G_1)$ *such that* $f_1 = t\, \mathrm{pr}_{\mathrm{im}\, f_2f_1}$ *and* $f_2\, t = \mathrm{in}_{\mathrm{im}\, f_2f_1}$. *Through univalence t gives rise to identifications*

$$f_1 =_{\mathrm{Epi}_G} \mathrm{pr}_{\mathrm{im}\, f_2f_1} \text{ and } f_2 =_{\mathrm{Mono}_{G'}} \mathrm{in}_{\mathrm{im}\, f_2f_1}.$$

$$\begin{tikzcd} G_0 \arrow[r, "\mathrm{pr}_{\mathrm{im}\, f_2f_1}"] \arrow[d, "f_1"'] & \mathrm{Im}\, f_2f_1 \arrow[d, "\mathrm{in}_{\mathrm{im}\, f_2f_1}"] \\ G_1 \arrow[r, "f_2"'] \arrow[ur, dotted, "t"] & G_2 \end{tikzcd}$$

Implementation of Construction 5.2.10. Consider the element

$$\mathrm{sh}_{\mathrm{Im}\, f} :\equiv (\mathrm{sh}_{G'}, |\mathrm{sh}_G, p_f|) : \sum_{z:BG'} \mathrm{coker}\, f(z)$$

and define the image group of f to be

$$\mathrm{Im}\, f :\equiv \mathrm{Aut}_{\sum_{z:BG'} \mathrm{coker}\, f(z)}(\mathrm{sh}_{\mathrm{Im}\, f}).$$

The first projection $\mathrm{fst} : B\,\mathrm{Im}\, f \to BG'$ is a set bundle (since $\mathrm{coker}\, f(z)$ is a set) giving the desired monomorphism $\mathrm{in}_{\mathrm{im}\, f}$ of $\mathrm{Im}\, f$ into G.

The homomorphism $\mathrm{pr}_{\mathrm{im}\, f} : \mathrm{Hom}(G, \mathrm{Im}\, f)$ is given on the level of classifying types by sending $x : BG$ to

$$B\mathrm{pr}_{\mathrm{Im}\, f} f(x) :\equiv (Bf(x), |x, \mathrm{refl}_{Bf(x)}|) : \mathrm{Im}\, f.$$

From this it is clear that Bf is definitionally equal to the composite of $B\mathrm{in}_{\mathrm{Im}\, f}$ and $B\mathrm{pr}_{\mathrm{Im}\, f}$. That $\mathrm{pr}_{\mathrm{im}\, f}$ is an epimorphims is best seen by using the equivalence between BG and $\sum_{z:BG'}(Bf)^{-1}(z)$ which translates $\mathrm{pr}_{\mathrm{im}\, f}$ to the sum over $z : BG'$ of the truncation $(Bf)^{-1}(z) \to \|(Bf)^{-1}(z)\|_0 \equiv \mathrm{coker}\, f(z)$ which has connected fiber (recall that a homomorphism is an epimorphism iff its classifying map has connected fibers).

and if the wrapping destroys this fact in some ugly manner, it is an argument against wrapping

Assume given homomorphisms $f_1\, \mathrm{Hom}(G_0, G_1)$ and $f_2 : \mathrm{Hom}(G_1, G_2)$. The claimed homomorphism $u : \mathrm{Hom}(\mathrm{Im}\, f_2 f_1, \mathrm{Im}\, f_2)$ has classifying map $Bu : \sum_{z:BG_2}(\mathrm{coker}\, f_2 f_1)(z) \to_* \sum_{z:BG_2}(coker f_2)(z)$ given by $F_2'(z) : (\mathrm{coker}\, f_2 f_1)(z) \to (\mathrm{coker}\, f_2)(z)$ from Corollary 5.2.7 (which was the truncation of the map of preimages $F_2 : (Bf_2 f_1)^{-1}(z) \to (Bf_2)^{-1}(z)$ with $F_2(x,p) \equiv (f_1 x, p)$). That $\mathrm{in}_{\mathrm{im}\, f_2 f_1} = \mathrm{in}_{\mathrm{im}\, f_2} u$ and $\mathrm{pr}_{\mathrm{im}\, f_2} f_1 = u\, prj_{\mathrm{im}\, f_2 f_1}$ follows by the definitions of the maps and the uniqueness of u follows from the fact that $\mathrm{in}_{\mathrm{im}\, f_2}$ is a monomorphism and the demand $\mathrm{in}_{\mathrm{im}\, f_2 f_1} = \mathrm{in}_{\mathrm{im}\, f_2} u$ – which also forces u to be a monomorphism since $\mathrm{in}_{\mathrm{im}\, f_2 f_1}$ is a monomorphism. Conversely, if f_1 is an epimorphism, then the composite $\mathrm{pr}_{\mathrm{im}\, f_2} f_1$ is an epimorphism and $\mathrm{pr}_{\mathrm{im}\, f_2} f_1 = u\, prj_{\mathrm{im}\, f_2 f_1}$ forces u to be an epimorphism, and so an isomorphims.

As for the uniqueness of the factorization, assume given $f_1\, \mathrm{Hom}(G_0, G_1)$ and $f_2 : \mathrm{Hom}(G_1, G_2)$ such that f_1 is an epimorphism and f_2 in a monomorphism, we let $t : \mathrm{Hom}(G_1, \mathrm{Im}\, f_2 f_1)$ be the composite $u^{-1}\mathrm{pr}_{\mathrm{im}\, f_2}$, where u is invertible since f_1 is an epimorphism. Since f_1 is an epimorphism, f_2 a monomorphism and the claimed diagram commutes this forces t to be an isomorphism.

□

In view of Exercise 5.2.11 below, the families

$$\mathrm{isepi}, \mathrm{ismono} : \mathrm{Hom}(G, G') \to \mathrm{Prop}$$

of propositions that a given homomorphism is an epimorphism or monomorphism have several useful interpretations (parts of the exercise have already been done).

ex:charsurinj

Exercise 5.2.11. Let $f : \mathrm{Hom}(G, G')$ Prove that

(1) the following are equivalent

a) f is an epimorphism,

b) Uf is a surjection

c) the cokernel of f is contractible,

d) the "inclusion of the image" $\mathrm{Im}_f : \mathrm{Hom}(\mathrm{Im}\, f, G')$ is an isomorphism,

(2) the following are equivalent

a) f is a monomorphism,

b) $\mathrm{U}f$ is an injection

c) the kernel of f is trivial

d) $Bf : BG \to BG'$ is a set bundle.

e) the projection onto the image $\mathrm{pr}_{\mathrm{im}\, f} : \mathrm{Hom}(G, \mathrm{Im}\, f)$ is an isomorphism.

⌟

lem:[illegible]

Lemma 5.2.12. *Let $f : \mathrm{Hom}(G, G')$ be a group homomorphism. The induced map $(B\mathrm{pr}_{\mathrm{im}\, f})^{-1}(\mathrm{sh}_{\mathrm{Im}\, f}) \to (Bf)^{-1}(\mathrm{sh}_{G'})$ gives an identification*

$$\ker \mathrm{pr}_{\mathrm{im}\, f} =_{\mathrm{Mono}_G} \ker f.$$

Proof. Using univalence, this is a special case of Corollary 5.2.7 with $f_2 :\equiv \mathrm{in}_{\mathrm{im}\, f}$ and $f_1 :\equiv \mathrm{pr}_{\mathrm{im}\, f}$. [6] □

Exercise 5.2.13. (1) If $f : \mathrm{Mono}_{G'}$, then $\mathrm{ua}(\mathrm{pr}_{\mathrm{im}\, f}) : f =_{\mathrm{Mono}_{G'}} \mathrm{in}_{\mathrm{im}\, f}$.

(2) If $f : \mathrm{Epi}_G$, then $\mathrm{ua}(\mathrm{in}_{\mathrm{im}\, f}) : f =_{\mathrm{Epi}_G} \mathrm{pr}_{\mathrm{im}\, f}$.
(True propositions suppressed). ⌟

Example 5.2.14. An example from linear algebra: let A be any $n \times n$-matrix with nonzero determinant and with integer entries, considered as a homomorphism $A : \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^n)$. "Nonzero determinant" corresponds to "monomorphism". Then the cokernel of A is a finite set with cardinality the absolute value of the determinant of A. You should picture A as a $|\det(A)|$-fold set bundle of the n-fold torus $(S^1)^{\times n}$ by itself.

In general, for an $m \times n$-matrix A, then the "nullspace" is given by the kernel and the "rowspace" is given by the image. ⌟

5.3 *The action on the set of subgroups*

sec:actiononsub

Not only is the type of subgroups of G a set, it is in a natural way (equivalent to the value at sh_G of) a G-set which we denote by the same name. We first do the monomorphism interpretation

Definition 5.3.1. If G is the group, the *G-set of monomorphisms into G* $\mathrm{Mono}_G : BG \to \mathrm{Set}$ is given by

$$\mathrm{Mono}_G(y) :\equiv \sum_{H : \mathrm{Group}} \sum_{f : \mathrm{Hom}(H,G)(y)} \mathrm{isSet}(Bf^{-1}(\mathrm{sh}_G))$$

for $y : BG$, where – as in Example 4.5.5 –

$$\mathrm{Hom}(H, G)(y) :\equiv \sum_{F : BH_{\div} \to BG_{\div}} (y = F(\mathrm{sh}_H))$$

is the G-set of homomorphisms from H to G. ⌟

The type of monomorphisms into G is $\mathrm{Mono}(\mathrm{sh}_G)$, and as $y : BG$ varies, the only thing that changes in $\mathrm{Mono}_G(y)$ is that $BG = (BG_{\div}, \mathrm{sh}_G)$ is replaced by $(BG_{\div}, y)$.

[6] Also show counting results for the finite group part somewhere.

def:conjactonmonos
def:conjugate

Definition 5.3.2. If G is a group, then the action of G on the set of monomorphisms into G is called *conjugation*.

If $(H, F, p, !) : \mathrm{Mono}_G(\mathrm{sh}_G)$ is a monomorphism into G and $g : \mathrm{U}G$, then the monomorphisms $(H, F, p, !), (H, F, p\, g^{-1}, !) : \mathrm{Mono}_G(\mathrm{sh}_G)$ are said to be *conjugate*. ⌟

rem:whyconjugate

Remark 5.3.3. The term "conjugation" may seem confusing as the action of $g : \mathrm{U}G$ on a monomorphism $(H, F, p, !) : \mathrm{Mono}_G(\mathrm{sh}_G)$ (where $p : x = F(\mathrm{sh}_H)$) is simply $(H, F, p\, g^{-1}, !)$, which does not seem much like conjugation. However, as we saw in Example 4.11.2, under the equivalence $\mathrm{abs} : \mathrm{Hom}(H, G) \xrightarrow{\sim} \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(H), \mathrm{abs}(G))$, the corresponding action on $\mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(H), \mathrm{abs}(G))$ is exactly (postcomposition with) conjugation $c^g : \mathrm{abs}(G) = \mathrm{abs}(G)$. [7] ⌟

rem:conjactiononmonos

Summing up the remark:

lem:conjugationabstractly

Lemma 5.3.4. *Under the equivalence of Lemma 4.11.1 between G-sets and* $\mathrm{abs}(G)$*-sets, the G-set* Mono_G *corresponds to the* $\mathrm{abs}(G)$*-set*

$$\sum_{H : \mathrm{Group}} \; \sum_{\phi : \mathrm{Hom}^{\mathrm{abs}}(\mathrm{abs}(H), \mathrm{abs}(G))} \mathrm{isProp}(\phi^{-1}(e_G))$$

of abstract monomorphisms *of* $\mathrm{abs}(G)$*, with action* $g \cdot (H, \phi, !) :\equiv (H, c^g\, \phi, !)$ *for* $g : \mathrm{abs}(G)$*, where* $c^g : \mathrm{abs}(G) = \mathrm{abs}(G)$ *is conjugation as defined in Example 4.3.21.*

rem:typeofsubgpstrivifab

Remark 5.3.5. We know that a group G is abelian if and only if conjugation is trivial: for all $g : \mathrm{U}G$ we have $c^g = \mathrm{id}$, and so we get that Mono_G is a trivial G-set if and only if G is abelian. ⌟

The subgroup analog of $y \mapsto \mathrm{Mono}_G(y)$ is

Definition 5.3.6. Let G be a group and $y : BG$, then the G-set of *subgroups of G* is

$$\mathrm{Sub}_G : BG \to \mathrm{Set}, \qquad \mathrm{Sub}_G(y) :\equiv \sum_{X : BG \to \mathrm{Set}} X(y) \times \mathrm{isTrans}(X).$$

⌟

The only thing depending on y in $\mathrm{Sub}_G(y)$ is where the "base"point is residing.

Extending the equivalence of sets we get an equivalence of G-sets $E : \mathrm{Mono}_G \to \mathrm{Sub}_G$ via

$$E(y) : \mathrm{Mono}_G(y) \to \mathrm{Sub}_G(y), \qquad E(H, F, p_F, !) = (F^{-1}, (\mathrm{sh}_H, p_F), !)$$

for $y : BG$ (where H is a group, $F : BH_\div \to BG_\div$ is a map and $p_F : y = F(\mathrm{sh}_H)$ an identity in BG; and $F^{-1} : BG \to \mathrm{Set}$ is G-set given by the preimages of F and $(\mathrm{sh}_H, p_F) : F^{-1}(y) :\equiv \sum_{x : BH} y = F(x)$ is the base point). If y is sh_G we follow our earlier convention of dropping it from the notation.

Since the families are equivalent (via E) we use Mono_G or Sub_G interchangeably.

5.4 *Normal subgroups*

sec:normal

In the study of groups, the notion of a normal subgroup is of vital important, and as any important concept it comes in many guises,

[7] The same phenomenon appeared in Exercise 4.5.6 where we gave an equivalence between the G-sets $\mathrm{Hom}(\mathbb{Z}, G)$ and Ad_G (where the action is very visibly by conjugation).

revealing the different aspects. For now we just state the definition in the form that it is a subgroup fixed under the action of G on the G-set of subgroups.

def:normalsubgroup

Definition 5.4.1. The set of *normal subgroups* is

$$\mathrm{Nor}_G :\equiv \prod_{y\,:\,BG} \mathrm{Sub}_G(y)$$

considered as a subset of Sub_G via the injection

$$i : \mathrm{Nor}_G \to \mathrm{Sub}_G, \qquad i(N) :\equiv N(\mathrm{sh}_G).$$

⌟

The corresponding set of fixed point in the G-set of monomorphisms

$$\prod_{y\,:\,BG} \mathrm{Mono}_G(y)$$

will not figure as prominently, since the focus shifts naturally to an equivalent set which we have already defined, namely the kernels.

Restricting the equivalence $E : \mathrm{Mono}_G \to \mathrm{Sub}_G$ to the fixed sets, we get an equivalence from $\prod_{y\,:\,BG} \mathrm{Mono}_G(y)$ to Nor_G

def:setofkernels

Definition 5.4.2. If G is a group, let

$$\begin{tikzcd}
\mathrm{Epi}_G \arrow[r, two heads, "\ker"] & \mathrm{Ker}_G \arrow[r, tail, "i"] & \mathrm{Mono}_G
\end{tikzcd}$$

be the surjection/injection factorization of the kernel function restricted to the epimorphisms from G. We call the subset Ker_G the *set of kernels*. ⌟

Our aim is twofold:

(1) we want to show that $\ker : \mathrm{Epi}_G \to \mathrm{Ker}_G$ is an equivalence, so that knowing that a monomorphism is *a* kernel is equivalent to knowing an epimorphism it is *the* kernel of.

(2) we want to show that the kernels correspond via the equivalence E to the fixed points under the G action on the G-set of subgroups.

For $x', y' : BG'$, recall the notation $\mathrm{P}_{y'}(x') :\equiv (y' = x')$.

We will achieve these goals by defining a function $\mathrm{nor} : \mathrm{Epi}_G \to \mathrm{Nor}_G$ which we show is an equivalence and, furthermore, that the two functions $i\mathrm{nor}, Ei\ker : \mathrm{Epi}_G \to \mathrm{Sub}_G$ are identical. Since $i\mathrm{nor}$ is an injection, this forces the surjection ker to be injective too and we are done.

def:ker2

Definition 5.4.3. Define

$$\mathrm{nor} : \mathrm{Epi}_G \to \mathrm{Nor}_G$$

by $\mathrm{nor}(G', f, !)(y) :\equiv (\mathrm{P}_{f(y)}f, \mathrm{refl}_{f(y)}, !)$ for $y : BG$. ⌟

Remark 5.4.4. The G-set $\mathrm{P}_{f(y)}f$ is not a G-torsor (except if f is an isomorphism). ⌟

lem:diagfornormal

Lemma 5.4.5. *The diagram*

$$\begin{tikzcd}
 & \mathrm{Ker}_G \arrow[r, tail, "i"] & \mathrm{Mono}_G \arrow[dd, "E", "\simeq"'] \\
\mathrm{Epi}_G \arrow[ur, two heads, "\ker"] \arrow[dr, "\mathrm{nor}"'] & & \\
 & \mathrm{Nor}_G \arrow[r, tail, "i"] & \mathrm{Sub}_G
\end{tikzcd}$$

commutes, where the top composite is the image factorization of the kernel and the bottom inclusion is the inclusion of fixed points.

Proof. Following $(G', f, !) : \mathrm{Epi}_G$ around the top to Sub_G yields the transitive G-set sending $y : BG$ to the set $\mathrm{sh}_{G'} = f(y)$ together with the point $p_f : \mathrm{sh}_{G'} = f(\mathrm{sh}_G)$ while around the bottom we get the transitive G-set sending $y : BG$ to the set $f(\mathrm{sh}_G) = f(y)$ together with the point $\mathrm{refl}_{f(\mathrm{sh}_G)} : f(\mathrm{sh}_G) = f(\mathrm{sh}_G)$. Hence, precomposition by p_f gives the identity proving that the diagram commutes. □

We will prove that both ker and nor in the diagram of Lemma 5.4.5 are equivalences, leading to the desired conclusion that the equivalence $E : \mathrm{Mono}_G \xrightarrow{\sim} \mathrm{Sub}_G$ takes the subset Ker_G identically to Nor_G. Actually, since $\mathrm{ker} : \mathrm{Epi}_G \to \mathrm{Ker}_G$ is a surjection, we only need to know it is an injection, and for this it is enough to show that nor is an equivalence; we'll spell out the details.

Since it has independent interest, we take a detour via a quotient group construction of Definition 5.4.7 which gives us the desired inverse of nor.

We start with a small, but crucial observation.

lem:evaliseqwhennormal

Lemma 5.4.6. *Let $N : \mathrm{Nor}_G$ be a normal subgroup with $N(y) \equiv (X_y, \mathrm{pt}_y, !)$ for $y : BG$. Then for any $y, z : BG$*

(1) *the evaluation map*

$$\mathrm{ev}_{yz} : (X_y = X_z) \to X_z(y), \qquad \mathrm{ev}_{yz}(f) = f_y(\mathrm{pt}_y)$$

is an equivalence and

(2) *the map $X : (y = z) \to (X_y = X_z)$ (given by induction via $X_{\mathrm{refl}_y} :\equiv \mathrm{refl}_{X_y}$) is surjective.*

Proof. To establish the first fact we need to do induction independently on $y : BG$ and $z : BG$ in $X_y(z)$ at the same time as we observe that it suffices (since BG is connected) to show that ev_{yy} is an equivalence.

The composite

$$\mathrm{ev}_{yy} X : (y = y) \to X_y y$$

is determined by $\mathrm{ev}_{yy} X(\mathrm{refl}_y) \equiv \mathrm{pt}_y$. By transitivity of X_y this composite is surjective, hence ev_{yy} is surjective too.

