Mathematical Structures in Computer Science (2022), **32**, pp. 982–1014
doi:10.1017/S0960129523000026

PAPER

A rewriting coherence theorem with applications in homotopy type theory

Nicolai Kraus[1] and Jakob von Raumer[2*]

[1]School of Computer Science, University of Nottingham, Nottingham NG7 2RD, UK and [2]Karlsruhe Institute of Technology, Karlsruhe, Germany

*Corresponding author. Email: jakob@von-raumer.de

(Received 30 April 2021; revised 9 November 2022; accepted 6 January 2023; first published online 6 February 2023)

Abstract

Higher-dimensional rewriting systems are tools to analyse the structure of formally reducing terms to normal forms, as well as comparing the different reduction paths that lead to those normal forms. This higher structure can be captured by finding a *homotopy basis* for the rewriting system. We show that the basic notions of *confluence* and *wellfoundedness* are sufficient to recursively build such a homotopy basis, with a construction reminiscent of an argument by Craig C. Squier. We then go on to translate this construction to the setting of *homotopy type theory*, where managing equalities between paths is important in order to construct functions which are *coherent* with respect to higher dimensions. Eventually, we apply the result to approximate a series of open questions in homotopy type theory, such as the characterisation of the homotopy groups of the *free group on a set* and the *pushout of 1-types*. This paper expands on our previous conference contribution *Coherence via Wellfoundedness* by laying out the construction in the language of higher-dimensional rewriting.

Keywords: Homotopy type theory; higher-dimensional rewriting; constructive mathematics; coherence of structure; well-founded relation; confluence; Squier theory; abstract rewriting system; polygraph

1. Introduction

1.1 Confluence and coherence in higher-dimensional rewriting

In classical mathematics and computer science, a relation $\rightsquigarrow$ on a set M is called *terminating* or *Noetherian* if there is no infinite sequence $x_0 \rightsquigarrow x_1 \rightsquigarrow x_2 \rightsquigarrow \ldots$. This is a standard property of term rewriting (or reduction) systems such as the *typed lambda calculus* (Barendregt et al. 1992) and ensures that any term can be fully reduced, that is, after finitely many steps an irreducible term (a *normal form*) is reached. Similarly, *confluence* is the property that, whenever one has $x \mathrel{{}^*{\leftsquigarrow}} w \rightsquigarrow^* y$, there is a z such that $x \rightsquigarrow^* z \mathrel{{}^*{\leftsquigarrow}} y$. This property expresses that the final result of a sequence of reductions is independent of the order in which reduction steps are performed. Together, termination and confluence guarantee that every term has a unique normal form.

We can go up one level: Rather than asking whether two reduction sequences give the same result, we can ask whether two parallel reductions $u, v \in (x \rightsquigarrow^* y)$ are equal, or related, in some appropriate sense; in other words, we can ask whether the system of reduction steps is *coherent*, that is, whether different steps 'fit together'. One way to give meaning to the question is to consider (higher) rewriting steps between reduction sequences of the form $\alpha \in (u \Rightarrow v)$ and identify a set of 'good' steps. If any reduction sequence u can be rewritten into any sequence v parallel to it by concatenating or pasting 'good' steps or their inverses, the set of 'good' reduction steps is called a *homotopy basis*.

Termination and confluence of the reduction relation $\rightsquigarrow$ also help with the construction of homotopy bases. As an example, Newman's lemma shows that two reduction sequences starting from a single object x can be completed to parallel reduction sequences ending in the same object, and Newman (1942) as well as Huet (1980) essentially demonstrate that the resulting shape can be filled up with smaller shapes which witness local confluence. Going further, Newman (1942) also shows that a closed zig-zag of reductions is always deformable into the empty cycle using local confluence diagrams,[1] and a generalisation of this observation will be central to the current paper.

An argument in the same direction was presented by Squier (Squier and Otto 1987; Squier 1987). He constructed homotopy bases to show that a monoid with a finite presentation which is terminating and confluent satisfies the homological finiteness condition known as $(\mathsf{FP})_3$. Squier's original application was that monoids can have a decidable word problem without having a finite, terminating, and confluent presentation. A further improvement was made by Otto and Kobayashi using new ideas by Squier et al. (1994). Later, constructions based on Squier's arguments were used to solve various coherence problems, for example for Gray (Forest and Mimram 2018) and monoidal categories (Guiraud and Malbos 2012) as well as for Artin (Gaussent et al. 2015), plactic (Hage and Malbos 2017), and general monoids (Guiraud et al. 2013). The aim of the current paper is to show that a rewriting result, reminiscent of Newman's and Squier's, can also be applied in homotopy type theory.

1.2 Higher-dimensional structures in type theory

Higher-dimensional structures also naturally appear in *Martin-Löf type theory* (Martin-Löf 1984), where types are ∞-groupoids (globular Lumsdaine 2009; van den Berg and Garner 2011, simplicial Kraus 2015*a*,b, internal Allioux et al.; Annenkov et al. 2019) and universes are $(\infty, 1)$-categories (cf. Annenkov et al. 2019 ∞-groupoids). Given a type A in a type-theoretic universe $\mathcal{U}$, terms $x, y : A$ are the objects (0-cells) of the higher structure determined by A.

What allows us to form the morphisms and higher morphisms is Martin-Löf's *identity type*. Given terms x and y of type A, we write this type as $\mathsf{Id}_A(x, y)$ or simply as $x = y$. The type theory community calls an element (or a term) $p : x = y$ a *(propositional) equality*, an *identification*, or (and this is the terminology we use in this paper) a *path*. The path type $x = y$ is the type of 1-cells, and for $p, q : x = y$, the type $p = q$ (the iterated path type) is the type of 2-cells, and so on. This view can be made precise further by interpreting type theory in the setting of abstract homotopy theory (Awodey and Warren 2009) and especially in simplicial sets (Kapulkin and Lumsdaine 2018) or cubical sets (Cohen et al. 2017), themselves models of topological spaces.

Homotopy type theory (The Univalent Foundations Program 2013) is a variation of Martin-Löf type theory which embraces this interpretation of types as spaces by adding internal principles that are justified by a range of models. The central such principle is Voevodsky's *univalence axiom* (Kapulkin and Lumsdaine 2018), presented as an assumption in the original formulation of homotopy type theory (cf. The Univalent Foundations Program 2013) but derivable internally in cubical type theories (cf. Cohen et al. 2017).

Most of the results establishing that types are ∞-groupoids are of meta-theoretic nature (with the exception of attempts to internalise meta-theoretic results Allioux et al.; Annenkov et al. 2019). When working internally, for example when using a proof assistant, the other side of the coin becomes visible: Attempting to treat types as sets, or the universe as an ordinary 1-category, often leads to problems which stem from the fact that paths are structure rather than property (i.e. they are not unique). In other words, a path $p : x = y$ should be seen as an isomorphism in a higher category rather than an equality between elements of a set. A collection of (iso-) morphisms in a higher category often requires coherences in order to be well-behaved. The same is the case in homotopy type theory, and constructing these coherences is often a central difficulty.

1.3 Applying rewriting arguments in homotopy type theory

The goal of this paper is to demonstrate how techniques from higher-dimensional rewriting can be applied in homotopy type theory and to derive several new and non-trivial results with these techniques. Although the most immediate applications of rewriting to type theory are of meta-theoretic nature (e.g. showing that β-reduction is confluent), this is explicitly *not* what we mean; as indicated in the previous paragraph above, we mean purely internal type-theoretic applications.

The arguments that we apply are very far removed from the idea of computing normal forms and similar concepts that we have described in the beginning of Section 1.1. In the type-theoretic setting, the role of the reduction relation is played by a type family $R : A \times A \to \mathcal{U}$, and the statement that $a : A$ reduces to $b : A$ simply becomes the type $R(a, b)$. It is standard to give a relation a name such as $\sim$ which is then used infix, that is, one writes $a \sim b$ or $a \rightsquigarrow b$. In our type-theoretic applications, it will in general be undecidable whether a given element a can be reduced, whether a reduces to b, and even whether a and b are path-equal. In particular, it is not possible to compute normal forms.