On the other hand, in Lemma 4.5.14 we used the transitivity of X_y to deduce that ev_{yy} was injective. Consequently ev_{yy} is an equivalence. But since ev_{yy} is an equivalence and $\mathrm{ev}_{yy} X$ is surjective we conclude that X is surjective □

def:normalquotient

Definition 5.4.7. Let $N : \mathrm{Nor}_G$ be a normal subgroup with $N(y) \equiv (X_y, \mathrm{pt}_y, !)$ for $y : BG$. The *quotient group* is

$$G/N :\equiv \mathrm{Aut}_{G\text{-Set}}(X_{\mathrm{sh}_G}).$$

The *quotient homomorphism* is the homomorphism $q_N : \mathrm{Hom}(G, G/N)$ defined by $Bq_N(z) = X_z$ (strictly pointed). By Lemma 5.4.6, q_N is an epimorphism and we have defined a map

$$q : \mathrm{Nor}_G \to \mathrm{Epi}_G, \qquad q(N) = (G/N, q_N, !).$$

⌟

Remark 5.4.8. It is instructive to see how the quotient homomorphism $Bq_N : BG \to BG/N$ is defined in the torsor interpretation of BG. If $Y : BG \to \mathcal{U}$ is a G-type we can define the quotient as

$$Y/N : BG \to \mathcal{U}, \qquad Y/N(y) :\equiv \sum_{z:BG} Y(z) \times X_z(y).$$

We note that in the case $\mathrm{Pr}_G(y) :\equiv (\mathrm{sh}_G = y)$ we get that $\mathrm{Pr}_G/N(y) :\equiv \sum_{z:BG}(\mathrm{sh}_G = z) \times X_z(y)$ is equivalent to X_{sh_G}. Consequently, if Y is a G-torsor, then Y/N is in the component of X_{sh_G} and we have

$$-/N : \mathrm{Torsor}_G :\equiv (G\text{-set})_{(\mathrm{Pr}_G)} \to (G\text{-set})_{(X_{\mathrm{sh}_G})}.$$

Our quotient homomorphism $q_N : \mathrm{Hom}(G, G/N)$ is the composite of the equivalence $\mathrm{P}^G : BG \xrightarrow{\sim} \mathrm{Torsor}_G$ of Theorem 4.6.6 and the quotient map $-/N$. ⌟

lem:qeq

Lemma 5.4.9. *The map* $\mathrm{nor} : \mathrm{Epi}_G \to \mathrm{Nor}_G$ *is an equivalence with inverse* $q : \mathrm{Nor}_G \to \mathrm{Epi}_G$.

Proof. Assume $N : \mathrm{Nor}_G$ with $N(y) :\equiv (X_y, \mathrm{pt}_y, !)$ for $y : BG$. Then $\mathrm{nor}\, q(N) : BG \to \mathrm{Set}$ takes $y : BG$ to $(\mathrm{nor}\, q(N))(y) \equiv (Y_y, \mathrm{refl}_{X_y}, !)$, where $Y_y(z) :\equiv (X_y = X_z)$. Noting that the equivalence $\mathrm{ev}_{yz} : (X_y = X_z) \xrightarrow{\sim} X_z(y)$ of Lemma 5.4.6 has $\mathrm{ev}_{yy}(\mathrm{refl}_{X_y}) :\equiv \mathrm{pt}_y$ we see that univalence gives us the desired identity $\mathrm{nor}\, q(N) = N$.[8]

Conversely, let $f : \mathrm{Hom}(G, G')$ be an epimorphism. Recall that the quotient group is $G/\mathrm{nor}(f) :\equiv \mathrm{Aut}_{G\text{-Set}}(\mathrm{P}_{f(\mathrm{sh}_G)} f)$ and the quotient homomorphism $q_{\mathrm{nor} f} : \mathrm{Hom}(G, G/\mathrm{nor} f)$ is given by sending $y : BG$ to $\mathrm{P}_{f(y)} f : BG \to \mathrm{Set}$ (strictly pointed – i.e., by $\mathrm{refl}_{\mathrm{P}_{f(\mathrm{sh}_G)} f}$). We define a homomorphism $Q : \mathrm{Hom}(G', G/\mathrm{nor} f)$ by sending $z : BG'$ to $\mathrm{P}_z f$ and using the identification $\mathrm{P}_{\mathrm{sh}_{G'}} f = \mathrm{P}_{f(\mathrm{sh}_G)} f$ induced by $pf : \mathrm{sh}_{G'} = f(\mathrm{sh}_G)$ and notice the definitional equality

$$Q f \equiv q_{\mathrm{nor} f} : \mathrm{Hom}(G, G/\mathrm{nor} f).$$

We are done if we can show that Q is an isomorphism. The preimage of the base point $\mathrm{P}_{f(\mathrm{sh}_G)} f$ is

$$\sum_{z:BG'} \prod_{y:BG} (z = f(y)) = (f(\mathrm{sh}_G) = f(y))$$

which by Lemma 4.11.4 is equivalent to

$$\sum_{z:BG'} \prod_{v:BG'} (z = v) = (f(\mathrm{sh}_G) = v)$$

which by Lemma 4.6.5 is equivalent to the contractible type $\sum_{z:BG'} z = f(\mathrm{sh}_G)$. □

cor:normalisnormal

Corollary 5.4.10. *The kernel* $\ker : \mathrm{Epi}_G \to \mathrm{Ker}_G$ *is an equivalence of sets.*

Proof. Since $\mathrm{nor} : \mathrm{Epi}_G \to \mathrm{Nor}_G$ and $E : \mathrm{Mono}_G \to \mathrm{Sub}_G$ are equivalences, the inclusion of fixed points $i : \mathrm{Nor} \to \mathrm{Sub}$ is an injection and the diagram in Lemma 5.4.5 commutes, the surjection $\ker : \mathrm{Epi}_G \to \mathrm{Ker}_G$ is also an injection. □

the diagram in Lemma 5.4.5

$$\begin{tikzcd}
 & \mathrm{Ker}_G \arrow[r, tail, "i"] & \mathrm{Mono}_G \arrow[dd, "\simeq"', "E"] \\
\mathrm{Epi}_G \arrow[ur, "\ker"] \arrow[dr, "\mathrm{nor}"', "\simeq"] & & \\
 & \mathrm{Nor}_G \arrow[r, tail, "i"] & \mathrm{Sub}_G
\end{tikzcd}$$

Summing up, using the various interpretations of subgroups, we get the following list of equivalent sets all interpreting what a normal subgroup is.

[8] fix so that it adhers to dogmatic language and naturality in N is clear

lem:characterizations of normal

Lemma 5.4.11. *Let G be a group, then the following sets are equivalent*

(1) *The set* Epi_G *of epimorphisms from* G*,*

(2) *the set* Ker_G *of kernels of epimorphisms from* G*,*

(3) *the set* Nor_G *of fixed points of the* G*-set* Sub_G *(aka. normal subgroups),*

(4) *the set of fixed points of the* G*-set* Mono_G*,*

(5) *the set of fixed points of the* G*-set of abstract subgroups of* $\mathrm{abs}(G)$ *of Lemma 5.3.4.*

sec:assker

5.4.12 *The associated kernel*

With this much effort in proving that different perspectives on the concept of "normal subgroups" (in particular, kernels and fixed points) are the same, it can be worthwhile to make the composite equivalence

$$\ker q : \mathrm{Nor}_G \xrightarrow{\simeq} \mathrm{Ker}_G$$

explicit – where the quotient group function $q : \mathrm{Nor}_G \to \mathrm{Epi}_G$ is the inverse of nor constructed in Definition 5.4.7 – and even wite out a simplification.

Let $N : \mathrm{Nor}_G$ be a normal subgroup with $N(y) \equiv (X_y, \mathrm{pt}_y, !)$ for $y : BG$ with $X_y : BG \to \mathrm{Set}$, $\mathrm{pt}_y : X_y(y)$ and $! : \mathrm{isTrans}(X_y)$. Then

$$\mathrm{Ker}\, q(N) :\equiv \mathrm{Aut}_{\sum_{x:BG}(X_x = X_{\mathrm{sh}_G})}(\mathrm{sh}_G, \mathrm{refl}_{X_{\mathrm{sh}_G}})$$

and with the monomorphism $\mathrm{in}_{\ker q(N)} : \mathrm{Hom}(\mathrm{Ker}\, q(N), G)$ given by the first projection from $\sum_{x:BG}(X_x = X_{\mathrm{sh}_G})$ to BG.

However, going the other way around the pentagon of Lemma 5.4.5, we see that $\mathrm{ass}(N) :\equiv E^{-1} i(N) : \mathrm{Mono}_G$ consists of the group

$$\mathrm{Ass}(N) :\equiv \mathrm{Aut}_{\sum_{x:BG} X_{\mathrm{sh}_G}(x)}(\mathrm{sh}_G, \mathrm{pt}_{\mathrm{sh}_G})$$

and the monomorphism into G given by the first projection (monomorphism because X_{sh_G} has values in sets). Since the pentagon commutes we know that $\mathrm{ass}(N)$ is the kernel of $q(N) : \mathrm{Epi}_G$, and the identification $\mathrm{ev} : i \ker q(N) =_{\mathrm{Mono}_G} \mathrm{ass}(N)$ is given via Lemma 5.4.6 and univalence by the equivalence

$$\mathrm{ev}_{x\,\mathrm{sh}_G} : (X_x = X_{\mathrm{sh}_G}) \to X_{\mathrm{sh}_G}(x).$$

Letting the proposition that $\mathrm{ass}(N)$ is a kernel be invisible in the notation we may summarize the above as follows:

def:associatednormal

Definition 5.4.13. If $N : \mathrm{Nor}_G$ is a normal subgroup we call the kernel $\mathrm{ass}(N) : \mathrm{Ker}_G$ the *kernel assocaited to* N. ⌟

Remark 5.4.14. In forming the kernel associated to N, where did we use that N was a fixed point of the G-set Sub_G? If $Y : BG \to \mathrm{Set}$ is a transitive G-set and $\mathrm{pt} : Y(\mathrm{sh}_G)$, then surely we could consider the group

$$W :\equiv \mathrm{Aut}_{X:BG\to\mathrm{Set}}(Y)$$

as a substitute for the quotient group (see Section 5.6). One problem is that we wouldn't know how to construct a homomorphism from G to W which we then could consider the kernel of. And even if we tried our hand inventing formulae for the outcomes, ignoring all subscripts, we'd be stuck at the very end where we used Lemma 5.4.6 to show that the evaluation map is an equivalencs; if we only had transitivity we could try to use a variant of Lemma 4.5.14 to pin down injectivity, but surjectivity needs the extra induction freedom. ⌟

lem:normalsarekernels

Lemma 5.4.15. *The diagram of equivalences*

$$\begin{tikzcd}
 & \mathrm{Ker}_G \arrow[r, hook, "i"] & \mathrm{Mono}_G \arrow[d, "E", "\simeq"'] \\
\mathrm{Epi}_G \arrow[ur, "\mathrm{ker}", "\simeq"'] \arrow[dr, "\mathrm{nor}", "\simeq"'] & & \\
 & \mathrm{Nor}'_G \arrow[uu, "\mathrm{ass}", "\simeq"'] \arrow[r, hook, "i"] & \mathrm{Sub}_G
\end{tikzcd}$$

commutes.

5.5 *The pullback*

sec:pullback
def:pullback

Definition 5.5.1. Let B, C, D be types and let $f : B \to D$ and $g : C \to D$ be two maps. The *pullback* of f and g is the type

$$\prod(f, g) :\equiv \sum_{(b,c) : B\times C} (f(b) =_D g(c))$$

together with the two projections $\prod(f, g) \to B$ and $\prod(f, g) \to C$ sending $(b, c, p) : \prod(f, g)$ to $b : B$ or $c : C$. If f and g are clear from the context, we may write $B \times_D C$ instead of $\prod(f, g)$ and summarize the situation by the diagram

$$\begin{tikzcd}
B \times_D C \arrow[r] \arrow[d] & C \arrow[d, "g"] \\
B \arrow[r, "f"] & D.
\end{tikzcd}$$

⌟

xca:univpropofpullback

Exercise 5.5.2. Let $f : B \to D$ and $g : C \to D$ be two maps with common target. If A is a type show that

$$\begin{aligned}
(A \to B) \times_{(A\to D)} (A \to C) \to& (A \to B \times_D C) \\
(\beta, \gamma, p : f\beta = g\gamma) \mapsto& (a \mapsto (f(a), g(a), p(a) : f\beta(a) = g\gamma(a)))
\end{aligned}$$

is an equivalence. ⌟

Example 5.5.3. If $g : \mathbb{1} \to D$ has value $d : D$ and $f : B \to D$ is any map, then $\prod(f, g) \equiv B \times_D \mathbb{1}$ is equivalent to the preimage $f^{-1}(d) :\equiv \sum_{b:B} d = f(b)$. ⌟

ex:pullbackandgcd

Example 5.5.4. Much group theory is hidden in the pullback. For instance, the greatest common divisor $\gcd(a, b)$ of $a, b : \mathbb{N}$ is another name for the number of components you get if you pull back the a-fold and the b-fold cover of the circle: as we will see in **??** we have a pullback

$$\begin{tikzcd}
S^1 \times C_{\gcd(a,b)} \arrow[r] \arrow[d] & S^1 \arrow[d, "(-)^b"] \\
S^1 \arrow[r, "(-)^a"] & S^1
\end{tikzcd}$$

(where C_n was the cyclic group of order n). To get a geometric idea, think of the circle as the unit circle in the complex numbers so that the a-fold cover is simply taking the a-fold power. With this setup, the pullback should consist of pairs (z_1, z_2) of unit length complex numbers with the property that $z_1^a = z_2^b$. Let $a = a'G$ and $b = b'G$ where $G = \gcd(a, b)$. Taking an arbitrary unit length complex number z, then the pair $(z^{b'}, z^{a'})$ is in the pull back (since $a'b = ab'$). But so is $(\zeta z^{b'}, z^{a'})$, where ζ is any G-th root of unity. Each of the G-choices of ζ contributes in this way to a component of the pullback. In more detail: identifying the cyclic group C_G of order G with the group of g-th roots of unity, the top horizontal map $S^1 \times C_G \to S^1$ sends (z, ζ) to $z^{a'}$ and the left vertical map sends (z, ζ) to the product $\zeta z^{b'}$.

Also the least common multiple is hidden in the pullback; in the present example it is demonstrated that the map(s) accross the diagram makes each component of the pullback a copy of the subgroup $a'b\mathbb{Z}$ of $\mathbb{Z}$. ⌟

def:intersectionand unionofsets

Definition 5.5.5. Let S be a set and consider two subsets A and B of S given by two families of propositions (for $s : S$) $P(s)$ and $Q(s)$. The *intersection* $A \cap B$ of the two subsets is given by the family of propositions $P(s) \times Q(s)$. The *union* $A \cup B$ is given by the set family of propositions $A(s) + B(s)$. ⌟

xca:intersectionpullbackofsets

xca:cardinalityintersectionunion

Exercise 5.5.6. Given two subsets A, B of a set S, prove that

(1) The pullback $A \times_S B$ maps by an equivalence to the intersection $A \cap B$,

(2) If S is finite, then the sum of the cardinalities of A and B is equal to the sum of the cardinalities of $A \cup B$ and $A \cap B$.

⌟

def:intersectionofgroups

Definition 5.5.7. Let $f : \mathrm{Hom}(H, G)$ and $f' : \mathrm{Hom}(H', G)$ be two homomorphisms with common target. The *pullback* $H \times_G H'$ is the group obtained as the (pointed) component of

$$\mathrm{pt}_{H \times_G H'} :\equiv (\mathrm{sh}_H, \mathrm{pt}_{H'}, p_{f'} p_f^{-1})$$

of the pullback $BH \times_{BG} BH'$ (where $p_f : \mathrm{sh}_G = f(\mathrm{sh}_H)$ is the name we chose for the data displaying f as a pointed map, so that $p_{f'} p_f^{-1} : f(\mathrm{sh}_H) = f'(\mathrm{pt}_{H'})$).

If $(H, f, !)$ and $(H', f', !)$ are monomorphisms into G, then the pullback is called the *intersection* and if the context is clear denoted simply $H \cap H'$. ⌟

Example 5.5.8. If $a, b : \mathbb{N}$ are natural number with least common multiple L, then $L\mathbb{Z}$ is the intesection $a\mathbb{Z} \cap b\mathbb{Z}$ of the subgroups $a\mathbb{Z}$ and $b\mathbb{Z}$ of $\mathbb{Z}$. ⌟

Exercise 5.5.9. Prove that if $f : \mathrm{Hom}(H, G)$ and $f' : \mathrm{Hom}(H', G)$ are homomorphisms, then the pointed version of Exercise 5.5.2 induces an equivalence

$$\mathrm{U}H \times_{\mathrm{U}G} \mathrm{U}H' \simeq (\mathrm{pt}_{H \times_G H'} = \mathrm{pt}_{H \times_G H'})$$

(hint: set $A :\equiv S^1$, $B :\equiv BH$, $C :\equiv BH'$ and $D :\equiv BG$). Elevate this equivalence to a statement about abstract groups. ⌟

Exercise 5.5.10. If $\mathcal{G}$ is an abstract group and $\mathcal{H}$ and $\mathcal{K}$ are abstract subgroups. Give a definition of the intersection $\mathcal{H} \cap \mathcal{K}$ is the abstract subgroup of $\mathcal{G}$ agreeing with our definition for groups. ⌟

lem:whatSylow2needs

Lemma 5.5.11. *Let* $(G', f, !) : \mathrm{Epi}_G$, *$N$ be the kernel of f and let* $(H, i, !) : \mathrm{Mono}_G$. *Then $N \cap H$ is the kernel of fi* $: \mathrm{Hom}(H, G')$. *and the induced homomorphism in* $\mathrm{Hom}(H/(N \cap H), G')$ *is a monomorphism.*

Proof. Now, N is the kernel of the epimorphism f, giving an equivalence between $BN_{\div}$ and the preimage

$$(Bf)^{-1}(\mathrm{sh}_{G'}) :\equiv \sum_{y : BG} (\mathrm{sh}_{G'} = Bf(y)).$$

Writing out the definition of the pullback (and using that for each $x : BH$ the type $\sum_{y : BG} y = Bi(x)$ is contractible), we get an equivalence between $BN \times_{BG} BH$ and

$$B(fi)^{-1}(\mathrm{sh}_{G'}) :\equiv \sum_{x : BH} \mathrm{sh}_{G'} = B(fi)x,$$

the preimage of $\mathrm{sh}_{G'}$ of the composite $B(f\,i) : BH \to BG'$. By definition, the intersection $B(N \cap H)$ is the pointed component of the pullback containing $(\mathrm{pt}_N, \mathrm{sh}_H)$. Under the equivalence with $B(f\,i)^{-1}(\mathrm{sh}_{G'})$ the intersection corresponds to the component of $(\mathrm{sh}_H, Bf(p_i)\,p_f)$. Since (by definition of the composite of pointed maps) $p_{f\,i} :\equiv Bf(p_i)\,p_f$ we get that the intersection $N \cap H$ is identified with the kernel of the composite $f\,i : \mathrm{Hom}(H, G')$.

Finally, since $N \cap H$ is the kernel of the composite $f\,i : \mathrm{Hom}(H, G')$, under the equivalence of Lemma 5.2.12, $N \cap H$ is equivalent to the kernel of the epimorphism $\mathrm{pr}_{\mathrm{im}(f\,i)} : \mathrm{Hom}(H, \mathrm{Im}(f\,i))$. Otherwise said, the quotient group $H/(N \cap H)$ is another name for the image $\mathrm{Im}(f\,i)$, and $\mathrm{in}_{\mathrm{im}(f\,i)}$ is indeed a monomorphism into G'. □

Exercise 5.5.12. Write out all the above in terms of the set Sub_G of subgroups of G instead of in terms of the set Mono_G of monomorphism into G. ⌟

Is the below misplaced?

Recall that if $X : BG \to \mathrm{Set}$ is a G-set, then the set of fixed points is the set $\prod_{v\,:\,BG} X(v)$, which is a subset of $X(\mathrm{sh}_G)$ via the evaluation map. If a homomorphism from a group H to G is given by $F : BH_{\div} \to BG_{\div}$ and $p_F : \mathrm{sh}_G = F(\mathrm{sh}_H)$, then precomposition ("restriction of scalars") by F gives an H-set

$$F^*X :\equiv X\,F : BH \to \mathrm{Set}.$$

In the case of inclusions of subgroups (or other situations where the homomorphism is clear from the context) it is not uncommon to talk about "the H-set X" rather than "F^*X". This can be somewhat confusing when it comes to fixed points: the fixed points of F^*X are given by $\prod_{v\,:\,BH} XF(v)$ which evaluates nicely to $XF(\mathrm{sh}_H)$, but in order to considered these as elements in $X(\mathrm{sh}_G)$ we need to apply $X(p_F^{-1}) : X(F(\mathrm{sh}_H)) = X(\mathrm{sh}_G)$.

Consequently, we'll say that $x : X(\mathrm{sh}_G)$ is an *H-fixed point* if there is an $f : \prod_{v\,:\,BH} XF(v)$ such that $x = X(p_F^{-1})f(\mathrm{sh}_H)$.

lem:thereisaconjugate

Lemma 5.5.13. *Let G be a group, $X : BG \to \mathit{Set}$ a G-set, $x : X(\mathrm{sh}_G)$, $g : UG$ and $H = (H, F, p, !) : \mathrm{Sub}_G$ a subgroup of G ($F : BH_{\div} \to BG_{\div}$ and $p : \mathrm{sh}_G = F(\mathrm{sh}_H)$).*

Then $g\,x$ is a fixed point for the H-action on X if and only if x is a fixed point for the action of the conjugate subgroup $g\,H :\equiv (H, F, g^{-1}p_F, !)$ on X.

Proof. Consider an $f : \prod_{v\,:\,BH} XF(v)$. Then $g \cdot x = X(p_F^{-1})(f(\mathrm{sh}_H))$ if and only if $x = g^{-1} \cdot X(p_F^{-1})(f(\mathrm{sh}_H)) \equiv X((g^{-1}p_F)^{-1}(f(\mathrm{sh}_H)))$. □

5.6 *The Weyl group*

sec:Weyl

In Definition 5.4.7 defined the quotient group of a normal subgroup. As commented in Definition 5.4.13, the definition itself never used that the subgroup was normal (but the quotient homomorphism did) and is important in this more general context.

Recall the equivalence E between the set Mono_G of monomorphisms and the set Sub_G of of subgroups of G (pointed transitive G-sets): The subgroup $(X, \mathrm{pt}_X, !) : \mathrm{Sub}_G$ where $X : BG \to \mathrm{Set}$ is a transitive G-set and $\mathrm{pt}_X : X(\mathrm{sh}_G)$ corresponds to $(H, i_H, !) : \mathrm{Mono}_G$ defined by

$$H :\equiv \mathrm{Aut}_{\sum_{y\,:\,BG} X(y)}(\mathrm{sh}_G, \mathrm{pt}_X)$$

together with the first projection from $\sum_{y:BG} X(y)$ to BG. Conversely, if $(H, i_H, !) : \mathrm{Mono}_G$, then the corresponding transitive G-set is $G/H :\equiv$ coker i_H pointed at $|\mathrm{sh}_H, p_{i_H}|$: coker $i_H(\mathrm{sh}_G) :\equiv \|\sum_{x:BH} \mathrm{sh}_G = Bi_H(x)\|_0$.

For the remainder of the section we'll consider a fixed group G, monomorphism $i_H : \mathrm{Hom}(H, G)$ and $(X, \mathrm{pt}_X, !)$ will be the associated pointed transitive G-set.

def:Weyl

Definition 5.6.1. The *Weyl group*

$$W_GH :\equiv \mathrm{Aut}_{G\text{-set}}(X)$$

is defined by the component BW_GH of the groupoid of G-sets pointed at X.

def:normalizer

The *normalizer subgroup*

$$N_GH :\equiv \mathrm{Aut}_{\sum_{y:BG}\mathrm{Sub}_G(y)}(\mathrm{sh}_G, X, \mathrm{pt}_X)$$

is defined by the component BN_GH of the groupoid $\sum_{y:BG} \mathrm{Sub}_G(y)$ pointed at $(\mathrm{sh}_G, X, \mathrm{pt}_X)$. ⌟

Unpacking, we find that

$$BN_GH_\div \equiv \sum_{y:BG} \sum_{Y:BG\to\mathrm{Set}} \sum_{\mathrm{pt}^y_Y:Y(y)} \|(\mathrm{sh}_G, X, \mathrm{pt}_X) = (y, Y, \mathrm{pt}^y_Y)\|.$$

While the projection $((\mathrm{sh}_G, X, \mathrm{pt}_X) = (y, Y, \mathrm{pt}^y_Y)) \to (X = Y)$ may not be an equivalence, the transitivity of X tells us that for any $\beta : X = Y$ there is a $g : \mathrm{sh}_G = y$ such that $X(g)\,p^y_Y = \beta_y^{-1}\mathrm{pt}_X$, and so the propositional truncation $\|(\mathrm{sh}_G, X, \mathrm{pt}_X) = (y, Y, \mathrm{pt}^y_Y)\| \to \|X = Y\|$ is an equivalence. Consequently, the projection

$$BN_GH_\div \to \sum_{y:BG} \sum_{Y:BG\to\mathrm{Set}} Y(y) \times \|X = Y\|$$

is an equivalence. With an innocent rewriting, we see that we have provided an equivalence

$$e : BN_GH_\div \xrightarrow{\sim} \sum_{(y\times Y):BG\times BW_GH} Y(y) \qquad e(y, Y, \mathrm{pt}^y_Y, !) :\equiv (y, Y, \mathrm{pt}^y_Y, !).$$

This formulation has the benefit of simplifying the analysis of the monomorphism

$$i_{N_GH} : \mathrm{Hom}(N_GH, G)$$

given by $Bi_{N_GH}(y, Y, \mathrm{pt}^y_Y, !) :\equiv y$, the "projection"

$$p^H_G : \mathrm{Hom}(N_GH, W_GH)$$

$Bp^H_G(y, Y, \mathrm{pt}^y_Y, !) :\equiv (Y, !)$ and the monomorphism

$$j_H : \mathrm{Hom}(H, N_GH)$$

given by $Bj_H(y, v) :\equiv (y, X, v, !))$.

Lemma 5.6.2. *The monomorphism* $i^H_G : \mathrm{Hom}(N_GH, G)$ *displays the normalizer as a subgroup of* G *and the projection* $p^H_G : \mathrm{Hom}(N_GH, W_GH)$ *is an epimorphism.*

The homomorphism $j_H : \mathrm{Hom}(H, N_GH)$ *defines* H *as a normal subgroup of the normalizer,*

$$\ker p^H_G =_{\mathrm{Mono}_{N_GH} for} (H, i_H, !)$$

and $i_H =_{\mathrm{Hom}(H,G)} i^H_G\, j_H$.

Proof. Immediate from (our rewriting of) the definitions. □

The Weyl group $W_G H$ has an important interpretation. It is defined as symmetries of the transitive G-set X, and so $\mathrm{pt}_{W_G H} = \mathrm{pt}_{W_G H}$ is nothing but $(X =_{G\text{-set}} X) = \prod_{y:BG}(X(y) = X(y))$. On the other hand, $BH_{\div}$ is equivalent to $\sum_{y:BG} X(y)$ and

$$\prod_{y:BG}(X(y) = X(y)) \simeq \prod_{\sum_{y:BG} X(y)} X(y),$$

so $\mathrm{pt}_{W_G H} = \mathrm{pt}_{W_G H}$ is equivalent to the set $\prod_{x:BH} X\, Bi_H x$ of fixed points of $X = G/H$ (regarded as an H-set through i_H).

Summing up

lem:WGHisHfixofG/H

Lemma 5.6.3. *The map $e:(X = X) \to \prod_{x:BH} X\, Bi_H x$ with $e(f)(y,v) = f(y)$ defines an equivalence*

$$e:(\mathrm{pt}_{W_G H} = \mathrm{pt}_{W_G H}) \xrightarrow{\sim} (G/H)^H.$$

6 Symmetry

ch:symmetry

sec:cayley-diagram

6.1 *Cayley diagram*

We have seen in the previous chapter how cyclic groups (those generated by a single generator) have neatly described torsors.

In this section we shall generalize this story to groups G generated by a (finite or just decidable) set of generators S.

$G \simeq \mathrm{Aut}(D_G) \to \mathrm{Sym}(\mathrm{Card}\, G)$

sec:actions

6.2 *Actions*

action

Definition 6.2.1. If G is any (possibly higher) group and A is any type of objects, then we define an *action* by G in the world of elements of A as a function

$$X : BG \to A. \qquad \lrcorner$$

The particular object of type A being acted on is $X(\mathrm{pt}) : A$, and the action itself is given by transport. This generalizes our earlier definition of G-sets, $X : BG \to \mathrm{Set}$.

std-action

Definition 6.2.2. The *standard action* of G on its designated shape sh_G is obtained by taking $A :\equiv \mathrm{BG}$ and $X :\equiv \mathrm{id}_{BG}$. ⌟

Example 6.2.3. An action of G on its set UG of symmetries is provided by taking X to be the principal torsor Pr_G as defined in Example 4.5.3. ⌟

Notice that the type $BG \to A$ is equivalent to the type

$$\sum_{a:A} \hom(G, \mathrm{Aut}_A(a)),$$

that is, the type of pairs of an element $a : A$, and a homomorphism from G to the automorphism group of A. The equivalence maps $X : BG \to A$

fig:cayley-s3

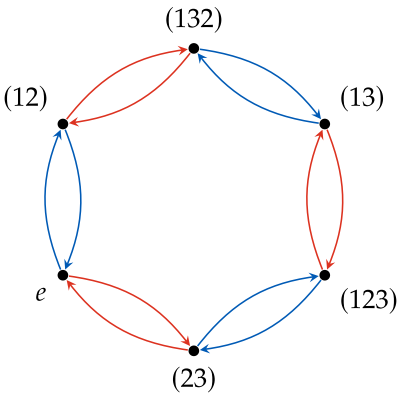


Figure 6.1: Cayley diagram for S_3 with respect to $S = \{(12), (23)\}$.

to the pair consisting of $X(\mathrm{pt})$ and the homomorphism represented by the pointed map arising from corestricting X to factor through the component of A containing a together with the trivial proof that this map takes $\mathrm{pt} : BG$ to a.

Because of this equivalence, we define a *G-action on $a : A$* to be a homomorphism from G to $\mathrm{Aut}_A(a)$.

Many times we are particularly interested in actions on types, i.e., A is a universe (or the universe of types-at-large):

$$X : BG \to \mathcal{U}.$$

In this case, we define *orbit type* of the action as

$$X_G :\equiv \sum_{z : BG} X(z),$$

and the type of *fixed points* as

$$X^G :\equiv \prod_{z : BG} X(z).$$

The set of orbits is the set-truncation of the orbit type,

$$X/G :\equiv \|X_G\|_0.$$

We say that the action is *transitive* if X/G is contractible.

6.3 *Heaps* (†)

sec:heaps

Recall that we in Remark 4.1.4 wondered about the status of general identity types $a =_A a'$, for a and a' elements of a groupoid A, as opposed to the more special loop types $a =_A a$. Here we describe the resulting algebraic structure and how it relates to groups.

This section has no implications for the rest of the book, and can thus safely be skipped on a first reading. (TODO: Move in place in Chapter 4?)

We proceed in a fashion entirely analogous to that of Section 4.1, but instead of looking a pointed types, we look at *bipointed types*.

def:bipt-conn-groupoid

Definition 6.3.1. The type of *bipointed, connected groupoids* is the type

$$\mathcal{U}^{=1}_{**} :\equiv \sum_{A : \mathcal{U}^{=1}} (A \times A). \qquad \lrcorner$$

Recall that $\mathcal{U}^{=1}$ is the type of connected groupoids A, and that we also write $A : \mathcal{U}$ for the underlying type. We write $(A, a, a') : \mathcal{U}^{=1}_{**}$ to indicate the two endpoints.

Analogous to the loop type of a pointed type, we have a designated identity type of a bipointed type, where we use the two points as the endpoints of the identifications: We set $\mathrm{I}(A, a, a') :\equiv (a =_A a')$.

def:heap

Definition 6.3.2. The type of *heaps*[1] is a wrapped copy (cf. Section 2.12.5) of the type of bipointed, connected groupoids $\mathcal{U}^{=1}_{**}$,

$$\mathrm{Heap} :\equiv \mathrm{Copy}_{\underline{\mathrm{I}}}(\mathcal{U}^{=1}_{**}),$$

with constructor $\underline{\mathrm{I}} : \mathcal{U}^{=1}_{**} \to \mathrm{Heap}$. $\lrcorner$

We call the destructor $\mathrm{B} : \mathrm{Heap} \to \mathcal{U}^{=1}_{**}$, and call BH the *classifying type* of the heap $H \equiv \underline{\mathrm{I}}BH$, just as for groups, and we call the first point in BH is *start shape* of H, and the second point the *end shape* of H.

[1] The concept of heap (in the abelian case) was first introduced by Prüfer[2] under the German name *Schar* (swarm/flock). In Anton Sushkevich's book Теория Обобщенных Групп (*Theory of Generalized Groups*, 1937), the Russian term груда (heap) is used in contrast to группа (group). For this reason, a heap is sometimes known as a "groud" in English.

[2] Heinz Prüfer. "Theorie der Abelschen Gruppen". In: *Math. Z.* 20.1 (1924), pp. 165–187. DOI: 10.1007/BF01188079.

The identity type construction $\mathrm{I} : \mathcal{U}^{=1}_{**} \to \mathrm{Set}$ induces a map $\mathrm{U} : \mathrm{Heap} \to \mathrm{Set}$, mapping $\underline{\mathrm{I}}X$ to $\mathrm{I}X$. These are the *underlying identifications* of the heaps.

These is an obvious map (indeed a functor) from groups to heaps, given by doubling the point. That is, we keep the classifying type and use the designated shape as both start and end shape of the heap. In fact, this map lifts to the type of heaps with a chosen identification.

xca:group+torsor-heap

Exercise 6.3.3. Define natural equivalences $\mathrm{Heap} \simeq \sum_{G : \mathrm{Group}} BG$, and $\mathrm{Group} \simeq \sum_{H : \mathrm{Heap}} (\mathrm{U}H)$. ⌟

Recalling the equivalence between BG and the type of G-torsors from Theorem 4.6.6, we can also say that a heap is the same as a group G together with a G-torsor.[3] It also follows that the type of heaps is a (large) groupoid.

In the other direction, there are *two* obvious maps (functors) from heaps to groups, taking either the start or the end shape to be the designated shape.

Here's an *a priori* different map from heaps to groups: For a heap H, consider all the symmetries of the underlying set of identifications $\mathrm{U}H$ that arise as $r \mapsto pq^{-1}r$ for $p, q \in \mathrm{U}H$.

Note that (p, q) and (p', q') determine the same symmetry if and only if $pq^{-1} = p'q'^{-1}$, and if and only if $p'^{-1}p = q'^{-1}q$.

For the composition, we have $(p, q)(p', q') = (pq^{-1}p', q') = (p, q'p'^{-1}q)$.

Exercise 6.3.4. Complete the argument that this defines a map from heaps to groups. Can you identify the resulting group with the symmetry group of the start or end shape? How would you change the construction to get the other endpoint? ⌟

Exercise 6.3.5. Show that the symmetry groups of the two endpoints of a heap are *merely* isomorphic.

Define the notion of an *abelian heap*, and show that for abelian heaps, the symmetry groups of the endpoints are (*purely*) isomorphic. ⌟

Now we come to the question of describing the algebraic structure of a heap. Whereas for groups we can define the abstract structure in terms of the reflexivity path and the binary operation of path composition, for heaps, we can define the abstract structure in terms of a *ternary operation*, as envisioned by the following exercise.

xca:heap-variety

Exercise 6.3.6. Fix a set S. Show that the fiber $\mathrm{U}^{-1}(S) \equiv \sum_{H : \mathrm{Heap}} (S = \mathrm{U}H)$ is a set.

Now fix in addition a ternary operation $t : S \times S \times S \to S$ on S. Show that the fiber of the map $\mathrm{Heap} \to \sum_{S : \mathrm{Set}} (S \times S \times S \to S)$, mapping H to $(\mathrm{U}H, (p, q, r) \mapsto pq^{-1}r)$, at (S, t) is a proposition, and describe this proposition in terms of equations. ⌟

6.4 *Semidirect products*

sec:Semidirect-products

In this section we describe a generalization of the product of two group, called the *semidirect* product, which can be constructed from an action of a group on a group. Like the product, it consists of pairs, both at the level of concrete groups and of abstract groups, as we shall see.

[3] But be aware that are *two* such descriptions, according to which endpoint is the designated shape, and which is the "twisted" torsor.

We start with some preliminaries on paths between pairs. Lemma Lemma 2.10.3 above takes a simpler form when y and y' are values of a family $x \mapsto f(x)$ of elements of the family $x \mapsto Y(x)$, as the following lemma shows.

lem:pathpairsection

Lemma 6.4.1. *Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. Suppose we are also given a function $f : \prod_{x:X} Y(x)$. For any elements x and x' of X, there is an equivalence of type*

$$((x, f(x)) = (x', f(x'))) \simeq (x = x') \times (f(x) = f(x)),$$

where the identity type on the left side is between elements of $\sum_{x:X} Y(x)$.

Proof. By Lemma 2.10.3 and by composition of equivalences, it suffices to establish an equivalence of type

$$\left(\sum_{p:x=x'} f(x) \overset{Y}{\underset{p}{=}} f(x') \right) \simeq (x = x') \times (f(x) = f(x)).$$

Rewriting the right hand side as a sum over a constant family, it suffices to find an equivalence of type

$$\left(\sum_{p:x=x'} f(x) \overset{Y}{\underset{p}{=}} f(x') \right) \simeq \sum_{p:x=x'} (f(x) = f(x)).$$

By Lemma 2.9.14 it suffices to establish an equivalence of type

$$\left(f(x) \overset{Y}{\underset{p}{=}} f(x') \right) \simeq (f(x) = f(x))$$

for each $p : x = x'$. By induction on x' and p we reduce to the case where x' is x and p is refl_x, and it suffices to establish an equivalence of type

$$\left(f(x) \overset{Y}{\underset{\mathrm{refl}_x}{=\!=}} f(x) \right) \simeq (f(x) = f(x)).$$

Now the two sides are equal by definition, so the identity equivalence provides what we need. □

The lemma above shows how to rewrite certain paths between pairs as pairs of paths. Now we wish to establish the formula for composition of paths, rewritten in terms of pairs of paths, but first we introduce a convenient definition for the transport of loops in $Y(x)$ along paths in X.

def:pathsectionaction

Definition 6.4.2. Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. Suppose we are also given a function $f : \prod_{x:X} Y(x)$. For any elements x and x' of X and for any identity $p : x = x'$, define a function $(f(x') = f(x')) \to (f(x) = f(x))$, to be denoted by $q' \mapsto q'^{p}$, by induction on p and x', reducing to the case where x' is x and p is refl_x, allowing us to set $q'^{\mathrm{refl}_x} :\equiv q'$. ⌟

We turn now to associativity for the operation just defined.

def:pathsectionactionassoc

Lemma 6.4.3. *Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. Suppose we are also given a function $f : \prod_{x:X} Y(x)$. For any elements x, x', and x'' of X, for any identities $p : x = x'$ and $p' : x' = x''$, and for any $q : fx'' = fx''$, there is an identification of type $(q^{p'})^{p} = q^{(p' \cdot p)}$.*

Proof. By induction on p and p', it suffices to show that $(q^{\mathrm{refl}_y})^{\mathrm{refl}_y} = q^{(\mathrm{refl}_y \cdot \mathrm{refl}_y)}$, in which both sides are equal to q by definition. □

Observe that the operation depends on f, but f is not included as part of the notation.