1.4 Set-quotients and the usefulness of a homotopy basis

Let us try to describe fairly concretely why a homotopy basis is useful in homotopy type theory. We assume basic familiarity with the contents and notations of the book (The Univalent Foundations Program 2013), the terminology of which we use. Recall that a type A is a *proposition* if any two of its elements are equal,

$$\mathsf{isProp}(A) :\equiv \Pi(x\,y : A).x = y, \tag{1}$$

and a *set* if any two parallel equalities are equal, that is, if every equality type $x = y$ is a proposition,

$$\mathsf{isSet}(A) :\equiv \Pi(x\,y : A).\mathsf{isProp}(x = y). \tag{2}$$

One also says that sets are the types that satisfy the principle of unique identity proofs (UIP).

Recall from The Univalent Foundations Program (2013, Chp 6.10) that, for a given type $A : \mathcal{U}$ together with a relation $(\sim) : A \to A \to \mathcal{U}$, the set-quotient can be implemented as the higher inductive type

$$\begin{aligned} &\text{inductive } A/{\sim} \text{ where} \\ &\quad \iota : A \to A/{\sim} \\ &\quad \mathsf{glue} : \Pi\{a, b : A\}.(a \sim b) \to \iota(a) = \iota(b) \\ &\quad \mathsf{trunc} : \Pi\{x, y : A/{\sim}\}.\Pi(p, q : x = y).p = q \end{aligned} \tag{3}$$

The last constructor trunc ensures that the type $A/{\sim}$ is a set. From the above representation, we can derive the usual elimination rule for the set-quotient: In order to get a function $f : (A/{\sim}) \to X$, we need to give a function $g : A \to X$ such that, whenever $a \sim b$, we have $g(a) = g(b)$. However, this only works if X is a set itself. If it is not, we have a priori no way of constructing the function f.

Let us look at one instance of the problem. We consider the following set-quotient, which we will use as a running example. It is a standard construction that has been discussed in The Univalent Foundations Program (2013, Chp 6.11).

Example 1 (free group)**.** Let M be a set. We construct the free group on M as a set-quotient. We consider lists over $M \uplus M$, where we think of the left copy of M as positive and the right copy as negative elements. For $x : M \uplus M$, we write x^{-1} for the 'inverted' element:

$$\mathsf{inl}(a)^{-1} :\equiv \mathsf{inr}(a) \qquad\qquad \mathsf{inr}(a)^{-1} :\equiv \mathsf{inl}(a) \tag{4}$$

We let the binary relation $\rightsquigarrow$ on $\mathsf{List}(M \uplus M)$ to be generated by the reduction steps for all lists $[\,\ldots, x_i, \ldots\,]$:

$$[\,\ldots, x_1, x_2, x_2^{-1}, x_3, \ldots\,] \rightsquigarrow [\,\ldots, x_1, x_3, \ldots\,]. \tag{5}$$

Then, the set-quotient $\mathsf{List}(M \uplus M)/\rightsquigarrow$ is the free group on M: It satisfies the correct universal property by The Univalent Foundations Program (2013, Thm 6.11.7).

Another way to construct the free group on M is to re-use the natural groupoid structure that every type carries; this can be seen as a typical 'homotopy type theory style' construction. It works as follows. The *wedge of M-many circles* is the (homotopy) coequaliser of two copies of the map M into the unit type, $\mathsf{hcolim}(M \rightrightarrows \mathbf{1})$. Using a higher inductive type, it can be explicitly constructed:

$$\begin{aligned}&\text{inductive } \mathsf{hcolim}(M \rightrightarrows \mathbf{1}) : \mathcal{U} \text{ where}\\ &\quad \mathsf{base} : \mathsf{hcolim}(M \rightrightarrows \mathbf{1})\\ &\quad \mathsf{loop} : M \to \mathsf{base} = \mathsf{base}\end{aligned} \tag{6}$$

Its loop space $\Omega(\mathsf{hcolim}(M \rightrightarrows \mathbf{1}))$ is by definition simply $\mathsf{base} = \mathsf{base}$. This loop space carries the structure of a group in the obvious way: the neutral element is given by reflexivity, multiplication is given by path composition, symmetry by path reversal, and every $a : M$ gives rise to a group element $\mathsf{loop}(a)$. This construction is works without the assumption that M is a set and defines the free *higher* group (cf. Kraus and Altenkirch 2018). Following Bezem et al., we write $\mathsf{F}(M)$ for this free higher group:

$$\mathsf{F}(M) :\equiv \Omega(\mathsf{hcolim}(M \rightrightarrows \mathbf{1})). \tag{7}$$

In contrast to this observation, the set-quotient of Example 1 ignores any existing higher structure (cf. The Univalent Foundations Program 2013, Rem 6.11.8) and thus really only defines the free 'ordinary' (set-level) group. If we do start with a set M, it is a natural question whether the free higher group and the free group coincide: There is a canonical function

$$\mathsf{F}(M) \to \mathsf{List}(M \uplus M)/\!\!/\rightsquigarrow, \tag{8}$$

defined analogously to $\Omega(\mathsf{S}^1) \to \mathbb{Z}$, cf. The Univalent Foundations Program (2013). Classically, this function is an equivalence. Constructively, it is an open problem to construct an inverse of (8).

The difficulties do not stem from the first two constructors of the set-quotient. Indeed, we have a canonical map

$$\omega_1 : \mathsf{List}(M \uplus M) \to \mathsf{F}(M) \tag{9}$$

which maps a list such as $[\mathsf{inl}(a_1), \mathsf{inr}(a_2), \mathsf{inl}(a_3)]$ to the composition of paths given as $\mathsf{loop}(a_1) \cdot (\mathsf{loop}(a_2))^{-1} \cdot \mathsf{loop}(a_3)$. For this map, we also have

$$\omega_2 : \Pi(\ell_1, \ell_2 : \mathsf{List}(M \uplus M)).(\ell_1 \rightsquigarrow \ell_2) \to \omega_1(\ell_1) = \omega_1(\ell_2) \tag{10}$$

since consecutive inverse loops cancel each other out. Therefore, if we define $(A/\!\!/\rightsquigarrow)$ to be the higher inductive type (3) *without* the constructor trunc, that is, the *untruncated quotient* or *coequaliser*, then there is a canonical map

$$\omega : \mathsf{List}(M \uplus M)/\!\!/\rightsquigarrow \to \mathsf{F}(M). \tag{11}$$

Thus, the difficulty with defining an inverse of (8) lies solely in the question whether $\mathsf{F}(M)$ is a set. This is an open problem which has frequently been discussed in the homotopy type theory community (a slight variation is recorded in The Univalent Foundations Program 2013, Ex 8.2). It is well-known in the community how to circumvent the problem if M has decidable equality. However, the only piece of progress on the general question that we are aware of is the result in Kraus and Altenkirch (2018), where it is shown that all fundamental groups (The Univalent Foundations Program 2013, Chp 6.11) are trivial. In other words: Instead of showing that *everything* above truncation level 0 is trivial, the result shows that a single level is trivial. The proof in

Kraus and Altenkirch (2018) uses a rather intricate construction which is precisely tailored to the situation.

The construction of functions $(A/\rightsquigarrow) \to X$ motivates the connection to higher-dimensional rewriting. We do not allow an arbitrary type X; however; instead, we assume that X is 1-truncated, that is, a *groupoid* a.k.a. a *1-type*, which means that all path spaces of X are sets,

$$\mathsf{isGrp}(A) :\equiv \Pi(x\, y : A).\mathsf{isSet}(x = y). \tag{12}$$

As an application, we will give a new proof for the theorem that the fundamental groups of $\mathsf{F}(M)$ are trivial. We will also show a family of similar statements, by proving a common generalisation.