The next lemma contains the formula we are seeking.

lem:pathpairsectionmult

Lemma 6.4.4. *Suppose we are given a type X and a family of types $Y(x)$ parametrized by the elements x of X. Suppose we are also given a function $f : \prod_{x:X} Y(x)$. For any elements x, x', and x'' of X, and for any two identities $e : (x, f(x)) = (x', f(x'))$ and $e' : (x', f(x')) = (x'', f(x''))$, if e corresponds to the pair (p, q) with $p : x = x'$ and $q : fx = fx$ under the equivalence of Lemma 6.4.1, and e' corresponds to the pair (p', q') with $p' : x' = x''$ and $q' : fx' = fx'$, then $e' \cdot e$ corresponds to the pair $(p' \cdot p, (q'^{p}) \cdot q)$.*

Proof. By induction on p and p' we reduce to the case where x' and x'' are x and p and p' are refl_x. It now suffices to show that $e' \cdot e$ corresponds to the pair $(\mathrm{refl}_x, q' \cdot q)$. Applying the definition of the map Φ in the proof of Lemma 2.10.3 to our three pairs, we see that it suffices to show that $\left(\mathrm{apap}_g(\mathrm{refl}_x)(q')\right) \cdot \left(\mathrm{apap}_g(\mathrm{refl}_x)(q)\right) = \mathrm{apap}_g(\mathrm{refl}_x)(q' \cdot q)$, with g, as there, being the function $g(x)(y) :\equiv (x, y)$. By Definition 2.7.6 it suffices to show that $\left(\mathrm{ap}_{g(x)}\, q'\right) \cdot \left(\mathrm{ap}_{g(x)}\, q\right) = \mathrm{ap}_{g(x)}\,(q' \cdot q)$, which follows from compatibility of $\mathrm{ap}_{g(x)}$ with composition, as in Lemma 2.6.2. □

The lemma above will be applied mostly in the case where x' and x'' are x, but if it had been stated only for that case, we would not have been able to argue by induction on p and p'.

def:semidirect-product

Definition 6.4.5. Given a group G and an action $\tilde{H} : BG \to \mathrm{Group}$ on a group $H :\equiv \tilde{H}(\mathrm{sh}_G)$, we define a group called the *semidirect product* as follows.

$$G \ltimes \tilde{H} :\equiv \underline{\Omega} \sum_{t:BG} \mathrm{B}\tilde{H}(t)$$

Here the basepoint of the sum is taken to be the point $(\mathrm{sh}_G, \mathrm{sh}_H)$. (We deduce from Lemma 2.15.4, Item (4), that $\sum_{t:BG} \mathrm{B}\tilde{H}(t)$ is a groupoid. See **??** for a proof that $\sum_{t:BG} \mathrm{B}\tilde{H}(t)$ is connected.) ⌟

Observe that if the action of G on H is trivial, then $\tilde{H}(t) \equiv H$ for all t and $G \ltimes \tilde{H} \equiv G \times H$.

Projection onto the first factor gives a homomorphism $p :\equiv \underline{\Omega}\,\mathrm{fst} : G \ltimes \tilde{H} \to G$. Moreover, there is a homomorphism $s : G \to G \ltimes \tilde{H}$ defined by $s :\equiv \underline{\Omega}\left(t \mapsto (t, \mathrm{sh}_{\tilde{H}(t)})\right)$, for $t : \mathrm{B}G$. The two maps are homomorphisms because they are made from basepoint-preserving maps. The map s is a *section* of p in the sense the $p \circ s = \mathrm{id}_G$. There is also a homomorphism $j : H \to G \ltimes \tilde{H}$ defined by $j :\equiv \underline{\Omega}(u \mapsto (\mathrm{sh}_G, u))$, for $u : \mathrm{B}H$.

Lemma 6.4.6. *The homomorphism j above is a monomorphism, and it gives the same (normal) subgroup of $G \ltimes \tilde{H}$ as the kernel* $\ker p$ *of p.*

Proof. See 5.2.2 for the definition of kernel. According to Lemma 2.25.1, the map $\mathrm{B}H \to (\mathrm{B}p)^{-1}(\mathrm{sh}_G)$ defined by $u \mapsto ((\mathrm{sh}_G, u), \mathrm{refl}_{\mathrm{sh}_G})$ is an equivalence. This establishes that the fiber $(\mathrm{B}p)^{-1}(\mathrm{sh}_G)$ is connected and thus serves as the classifying type of $\ker p$. Pointing out that the composite map $H \xrightarrow{\cong} \ker p \to G \ltimes \tilde{H}$ is j and using univalence to promote the equivalence to an identity gives the result. □

Our next goal is to present the explicit formula for the multiplication operation in $UG \ltimes \tilde{H}$. First we apply Lemma 6.4.1 to get a bijection $UG \ltimes \tilde{H} \simeq UG \times UH$. Now use that to transport the multiplication operation of the group $UG \ltimes \tilde{H}$ to the set $UG \times UH$. Now Lemma 6.4.4 tells us the formula for that transported operation is given as follows.

$$(p', q') \cdot (p, q) = (p' \cdot p, (q'^{p}) \cdot q)$$

In a traditional algebra course dealing with abstract groups, this formula is used as the definition of the multiplication operation on the set $UG \times UH$, but then one must prove that the operation satisfies the properties of Definition 4.2.1. The advantage of our approach is that the formula emerges from the underlying logic that governs how composition of paths works.

6.5 *Orbit-stabilizer theorem*

sec:orbit-stabilizer-theorem

Given an action $X : BG \to \mathcal{U}$ and a point $x : X(\mathrm{pt})$, we define the *orbit* through x as the subtype of $X(\mathrm{pt})$ consisting of all $y : X(\mathrm{pt})$ that are merely equal to x in the orbit type:

$$G \cdot x :\equiv \mathcal{O}_x :\equiv \sum_{y : X(\mathrm{pt})} \|[x] = [y]\|_{-1}$$

(Note the unfortunate terminology: an orbit is not an element in the orbit type!) Note that this only depends on the image of x in the set of orbits, thus justifying the names.

In this way, the type $X(\mathrm{pt})$ splits as a disjoint union of orbits, parametrized by the set of orbits

$$X(\mathrm{pt}) \simeq \amalg_{z : X/G} \mathcal{O}_z.$$

The *stabilizer group* G_x of $x : X(\mathrm{pt})$ is the automorphism group of $[x]$ in the orbit type. Different points in the same orbit have conjugate stabilizer groups.

We say that the action is *free* if all stabilizer groups are trivial.

thm:orbitstab

Theorem 6.5.1 (Orbit-stabilizer theorem). *Fix a G-type X and a point $x : X(\mathrm{pt})$. There is a canonical action $\tilde{G} : BG_x \to \mathcal{U}$, acting on $\tilde{G}(\mathrm{pt}) \simeq G$ with orbit type $\tilde{G} /\!\!/ G_x \simeq \mathcal{O}_x$.*

Proof. Define $\tilde{G}(x, y, !) :\equiv (\mathrm{pt} = x)$. □

Now suppose that G is a 1-group acting on a set. We see that the orbit type is a set (and is thus equivalent to the set of orbits) if and only if all stabilizer groups are trivial, i.e., if and only if the action is free.

If G is a 1-group, then so is each stabilizer-group, and in this case (of a set-action), the orbit-stabilizer theorem tells us that

thm:lagrange

Theorem 6.5.2 (Lagrange's Theorem). [4] *If $H \to G$ is a subgroup, then H has a natural action on G, and all the orbits under this action are equivalent.*

6.6 *The isomorphism theorems*

sec:noether-theorems

Cf. Section 2.26

Group homomorphisms provide examples of forgetting stuff and structure. For example, the map from cyclically ordered sets with

[4] insert that the action is free (referred to)

cardinality n to the type of sets with cardinality n forgets structure, and represents an injective group homomorphism from the cyclic group of order n to the symmetric group Σ_n.

And the map from pairs of n-element sets to n-element sets that projects onto the first factor clearly forgets stuff, namely, the other component. It represents a surjective group homomorphism.

More formally, fix two groups G and H, and consider a homomorphism φ from G to H, considered as a pointed map $\mathrm{B}\varphi : BG \to_{\mathrm{pt}} BH$. Then $\mathrm{B}\varphi$ factors as

$$\begin{aligned} BG &= \sum_{w:BH} \sum_{z:BG} (\mathrm{B}\varphi(z) = w) \\ &\to_{\mathrm{pt}} \sum_{w:BH} \left\| \sum_{z:BG} (\mathrm{B}\varphi(z) = w) \right\|_0 \\ &\to_{\mathrm{pt}} \sum_{w:BH} \left\| \sum_{z:BG} (\mathrm{B}\varphi(z) = w) \right\|_{-1} = BH. \end{aligned}$$

The pointed, connected type in the middle represents a group that is called the *image* of φ, $\mathrm{Im}(\varphi)$.

(FIXME: Quotient groups as automorphism groups, normal subgroups/normalizer, subgroup lattice)

lem:aut-orbit

Lemma 6.6.1. *The automorphism group of the G-set G/H is isomorphic to* $\mathrm{N}_G(H)/H$.

6.7 (*the lemma that is not*) *Burnside's lemma*

sec:burnsides-lemma
lem:burnside

Lemma 6.7.1. *Let G be a finite group acting on a finite set X. Then the set of orbits is finite with cardinality*

$$\mathrm{Card}(X/G) = \frac{1}{\mathrm{Card}(G)} \sum_{g\,:\,\mathrm{El}\,G} \mathrm{Card}(X^g),$$

where $X^g = \{\, x : X \mid gx = x \,\}$ is the set of elements that are fixed by g.

Proof. Since X and G is finite, we can decide equality of their elements. Hence each X^g is a finite set, and since G is finite, we can decide whether x, y are in the same orbit by searching for a $g : \mathrm{El}\,G$ with $gx = y$. Hence the set of orbits is a finite set as well.

Consider now the set $R :\equiv \sum_{g\,:\,\mathrm{El}\,G} X^g$, and the function $q : R \to X$ defined by $q(g, x) :\equiv x$. The map $q^{-1}(x) \to G_x$ that sends (g, x) to g is a bijection. Thus, we get the equivalences

$$R \equiv \sum_{g\,:\,\mathrm{El}\,G} X^g \simeq \sum_{x:X} G_x \simeq \sum_{z:X/G} \sum_{x:\mathcal{O}_z} G_x,$$

where the last step writes X as a union of orbits. Within each orbit $\mathcal{O}_z$, the stabilizer groups are conjugate, and thus have the same finite cardinality, which from the orbit-stabilizer theorem (**??**), is the cardinality of G divided by the cardinality of $\mathcal{O}_z$. We conclude that $\mathrm{Card}(R) = \mathrm{Card}(X/G)\,\mathrm{Card}(G)$, as desired. □

6.8 *More about automorphisms*

sec:automorphisms

For every group G (which for the purposes of the discussion in this section we allow to be a higher group) we have the automorphism group $\mathrm{Aut}(G)$. This is of course the group of self-identifications $G = G$ in the type of groups, Group. If we represent G by the pointed connected classifying type BG, then $\mathrm{Aut}(G)$ is the type of pointed self-equivalences of BG.

We have a natural forgetful map from groups to the type of connected groupoids. Define the type Bunch to be the type of all connected groupoid. If $X :$ Bunch, then all the elements of X are merely isomorphic, that is, they all look alike, so it makes sense to say that X consists of a *bunch* of alike objects.

For every group G we have a corresponding bunch, $BG_{\div}$, i.e., the collection of G-torsors, and if we remember the basepoint $\mathrm{pt} : BG_{\div}$, then we recover the group G. Thus, the type of groups equivalent to the type $\sum_{X : \mathrm{Bunch}} X$ of pairs of a bunch together with a chosen element. (This is essentially our definition of the type Group.)

Sometimes we want to emphasize that we $BG_{\div}$ is a bunch, so we define $\mathrm{bunch}(G) :\equiv BG_{\div} : \mathrm{Bunch}$.

def:center

Definition 6.8.1 (The center as an abelian group). Let $Z(G) :\equiv \prod_{z : BG}(z = z)$ denote the type of fixed points of the adjoint action of G on itself. This type is equivalent to the automorphism group of the identity on $\mathrm{bunch}(G)$, and hence the loop type of

$$\mathrm{B}Z(G) :\equiv \sum_{f : BG \to BG} \|f \sim \mathrm{id}\|_{-1}.$$

This type is itself the loop type of the pointed, connected type

$$\mathrm{B}^2 Z(G) :\equiv \sum_{X : \mathrm{Bunch}} \|\mathrm{bunch}(G) = X\|_0,$$

and we use this to give $Z(G)$ the structure of an *abelian* group, called the *center* of G. ⌟

There is a canonical homomorphism from $Z(G)$ to G given by the pointed map from $\mathrm{B}Z(G)$ to BG that evaluates at the point pt. The fiber of the evaluation map $e : \mathrm{B}Z(G) \to_{\mathrm{pt}} BG$ is

$$\begin{aligned} \mathrm{fiber}_e(\mathrm{pt}) &\equiv \sum_{f : BG \to BG} \|f \sim \mathrm{id}\|_{-1} \times (f\,\mathrm{pt} = \mathrm{pt}) \\ &\simeq \sum_{f : BG \to_{\mathrm{pt}} BG} \|f \sim \mathrm{id}\|_{-1}, \end{aligned}$$

and this type is the loop type of the pointed, connected type

$$\mathrm{B\,Inn}(G) :\equiv \sum_{H : \mathrm{Group}} \|\mathrm{bunch}(G) = \mathrm{bunch}(H)\|_0,$$

thus giving the homomorphism $Z(G)$ to G a normal structure with quotient group $\mathrm{Inn}(G)$, called the *inner automorphism group*.

Note that there is a canonical homomorphism from $\mathrm{Inn}(G)$ to $\mathrm{Aut}(G)$ given by the pointed map $i : \mathrm{B\,Inn}(G) \to \mathrm{B\,Aut}(G)$ that forgets the component. On loops, i gives the inclusion into $\mathrm{Aut}(G)$ of the subtype of automorphisms of G that become merely equal to the identity automorphism of $\operatorname{bunch}(G)$. The fiber of i is

$$\begin{aligned}\mathrm{fiber}_i(\mathrm{pt}) &\equiv \sum_{H\,:\,\mathrm{Group}} \|\operatorname{bunch}(G) = \operatorname{bunch}(H)\|_0 \times (H = G)\\ &\simeq \|\operatorname{bunch}(G) = \operatorname{bunch}(G)\|_0.\end{aligned}$$

This is evidently the type of loops in the pointed, connected groupoid

$$\mathrm{B}\,\mathrm{Out}(G) :\equiv \left\| \sum_{X\,:\,\mathrm{Bunch}} \|\operatorname{bunch}(G) = X\|_{-1} \right\|_1,$$

thus giving the homomorphism $\mathrm{Inn}(G)$ to $\mathrm{Aut}(G)$ a normal structure with quotient group $\mathrm{Out}(G)$, called the *outer automorphism group*. Note that $\mathrm{Out}(G)$ is always a 1-group, and that it is the decategorification of $\mathrm{Aut}(\operatorname{bunch}(G))$.

THEOREM 6.8.2. *Let two groups G and H be given. There is a canonical action of* $\mathrm{Inn}(H)$ *on the set of homomorphisms from G to H, $\|BG \to_{\mathrm{pt}} BH\|_0$. This gives rise to an equivalence*

$$\|BG_{\div} \to BH_{\div}\|_0 \simeq \big\| \|BG \to_{\mathrm{pt}} BH\|_0 /\!\!/ \mathrm{Inn}(H) \big\|_0$$

between the set of maps from $\operatorname{bunch}(G)$ *to* $\operatorname{bunch}(H)$ *and the set of components of the orbit type of this action.*

Proof. We give the action by defining a type family $X : \mathrm{B}\,\mathrm{Inn}(H) \to \mathcal{U}$ as follows

$$X\,\langle K, \phi\rangle :\equiv \|\mathrm{Hom}(G, K)\|_0 \equiv \|BG \to_{\mathrm{pt}} BK\|_0,$$

for $\langle K, \phi\rangle : \mathrm{B}\,\mathrm{Inn}(H) \equiv \sum_{K\,:\,\mathrm{Group}} \|\operatorname{bunch}(H) = \operatorname{bunch}(K)\|_0$. Now we can calculate

$$\begin{aligned}\|X_{\mathrm{Inn}(H)}\|_0 &\equiv \left\| \sum_{K\,:\,\mathrm{Group}} \|\operatorname{bunch}(H) = \operatorname{bunch}(K)\|_0 \times \|\mathrm{Hom}(G, K)\| \right\|_0\\ &\simeq \left\| \sum_{K\,:\,\mathrm{Group}} (\operatorname{bunch}(H) = \operatorname{bunch}(K)) \times \mathrm{Hom}(G, K) \right\|_0\\ &\simeq \left\| \sum_{K\,:\,\mathrm{Bunch}} \sum_{k\,:\,K} (\operatorname{bunch}(H) = K) \times \sum_{f\,:\,\operatorname{bunch}(G)\to K} f\,\mathrm{pt} = k \right\|_0\\ &\simeq \left\| \sum_{K\,:\,\mathrm{Bunch}} (\operatorname{bunch}(H) = K) \times (\operatorname{bunch}(G) \to K) \right\|_0\\ &\simeq \|\operatorname{bunch}(G) \to \operatorname{bunch}(H)\|_0 \equiv \|BG_{\div} \to BH_{\div}\|_0.\end{aligned}$$

□

6.9 *Orbit type as a groupoid completion(*)*

This is a somewhat advanced topic that should occur much later, if at all.

Suppose G is a group acting on a groupoid X, given by a map $X : BG \to_{\mathrm{pt}} \mathrm{BAut}(X_0)$, with $e_X : X(\mathrm{pt}) = X_0$. By induction on e_X we may assume that $X_0 \equiv X(\mathrm{pt})$ and $e_X \equiv \mathrm{refl}$.

We have the orbit type $X /\!\!/ G \equiv \sum_{T\,:\,BG} X(T)$. We think of this as identifying elements of X_0 that are in the same orbit, in the sense that there are new identifications of x and y for group elements g with $g \cdot x = y$.

In this section we study one way of making this intuition precise.

Consider the pregroupoid $C_G(X)$ with object type X_0 and morphism sets

$$\hom(x, y) :\equiv \sum_{g : G} (g \cdot x = y),$$

where $g \cdot x :\equiv g_*(x)$. The identity at x is $(1, \mathrm{id})$, while the composite of $(g, p) \colon \hom(x, y)$ with $(h, q) \colon \hom(y, z)$ is (hg, r), where r is built from p and q as follows:

$$hg \cdot x = h \cdot (g \cdot x) = h \cdot y = z.$$

We have a functor of pregroupoids $F \colon C_G(X) \to X /\!\!/ G$ defined on objects by $F(x) := (\mathrm{pt}, x)$ and on morphisms $(g, p) \colon \hom(x, y)$ by $F(g, p) := (g, p)^=$.

This functor is essentially surjective on objects (by connectivity of BG) and fully faithful by the characterization of paths in $\sum$-types. Hence it induces an equivalence from the completion of $C_G(X)$ to $X /\!\!/ G$.

As a corollary, the orbit set $X/G \equiv \|X /\!\!/ G\|_0$, is the set quotient of X_0 modulo the equivalence relation $x \sim y :\equiv \exists g : G, g \cdot x = y$.

7
Finitely generated groups

ch:fggroups

TODO:

- Make a separate chapter on combinatorics? Actions and Burnside and counting colorings?
- Cayley actions: G acts on $\Gamma(G,S)$: Action on vertices is the left action of G on itself: $t \mapsto (t =_{BG} \mathrm{pt})$, on vertices, for $s : S$, have edge $t = \mathrm{pt}$ to $t = \mathrm{pt}$
- Recall universal property of free groups: If we have a map $\varphi : S \to H$, then we get a homomorphism $\bar{\varphi} : F(S) \to H$, represented by $BF(S) \to_{\mathrm{pt}} BH$ defined by induction, sending pt to pt and s to $\varphi(s)$.
- define different types of graphs (S-digraphs, $\tilde{S}$-graphs, (partial) functional graphs, graph homomorphisms, quotients of graphs)
- define (left/right) Cayley graphs of f.g. groups – $\mathrm{Aut}(\Gamma_G) = G$ (include $\alpha : F(S) \to G$ in notation?) – Cayley graphs are vertex transitive
- Cayley graphs and products, semi-direct products, homomorphisms
- Some isomorphisms involving semi-direct products – Exceptional automorphism of Σ_6: – Exotic map $\Sigma_5 \to \Sigma_6$. (Conjugation action of Σ_5 on 6 5-Sylow subgroups.) A set bundle $X : \mathrm{B}\Sigma_6 \to \mathrm{B}\Sigma_6$.
- https://math.ucr.edu/home/baez/six.html Relating Σ_6 to the icosahedron. The icosahedron has 6 axes. Two axes determines a golden rectangle (also known as a *duad*,[1] so there are 15 such. A symmetry of the icosahedron can be described by knowing there a fixed rectangle goes, and a symmetry of the rectangle. Picking three rectangles not sharing a diagonal gives a *syntheme*: three golden rectangles whose vertices make up the icosahedron. Some synthemes (known as *true crosses* have the rectangles orthogonal to each other, as in Fig. 7.1. Fact: The symmetries of the icosahedron form the alternating symmetries of the 5 true crosses. Of course, we get an action on the 6 axes, thus a homomorphism $\mathrm{A}_5 \to \Sigma_6$. Every golden rectangle lies in one true cross and two skew crosses. The combinatorics of duads, synthemes, and synthematic totals are illustrated in the Cremona-Richardson configuration and the resulting Tutte-Coxeter graph. The automorphism group of the latter is in fact $\mathrm{Aut}(\Sigma_6)$. If we color the vertices according to duad/syntheme, we get Σ_6 itself.
- words and reduced words
- define (left/right) presentation complex of group presentation

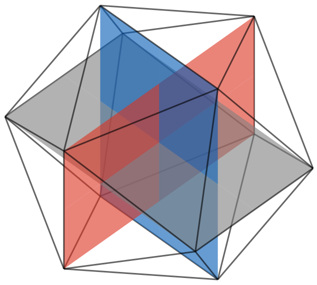


FIGURE 7.1: Icosahedron with an inscribed true cross

fig:true-cross

[1] These names come from Sylvester.

- define Stallings folding
- deduce Nielsen–Schreier and Nielsen basis
- deduce algorithms for generalized word problem, conjugation, etc.
- deduce Howson's theorem
- think about 2-cell replacement for folding; better proofs in HoTT?
- move decidability results to main flow
- include undecidability of word problem in general – doesn't depend on presentation (for classes closed under inverse images of monoid homomorphisms)
- describe $F(S)/H$ in the case where H has infinite index
- describe normal closure of R in $F(S)$ – still f.g.? – get Cayley graph of $F(S)/\langle R\rangle$. – Todd-Coxeter algorithm?
- in good cases we can recognize $\mathcal{S}(R)$ as a "fundamental domain" in Cayley graph of $\langle S \mid R\rangle$.

Remark 7.0.1. In this chapter, we use letters from the beginning of the alphabet $a, b, c, \ldots$ to denote generators, and we use the corresponding capital letters A, B, C to denote their inverses, so, e.g., $aA = Aa = 1$. This cleans up the notational clutter significantly. ⌟

Do we fix S, a finite set $S = \{a, b, \ldots\}$? Mostly F will denote the free group on S. And for almost all examples, we take $S = \{a, b\}$.

7.1 *Cayley diagrams*

sec:cayley-diagrams

We have seen in the previous chapter how cyclic groups (those generated by a single generator) have neatly described types of torsors. Indeed, $\mathrm{BC}\, n \simeq \mathrm{Cyc}_n$, where Cyc_n is the type of n-cycles, And the BZ, and equivalently, the circle, is equivalent to the type of infinite cycles. In Chapter 3, we defined the types of (finite or infinite) cycles as certain components of $\sum_{X:\mathcal{U}}(X = X)$, but we can equivalently consider components of $\sum_{X:\mathcal{U}}(X \to X)$, since the former is a subtype of the latter. By thinking of functions in terms of their graphs, we might as well look at components of $\sum_{X:\mathcal{U}}(X \to X \to \mathcal{U})$.

In this section we shall generalize this story to groups G generated by a (finite or just decidable) set of generators S.

$G \simeq \mathrm{Aut}(D_G) \to \mathrm{Sym}(\mathrm{Card}\, G)$

7.2 *Free groups*

sec:freegroups

7.3 *Examples*

sec:fg-examples

7.4 *Subgroups of free groups*

sec:subgroups-free

The Nielsen-Schreier theorem.

Corollary 7.4.1. *A subgroup of finite index of a finitely generated group is finitely generated.*

(This also has an automata theoretic proof, see below.)

fig:cayley-s3

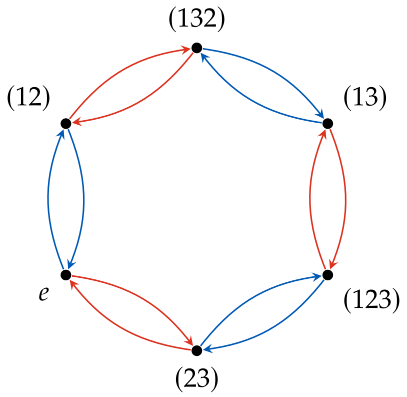


Figure 7.2: Cayley diagram for S_3 with respect to $S = \{(12), (23)\}$.

fig:cayley-a5

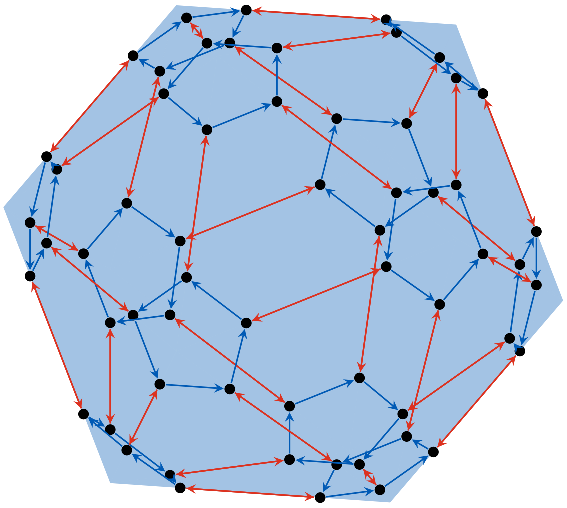


Figure 7.3: Cayley diagram for A_5 with respect to $S = \{a, b\}$, where a is a $1/5$-rotation about a vertex and b is a $1/2$-rotation about an edge in an icosahedron.

7.5 *Intersecting subgroups*

sec:intersecting-subgroups

Stallings folding[2].

Theorem 7.5.1. *Let H be a finitely generated subgroup of $F(S)$ and let $u \in \tilde{S}^*$ by a reduced word. Then u represents an element of H if and only if u is recognized by the Stallings automaton $\mathcal{S}(H)$.*

Theorem 7.5.2. *Let H be a finitely generated subgroup of $F(S)$. Then H has finite index if and only if $\mathcal{S}(H)$ is total.*

Furthermore, in this case the index equals the number of vertices of $\mathcal{S}(H)$.

Corollary 7.5.3. *If H has index n in $F(S)$, then* $\operatorname{rk} H = 1 + n(\operatorname{card} S - 1)$.

thm:howson-neumann

Theorem 7.5.4. *Suppose H_1, H_2 are two subgroups of F with finite indices h_1, h_2. Then the intersection $H_1 \cap H_2$ has finite index at most $h_1 h_2$.*

The qualitative part of Theorem 7.5.4 is known as *Howson's theorem*, while the inequality is known as *Hanna Neumann's inequality*. Hanna's son, Walter Neumann, conjectured that the 2 could be removed, and this was later proved independently by Joel Friedman and Igor Mineyev.

7.6 *Connections with automata* (*)

(S is still a fixed finite set.)

Let $\iota : F(S) \longrightarrow \tilde{S}^*$ map an element of the free group to the corresponding reduced word. The kernel of ι is the 2*-sided Dyck language* D_S.[3]

The following theorem is due to Benois.

Theorem 7.6.1. *A subset X of $F(S)$ is rational if and only if $\iota(X) \subseteq \tilde{S}^*$ is a regular language.*

Lemma 7.6.2. *Let $\rho : \tilde{S}^* \longrightarrow \tilde{S}^*$ map a word to its reduction. Then ρ maps regular languages to regular languages.*

[2] John R. Stallings. "Foldings of G-trees". In: *Arboreal group theory (Berkeley, CA, 1988)*. Vol. 19. Math. Sci. Res. Inst. Publ. Springer, New York, 1991, pp. 355–368. doi: 10.1007/978-1-4612-3142-4_14.

[3] Perhaps the 1*-sided Dyck language* is more familiar in language theory: Here, S is considered as a set of "opening parentheses", while the complementary elements of S^* are "closing parentheses". It's customary to pick $S^* = \{(,)\}$. The 1-sided Dyck language for this alphabet consists of all *balanced* words of opening and closing parentheses, like (),(()),()(), etc.

The following is due to Sénizergues:

THEOREM 7.6.3. *A rational subset of $F(S)$ is either disjunctive or recognizable.*

Given a surjective monoid homomorphism $\alpha : S^* \to G$, we define the corresponding *matched homomorphism* $\tilde{\alpha} : \tilde{S}^* \to G$ by $(\tilde{\alpha}(a^{-1}) :\equiv \alpha(a)^{-1}$.

THEOREM 7.6.4 (?). *Consider a f.g. group G with a surjective homomorphism $\alpha : F(S) \to G$. A subset X of G is recognisable by a finite G-action if and only if $\tilde{\alpha}^{-1}(X) \subseteq \tilde{S}^*$ is rational (i.e., regular).*

THEOREM 7.6.5 (Chomsky–Schützenberger). *A language $L \subseteq T^*$ is context-free if and only if $L = h(R \cap D_S)$ for some finite S, where $h : T^* \to \tilde{S}^*$ is a hommormorphism, $R \subseteq \tilde{S}^*$ is a regular language, and D_S is the Dyck language for S.*[4]

THEOREM 7.6.6 (Muller–Schupp, ?). *Suppose $\tilde{\alpha} : \tilde{S}^* \to G$ is a surjective matched homomorphism onto a group G. Then G is virtually free (i.e., G has a normal free subgroup of finite index) if and only if* $\ker(\tilde{\alpha})$ *is a context-free language.*

THEOREM 7.6.7.

The Stallings automaton is an *inverse automaton*: it's deterministic, and there's an edge (p, a, q) if and only if there's one (q, A, p). We can always think of the latter as the *reverse* edge. (It's then also deterministic in the reverse direction.)

Two vertices p, q get identified in the Stallings graph/automaton if and only if there is a run from p to q with a word w whose reduction is 1. (So a word like $aAAaBBbb$.)

THEOREM 7.6.8. *Let $X \subseteq F(S)$. Then Y is a coset Hw with H a finitely generated subgroup, if and only if there is a finite state inverse automaton whose language (after reduction) is Y.*

COROLLARY 7.6.9. *The generalized word problem in $F(S)$ is solvable: Given a finitely generated subgroup H, and a word $u : \tilde{S}^*$, we can decide whether u represents an element of H.*

As above, we get a basis for H as a free group from a spanning tree in $\mathcal{S}(H)$.

The Stallings automaton for H can be constructed in time $O(n \log^* n)$, where n is the sum of the lengths of the generators for H. [Cite: Touikan: A fast algorithm for Stallings' folding process.] Once this has been constructed, we can solve membership in H in linear time.

THEOREM 7.6.10. *We can decide whether two f.g. subgroups of $F(S)$ are conjugate. Moreover, a f.g. subgroup H is normal if and only if $\mathcal{S}(H)$ is vertex-transitive.*

Proof. G, H are conjugate of and only if their cores are equal. □

There are other connections between group theory and language theory:

THEOREM 7.6.11 (Anisimov and Seifert). *A subgroup H of G is rational if and only if H is finitely generated.*

THEOREM 7.6.12. *A subgroup H of G is recognizable if and only if it has finite index.*

[4] References TODO. The theorem is also true if we replace D_S by its one-sided variant, but in this case it reduces to the well-known equivalence between context-free languages and languages recognizable by push-down automata.

8

Finite group theory

ch:fingp
sec:fingp

8.0.1 *Finite groups*

Objects having only a finite number of symmetries can be analyzed through counting arguments. The strength of this approach is stunning. The orbit-stabilizer theorem Section 6.5 is at the basis of this analysis: if G is a group and $X : BG \to \mathrm{Set}$ is a G-set, then

$$X(\mathrm{sh}_G) \simeq \amalg_{x : X/G} \mathcal{O}_x$$

and each orbit set $\mathcal{O}_x$ is equivalent to the cokernel of the inclusion $G_x \subseteq G$ of the stabilizer subgroup of x. Consequently, if $X(\mathrm{sh}_G)$ is a finite set, then its cardinality is the sum of the cardinality of these cokernels. If also the set UG is finite much more can be said and simple arithmetical considerations often allow us to deduce deep statements like the size of a certain subset of $X(\mathrm{sh}_G)$ and in particular whether or not there are any fixed points.

Example 8.0.2. A typical application could go like this. If $X(\mathrm{sh}_G)$ is a finite set with 13 elements and for some reason we know that all the orbits have cardinalities dividing 8 – which we'll see happens if UG has 8 elements – then we must have that some orbits are singletons (for a sum of positive integers dividing 8 to add up to 13, some of them must be 1). That is, X has fixed points. ⌟

The classical theory of finite groups is all about symmetries coupled with simple counting arguments. Lagrange's Theorem 6.5.2 gives the first example: if H is a subgroup of G, then the cardinality "$|G|$" of UG is divisible by $|H|$, putting severe restrictions on the possible subgroups. For instance, if $|G|$ is a prime number, then G has no notrivial proper subgroups! (actually, G is necessarily a cyclic group). To prove this result we interpret G as an H-set.

Further examples come from considering the G-set Sub_G of subgroups of G from Section 5.1. Knowledge about the G-set of subgroups is of vital importance for many applications and Sylow's theorems in Section 8.3 give the first restriction on what subgroups are possible and how they can interact. The first step is Cauchy's Theorem 8.2.2 which says that if $|G|$ is divisible by a prime p, then G contains a cyclic subgroup of order p. Sylow's theorems goes further, analyzing subgroups that have cardinality powers of p, culminating in very detailed and useful information about the structure of the subgroups with cardinality the maximal possible power of p.

Example 8.0.3. For instance, for the permutation group Σ_3, Sylow's theorems will deduce from the simple fact $|\Sigma_3| = 6$ that Σ_3 contains a unique subgroup $|H|$ with $|H| = 3$. Since it is unique, H must be a normal subgroup.

On the other hand, for Σ_4 the information $|\Sigma_4| = 24$ only suffices to tell us that there are either 1 or 4 subgroups K with $|K| = 3$, but that all of them are conjugate. However, the inclusion of Σ_3 in Σ_4 shows that the $H \subseteq \Sigma_3$ above (which is given by the cyclic permutations of three letters) can be viewed as a subgroup of Σ_4, and elementary inspection gives that this subgroup is not normal. Hence there must be more than one subgroup K with $|K| = 3$, pinning the number of such subgroups down to 4.

Indeed, Σ_n has $n(n-1)(n-2)/6$ subgroups of order 3 (for $n > 2$), but when $n > 5$ something like a phase transformation happens: the subgroups of order 3 are no longer all conjugate. This can either be seen as a manifestation of the fact that $3^2 = 9$ divides $n! = |\Sigma_n|$ for $n > 5$ or more concretely by observing that there is room for "disjoint" cyclic permutations. For instance the subgroup of cyclic permutations of $\{1, 2, 3\}$ will not be conjugate to the subgroup of cyclic permutations of $\{4, 5, 6\}$. Together these two cyclic subgroups give a subgroup K with $|K| = 9$ and there are 10 of these (one for each subset of $\{1, 2, 3, 4, 5, 6\}$ of cardinality 3). ⌟

rem:noofsubgps

Remark 8.0.4. One should observe that the number of subgroups is often very large and the structure is often quite involved, even for groups with a fairly manageable size and transparent structure (for instance, the number of subgroups of the group you get by taking the product of the cyclic group C_2 with itself n times grows approximately as $7 \cdot 2^{n^2/4}$ – e.g., $C_2^{\times 18}$ has 17741753171749626840952685 subgroups, see https://oeis.org/A006116). ⌟

8.1 *Lagrange's theorem, counting version*

sec:Lagrangecounting

We start our investigation by giving the version of Lagrange's theorem which has to do with counting, but first we pin down some language.

def:finitegrd

Definition 8.1.1. A *finite group* is a group such that the set UG is finite. If G is a finite group, then the *cardinality* $|G|$ is the cardinality of the finite set UG (i.e., $UG : \mathrm{fin}_{|G|}$). ⌟

Example 8.1.2. The trivial group has cardinality 1, the cyclic group C_n of order n has cardinality n and the permutation group Σ_n has cardinality $n!$. ⌟

In the literature, "order" and "cardinality" are used interchangeabley for groups.

For finite groups, Lagrange's Theorem 6.5.2 takes on the form of a counting argument

lem:Lagrangeascounting

Lemma 8.1.3 (Lagrange's theorem: counting version). *Let $i : \mathrm{Hom}(H, G)$ be a subgroup of a finite group G. Then*

$$|G| = |G/H| \cdot |H|.$$

If $|H| = |G|$, then $H = G$ (as subgroups of G).

Proof. Consider the H action of H on G, i.e., the H-set $i^*G : BH \to \mathrm{Set}$ with $i^*G(x) :\equiv (\mathrm{sh}_G = Bi(x))$, so that G/H is just another name for the orbits $i^*G/H :\equiv \sum_{x:BH} i^*G(x)$. Note that composing with the structure identity $p_i : \mathrm{sh}_G = Bi(\mathrm{sh}_H)$ gives an equivalence $i^*G(\mathrm{sh}_H) \simeq UG$, so that $|i^*G(\mathrm{sh}_H)| = |G|$.

Lagrange's Theorem 6.5.2 says that i^*G is a free H-set [1] and so all orbits $\mathcal{O}_x$ are equivalent to the H-set $\tilde{H}(x) = (\mathrm{sh}_H = x)$. Consequently, the equivalence

$$i^*G(\mathrm{sh}_H) \simeq \sum_{x : i^*G/H} \mathcal{O}_x$$

of Section 6.5 gives that G/H and H are finite and that $|G| = |G/H| \cdot |H|$.[2]

Finally, since we are considering a subgroup, the preimage $Bi^{-1}(\mathrm{pt})$ is equivalent to the set G/H. If $|H| = |G|$, then $|G/H| = 1$ and so the set G/H is contractible. □

cor:cyclicgroupsaresimple

Corollary 8.1.4. *If p is a prime, then the cyclic group C_p has no non-trivial proper subgroups.*

Proof. By Lagrange's counting Lemma 8.1.3 a subgroup of C_p has cardinality dividing $p = |C_p|$, i.e., either 1 or p. □

cor:whatSylow2needs

Corollary 8.1.5. *Let $f : \mathrm{Hom}(G, G')$ be a surjective homomorphism with kernel N and let H be a subgroup of G. If H and G' are finite with coprime cardinalities, then H is a subgroup of N.*

Proof. Let $i : \mathrm{Hom}(H, G)$ be the inclusion. By Lemma 5.5.11 the intersection $N \cap H$ is the kernel of the composite $fi : \mathrm{Hom}(H, G')$. Let H' be the image of fi. Now, Lagrange's counting Lemma 8.1.3 gives that $|H| = |H'| \cdot |N \cap H|$ and $|G'| = |G'/H'| \cdot |H'|$. This means that $|H'|$ divides both $|H|$ and $|G'|$, but since these numbers are coprime we must have that $|H'| = 1$, and finally that $|H| = |N \cap H|$. This imples that $N \cap H = H$, or in other words, that H is a subgroup of N ((elaborate)). □

Corollary 8.1.6. *If G and G' are finite groups, then the cardinality $|G \times G'|$ of the product is the product $|G| \cdot |G'|$ of the cardinalities.*

Remark 8.1.7. Hence the cardinality of the n-fold product of Remark 8.0.4 of C_2 with itself is (2^n and so grows quickly, but is still) dwarfed by the number of subgroups as n grows. ⌟

8.2 *Cauchy's theorem*

lem:fixedptsize

Lemma 8.2.1. *Let p be a prime and G a group of cardinality p^n for some positive $n : \mathbb{N}$. If $X : BG \to \mathrm{Set}$ is a non-empty finite G-set such that the cardinality of $X(\mathrm{sh}_G)$ is divisible by p, then the cardinality of the set of fixed points $X^G :\equiv \prod_{z:BG} X(z)$ is divisible by p.*

Proof. Recall that the evaluation at sh_G gives an injection of sets $X^G \to X(\mathrm{sh}_G)$ through which we identify X^G with the subset "$X(\mathrm{sh}_G)^G$" of all trivial orbits of $X(\mathrm{sh}_G)$. The orbits of $X(\mathrm{sh}_G)$[3] all have cardinalities that divide the cardinality p^n of G. This means that all the the cardinalities of the non-trivial orbits (as well as of $X(\mathrm{sh}_G)$) are positive integers divisible by p.