The characterisation of the equality types of $(A/\rightsquigarrow)$ makes it necessary to consider *closed zig-zags* in A. A closed zig-zag is simply an element of the symmetric-reflexive-transitive closure, for example:

$$s : a \rightsquigarrow b \qquad p : d \rightsquigarrow c$$
$$t : c \rightsquigarrow b \qquad q : a \rightsquigarrow d$$

$$\begin{tikzcd} a \arrow[r, "q"] \arrow[d, "s"'] & d \arrow[d, "p"] \\ b & c \arrow[l, "t"] \end{tikzcd} \tag{13}$$

Our first result related to quotients (Theorem 38) says: We get a function $(A/\rightsquigarrow) \to X$ into a groupoid X if we have $f : A \to X$ and $h : (a \rightsquigarrow b) \to f(a) = f(b)$, together with the coherence condition stating that h maps any closed zig-zag to a 'commuting cycle' in X. In the case of the example (13) above, this means that the composition $h(s) \cdot h(t)^{-1} \cdot h(p)^{-1} \cdot h(q)^{-1}$ equals $\mathsf{refl}_{f(a)}$. Theorem 38 is fairly simple, and we do not consider it a major contribution of this paper.

The actual contribution of the paper is to make Theorem 38 usable, since, on its own, it is virtually impossible to apply in any non-trivial situation. The reason for this is that the coherence condition talks about *closed* zig-zags. Zig-zags are inductively generated (they are simply a chain of segments), but closed zig-zags are not. If we have a property of closed zig-zags which we cannot generalise to arbitrary zig-zags, then there is no obvious inductive strategy to show the property in general: if we remove a segment of a closed zig-zag, it is not closed any more. In all our examples, it seems not possible to formulate an induction hypothesis based on isolated not necessarily closed zig-zags.[2]

Thus, how can Theorem 38 be made usable? This is where the construction of a homotopy basis comes into play. In the example of the free group, the relation $\rightsquigarrow$ on $\mathsf{List}(M \uplus M)$ can be presented in a way that ensures that it is Noetherian and confluent, conditions that are (stronger than) necessary in order to construct the homotopy basis, and we have full control over how we want this basis to look like. While it is very hard to show a property directly for *all* closed zig-zags, it is much more manageable to show the property for closed zig-zags in the homotopy basis, and if the property is nice enough (which it is in all our examples), then this is sufficient.

Let us get back to homotopy type theory. The combination of the two mentioned results (and Theorems 38 and 31) gives us Theorem 40: Given a groupoid X and $f : A \to X$ such that $a \rightsquigarrow b$ implies $f(a) = f(b)$, it suffices to show that closed zig-zags in the basis are mapped to trivial equality proofs. We apply this to show that the free higher group over a set has trivial fundamental groups. There is a family of similar statements that we also discuss and prove.

1.5 Structure of this paper and formalisation

The ideas and developments in this paper can be split into three categories:

(1) The first category consists of the plain rewriting arguments that are to some degree independent of the foundation in which they are formulated. These arguments work in set-theoretic settings as well as in various forms of type theory.

(2) The second category addresses the choices that have to be made when choosing homotopy type theory as the foundational setting. A central question is whether one works with general types or *h-sets*, that is, types of truncation level 0.
(3) The final category consists of our concrete applications in homotopy type theory.

We strive to separate the rewriting arguments from the type-theoretic arguments as much as possible and structure the development as follows:

- In Section 2, we show the specific construction of a homotopy basis in a standard generic (unspecified) set-theoretic framework. This corresponds to category 1 above. We attempt to follow the style and presentation of other papers in the field on higher-dimensional rewriting.
- The fairly short Section 3 points out which parts of the development presented in Section 2 require particular attention when switching to homotopy type theory, and in which ways our type-theoretic formulation is more general. In this section, we give a high-level explanation of these points that corresponds to category 2.
- Section 4 gives the complete type-theoretic translation.
- We have formalised the development of Section 4 in Agda. The source code is available at `bitbucket.org/fplab/confluencecoherence` and requires Agda 2.6.2.2 to be installed. In addition, we make a browsable `html` version available at `cs.nott.ac.uk/~psznk/agda/confluence/`. This `html` version requires no software to be installed, is fully interlinked (i.e. everything can be clicked on to reach its definition), and benefits from complete syntax highlighting.
- Finally, we explain the applications in homotopy type theory in Section 5 (category 3).

This structure allows a reader who is interested in the rewriting argument, but less so in type theory, to only study Section 2. At the same time, an expert in the field of homotopy type theory will probably understand all ideas by additionally reading Section 3 without going through Section 4 in full.

$$\begin{tikzcd}
 & \boxed{\text{Section 1}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\boxed{\text{Section 2}} \arrow[d] & \boxed{\text{Section 4}} \arrow[r, dash] \arrow[dd] & \boxed{\text{Agda}} \arrow[ddl] \\
\boxed{\text{Section 3}} \arrow[dr] & & \\
 & \boxed{\text{Section 5}} &
\end{tikzcd}$$

However, a reader interested in the full type-theoretic development and all the nitty-gritty details may wish to skip Sections 2 and 3 completely and immediately jump to Section 4 as well as the accompanying Agda formalisation: Section 4 can be seen as a high-level guide through the Agda code, and it comes with links to the `html` version of all the important definitions and theorems.

Finally, Section 5 explains our applications in homotopy type theory.

1.6 Background of the paper

The core observation on which this article is based is that confluence and wellfoundedness can be used to prove coherence results internally in homotopy type theory. We (the current authors) originally presented this insight at the LICS'20 conference (Kraus and von Raumer 2020). We received very helpful feedback. In particular, Vincent van Oostrom explained to us the connection to several lines of work in the rewriting community, especially the relationship to Squier's work. The current article extends and improves the conference paper (Kraus and von Raumer 2020) and attempts to more cleanly separate the rewriting arguments from the type-theoretic applications.

The arguments of the current article are somewhat different and the results in a certain way more general than the results in the conference paper. By specialising the main result of Section 4, that is, the construction of a homotopy basis in type theory (Theorem 30), we can derive a version of the main result of the conference paper (Theorem 31); this then turns out to actually be slightly weaker in a subtle sense, as explained in Remark 32. However, these differences are insignificant from the point of view of the applications that we present.

The conference paper was presented with a formalisation in the Lean theorem prover, available at `gitlab.com/fplab/freealgstr`. This formalisation is independent of and follows a different strategy from, our Agda implementation.

2. A Homotopy Basis for Noetherian 2-Polygraphs

In this section, we explain the construction of a homotopy basis in a generic set-theoretic framework. In later sections, we will show how the development can be translated to and generalised in homotopy type theory and used to address open questions in the field.

2.1 Notations for 1-polygraphs

A *1-polygraph* is given by two sets Σ_0 and Σ_1 together with two functions $s_0, t_0 : \Sigma_1 \to \Sigma_0$. This structure is sometimes known as a *quiver* (Gabriel 1972), a *directed pseudograph*, or simply a *directed graph*. Alternatively, it may be described as a low-dimensional special case of Street's *computads* (Street 1987), analogously to how it is a special case of Burroni's *polygraphs* (Burroni 1993).

We refer to elements of Σ_0 as *objects* and elements of Σ_1 as *reduction steps* or simply *steps*. Given $u \in \Sigma_1$, we call $s_0(u)$ the *source* and $t_0(u)$ the *target* of u. For $x, y \in \Sigma_0$, we write $(x \rightsquigarrow y)$ for the subset of Σ_1 containing those reduction steps which have x as source and y as target.