Burnside's Lemma Section 6.7 states that $X(\mathrm{sh}_G)$ is the sum of its orbits. Hence the cardinality of the set of all trivial orbits, i.e., of X^G, is the difference of two numbers both divisible by p. □

[1] Theorem 6.5.2 doesn't say this at present: fix it

[2] somewhere: prove that if A is a finite set and $B(a)$ is a family of finite sets indexed over $a : A$, then $\sum_{a:A} B(a)$ is a finite set of cardinality $\sum_{i:\mathfrak{m}} |B(f(i))|$ for any $f : \mathfrak{m} = A$, hence if $m = |B(a)|$ for all a then $|\sum_A B(a)| = n \cdot m$.

[3] or of X? Reference for identification of orbits with quotiens by stabilizers

thm:cauchys

Theorem 8.2.2. *Let p be a prime and let G be a finite group of cardinality divisible by p. Then G has a subgroup which is cyclic of cardinality p.*

Proof. Recall the cyclic group C_p of cardinality p given by the pointed connected groupoid

$$BC_p :\equiv (\sum_{S:\mathrm{Set}} \sum_{j:S=S} \|(S,j) = \mathrm{Z}/p\|, (\mathrm{Z}/p, !)),$$

where $\mathrm{Z}/p : \sum_{S:\mathrm{Set}} S = S$ was a particular model of a set with p element together with a successor modulo p. Informally, BC_p consists of pairs (S,j), where S is a set of cardinality p and $j : S = S$ is a cyclic permutation in the sense that for $0 < k < p$ we have that j^k is not refl while $j^p = \mathrm{refl}$. Note also that $j^? : \mathbb{p} \to ((S,j) = (S,j))$ given by $j^?(k) = j^k$ is an equivalence (just as for the integers, a symmetry of (S,j), i.e., an $f : S = S$ so that $fj = jf$, must be j^k for some $k : \mathbb{p}$, and if $k \neq l$, then $j^k \neq j^l$)

If $(S,j) : BC_p$ let

$$A(S,j) :\equiv ((S,j) = (S,j) \to \mathrm{U}G).$$

Since we have an equivalence $j^? : \mathbb{p} \to ((S,j) = (S,j))$ we get that $J : A(S,j) \to \prod_{\mathbb{p}} \mathrm{U}G$ given by $J(g) = (g_{j^0}, g_{j^1}, \dots, g_{j^{p-1}})$ is an equivalence. Define $\mu : \prod_{(S,j):BC_p}(A(S,j) \to \mathrm{U}G)$ by $\mu_{(S,j)}(g) :\equiv g_{j^0} \cdot \dots \cdot g_{j^{p-1}}$ and let $X : BC_p \to \mathrm{Set}$ be the G-set defined by

$$X(S,j) :\equiv \sum_{g:A(S,j)} \mu_{(S,j)}g = e_G.$$

The map from $X(S,j)$ to the $p-1$-fold product of $\mathrm{U}G$ with itself sending $(g,!)$ to $(g_{j^1}, \dots, g_{j^{p-1}})$ is an equivalence ($\mu_{(S,j)}g = e_G$ says exactly that g_{j^0} can be reconstructed as $(g_{j^1} \cdot \dots \cdot g_{j^{p-1}})^{-1}$), so $X(S,j)$ is a set of cardinality $p-1$ times the cardinality of G. In particular, p divides the cardinality of $X(S,j)$.

Specializing to (S,j) being Z/p and allowing to index the elements in $A(\mathrm{Z}/p)$ with $i : \mathbb{p}$ (instead of the very awkward "$(\sqrt[p]{\mathrm{id}})^i$" as purism would dictate) we proceed as follows.

Now, a C_p-fixed point of $X(\mathrm{Z}/p)$ is an element $(g_0, \dots, g_{p-1}, !)$ such that $(g_0, \dots, g_{p-1}, !) = (g_1, \dots, g_{p-1}, g_0, !)$, i.e., $g_0 = g_1 = g_2 = \dots = g_{p-1}$[4]. In other words, a fixed point is of the form $(g, \dots, g, !)$, where ! expresses that $g^p = e_G$: $X(\mathrm{Z}/p)^{C_p}$ is equivalent to $\sum_{g:\mathrm{U}G} g^p = e_G$. If we can show that $(g,!) : X(\mathrm{Z}/p)^{C_p}$ is nonempty, we'd have established an abstract cyclic subgroup consisting of the powers of g. Of course, setting $g = e_G$ will give us such a fixed point, but if $g \neq e_G$ we get a cyclic subgroup of cardinality p of G.

Now, Lemma 8.2.1 claims that p divides the cardinality of $X(\mathrm{Z}/p)^{C_p}$, and since there *are* fixed points, there must be at lest p fixed points. One of them is the trivial one (given by $g = e_G$ above), but the others are nontrivial.

[5] □

lem:nontrivcenter

Lemma 8.2.3. *Let be G be a finite subgroup of cardinality p^n, where p is prime and n a positive integer. Then the center $Z(G)$ of G is nontrivial.* (*point to center in the symmetry chapter*)

[4] if I am allowed to write that

[5] Two slight variations commented away. Have to choose one. The first needs some background essentially boiling down to BC_n being the truncation of the nth Moore space.

Proof. Recall the G-set $\mathrm{Ad}_G : BG \to \mathrm{Set}$ given by $\mathrm{Ad}_G(z) = (z = z)$. Then the map

$$\mathrm{ev}_{\mathrm{sh}_G} : \prod_{z\,:\,BG} (z = z) \to \mathrm{U}G, \quad \mathrm{ev}_G(f) = f(\mathrm{sh}_G)$$

has the structure of a (n abstract) inclusion of a subgroup; namely the inclusion of the center $Z(G)$ in G. The center thus represents the fixed points of the G-set Ad_G. Since G has cardinality a power of p, all orbits but the fixed points have cardinality divisible by p. Consequently, Burnside's lemma states that the number of fixed points, i.e., the cardinality of $Z(G)$, must be divisible by p. □

cor:orderpsquaredgroups

COROLLARY 8.2.4. *If G is a noncyclic group of cardinality p^2, then G of the form $C_p \times C_p$.*

Proof. The center $Z(G)$ is by Lemma 8.2.3 of cardinality p or p^2. Since G is not cyclic we have that $g^p = e_G$ for all $g : \mathrm{U}G$. [6] □

sec:sylow

8.3 *Sylow's Theorems*

thm:sylow1

THEOREM 8.3.1. *If p is a prime, $n : \mathbb{N}$ and G a finite group whose cardinality is divisible by p^n, then G has a subgroup of cardinality p^n.*

Proof. We prove the result by induction on n. If $n = 0$ we need to have a subgroup of cardinality 1, which is witnessed by the trivial subgroup. If $n > 0$, assume by induction that G contains a subgroup K of cardinality p^{n-1}. Now, K acts on the set G/K. The cardinality of G/K is divisible by p (since p^n divides the cardinality of G), and so by Lemma 8.2.1 the fixed point set $(G/K)^K$ has cardinality divisible by p.

Recall the Weyl group $W_G K$. By Lemma 5.6.3,

$$|W_G K| = |(G/K)^K|,$$

and so $W_G K$ has cardinality divisible by p.

Recall the normalizer subgroup $N_G(K)$ of G from Definition 5.6.1 and Section 6.6 and the surjective homomorphism p^H_G from $N_G H$ to $W_G H$, whose kernel may be identified with H so that $|N_G H| = |W_G H| \cdot |H|$ by Lagrange's theorem.

By Cauchy's Theorem 8.2.2 there is a subgroup L of $W_G K$ of cardinality p. Taking the preimage of L under the projection $p^H_G : \mathrm{Hom}(N_G H, W_G H)$, or, equivalently, the pullback

$$BH :\equiv BL \times_{BW_G K} BN_G K,$$

we obtain a subgroup H of $N_G(K)$ of cardinality p^n (H is a free K-set with p orbits). The theorem is proven by considering H as a subgroup of G. □

def:sylowsubgroup

DEFINITION 8.3.2. Let p^n be the largest power of p which divides the cardinality of G. A subgroup of G of cardinality p^n is called a *p-Sylow subgroup* of G and Syl^p_G is the G-subset of Sub_G of p-Sylow subgroups of G. $\lrcorner$

lem:numberofconjofSylow

LEMMA 8.3.3. *Let G be a finite group and P a p-Sylow subgroup. Then the number of conjugates of P is not divisible by p.*

[6] ((To be continued: the classical proof involves choosing nontrivial elements – see what can be done about that. At present this corollary is not used anywhere))

Proof. Let X be the G-set of conjugates of P. Being a G-orbit, X is equivalent G/Stab_P, where P is the stabilizer subgroup of P. Now, P is contained in the stabilizer so the highest power of p dividing the cardinality of G also divides the cardinality of Stab_P. □

((the approach below is on the abstract G-sets which may be ok given that this is what we're counting, but consider whether there is a more typie approach))

lem:sylowsareconjugate thm:sylow2

Theorem 8.3.4. *Let G be a finite group. Then any two p-Sylow subgroups are conjugate, or in other words, the G-set Syl^p_G is transitive.*

Furthermore, if H a subgroup of G of cardinality p^s and P a p-Sylow subgroup of G. Then H is conjugate to a subgroup of P.

Proof. We prove the last claim first. Consider the set $\mathcal{O}_P$ of conjugates of P as an H-set. Since the cardinality of $\mathcal{O}_P \simeq G/Stab_P$ is prime to p there must be an H-fixed point Q. In other words, $H \subseteq Stab_Q$. By Lemma 5.5.13 there is a conjugate H' of H with $H' \subseteq Stab_P$. Now, $P \subseteq Stab_P$ (ref) is a normal subgroup and so by **??**.

The first claim now follows, since if both H and P are p-Sylow subgroup, then a conjugate of H is a subgroup of P, but since these have the same cardinalities they must be equal. □

thm:sylow3

Theorem 8.3.5. *Let G be a finite group and let P be a p-Sylow subgroup of G. Then the cardinality of* Syl^p_G

(1) *divides $|G|/|P|$ and*

(2) *is 1 modulo p.*

Proof. Theorem 8.3.4 claims that Syl^p_G is transitive, so as a G-set it is equivalent to G/N_GP (N_GP is the stabilizer of P in Sub_G. Since P is a subgroup of N_GP we get that $|P|$ divides N_GP and so $|\mathrm{Syl}^p_G| = |G|/|N_GP|$ divides $|G|/|P|$.

Let i be the inclusion of P in G and consider the P-set $i^*\mathrm{Syl}^p_G$ obtained by restricting to P. Since the cardinality only depends on the underlying set we have that $|i^*\mathrm{Syl}^p_G| = |\mathrm{Syl}^p_G|$ and we analyze the decomposition into P-orbits to arrive at our conclusion.

Let $Q : i^*\mathrm{Syl}^p_G$ be a fixed point, i.e., $P \subseteq N_GQ$. Now, since N_GQ is a subgroup of G, we get that $|N_GQ|$ divides $|G$, so this proves that P is a p-Sylow subgroup of N_GQ. However, the facts that Q is normal in N_GQ and that all Sylow subgroups being conjugates together conspire to show that $P = Q$. That is, the number of fixed points in $i^*\mathrm{Syl}^p_G$ is one. Since P is a p-group, all the other orbits have cardinalities divisible by p, and so

$$|\mathrm{Syl}^p_G| = |i^*\mathrm{Syl}^p_G| \equiv 1 \mod p.$$

□

((Should we include standard examples, or is this not really wanted in this book?))

8.4 *cycle decompositions*

8.5 *Lagrange*

8.6 *Sylow stuff?*

9
Euclidean geometry

ch:euclidean

In this chapter we study Euclidean geometry. We assume some standard linear algebra over real numbers, including the notion of finite dimensional vector space over the real numbers and the notion of inner product. In our context, the field of real numbers, $\mathbb{R}$, is a set, and so are vector spaces over it. Moreover, a vector space V has an underlying additive abstract group, and we will feel free to pass from it to the corresponding group.

9.1 *Inner product spaces*

def:InnerProductSpace

Definition 9.1.1. An *inner product space V* is a real vector space of finite dimension equipped with an inner product $H : V \times V \to \mathbb{R}$. ⌟

Let $\tilde{\mathbb{V}}$ denote the type of inner product spaces. It is a type of pairs whose elements are of the form (V, H). For $n : \mathbb{N}$, let $\tilde{\mathbb{V}}_n$ denote the type of inner product spaces of dimension n.

For each natural number n, we may construct the *standard* inner product space $\mathbb{V}^n :\equiv (V, H)$ of dimension n as follows. For V we take the vector space $\mathbb{R}^n$, and we equip it with the standard inner product given by the dot product

$$H(x, y) :\equiv x \cdot y,$$

where the dot product is defined as usual as

$$x \cdot y :\equiv \sum_i x_i y_i.$$

thm:GramSchmidt

Theorem 9.1.2. *Any inner product space V is merely equal to $\mathbb{V}^n$, where n is $\dim V$.*

For the definition of the adverb "merely", refer to Definition 2.16.7.

Proof. Since any finite dimensional vector space merely has a basis, we may assume we have a basis for V. Now use Gram-Schmidt orthonormalization to show that $V = \mathbb{V}^n$. □

lem:InnerProductSpace1Type

Lemma 9.1.3. *The type $\tilde{\mathbb{V}}$ is a 1-type.*

Proof. Given two inner product spaces V and V', we must show that the type $V = V'$ is a set. By univalence, its elements correspond to the linear isomorphisms $f : V \xrightarrow{\simeq} V'$ that are compatible with the inner products. Those form a set. □

def:OrthogonalGroup

Definition 9.1.4. Given a natural number n, we define the *orthogonal group* $\mathrm{O}(n)$ as follows.

$$\mathrm{O}(n) :\equiv \underline{\Omega}\tilde{\mathbb{V}}_n$$

Here $\tilde{\mathbb{V}}_n$ is equipped with the basepoint provided by $\mathrm{sh}_{\mathrm{O}(n)} :\equiv \mathbb{V}^n$, and with the proof that it is a connected groupoid provided by Theorem 9.1.2 and Lemma 9.1.3. ⌟

The standard action (in the sense of Definition 6.2.2) of $\mathrm{O}(n)$ is an action of it on its designated shape $\mathbb{V}^n$. Letting $\mathrm{Vect}_{\mathbb{R}}$ denote the type of finite dimensional real vector spaces, we may compose the standard action with the projection map $\mathrm{BO}(n) \to \mathrm{Vect}_{\mathbb{R}}$ that forgets the inner product to get an action of $\mathrm{O}(n)$ on the vector space $\mathbb{R}^n$.

9.2 *Euclidean spaces*

In high school geometry courses, one encounters the Euclidean plane (of dimension 2) and the Euclidean space of dimension 3. The vectors and the points of Euclidean geometry are the basic ingredients, from which the other concepts are derived. Those concepts include such things as lines, line segments, triangles, tetrahedra, spheres, and so on. Symmetries of those objects are also studied: for example, an isosceles non-equilateral triangle has a total of 2 symmetries: the identity and the reflection through the midline.

So, a Euclidean space will come with two sets: a set of points and a set of vectors. The structure on the two sets includes the following items.

(1) If v and w are vectors, then there is a vector $v + w$ called its *sum*.

(2) If v is a vector and r is a real number, then there is a vector rv called the *scalar multiple* of v by r.

(3) If v is a vector, then there is a real nonnegative number called its *length*.

(4) If P and Q are points, then there is a unique vector v which can be "positioned" so its tail is "at" P and its head is "at" Q. It is called the vector *from P to Q*. The *distance* from P to Q is the length of v.

(5) If P is a point and v is a vector, then there is a unique point Q so that v which can be positioned so its tail is at P and its head is at Q. It is called the point obtained from P by *translation along v*.

We introduce the (new) notation $v + P$ for the point Q obtained from P by translation along v. Another fact from high school geometry is that if w is a vector, too, then the associative rule $v + (w + P) = (v + w) + P$ holds. This suggests that the essential features of high school geometry can be captured by describing the set of points as a torsor for the group of vectors.

We use that idea now to give a precise definition of *Euclidean space of dimension n*, together with its points and vectors. More complicated geometric objects will be introduced in subsequent sections.

def:EuclideanSpace

Definition 9.2.1. A *Euclidean space E* is an torsor A for the additive group underlying an inner product space V. (For the definition of torsor, see Definition 4.7.1.) ⌟

We will write V also for the additive group underlying V. Thus an expression such as $\mathrm{B}V$ or Torsor_V will be understood as applying to the underlying additive group[1] of V.

Definition 9.2.2. We denote the type of all Euclidean spaces of dimension n by $\tilde{\mathbb{E}}_n :\equiv \sum_{V : \tilde{\mathbb{V}}_n} \mathrm{Torsor}_V$. The elements of $\mathrm{Pts}\, E$ will be the *points* in the geometry of E, and the elements of $\mathrm{Vec}\, E$ will be the *vectors* in the geometry of E. We let $\tilde{\mathbb{E}}$ denote the type of all Euclidean spaces; it is equivalent to the sum $\sum_{n : \mathbb{N}} \tilde{\mathbb{E}}_n$. ⌟

The torsor $\mathrm{Pts}\, E$ is a nonempty set upon which V acts. Since V is an additive group, we prefer to write the action additively, too: given $v : V$ and $P : \mathrm{Pts}\, E$ the action provides an element $v + P : \mathrm{Pts}\, E$. Moreover, given $P, Q : \mathrm{Pts}\, E$, there is a unique $v : V$ such $v + P = Q$; for it we introduce the notation $Q - P :\equiv v$, in terms of which we have the identity $(Q - P) + P = Q$.

For each natural number n, we may construct the *standard* Euclidean space $\mathbb{E}^n : \tilde{\mathbb{E}}_n$ of dimension n as follows. For $\mathrm{Vec}\, E$ we take the standard inner product space $\mathbb{V}^n$, and for $\mathrm{Pts}\, E$ we take the corresponding principal torsor $\mathrm{Pr}_{\mathbb{R}^n}$.

thm:EuclideanNormalization

Theorem 9.2.3. *Any Euclidean space E is merely equal to $\mathbb{E}^n$, where n is* $\dim E$.

Proof. Since we are proving a proposition and any torsor is merely trivial, by Theorem 9.1.2 we may assume $\mathrm{Vec}\, E$ is $\mathbb{V}^n$. Similarly, we may assume that $\mathrm{Pts}\, E$ is the trivial torsor. □

lem:EuclideanSpace1Type

Lemma 9.2.4. *The type $\tilde{\mathbb{E}}_n$ is a 1-type.*

Proof. Observe using Theorem 4.6.6 that $\tilde{\mathbb{E}}_n \simeq s \sum_{V : \mathrm{BO}(n)} \mathrm{B}V$. The types $\mathrm{BO}(n)$ and $\mathrm{B}V$ are 1-types, so the result follows from Item (4). □

def:EuclideanGroup

Definition 9.2.5. Given a natural number n, we define the *Euclidean group* by

$$\mathrm{E}(n) :\equiv \underline{\Omega} \tilde{\mathbb{E}}_n.$$

Here we take the basepoint of $\tilde{\mathbb{E}}_n$ to be $\mathbb{E}^n$, and we equip $\tilde{\mathbb{E}}_n$ with the proof that it is a connected groupoid provided by Theorem 9.2.3 and Lemma 9.2.4. ⌟

The *standard action* of $\mathrm{E}(n)$ (in the sense of Definition 6.2.2) is an action of it on the Euclidean space $\mathbb{E}^n$.

thm:EuclideanGroupSemidirect

Theorem 9.2.6. *For each n, the Euclidean group $\mathrm{E}(n)$ is equivalent to a semidirect product $\mathrm{O}(n) \ltimes \mathbb{R}^n$.*

Proof. Recall Definition 6.4.5 and apply it to the standard action $\tilde{H} : \mathrm{BO}(n) \to \mathrm{Group}$ of $\mathrm{O}(n)$ on the additive group underlying $\mathbb{R}^n$, as defined in **??**. The semidirect product $\mathrm{O}(n) \ltimes \mathbb{R}^n$ has $\sum_{V : \mathrm{BO}(n)} \mathrm{B}V$ as its underlying pointed type. Finally, observe that $\mathrm{E}(n) \simeq \sum_{V : \mathrm{BO}(n)} \mathrm{B}V$, again using Theorem 4.6.6. □

9.3 *Geometric objects*

In this section, we discuss the notion of "object" in Euclidean space, but much of what we say is more general and applies equally well to other sorts of geometry, such as projective geometry or hyperbolic geometry.

[1] We are careful not to refer to the group as an Abelian group at this point, even though it is one, because the operator B may be used in some contexts to denote a different construction on Abelian groups.

Let E be a Euclidean space, as defined in Definition 9.2.1. The points of E are the elements of $\mathrm{Pts}\,E$, and intuitively, a geometric object in E ought to come with a way to tell which points of E are inside the object.

For example, in the standard Euclidean plane with coordinates labelled x and y, the x-axis is described by the equation $y = 0$. In other words, we have a function of type $g : \mathrm{Pts}\,E \to \mathrm{Prop}$ defined by $(x, y) \mapsto y = 0$. It's the predicate that defines the line as a subset of the plane. More complicated objects can also be specified as sets of points of E by other functions $\mathrm{Pts}\,E \to \mathrm{Prop}$. Now consider a typical Euclidean symmetry of the line, for example, the symmetry given by the function $t : (x, y) \mapsto (x + 3, y)$. It is compatible with the action of $\mathrm{Vec}\,E$ on $\mathrm{Pts}\,E$, and it sends the line to itself. If we consider the pair (E, g) as an element of the type $\sum_{E : \tilde{\mathbb{E}}}(\mathrm{Pts}\,E \to \mathrm{Prop})$, then, by univalence, we see that the translation t gives rise to an identification of type $(E, g) = (E, g)$.

Now suppose the object to be described is a car, as an object in a 3-dimensional Euclidean space. Then presumably we would like to give more information than just whether a point is inside the car: we may wish to distinguish points of the car by the type of material found there. For example, to distinguish the windshield (made of glass) from the hood (made of steel). Thus, letting M denote the set of materials found in the car, with one extra element for the points not in the car, we may choose to model the car as a function of type $\mathrm{Pts}\,E \to M$.

In order to unify the two examples above into a general framework, one may observe that Prop is a set (with 2 distinguished elements, True and False). That motivates the following definition.

Definition 9.3.1. Let M be a set. A *geometric object* is a pair (E, g) of type $\mathrm{EucObj} :\equiv \sum_{E : \tilde{\mathbb{E}}}(\mathrm{Pts}\,E \to M)$. If one wishes to emphasize the role played by the set M, we may refer to (E, g) as a geometric object *with materials drawn from the set* M.[2] We may also say that (E, g) is a geometric object *in* E. When M is Prop, we will think of the object as the subset of $\mathrm{Pts}\,E$ consisting of those points P such that $g(P)$ holds. ⌟

Exercise 9.3.2. Show that EucObj is a groupoid. ⌟

The exercise above allows us to speak of the symmetry group of a geometric object.

Exercise 9.3.3. Show that the symmetry group of a geometric object in $\mathbb{E}^n$ is a subgroup of $\mathrm{E}(n)$. ⌟

Exercise 9.3.4. Let E be a Euclidean space of dimension n, and let P be a point of E. The subset of $\mathrm{Pts}\,E$ containing just the point P is defined by the predicate $Q \mapsto (Q = P)$. Show that its symmetry group is isomorphic to $\mathrm{O}(n)$. ⌟

One often considers situations in geometry with multiple objects in the same space. For example, one may wish to consider two lines in the plane, or a point and a plane in space. This prompts the following definitions.

Definition 9.3.5. Suppose we are given an parameter type I and a set M_i for each $i \in I$. A *configuration* of geometric objects relative to that data is a Euclidean space E together with a function $p_i : \mathrm{Pts}\,E \to M_i$ for each $i \in I$. Its *consituents* are the geometric objects of the form (E, p_i), for each $i \in I$. If n is a natural number, and we let I be the finite type with n

[2] It would be a mistake to regard a geometric object as a triple (E, M, g), for then symmetries would be allowed to permute the materials.

fig:icosahedron

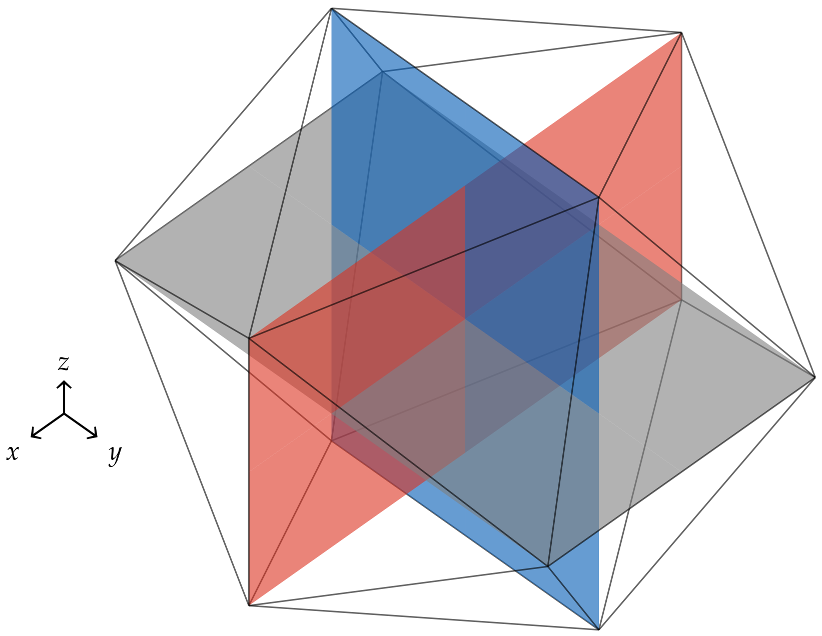


Figure 9.1: Icosahedron with its golden rectangles.

elements, then we may refer to the configuration as a configuration of n objects. ⌟

Definition 9.3.6. Given an type I and a family of geometric objects T_i parametrized by the elements of I, an *arrangement* of the objects is a configuration, also parametrized by the elements of I, whose i-th consituent is merely equal to T_i. ⌟

For example, suppose we consider arrangements consisting of a point and a line in the plane. The arrangements where the point is at a distance d from the line, where $d \geq 0$, are all merely equal to each other, because there is a Euclidean motion that relates any two of them. Hence, in some sense, the arrangements are classified by the set of nonnegative real numbers d. This motivates the following definition.

Definition 9.3.7. Given an parameter type I and a collection of geometric objects T_i parametrized by the elements of I, then an *incidence type* between them is a connected component of the type of all arrangements of the objects. ⌟

9.4 *The icosahedron*

Definition 9.4.1. The *icosahedron* (with side length 2) is the regular solid in standard euclidean three-space $\mathbb{E}^3$ with vertices at cyclic permutations of $(0, \pm 1, \pm\varphi)$, where $\varphi = (1 + \sqrt{5})/2$ is the golden ratio. ⌟

Remark 9.4.2. The four vertices $(0, \pm 1, \pm\varphi)$ make up a *golden rectangle* with short side length equal to 2. To check that the above vertices really form a regular polyhedron, we just need to calculate the length between to adjacent corners of golden rectangles:

$$\|(0, 1, \varphi) - (1, \varphi, 0)\| = \sqrt{1 + (\varphi - 1)^2 + \varphi^2} = \sqrt{4} = 2 \qquad ⌟$$

10
Geometry (first look)

10.1 *incidence geometries and the Levi graph*

10.2 *euclidean planes*

10.3 *ruler and compass constructions*

10.4 *affine planes and Pappus' law*

10.5 *projective planes*

11

Vector spaces and linear groups

Quotients; subspaces (= ?). Bases and so. Dual space; orthogonality. (all of this depends on good implementations of subobjects). Eigen-stuff. Characteristic polynomials; Hamilton-Cayley.

11.1 *the algebraic hierarchy*: *groups, abelian groups, rings, fields*

11.2 *vector spaces*

11.3 *the general linear group as automorphism group*

11.4 *determinants*(†)

12
Field theory

12.1 *examples*: *rationals, polynomials, adding a root, field extensions*

12.2 *ordered fields, real-closed fields, pythagorean fields, euclidean fields*

12.3 *complex fields, quadratically closed fields, algebraically closed fields*

12.4 *Diller-Dress theorem*(†)

13
Classification of wallpaper groups(†)

14
Affine geometry

Barycentric calculus. Affine transformations. Euclidean / Hermitian geometry (isometries, conformity...)

14.1 *affine frames, affine planes*

14.2 *the affine group as an automorphism group*

14.3 *the affine group as a semidirect product*

14.4 *affine properties* (*parallelism, length ratios*)

15
Bilinear forms

16
Inversive geometry (Möbius)

16.1 *residue at a point is affine*

16.2 *Miquel's theorem*

17
Projective geometry

Projective spaces (projective invariance, cross ratio, harmonic range...). Conics/quadrics. (Classification in low dimensions?)

complex algebraic plane projective curves (tangent complexes, singular points, polar, hessian, ...).

17.1 *projective frames*

17.2 *the projective group and projectivities*

17.3 *projective properties* (*cross-ratio*)

17.4 *fundamental theorem of projective geometry*

18
Minkowski space-time geometry

Affine spaces + vector spaces + quadric with some signature.

19
Kleinian geometries

19.1 *conics and dual conics*

19.2 *elliptic geometry*

20

Galois theory

chap:galois-theory

The goal of Galois theory is to study how the roots of a given polynomial can be distinguished from one another. Take for example $X^2 + 1$ as a polynomial with real coefficients. It has two distincts roots in $\mathbb{C}$, namely i and $-i$. However, an observer, who is limited to the realm of $\mathbb{R}$, can not distinguish between the two. Morally speaking, from the point of view of this observer, the two roots i and $-i$ are pretty much the same. Formally speaking, for any polynomial $Q : \mathbb{R}[X, Y]$, the equation $Q(i, -i) = 0$ is satisfied if and only if $Q(-i, i) = 0$ also. This property is easily understood by noticing that there is a automorphism of fields $\sigma : \mathbb{C} \to \mathbb{C}$ such that $\sigma(i) = -i$ and $\sigma(-i) = i$ which also fixes $\mathbb{R}$. The goal of this chapter is to provide the rigourous framework in which this statement holds. TODO: complete/rewrite the introduction

20.1 *Covering spaces and field extensions*

sec:cover-spac-fields

Recall that a field extension is simply a morphism of fields $i : k \to K$ from a field k to a field K. Given a fixed field k, the type of fields extensions of k is defined as

$$k\backslash\mathbf{Fields} :\equiv \sum_{K : \mathbf{Fields}} \hom_{\mathbf{Fields}}(k, K)$$

def:galois-group

Definition 20.1.1. The Galois group of an extension (K, i) of a field K, denoted $\mathrm{Gal}(K, i)$ or $\mathrm{Gal}(K/k)$ when i is clear from context, is the group $\mathrm{Aut}_{k\backslash\mathbf{Fields}}(K, i)$. ⌟

rem:sip-univalence

Remark 20.1.2. The Structure Identity Principle holds for fields, which means that for $K, L : \mathbf{Fields}$, one has

$$(K = L) \simeq \mathrm{Iso}(K, L)$$

where $\mathrm{Iso}(K, L)$ denotes the type of these equivalences that are homomorphisms of fields. Indeed, if one uses K and L also for the carrier types of the fields, one gets:

$$\begin{aligned}(K = L) \simeq \sum_{p : K =_{\mathcal{U}} L} &(\mathrm{trp}_p(+_K) = +_L) \times (\mathrm{trp}_p(\cdot_K) = \cdot_L) \\ &\times (\mathrm{trp}_p(0_K) = 0_L) \times (\mathrm{trp}_p(1_K) = 1_L)\end{aligned}$$

Any $p : K =_{\mathcal{U}} L$ is the image under univalence of an equivalence $\phi : K \simeq L$, and then:

$$\begin{aligned}
\operatorname{trp}_p(+_K) &= (x,y) \mapsto \phi(\phi^{-1}(x) +_K \phi^{-1}(y))\\
\operatorname{trp}_p(\cdot_K) &= (x,y) \mapsto \phi(\phi^{-1}(x) \cdot_K \phi^{-1}(y))\\
\operatorname{trp}_p(0_K) &= \phi(0_K)\\
\operatorname{trp}_p(1_K) &= \phi(1_K)
\end{aligned}$$

It follows that:

$$\begin{aligned}
(K = L) \simeq \sum_{\phi : K \simeq L} &(\phi(x +_K y) = \phi(x) +_L \phi(y))\\
&\times(\phi(x \cdot_K y) = \phi(x) \cdot_L \phi(y))\\
&\times(\phi(0_K) = 0_L) \times (\phi(1_K) = 1_L)
\end{aligned}$$

The type on the right hand side is the same as $\operatorname{Iso}(K, L)$ by definition.

In particular, given an extension (K, i) of K:

$$\operatorname{UGal}(K, i) \simeq \sum_{p : K = K} \operatorname{trp}_p i = i \simeq \sum_{\sigma : \operatorname{Iso}(K,K)} \sigma \circ i = i$$

This is how the Galois group of the extension (K, i) is defined in ordinary mathematics. ⌟

Given an extension (K, i) of field k, there is a map of interest:

$$i^* : K\backslash\mathbf{Fields} \to k\backslash\mathbf{Fields}, \quad (L, j) \mapsto (L, ji)$$

lem:field-ext-restriction-set-bundle

Lemma 20.1.3. *The map i^* is a set-bundle.*

Proof. Given a field extension (K', i') in $k\backslash\mathbf{Fields}$, one wants to prove that the fiber over (K', i') is a set. Suppose (L, j) and (L', j') are extensions of K, together with paths $p : (K', i') = (L, ji)$ and $p' : (K', i') = (L', j'i)$. Recall that p and p' are respectively given by equivalences $\pi : K' = L$ and $\pi' : K' = L'$ such that $\pi i' = ji$ and $\pi' i' = j'i$. A path from $((L, j), p)$ to $((L', j'), p')$ in the fiber over (K', i') is given a path $q : (L, j) = (L', j')$ in $K\backslash\mathbf{Fields}$ such that $\operatorname{trp}_q p = p'$. However, such a path q is the data of an equivalence $\varphi : L = L'$ such that $\varphi j = j'$, and then the condition $\operatorname{trp}_q p = p'$ translates as $\varphi\pi = \pi'$. So it shows that φ is necessarily equal to $\pi'\pi^{-1}$, hence is unique. □

The fiber of this map at a given extension (L, j) of k is:

$$\begin{aligned}
(i^*)^{-1}(L, j) &\simeq \sum_{L' : \mathbf{Fields}} \sum_{j' : K \to L'} (L, j) = (L', j'i)\\
&\simeq \sum_{L' : \mathbf{Fields}} \sum_{j' : K \to L'} \sum_{p : L = L'} pj = j'i\\
&\simeq \sum_{j' : K \to L} j = j'i\\
&\simeq \hom_k(K, L)
\end{aligned}$$

where the last type denotes the type of homomorphisms of k-algebra (the structure of K and L being given by i and j respectively).

In particular, the map $t : \operatorname{UGal}(K, i) \to (i^*)^{-1}(K, i)$ mapping g to $\operatorname{trp}_g(\mathrm{id}_K)$ identifies with the inclusion of the k-automorphisms of K into the k-endomorphisms of K.

TODO: write a section on polynomials in chapter 12

Definition 20.1.4. Given an extension $i : k \to K$, an element $\alpha : K$ is algebraic if α is merely a root of a polynomial with coefficients in k. That is if the following proposition holds:

$$\| \sum_{n : \mathbb{N}} \sum_{a : \mathbb{n}+1 \to k} i(a(0)) + i(a(1))\alpha + \cdots + i(a(n))\alpha^n = 0 \|$$

⌟

defn:algebraic-element

defn:algebraic-extension

Definition 20.1.5. A field extension (K, i) is said to be algebraic when each $a : K$ is algebraic. ⌟

Remark 20.1.6. Note that when the extension (K, i) is algebraic, then t is an equivalence. However, the converse is false, as shown by the non-algebraic extension $\mathbb{Q} \hookrightarrow \mathbb{R}$. We will prove that every $\mathbb{Q}$-endomorphism of $\mathbb{R}$ is the identity function. Indeed, any $\mathbb{Q}$-endormorphism $\varphi : \mathbb{R} \to \mathbb{R}$ is linear and sends squares to squares, hence is non-decreasing. Let us now take an irrational number $\alpha : \mathbb{R}$. For any rational $p, q : \mathbb{Q}$ such that $p < \alpha < q$, then $p = \varphi(p) < \varphi(\alpha) < \varphi(q) = q$. Hence $\varphi(\alpha)$ is in any rational interval that α is. One deduces $\varphi(\alpha) = \alpha$. ⌟

aic-endomorphisms-are-automorphisms

defn:degree-field-extension

Definition 20.1.7. A field extension $i : k \to K$ is said finite when K as a k-vector space, the structure of which is given by i, is of finite dimension. In that case, the dimension is called the degree of i, denoted $[(K, i)]$ or $[K : k]$ when i is clear from context. ⌟

20.2 *Intermediate extensions and subgroups*

Given two extensions $i : k \to K$ and $j : K \to L$, the map i^* can be seen as a pointed map

$$i^* : \mathrm{BGal}(L, j) \to \mathrm{BGal}(L, ji), \quad x \mapsto x \circ i.$$

Then, through Lemma 20.1.3, i^* presents $\mathrm{Gal}(L, j)$ as a subgroup of $\mathrm{Gal}(L, ji)$. One goal of Galois theory is to characterize those extensions $i' : k \to L$ for which all subgroups of $\mathrm{Gal}(L, i')$ arise in this way.

Given any extension $i : k \to L$, there is an obivous $\mathrm{Gal}(L, i)$-set X given by

$$(L', i') \mapsto L'.$$

For a pointed connected set-bundle $g : B \to \mathrm{BGal}(L, i)$, one can consider the type of fixed points of the $\underline{\Omega} B$-set Xf:

$$K :\equiv (Xg)^{\underline{\Omega} B} \equiv \prod_{x : B} X(g(x))$$

It is a set, which can be equipped with a field structure, defined pointwise. Morevover, if one denotes b for the distinguished point of B, and (L'', j'') for $g(b)$, then, because g is pointed, one has a path $p : L = L''$ such that $pi' = j''$. There are fields extensions $i' : k \to K$ and $j' : K \to L$ given by:

$$i'(a) :\equiv x \mapsto \mathrm{snd}(g(x))(a), \quad j'(f) :\equiv p^{-1} f(b)$$

In particular, for all $a : k$, $j'i'(a) = p^{-1}\,\mathrm{snd}(g(b))(a) = p^{-1} j''(a) = i'(a)$.

Galois theory is interested in the settings when these two contructions are inverse from each other.

20.3 *separable/normal/etc.*

20.4 *fundamental theorem*

sec:cover-spac-fields-1

sec:fundamental-theorem

21
Impossible constructions

21.1 *doubling the cube*

21.2 *trisecting the angle*

21.3 *squaring the circle*

21.4 *7-gon*

21.5 *quintic equations*

22 Possible constructions

22.1 5-gon and the icosahedron

(cyclotomic field of deg 5 has galois group $(\mathbb{Z}/5\mathbb{Z})^\times \simeq \mathbb{Z}/4\mathbb{Z}$)

22.2 17-gon and 257-gon

22.3 cubics and quartics

23
Witt theory, SOSs, Artin-Schreier

23.1 *quadratic forms*

23.2 *Grothendieck-Witt ring*

24
Dual numbers and split-complex numbers

24.1 *minkowski and galilaean spacetimes*

A
Historical remarks

ch:grouphistory

Here we briefly sketch some of the history of groups. See the book by Wussing[1] for a detailed account, as well as the shorter survey by Kleiner[2].