We write $(x \rightsquigarrow^* y)$ for the set of composable (if $x = y$ possibly empty) finite sequences of reduction steps which start in x and end in y. Elements of $(x \rightsquigarrow^* y)$ are *reduction sequences* or simply *sequences* from x to y. Formally, such a sequence consists of an *object sequence* $(x = x_0, x_1, \ldots, x_n = y)$, with $x_i \in \Sigma_0$, and a list $(u_1, \ldots, u_n)$ with $u_i \in (x_{i-1} \rightsquigarrow x_i)$. Given two composable sequences, say $u \in (x \rightsquigarrow^* y)$ and $v \in (y \rightsquigarrow^* z)$, we write $u \cdot v$ for their composition, $u \cdot v \in (x \rightsquigarrow^* z)$. We use the same notation if u and/or v is a single step instead of a sequence.

$(y \leftsquigarrow x)$ denotes a copy of the set $(x \rightsquigarrow y)$. We write $(x \leftrightsquigarrow y)$ for the disjoint sum $(x \rightsquigarrow y) \uplus (x \leftsquigarrow y)$ and $(x \leftrightsquigarrow^* y)$ for the set of *reduction zig-zags* (or simply *zig-zags*) from x to y. Just as a reduction sequence, a reduction zig-zag is given by an object sequences and a list of steps $(u_1, \ldots, u_n)$, where however we allow that either $u_i \in (x_{i-1} \rightsquigarrow x_i)$ or $u_i \in (x_{i-1} \leftsquigarrow x_i)$. If for all steps the first (second) option is the case, the zig-zag is called *positive* (*negative*). Based on this, we use notations such as $(y \leftsquigarrow x \rightsquigarrow z)$, which is the set of pairs (u, v) with $u \in (x \rightsquigarrow y)$ and $v \in (x \rightsquigarrow z)$. In this case, we write $u^{-1} \cdot v \in (y \leftsquigarrow x \rightsquigarrow z)$ to make clear that u has been formally inverted, and $u^{-1} \cdot v$ is seen as a reduction zig-zag.

Finally, we denote by Σ_1^* the union of all sets of the form $(x \rightsquigarrow^* y)$, that is, the set of all sequences. Note that we have one trivial sequence of length 0 for each $x \in \Sigma_0$, which we denote by ε_x. Similarly, $(\Sigma_1 \uplus \Sigma_1^{-1})^*$ denotes the union of all sets of the form $(x \leftrightsquigarrow^* y)$, that is, the set of all zig-zags. For $u \in (\Sigma_1 \uplus \Sigma_1^{-1})^*$, we write $s_0(u)$ for the starting element of the object sequence (i.e. x_0 in the description above) and $t_0(u)$ for the last (i.e. x_n).

2.2 Terminating 1-polygraphs

Let $>$ be a relation on a set M. We call an element $n \in M$ *accessible* if all $m \in M$ with $n > m$ are accessible. Recall that $>$ is called *Noetherian* (or *co-wellfounded*) if all elements of M are accessible. Recall further that, for such a relation, we can perform *Noetherian induction*: Given a property on M, if the property holds for an $n \in M$ as soon as it holds for all m with $n > m$, then the property holds for all n. Note that a relation is Noetherian if and only if its transitive closure is.

Let a 1-polygraph $\Sigma_0 \leftleftarrows \Sigma_1$ be given. We say that the polygraph is equipped with a (Noetherian) order if we have a (Noetherian) order $>$ on the set Σ_0. We say that the polygraph is *terminating* if it is equipped with a transitive Noetherian order $>$ and, whenever there is a $u \in (x \rightsquigarrow y)$, we have $x > y$. We furthermore extend an ordering to zig-zags by, for example, saying that for $u \in \Sigma_1$ we have $x > u$ if and only if $x > y_i$ for all $y_i \in \Sigma_0$ in the object sequence of u.

Remark 2. Note that the above definitions of *accessibility* and *Noetherian* are phrased in a way that makes them usable in a constructive setting. It is a consequence that there exists no infinite sequence $x_0 > x_1 > \ldots$; however, that statement taken as a definition would not allow us to perform the constructions we do in this paper.

Remark 3. Accessibility could be formulated directly in terms of Σ_1 instead of referring to a relation $>$. The reason for introducing $>$ is that requiring $x > y$ can be a weaker condition than requiring $x \rightsquigarrow^* y$, cf. Remark 26.

2.3 The list extension of a Noetherian relation

Given a set M, we write M^* for the set of finite (possibly empty) lists. Given a relation $>$ on M, we extend it to a relation $>_L$ on M^*, mirroring the *multiset extension* by Dershowitz and Manna (1979). This *list extension* is defined as follows, where we choose to build in the transitive closure. For lists $\vec{n}, \vec{m} \in M^*$, we have $\vec{n} >_L \vec{m}$ if it is possible to transform $\vec{n}$ into $\vec{m}$ by applying the following operation one or multiple times: remove one element n of the list and replace it by a finite list, where each new list element has to be smaller than n. In other words, the list extension of $>$ is the smallest relation $>_L$ on M^* which is:

(1) transitive;
(2) closed under congruence, that is, if $\vec{k}, \vec{l}, \vec{m}, \vec{n} \in M^*$ are four lists such that $\vec{n} > \vec{m}$, then we also have $(\vec{k} \cdot \vec{n} \cdot \vec{l}) > (\vec{k} \cdot \vec{m} \cdot \vec{l})$;
(3) and, if $n \in M$ and $(n_1, \ldots, n_e) \in M^*$ such that $\forall i \in \{1, \ldots, e\}. n > n_i$, then $(n) > (n_1, \ldots, n_e)$. Here, (n) is the list of length 1 with the single element n and $\cdot$ donates list concatenation.

Recall that the multiset extension of a Noetherian relation is Noetherian (Dershowitz and Manna 1979). As the multiset extension subsumes the list extension, the same statement holds for the list extension. The proof that we give is an adaption of an argument by Nipkow (1998).

Lemma 4. *Let M be a set with a relation $>$. If $>$ is Noetherian on M, then the list extension $>_L$ on the set M^* is also Noetherian.*

Proof. We need to show that every list in M^* is accessible. Since accessibility is closed under transitive closure, we can without loss of generality assume the definition of $>_L$ to lack the transitive closure and only contain the other two closure properties. We do this in three steps:

(1) If lists ℓ_1 and ℓ_2 are accessible, then so is their concatenation $\ell_1 \cdot \ell_2$.
(2) Single element lists (m), with $m \in M$, are accessible.
(3) Arbitrary lists of objects are accessible.

Point (1) holds because, in order to make arrive at a list smaller than $\ell_1 \cdot \ell_2$, one has to make ℓ_1 smaller or ℓ_2 smaller. More precisely, any list smaller than $\ell_1 \cdot \ell_2$ is of the form $\ell_1' \cdot \ell_2$ or of the form $\ell_1 \cdot \ell_2'$ or of the form $\ell_1' \cdot \ell_2'$, with $\ell_1 >_L \ell_1'$ and $\ell_2 >_L \ell_2'$. Since ℓ_1 and ℓ_2 are individually accessible, this shows that their concatenation is.[3]

To prove (2), we apply Noetherian induction on $x \in M$. To show that (x) is accessible, assume $(x) >_L \ell$ with $\ell = (x_1, \dots x_n)$. We have to show that ℓ is accessible. By the induction hypothesis, each (x_i) is accessible. Further, we have $\ell = (x_1) \cdot \dots \cdot (x_n)$. Thus, the statement is given by point (1).

For the proof of (3), we now combine the two previous steps and use the same argument as before: Every list can be written as a concatenation of singleton lists and is therefore accessible. □

2.4 Generalised 2-polygraphs

Burroni's notion of a *2-polygraph* (Burroni 1993) extends a 1-polygraph $\Sigma_0 \leftleftarrows \Sigma_1$. The extension consists of a set Σ_2 together with two functions $s_1, t_1 : \Sigma_2 \to \Sigma_1^*$ (i.e. a second 1-polygraph $\Sigma_1^* \leftleftarrows \Sigma_2$) subject to the condition that, for each $\alpha \in \Sigma_2$, we have $s_0(s_1(\alpha)) = s_0(t_1(\alpha))$ and $t_0(s_1(\alpha)) = t_0(t_1(\alpha))$. The data that we want to work with are captured by a slight generalisation of a 2-polygraph that is obtained by replacing Σ_1^* by $(\Sigma_1 \uplus \Sigma_1^{-1})^*$, that is, we generalise sequences to zig-zags. Given a generalised 2-polygraph with two zig-zags $u, v \in (x \leftrightsquigarrow^* y)$, we write $(u \Rightarrow v)$ for the set of all $\alpha \in \Sigma_2$ with $s_1(\alpha) = u$ and $t_1(\alpha) = v$ and call α a *rewrite step* from u to v. We use the notations $(u \overset{*}{\Rightarrow} y)$ as well as $(u \Leftrightarrow v)$ and $(u \overset{*}{\Leftrightarrow} v)$ analogously to $(x \rightsquigarrow^* y)$, $(x \leftrightsquigarrow y)$, and $(x \leftrightsquigarrow^* y)$, respectively.

We say that a generalised 2-polygraph is terminating if the underlying 1-polygraph $\Sigma_0 \leftleftarrows \Sigma_1$ is terminating. Further, we say that a generalised 2-polygraph is *closed under congruence* if, for any $\alpha \in (\Sigma_2 \uplus \Sigma_2^{-1})^*$ and $u, v \in (\Sigma_1 \uplus \Sigma_1^{-1})^*$ with $s_0(s_1(\alpha)) = t_0(u)$ and $t_0(t_1(\alpha)) = s_0(v)$, we have a chosen zig-zag of 2-cells $u \cdot \alpha \cdot v \in (\Sigma_2 \uplus \Sigma_2^{-1})^*$ with $s_1(u \cdot \alpha \cdot v) = u \cdot s_1(\alpha) \cdot v$ and $t_1(u \cdot \alpha \cdot v) = u \cdot t_1(\alpha) \cdot v$.[4]

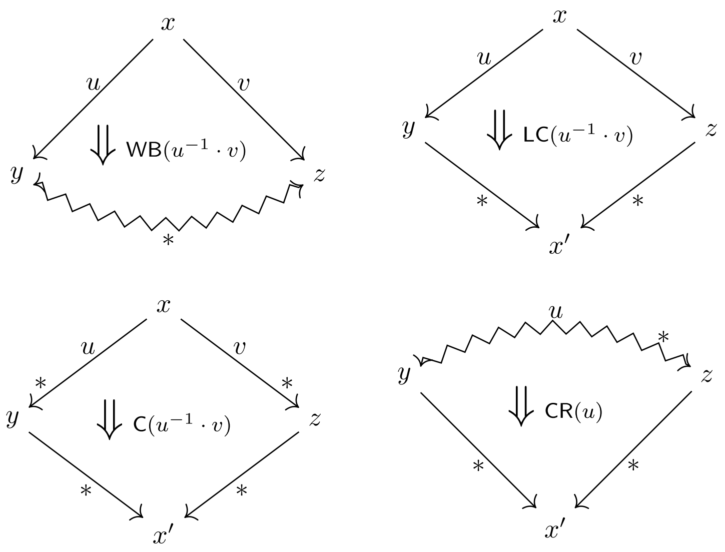
Remark 5. The fact that a *rewrite step* in our setting is simply an element of Σ_2 is in contrast with the terminology of other authors (e.g. Alleaume and Malbos 2016), for whom a rewrite step is a compositions of the form $u \cdot \alpha \cdot v$. For polygraphs that are closed under congruence, this distinction becomes essentially irrelevant since we are mostly interested in $(\Sigma_2 \uplus \Sigma_2^{-1})^*$ rather than Σ_2.

Remark 6. Note that the data $(\Sigma_0, \Sigma_1, \Sigma_2, s_0, t_0, s_1, t_1)$ of a generalised 2-polygraph can equivalently be described as a 1-polygraph $\Sigma_0 \leftleftarrows \Sigma_1$ and, for each pair $x, y \in \Sigma_0$, another 1-polygraph $(x \leftrightsquigarrow^* y) \leftleftarrows \Sigma_2^{x,y}$. Unfolding further, this data consists of a set Σ_0; for each pair $x, y \in \Sigma_0$, a set $(x \rightsquigarrow y)$ of reduction steps, and, for each pair $x, y \in \Sigma_0$ and $u, v \in (x \leftrightsquigarrow^* y)$, a set $(u \Rightarrow v)$ of rewrite steps. This presentation is much more natural in type theory and will be used in Section 4.

2.5 Generalisations of Newman's Lemma

Various notions of *confluence* have been studied to characterise well-behaved rewriting systems. In this section, we will recall four of these definitions, adapt them to our constructive meta-theory, and compare them with each other. The notions that we consider are shown in Figure 1.

We will use the following terminology to refer to the global and local 'topography' of reduction sequences: We call $u \in (y \,{}^*\!\leftsquigarrow x \rightsquigarrow^* z)$ a *peak* and $v \in (y \rightsquigarrow^* x \,{}^*\!\leftsquigarrow z)$ a *valley*. If we replace the respective reduction sequences by single reduction steps, $u \in (y \leftsquigarrow x \rightsquigarrow z)$ and $v \in (y \rightsquigarrow x \leftsquigarrow z)$ will be called a *local peak* (or *span*) and *local valley* (or *co-span*), respectively.



$$\begin{tikzcd}
& x \arrow[dl, "u"'] \arrow[dr, "v"] \arrow[d, phantom, "\Downarrow \mathsf{WB}(u^{-1}\cdot v)"] & \\
y \arrow[rr, leftrightsquigarrow, "*"'] & & z
\end{tikzcd}
\qquad
\begin{tikzcd}
& x \arrow[dl, "u"'] \arrow[dr, "v"] \arrow[dd, phantom, "\Downarrow \mathsf{LC}(u^{-1}\cdot v)"] & \\
y \arrow[dr, "*"'] & & z \arrow[dl, "*"] \\
& x' &
\end{tikzcd}$$

$$\begin{tikzcd}
& x \arrow[dl, "u"', "*"] \arrow[dr, "v", "*"'] \arrow[dd, phantom, "\Downarrow \mathsf{C}(u^{-1}\cdot v)"] & \\
y \arrow[dr, "*"'] & & z \arrow[dl, "*"] \\
& x' &
\end{tikzcd}
\qquad
\begin{tikzcd}
y \arrow[rr, leftrightsquigarrow, "u", "*"'] \arrow[dr, "*"'] \arrow[rr, phantom, bend right=40, "\Downarrow \mathsf{CR}(u)"'] & & z \arrow[dl, "*"] \\
& x' &
\end{tikzcd}$$

Figure 1. Different notions of confluence.

The first property we want to take a glance at is the arguably weakest notion of confluence we want to consider. It requires that each local peak can be rewritten into a reduction zig-zag below that peak. It was originally formulated as a property of a rewriting system by Winkler and Buchberger (Buchberger and Winkler 1983).

Definition 7. A *Winkler-Buchberger structure* on a 2-polygraph with an order $>$ on its objects consists of, for each local peak, $u \in (y \leftsquigarrow x \rightsquigarrow z)$, a rewrite zig-zag $\mathsf{WB}(u) \in (\Sigma_2 \uplus \Sigma_2^{-1})^*$ such that $s_1(\mathsf{WB}(u)) = u$ and, writing $t_1(\mathsf{WB}(u)) = (y \leftrightsquigarrow x_1 \ldots x_n \leftrightsquigarrow z)$, we have $x > x_i$ for all $1 \leq i \leq n$.