Some waypoints we might mention include:

- Early nineteenth century geometry, the rise of projective geometry, Möbius and Plücker
- Early group theory in number theory, forms, power residues, Euler and Gauss.
- Permutation groups, Lagrange and Cauchy, leading (via Ruffini) to Abel and Galois.
- Liouville and Jordan[3] ruminating on Galois.
- Cayley, Klein and the Erlangen Program[4].
- Lie and differentiation.
- von Dyck and Hölder.
- J.H.C. Whitehead and crossed modules.
- Artin and Schreier theory.
- Algebraic groups (Borel and Chevalley et al.)
- Feit-Thompson and the classification of finite simple groups.
- Grothendieck and the homotopy hypothesis.
- Voevodsky and univalence.

[1] Hans Wussing. *The genesis of the abstract group concept*. A contribution to the history of the origin of abstract group theory, Translated from the German by Abe Shenitzer and Hardy Grant. MIT Press, Cambridge, MA, 1984, p. 331.

[2] Israel Kleiner. "The evolution of group theory: a brief survey". In: *Math. Mag.* 59.4 (1986), pp. 195–215. DOI: 10.2307/2690312.

[3] Camille Jordan. *Traité des substitutions et des équations algébriques*. Les Grands Classiques Gauthier-Villars. Reprint of the 1870 original. Éditions Jacques Gabay, Sceaux, 1989, pp. xvi+670.

[4] Felix Klein. "Vergleichende Betrachtungen über neuere geometrische Forschungen". In: *Math. Ann.* 43.1 (1893), pp. 63–100. DOI: 10.1007/BF01446615.

B

Metamathematical remarks

app:metamath

[TODO: Give precise rules for our type theory and discuss some syntactic matters: definitional equality, more precision around normalization and canonicity, etc.]

The statement that something is not provable is a statement *about*, and not *in*, a theory. Proving such a statement often requires properties of the theory that cannot be formulated nor proved in the theory itself.

One such 'metaproperty' that we use here is *canonicity*. We call an element *closed* if it does not contain free variables. An example of canonicity is that every closed expression of type $\mathbb{N}$ is a *numeral*, that is, either 0 or $S(n)$ for some numeral n. Another example of canonicity is that every closed expresssion of type $L \amalg R$ is either of the form inl_l for some $l : L$ or of the form inr_r for some $r : R$. A second important metaproperty of our theory is that one can compute canonical forms.

[TODO: Write more about what metamathematics is.]

B.1 Definitional equality

sec:defeq

B.1.1 Basics

sec:defeq-basics

The concept of definition was introduced in Section 2.2, together with what it means to be *the same by definition*. Being the same by definition, or being definitionally equal, (NB appears for the first time on p. 26!) is a relationship between syntactic expressions. In this section we provide more details about this relationship.

There are four basic forms of definitional equality:

it:exp-defeq

(1) Resulting from making an explicit definition, e.g., $1 :\equiv \mathrm{succ}(0)$, after which we have $1 \equiv \mathrm{succ}(0)$;[1]

it:imp-defeq

(2) Resulting from making an implicit definition, like we do in inductive definitions, e.g., $n + 0 :\equiv n$ and $n + \mathrm{succ}(m) :\equiv \mathrm{succ}(n + m)$, after which we have $n + 0 \equiv n$ and $n + \mathrm{succ}(m) \equiv \mathrm{succ}(n + m)$;

it:beta

(3) Simplifying the application of an explicitly defined function to an argument, e.g., $(x \mapsto e_x)(a) \equiv e_a$;

it:eta

(4) Simplifying $(x \mapsto e_x)$ to f when e_x is the application of the function f to the variable x, e.g., $(x \mapsto S(x)) \equiv S$.

Definitional equality is the *congruence closure* of these four basic forms, that is, the smallest reflexive, symmetric, transitive and congruent relation that contains all instances of the four basic forms. Here a congruent relation is a relation that is closed under all syntactic operations

[1] The notation $:\equiv$ tells the reader that we make a definition (or reminds the reader that this definition has been made).

of type theory. One such operation is substitution, so that we get from the examples above that, e.g., $1 + 0 \equiv 1$ and $n + \mathrm{succ}(\mathrm{succ}(m)) \equiv \mathrm{succ}(n + \mathrm{succ}(m))$. Another important operation is application. For example, we can apply succ to each of the sides of $n + \mathrm{succ}(m) \equiv \mathrm{succ}(n + m)$ and get $\mathrm{succ}(n + \mathrm{succ}(m)) \equiv \mathrm{succ}(\mathrm{succ}(n + m))$, and also $n + \mathrm{succ}(\mathrm{succ}(m)) \equiv \mathrm{succ}(\mathrm{succ}(n + m))$ by transitivity.

Let's elaborate $\mathrm{id} \circ f \equiv f$ claimed on page 8. The definitions used on the left hand side are $\mathrm{id} :\equiv (y \mapsto y)$ and $g \circ f :\equiv (x \mapsto g(f(x)))$. In the latter definition we substitute id for g and get $\mathrm{id} \circ f \equiv (x \mapsto \mathrm{id}(f(x)))$. Unfolding id we get $(x \mapsto \mathrm{id}(f(x))) \equiv (x \mapsto (y \mapsto y)(f(x)))$. Applying (3) we can substitute $f(x)$ for $(y \mapsto y)(f(x)))$ and get $(x \mapsto (y \mapsto y)(f(x))) \equiv (x \mapsto f(x))$. By (4) the right hand side is definitionally equal to f. Indeed $\mathrm{id} \circ f \equiv f$ by transitivity.

Definitional equality is also relevant for typing. For example, let $A : \mathcal{U}$ and $P : A \to \mathcal{U}$. If $B \equiv A$, then $(B \to \mathcal{U}) \equiv (A \to \mathcal{U})$ by congruence, and also $P : B \to \mathcal{U}$, and even $\prod_{x:B} P(x) \equiv \prod_{x:A} P(x)$.

B.1.2 *Deciding definitional equality (not updated yet)*

sec:defeq-computation

By a *decision procedure* we mean a terminating algorithmic procedure that answers a yes/no question. Although it is possible to enumerate all true definitional equalities, this does not give a test that answers whether or not a given instance $e \equiv e'$ holds. In particular when $e \equiv e'$ does not hold, such an enumeration will not terminate. A test of definitional equality is important for type checking, as the examples in the last paragraph of the previous section show.

A better approach to a test of definitional equality is the following. First direct the four basic forms of definitional equality from left to right as they are given.[2] For the first two forms this can be viewed as unfolding definitions, and for the last two forms as simplifying function application and (unnecessary) abstraction, respectively. This defines a basic reduction relation, and we write $e \longrightarrow e'$ if e' can be obtained by a basic reduction of a subexpression in e. The reflexive transitive closure of $\longrightarrow$ is denoted by $\longrightarrow^*$. The symmetric closure of $\longrightarrow^*$ coincides with $\equiv$.

We mention a few important properties of the relations $\longrightarrow$, $\longrightarrow^*$ and $\equiv$. The first is called the Church–Rosser property, and states that, if $e \equiv e'$, then there is an expression c such that $e \longrightarrow^* c$ and $e' \longrightarrow^* c$. The second is called type safety and states that, if $e : T$ and $e \longrightarrow e'$, then also $e' : T$. The third is called termination and states that for well-typed expressions e there is no infinite reduction sequence starting with e. The proofs of Church–Rosser and type safety are long and tedious, but pose no essential difficulties. For a non-trivial type theory such as in this book the last property, termination, is extremely difficult and has not been carried out in full detail. The closest come results on the Coq[3] (TODO: find good reference).

Testing definitional equality of given well-typed terms e and e' can now be done by reducing them with $\longrightarrow$ until one reaches irreducible expressions n and n' such that $e \longrightarrow^* n$ and $e' \longrightarrow^* n'$, and then comparing n and n'. Now we have: $e \equiv e'$ iff $n \equiv n'$ iff (by Church–Rosser) there exists a c such that $n \longrightarrow^* c$ and $n' \longrightarrow^* c$. Since n and n' are irreducible the latter is equivalent to n and n' being identical syntactic expressions.

[2] TODO: think about the last, η.

[3] The Coq Development Team. *The Coq Proof Assistant*. Available at `https://coq.inria.fr/`.

B.2 *The Limited Principle of Omniscience*

sec:LPO rem:LPO-solves-halting problem

Remark B.2.1. Recall the Limited Principle of Omniscience (LPO), Principle 3.6.16: for any function $P : \mathbb{N} \to \mathbb{2}$, either there is a smallest number $n_0 : \mathbb{N}$ such that $P(n_0) = 1$, or P is a constant function with value 0. We will show that LPO is not provable in our theory.

The argument is based on the halting problem: given a Turing machine M and an input n, determine whether M halts on n. It is known that the halting problem cannot be solved by an algorithm that can be implemented on a Turing machine.[4]

We use a few more facts from computability theory. First, Turing machines can be enumerated. We denote the n^{th} Turing machine M_n, so we can list the Turing machines in order: $M_0, M_1, \ldots$. Secondly, there exists a function $T(e, n, k)$ such that $T(e, n, k) = 1$ if M_e halts on input n in at most k steps, and $T(e, n, k) = 0$ otherwise. This function T can be implemented in our theory.

Towards a contradiction, assume we have a closed proof t of LPO in our theory. We assume as well that t does not depend on any axiom.[5] It is clear that $k \mapsto T(e, n, k)$ is a constant function with value 0 if and only if M_e does not halt on input n. Now consider $t(k \mapsto T(e, n, k))$, which is an element of a type of the form $L \amalg R$.

We now explain how to solve the halting problem. Let e and n be arbitrary numerals. Then $t(k \mapsto T(e, n, k))$ is a closed element of $L \amalg R$. Hence we can compute its canonical form. If $t(k \mapsto T(e, n, k)) \equiv \mathrm{inr}_r$ for some $r : R$, then $k \mapsto T(e, n, k)$ is a constant function with value 0, and M_e does not halt on input n. If $t(k \mapsto T(e, n, k)) \equiv \mathrm{inl}_l$ for some $l : L$, then M_e does halt on input n. Thus we have an algorithm to solve the halting problem for all e and n. Since this is impossible, we have refuted the assumption that there is a closed proof t of LPO in our theory. ⌟

B.3 *Topology*

In this section we will explain how our intuition about types relates to our intuition about topological spaces.

INSERT AN INTRODUCTORY PARAGRAPH HERE.

injectionsurjectionisnotwhatyouthink

Remark B.3.1. Our definitions of injections and surjections are lifted directly from the intuition about sets. However, types need not be sets, and thinking of types as spaces may at this point lead to a slight confusion.

The real line is contractible and the inclusion of the discrete subspace $\{0, 1\}$ is, well, an inclusion (of sets, which is the same thing as an inclusion of spaces). However, $\{0, 1\}$ is not connected, seemingly contradicting the next result.

ft:cohesive

This apparent contradiction is resolved once one recalls the myopic nature of our setup: the contractibility of the real line means that "all real numbers are identical", and *our* "preimage of 3.25" is not a proposition: it contains *both* 0 and 1. Hence "$\{0, 1\} \subseteq \mathbb{R}$" would not count as an injection in our sense.

We should actually have been more precise above: we were referring to the *homotopy type* of the real line, rather than the real line itself.[6] We

[4] It's commonly accepted that every algorithm *can* be thus implemented.

[5] It is possible to weaken the notion of canonicity so that the argument still works even if the proof t uses the Univalence Axiom. Of course, the argument must fail if we allow t to use LEM!

[6] We don't define this formally here, see Shulman[7] for a synthetic account. The idea is that the homotopy type $\mathrm{h}(X)$ of a type X has a map from X, $\iota : X \to \mathrm{h}(X)$, and any continuous function $f : [0, 1] \to X$ gives rise to a path $\iota(f(0)) = \iota(f(1))$ in $\mathrm{h}(X)$.

[7] Michael Shulman. "Brouwer's fixed-point theorem in real-cohesive homotopy type theory". In: *Mathematical Structures in Computer Science* 28.6 (2018), pp. 856–941. doi: 10.1017/S0960129517000147. arXiv: 1509.07584.

shall later (in the chapters on geometry) make plenty of use of the latter, which is as usual a set with uncountably many elements. ⌟

Bibliography

Brouwer-1908 Atten, Mark Van and Göran Sundholm. “L.E.J. Brouwer’s ‘Unreliability of the Logical Principles’: A New Translation, with an Introduction”. In: *History and Philosophy of Logic* 38.1 (2017), pp. 24–47. DOI: 10.1080/01445340.2016.1210986. arXiv: 1511.01113 (page 35).

Baez-Shulman Baez, John C. and Michael Shulman. “Lectures on n-categories and cohomology”. In: *Towards higher categories*. Vol. 152. IMA Vol. Math. Appl. Springer, New York, 2010, pp. 1–68. DOI: 10.1007/978-1-4419-1524-5_1. arXiv: math/0608420 (page 47).

Connes1983 Connes, Alain. “Cohomologie cyclique et foncteurs Ext^n”. In: *C. R. Acad. Sci. Paris Sér. I Math.* 296.23 (1983), pp. 953–958 (page 63).

sep-type-theory Coquand, Thierry. “Type Theory”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Metaphysics Research Lab, Stanford University, 2018. URL: https://plato.stanford.edu/archives/fall2018/entries/type-theory/ (page 8).

Jordan Jordan, Camille. *Traité des substitutions et des équations algébriques*. Les Grands Classiques Gauthier-Villars. Reprint of the 1870 original. Éditions Jacques Gabay, Sceaux, 1989, pp. xvi+670 (page 188).

Klein-EP-de Klein, Felix. “Vergleichende Betrachtungen über neuere geometrische Forschungen”. In: *Math. Ann.* 43.1 (1893), pp. 63–100. DOI: 10.1007/BF01446615 (page 188).

Kleiner-group-survey Kleiner, Israel. “The evolution of group theory: a brief survey”. In: *Math. Mag.* 59.4 (1986), pp. 195–215. DOI: 10.2307/2690312 (page 188).

peano-principia Peano, Giuseppe. *Arithmetices principia: nova methodo*. See also https://github.com/mdnahas/Peano_Book/ for a parallel translation by Vincent Verheyen. Fratres Bocca, 1889. URL: https://books.google.com/books?id=z80GAAAAYAAJ (page 9).

Pruefer-AG Prüfer, Heinz. “Theorie der Abelschen Gruppen”. In: *Math. Z.* 20.1 (1924), pp. 165–187. DOI: 10.1007/BF01188079 (page 146).

Rijke-Join Rijke, Egbert. *The join construction*. 2017. arXiv: 1701.07538 (page 36).

Shulman-Real-Cohesive Shulman, Michael. “Brouwer’s fixed-point theorem in real-cohesive homotopy type theory”. In: *Mathematical Structures in Computer Science* 28.6 (2018), pp. 856–941. DOI: 10.1017/S0960129517000147. arXiv: 1509.07584 (page 191).

Stallings1991 Stallings, John R. “Foldings of G-trees”. In: *Arboreal group theory (Berkeley, CA, 1988)*. Vol. 19. Math. Sci. Res. Inst. Publ. Springer, New York, 1991, pp. 355–368. DOI: 10.1007/978-1-4612-3142-4_14 (page 157).

Coq Team, The Coq Development. *The Coq Proof Assistant*. Available at https://coq.inria.fr/ (page 190).

hottbook Univalent Foundations Program, The. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013 (pages 19, 20, 39, 41, 50).

Wussing-genesis Wussing, Hans. *The genesis of the abstract group concept*. A contribution to the history of the origin of abstract group theory, Translated from the German by Abe Shenitzer and Hardy Grant. MIT Press, Cambridge, MA, 1984, p. 331 (page 188).

Glossary

p^{-1} · reverse identification, path inverse, Definition 2.5.1, 12

$\bar{f}$ · ua(f), 26

$\tilde{p}$ · cast along a path between types, 26

$:$ · element judgment, 5

$:\equiv$ · definition, 7

$=_X$ · identity type, Item (E1), 11

$\equiv$ · definitional equality, 8

$=^Y_p$ · path-over type, Definition 2.7.1, 16

$\circ$ · function composition, 8

$(X \xrightarrow{f} Y \xrightarrow{g} Z)$ · function composition, 8

$p * q, q \cdot p, qp, q \circ p$ · path concatenation or composition, 13

$\to$ · function type, 6

$\mapsto$ · “maps to”, function definition, 8

η · η-rule, 8

ι_+ · embedding of $\mathbb{N}$ into Z, 52

ι_- · embedding of $\mathbb{N}^-$ into Z, 52

Π · product type, 6

Ω · loop type, Definition 4.1.1, 82

$\underline{\Omega}$ · group constructor, Definition 4.1.9, 84

$\underline{\Omega}$ · homomorphism constructor, Definition 4.3.2, 94

0 · the natural number zero, Peano’s rules, Item (P2), 9

1 · the natural number 1, 9

2 · the natural number 2, 9

3 · the natural number 3, 9

ap_f · application of a function to a path, Definition 2.6.1, 15

A_+ · A together with a disjoint base point, 39

$\mathrm{cast}_{X,Y}(p)$ · cast along a path between types, Definition 2.13.1, 26

id · identity function, 8

$\mathrm{im}(f)$ · the (propositional) image of f, 32

$\mathbb{N}$ · the type of natural numbers, Peano’s rules, Item (P1), 9

$\mathbb{N}^-$ · the type of negated natural numbers, Example 2.12.6, 25

ns · naturality square, Definition 2.6.5, 16

po_p · convert path over path, Definition 2.7.2, 16

refl_a · reflexivity, identity type, Item (E2), 11

s · successor function on Z, 53

succ · the successor function on $\mathbb{N}$, Peano’s rules, Item (P3), 9

trp^T_e · transport function, Definition 2.5.4, 14

twist · interchange the elements of Bool, Exercise 2.13.3, 26

ua · the inverse of cast, from univalence, 26

$\mathcal{U}$ · universe, 8

$\mathcal{U}^{=1}_*$ · pointed, connected groupoids, Definition 4.1.6, 83

$\mathcal{U}_*$ · universe of pointed types, Definition 2.21.1, 39

Z · the set of integers, Definition 3.2.1, 52

Index

cardinality, 160

abstract monomorphisms, 135

bound variable, 7

cast, 26
cokernel, 130
composition
  of paths, 13
concatenation
  of paths, 13
congruence, 11
conjugate, 135
conjugation, 93, 100, 135
connected set bundle, 53

decidable set bundle, 54
decidable proposition, 35
decidable set, 38
definition, 7
degree
  function, 63
dummy variable, 7

element, 5
epimorphism
  of groups, 112
equality, *see* identification
equality type, *see* identity type
equation, *see* identity type

family
  of elements, 6
  of types, 5
finite set bundle, 54
finite group, 160
forget, 47
  higher structure, 48
  properties, 48
  structure, 48
function, 6
  degree m, 63
function extensionality, 15

groupoid, 28
$\mathcal{G}$-set (of abstract group, 106
G-set of monomorphisms into G, 134

Hedberg's theorem, 39

identification, 11
identity, *see* identification
identity type, 11
image, 48
  function, 132
  group, 132
  projection to, 132
injection, 37
injective, 37
intersection
  of monomorphisms, 141
  of sets, 141
isomorphism
  of groups, 95
iteration, 53

kernel, 130
  associated to, 139
  group, 130
Klein four-group, 89

Limited Principle of Omniscience, 68
loop type, 82

map, 6
mathematics
  univalent, 5
monomorphism
  of groups, 112

naturality square, 16
normal, 130
normal subgroup, 136
normalizer, 143

parameter type, 6, 14
path over, 16
pointed type, 39

predicate, 36
product type, 6, 14
proof, 11
    of a proposition, 28
proper monomorphism, 127
proper subgroup, 128
proposition, 28
Propositional resizing, 35
pullback, 140
    of groups, 141

quotient group, 137
quotient homomorphism, 137

Replacement principle, 36

set, 28
    of kernels, 136
substitution, 13
subtype, 37
successor, 9
sum of groups, 122
Sylow subgroup, 163
symmetry
    in a type, 50
    of an element, 12

transitive *G*-set, 103
transport, 14
trivial monomorphism, 127
trivial subgroup, 128
type, 4
    n-truncated, 28
    of normal subgroups, 136

union of sets, 141
univalence axiom, 26
universal set bundle, 54
universe, 8

Vierergruppe, 89

wedge of pointed types, 122
Weyl group, 143

zero, 9