Just like we represent the Winkler-Buchberger property by a structure on the polygraph, we will proceed to present (local) confluence as a choice of rewrite zig-zags:

Definition 8. A *local confluence structure* on the assumed 2-polygraph consists of, for each local peak $u \in (y \leftsquigarrow x \rightsquigarrow z)$, a rewrite zig-zag $\mathsf{LC}(u) \in (\Sigma_2 \uplus \Sigma_2^{-1})^*$ such that $s_1(\mathsf{LC}(u)) = u$ and $t_1(\mathsf{LC}(u))$ takes the form $(y \rightsquigarrow^* x' \; {}^*\!\!\leftsquigarrow z)$ for some $x' \in \Sigma_0$ which we will call the *reduct* of the local peak u.

A *confluence structure* is defined analogously, with a rewrite zig-zag $\mathsf{C}(u)$ for an arbitrary (not necessarily local) peak $u \in (y \; {}^*\!\!\leftsquigarrow x \rightsquigarrow^* z)$.

While sometimes conflated with confluence, we will call *Church-Rosser structure* the generalisation of confluence where objects are connected by an arbitrary reduction zig-zag.

Definition 9. A choice of rewrite zig-zag $\mathsf{CR}(u)$ for every $u \in (y \leftrightsquigarrow^* z)$ is called a *Church-Rosser structure* if $s_1(\mathsf{CR}(u)) = u$ and $t_1(\mathsf{CR}(u)) \in (y \rightsquigarrow^* x' \; {}^*\!\!\leftsquigarrow z)$, for some $x' \in \Sigma_0$. Like with confluence, we will call x' the reduct of u.

A Church-Rosser structure always contains a confluence structure (since every peak is a zig-zag). Similarly, a confluence structure contains a local confluence structure. If $u \in (x \rightsquigarrow y)$ implies $f(x) > f(y)$ (and $>$ is transitive), which happens in particular if the generalised 2-polygraph is terminating, then every local confluence structure trivially is a Winkler-Buchberger structure as well.

We will now show that, if a 2-polygraph is terminating and closed under congruence (see Section 2.4), then these implications can be reversed. In its core, the proof is simply the standard argument for Newman's Lemma (Huet 1980).

Lemma 10. *Assume we have a generalised 2-polygraph that is closed under congruence and is terminating, with underlying transitive relation $>$. Then, a Winkler-Buchberger structure* WB *allows us to construct a Church-Rosser structure* CR.

Proof. Let $y, z \in \Sigma_0$ be given. The goal is to construct, for any reduction zig-zag $u \in (y \leftrightsquigarrow^* z)$, a sequence $\mathsf{CR}(u) \in (\Sigma_2 \uplus \Sigma_2^{-1})^*$ such that $s_1(\mathsf{CR}(u)) = u$ and such that $t_1(\mathsf{CR}(u))$ is a valley.

The list extension of $>$ gives an order $>_L$ on $\Sigma_0,^*$ and, by taking the underlying object sequence of a zig-zag, this induces a relation on $(y \leftrightsquigarrow^* z)$ which, by Lemma 4, is Noetherian. We show the goal by Noetherian induction on this relation.

A given $u \in (y \leftrightsquigarrow^* z)$ is either of the form $(y \rightsquigarrow^* x' \,{}^*\!\leftsquigarrow z)$, in which case we are done ($\mathsf{CR}(u)$ is the empty sequence), or it contains a local peak and can be written as $u = v \cdot u' \cdot w$ with $v \in (y \leftrightsquigarrow^* y')$, $u' \in (y' \leftsquigarrow x \rightarrow z')$, $w \in (z' \leftrightsquigarrow^* z)$. The Winkler-Buchberger structure and the closure under congruence gives us a rewrite step

$$\left(v \cdot \mathsf{WB}(u') \cdot w\right) \in \left((v \cdot u' \cdot w) \Rightarrow (v \cdot t_1(\mathsf{WB}(u')) \cdot w)\right). \tag{14}$$

By construction of the list extension and by the condition on the Winkler-Buchberger structure, the induction hypothesis lets us assume that we already have a suitable $\mathsf{CR}(v \cdot t_1(\mathsf{WB}(u')) \cdot w) \in (\Sigma_2 \uplus \Sigma_2^{-1})^*$. Concatenating (14) with that rewrite zig-zag gives $\mathsf{CR}(u)$. □

2.6 A homotopy basis

The goal of this section is to show that, with the help of a few assumptions, we can construct a *homotopy basis* for a generalised 2-polygraph.

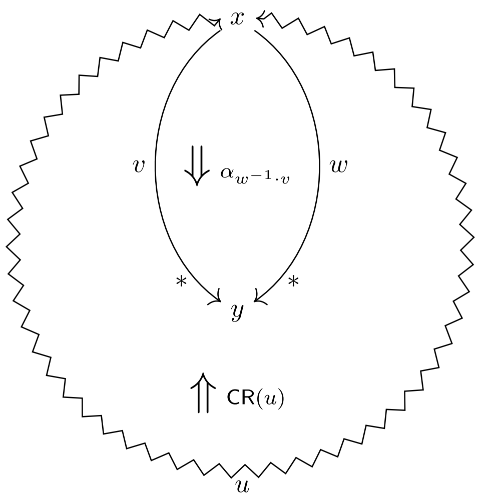
How does the concept of homotopy come into play here? Imagine a topological realisation of the 2-polygraph where objects are represented by points, reduction steps by the interval space, and rewrite steps by surfaces between the zig-zags corresponding to their source and target. Then, we can consider the fundamental group of this topological space. The fundamental group is trivial (at any point) if, for reduction zig-zags $u, v \in (x \leftrightsquigarrow^* y)$, there is always a 'filler' $\alpha \in (u \overset{*}{\Leftrightarrow} v)$. A subset of Σ_2 which can be used to 'fill' every loop is a homotopy basis:

Definition 11 (homotopy basis)**.** Let a generalised 2-polygraph $\Sigma = (\Sigma_0, \Sigma_1, \Sigma_2)$ be given. A *homotopy basis* consists of, for all $x, y \in \Sigma_0$ and $u, v \in (x \leftrightsquigarrow^* y)$, a rewrite zig-zag $\alpha_{u,v} \in (u \overset{*}{\Leftrightarrow} v)$.

Remark 12 (alternative definition of homotopy basis)**.** Instead of calling the set of all $\alpha_{u,v}$ a homotopy basis, it may seem more natural to refer to a subset $\mathcal{B} \subseteq \Sigma_2$ as a homotopy basis if one can construct all the $\alpha_{u,v}$ from rewrite steps in $\mathcal{B}$ (or their closure under congruence). That variation would in particular allow the formulation of properties or statements with respect to the cardinality of a basis. However, we are only interested in the collection of all the $\alpha_{u,v}$ itself, which is why we use the simplified Definition 11.

An additional but intuitive property that we need it the following:

Definition 13. We say that a generalised 2-polygraph $(\Sigma_0, \Sigma_1, \Sigma_2)$ *cancels inverses* if, for any reduction step $s \in (x \rightsquigarrow y)$, we have rewrite zig-zags $\mathsf{RINV}(s) \in (s \cdot s^{-1} \overset{*}{\Leftrightarrow} \varepsilon_x)$ and $\mathsf{LINV}(s) \in (s^{-1} \cdot s \overset{*}{\Leftrightarrow} \varepsilon_y)$.



x
v
$\Downarrow \alpha_{w^{-1}\cdot v}$
w
$*$
$*$
y
$\Uparrow \mathsf{CR}(u)$
u

Figure 2. The induction step for the construction of the homotopy basis.

Theorem 14. *Let* $(\Sigma_0, \Sigma_1, \Sigma_2)$ *be a terminating generalised 2-polygraph which is closed under congruence and which cancels inverses. If it has a Winkler-Buchberger structure* WB*, then it has a homotopy basis.*

Proof. By Lemma 10, we can construct a Church-Rosser structure CR.

Moreover, using RINV and LINV together with closure under congruence we can, by straightforward induction on the length of a zig-zag $u \in (y \leftrightsquigarrow^* z)$, construct a sequence $\mathsf{INV}(u) \in (u \cdot u^{-1} \Rightarrow \varepsilon_y)$, where ε_y is the empty zig-zag. Given $u, v \in (y \leftrightsquigarrow^* z)$ and a rewrite zig-zag $\alpha \in (u \cdot v^{-1} \overset{*}{\Leftrightarrow} \varepsilon_y)$, closure under congruence shows that we also have

$$\begin{aligned} u &\overset{*}{\Leftrightarrow} u \cdot v^{-1} \cdot v && \text{by } \mathsf{INV}(v^{-1}), && (15)\\ &\overset{*}{\Leftrightarrow} v && \text{by } \alpha && (16)\end{aligned}$$

By the above argument, it suffices to show the goal for the case that u is a *closed* reduction zig-zag ($u \in (y \leftrightsquigarrow^* y)$) and v is empty ($v = \varepsilon_y$). By Noetherian induction on the object y, we show the following statement:

$$P(y) := \text{'For all closed reduction zig-zags } u \in (y \leftrightsquigarrow^* y)\text{, we have } \alpha_u \in (u \overset{*}{\Leftrightarrow} \varepsilon_y).\text{''}$$

To this end, we assume $P(z)$ for all $y > z$ and take an arbitrary reduction zig-zag $u \in (y \leftrightsquigarrow^* y)$ for which we want to construct $\alpha_u \in (u \overset{*}{\Leftrightarrow} \varepsilon_y)$. Now consider the rewrite zig-zag $\mathsf{CR}(u) \in (u \overset{*}{\Leftrightarrow} v \cdot w^{-1})$, where $v, w \in (y \rightsquigarrow^* z)$ for the reduct z of $\mathsf{CR}(u)$. If either of v or w is an empty sequence, then so is the other and we have $y = z$, in which case we set $\alpha_u := \mathsf{CR}(u)$. Otherwise, we have $y > z$. In this case, we consider the closed reduction zig-zag $w^{-1} \cdot v \in (z \leftrightsquigarrow^* z)$.

From the induction hypothesis, we obtain a rewrite zig-zag $\alpha_{w^{-1}\cdot v} \in (w^{-1} \cdot v \overset{*}{\Leftrightarrow} \varepsilon_z)$. We can now use this rewrite zig-zag to construct the following chain of rewrites, again heavily relying on the closure under congruence; see Figure 2 for an illustration:

$$\begin{aligned} u &\overset{*}{\Leftrightarrow} v \cdot w^{-1} && \text{by } \mathsf{CR}(u), && (17)\\ &\overset{*}{\Leftrightarrow} v \cdot w^{-1} \cdot v \cdot v^{-1} && \text{by } \mathsf{INV}(v), && (18)\\ &\overset{*}{\Leftrightarrow} v \cdot v^{-1} && \text{by } v \cdot \alpha_{w^{-1}\cdot v} \cdot v^{-1}, && (19)\\ &\overset{*}{\Leftrightarrow} \varepsilon_y && \text{by } \mathsf{INV}(v), && (20)\end{aligned}$$

which completes the construction. □

3. A Translation to Type Theory: Caveats and Generalisations

While the above constructions are formulated in an (unspecified) standard set-theoretic framework, the main motivation for the development is applications in homotopy type theory. Large parts of the translation of Section 2 into homotopy type theory follow standard strategies and are to some degree mechanical, but some points require further attention. Moreover, our type-theoretic construction is, due to the choices we make, more general than the set-theoretic development presented above. In this (short) section, we discuss these key points. A reader working in homotopy type theory will likely find these explanations sufficient to understand how the type-theoretic formulation works and can then jump to Section 5, while someone interested in all details may wish to skip the current section and immediately go to Section 4.

3.1 Constructivity

When translating a development into (constructive) type theory, the natural first consideration is whether any possibly implicit use of the law of excluded middle, and its consequences, especially proof by double negation and decidability of equality, can be avoided. This is the case here but, as it is often happens, relies on formulating the definitions correctly; an obvious example is the definition of *wellfounded* (and *Noetherian*), where the classical negative phrasing ‘there is no infinite sequence’ would not allow a constructive argument, while a well-known inductive definition works (cf. Section 4.2).

3.2 Coherence

More specific to the setting of homotopy type theory is the phenomenon of higher equalities. In many cases, translations of standard mathematical concepts into homotopy type theory are phrased using sets (i.e. types satisfying UIP, see equation (2) on p. 984) in order to faithfully represent the corresponding theory, and formulating the same concepts for arbitrary types can be extremely difficult or impossible.

We choose to not restrict ourselves to sets for the mere purpose of being more general (although the set-case would suffice for the applications in Section 5). This is not particularly difficult, but a possibly surprising consequence is that there may be rewrite zig-zags $u : x \overset{*}{\Leftrightarrow} x$ of length zero which are *not* the trivial sequence ε_x. To be precise, the type of zig-zags of length zero from x to y is equivalent to the equality type $x = y$. The analogue to Theorem 14 thus needs to include the assumption that the rewrite system *cancels empty sequences*, a condition which becomes trivial for sets (cf. Theorem 40).

Similarly, the faithful translation of the relations of Section 2 would be to consider families of *propositions*, that is, types with at most one inhabitant. Again, we aim for greater generality and avoid this assumption, which however has little consequences for the overall argument. There are several further points where the type-theoretic formulation is, strictly speaking, a generalisation of the results of Section 2.

3.3 Formulation of results

As we will discuss in Section 4.4 below, the construction of a homotopy basis is reminiscent of an induction principle, and it is natural from a type-theoretic point of view to phrase it as such. We will derive the following principle:

Theorem (simplified formulation of Theorem 31). *Let A be a type with a binary relation $\rightsquigarrow$ that is Noetherian and locally confluent, and let P be a type family indexed over closed zig-zags. To prove (inhabit) P for all closed zig-zags, it suffices to prove P for empty zig-zags and for the diamonds that* *come from the local confluence property and to check that P is closed under standard groupoidal constructions.*

4. A Homotopy Basis in Homotopy Type Theory

This section serves as a high-level description of our Agda formalisation, which in turn presents the type-theoretic development in full detail. The main mathematical ideas which make the arguments work correspond to those presented in Section 2, and, keeping the caveats and remarks of Section 3 in mind, the type-theoretic formulation follows standard strategies. The expert reader may wish to immediately jump to Section 5.

We use Section 4.1 to specify the type theory that we work in. For any such translation to type theory, the specific choices one makes determine whether the procedure is straightforward or includes (mathematical) challenges. We explain in Section 3 which choices we make and how they make the type-theoretic statement a generalisation of the results in Section 2. The concrete step-by-step translation is split into two subsections: In Section 4.2, we examine how closures of binary relations behave in homotopy type theory, and in Section 4.3, we discuss the construction of a homotopy basis of a generalised 2-polygraph. We then show in Section 4.4 how this work allows us to derive an induction-like statement similar to the one proved in our previous conference paper (Kraus and von Raumer 2020).

As explained in Section 1.5 above, we have formalised this part of the paper in Agda We link the main constructions and proofs to the `html` version of the formalisation.

4.1 Homotopy type theory as the setting

The type theory we work in is homotopy type theory as developed in the book (The Univalent Foundations Program 2013). However, for the translation of the results of Section 2 itself, we do not rely on any features specific to homotopy type theory: All of what we do here (i.e. in Section 4) works in intensional Martin-Löf type theory with function extensionality. Only for the applications that we discuss in Section 5, we rely on having set-quotients and a univalent universe.

Regarding notation, we mostly follow the book (The Univalent Foundations Program 2013). For *judgmental* (also known as *definitional*) equality between expressions, we use the symbol $\equiv$, and for definitions, we write $:\equiv$.

In detail, the type theory that we consider features the following components:

- We assume that the type theory has a (Russell-style) *universe* $\mathcal{U}$. It is standard to assume a hierarchy
$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \ldots$$
of universes, and in such a theory, $\mathcal{U}$ may denote any universe $\mathcal{U}_i$ (i.e. our constructions are *universe polymorphic*).
- Besides types of non-dependent functions, we require for any type $A : \mathcal{U}$ and for any *type family* $B : A \to \mathcal{U}$ the type $\Pi(a : A).B(a)$ of *dependent functions*.
- For any type $A : \mathcal{U}$ with elements $a, b : A$, we have the Martin-Löf *equality type* which, following The Univalent Foundations Program (2013), we denote by $(a = b) : \mathcal{U}$. The equality type is generated inductively on a witness for its reflexivity, which means that we have $\mathsf{refl}_a : (a = a)$ for each $a : A$ (we will sometimes omit the subscript). Its induction principle, called the *J-rule*, states that to produce an element of the type $\Pi(b : A).\Pi(p : a = b).C(b, p)$ it suffices to provide an element of $C(a, \mathsf{refl}_a)$.
Caveat: In Agda, the roles of $=$ and $\equiv$ are reversed. Definitions are written using $=$, while the equality type is written using $\equiv$.

- For $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, we will need the type of dependent pairs, written $\Sigma(a : A).B(a)$, with the non-dependent version (for $C : \mathcal{U}$) written $A \times C$.
- We require the disjoint sum $A \uplus C$ of types, with obvious functions $\mathsf{inl} : A \to A \uplus C$ and $\mathsf{inr} : C \to A \uplus C$.
- We assume that the type theory has inductive types and inductive families, examples for which we will see in Section 4.2.
- Finally, for Π-types, we assume function extensionality which, in its simplest form, says that for two functions $f, g : \Pi(a : A).B(a)$ we have $f = g$ as soon as they are pointwise equal: $\Pi(a : A).f(a) = g(a)$.[5]

To improve readability and usability, many proof assistants implement implicit arguments. We use them in writing as well, as a purely notational device, as we write implicit arguments in curly brackets $\{\}$. For example, $\Pi\{a : A\}.B(a) \to C$ denotes the same type as $\Pi(x : A).B(a) \to C$. For $a : A$ and $b : B(a)$, if we have $f : \Pi\{a : A\}.B(a) \to C$ and $g : \Pi(a : A).B(a) \to C$, we implicitly uncurry and write $g(a, b) : C$ but can omit the implicit argument in the other case and write $f(b) : C$. Instead of $\Pi(a : A).\Pi(b : B(a)).C(a, b)$, we write $\Pi(a : A), (b : B(a)).C(a, b)$.

The inductive definition of equality induces the structure of a (higher) groupoid on types, which forms the basis of the synthetic kind of topology performed in homotopy type theory. We will not introduce the corresponding terminology in full, but we will in the following go through the notions we will use in this paper.

Besides transitivity and symmetry, equality has the property that any function becomes a functor with respect to the induced groupoids: Given $f : A \to B$, $a, a' : A$ and an equality $p : a = a'$, we have $\mathsf{ap}_f(p) : f(a) = f(a')$. Also, equal elements are indiscernible in the sense that if $B : A \to \mathcal{U}$ is a type family over A and we have $a, a' : A$ with $p : a = a'$, we have the implication $p_* : B(a) \to B(a')$, the so called *transport* along p.

Taking iterated equalities (i.e. equalities on equality types), we discover a type's *higher structure*. The number of iterations of equality types we have to take after which equalities do not carry information is called a types *h-level*. In particular, a type $A : \mathcal{U}$ in which we have

- $\Pi(x, y : A).x = y$ is called a *proposition* or *(-1)-type*,
- $\Pi(x, y : A)(p, q : x = y).p = q$ is called an *(h-)set* or *0-type*,
- $\Pi(x, y : A)(p, q : x = y)(\alpha, \beta : p = q).\alpha = \beta$ is called a *1-type*, and so on.

Finally, another notion which makes an appearance in this section paper is the one of *equivalent types*. We denote with $A \simeq B$ the type of *equivalences* between A and B, that is, functions $f : A \to B$ which have a two-sided inverse $g : B \to A$ such that for each $x : A$ the two ways of proving the equality $f(g(f(x))) = x$ (by cancelling $g \circ f$ and by cancelling $f \circ g$) coincide.

We refer to The Univalent Foundations Program (2013) for the details of the concepts discussed above.

4.2 *Properties and closures of binary relations*

(files `graphclosures`, accessibility, and listextension). By a *binary relation* on a type A we mean, in the type-theoretic setting, simply a type family $R : A \to A \to \mathcal{U}$. This is sometimes called a *proof-relevant* binary relation since $R\ a\ b$ can potentially have many (different) inhabitants.

Thus, a *1-polygraph* in type theory is simply a type $A : \mathcal{U}$ together with a binary relation $(\_ \rightsquigarrow \_) : A \to A \to \mathcal{U}$, where the blanks reserve the places for arguments; that is, we write $(x \rightsquigarrow y) : \mathcal{U}$, mirroring the notation used in Section 2. As before, we will write

- $\rightsquigarrow^*$ for the reflexive-transitive closure,
- $\leftrightsquigarrow$ for the symmetric closure,
- $\leftrightsquigarrow^*$ for the symmetric-reflexive-transitive closure

of the relation $\rightsquigarrow$. The type-theoretic implementation of these closures is standard. The reflexive-transitive closure is constructed as an inductive family with two constructors:

$$\begin{aligned}
&\text{inductive } (\_ \rightsquigarrow^* \_) : A \to A \to \mathcal{U} \text{ where}\\
&\quad \mathsf{nil} : \Pi(x:A).\,(x \rightsquigarrow^* x)\\
&\quad \mathsf{snoc} : \Pi\{x,y,z:A\}.\,(x \rightsquigarrow^* y) \to (y \rightsquigarrow z) \to (x \rightsquigarrow^* z)
\end{aligned} \tag{21}$$

The symmetric closure is the obvious disjoint sum,

$$(x \leftrightsquigarrow y) :\equiv (x \rightsquigarrow y) \uplus (y \rightsquigarrow x). \tag{22}$$

The symmetric-reflexive-transitive closure $\leftrightsquigarrow^*$ is constructed by first taking the symmetric and then the reflexive-transitive closure. The transitive closure $\rightsquigarrow^+$ is defined analogously to $\rightsquigarrow^*$, but with nil replaced by a constructor taking a single step (cf. TransReflClosure, SymClosure, TransClosure in the formalisation).

Since the relation $\rightsquigarrow$ is proof-relevant, the same is the case for its closures. As a consequence, we have functions $(x \rightsquigarrow^* y) \to (x \rightsquigarrow^{**} y)$ and $(x \rightsquigarrow^{**} y) \to (x \rightsquigarrow^* y)$, but these are in general not inverse to each other. The analogous caveat holds for the other closure operations. However, this observation has no consequences for our constructions and proofs.

Mirroring the earlier terminology, we call an element of $x \rightsquigarrow^* y$ a *sequence*, and an inhabitant of $x \leftrightsquigarrow^* y$ a *zig-zag*. Let us write ε_x instead of $\mathsf{nil}\, x$ for the trivial sequence at point x. Given two sequences (or zig-zags) $u : x \rightsquigarrow^* y$ and $v : y \rightsquigarrow^* z$, the definition of their concatenation (by induction on u) is standard. Adopting the notation in Section 2, we write $u \cdot v$ for this concatenation, no matter whether u, v are single steps, sequences, or elements of the symmetric closure. It is standard that the operation $\cdot$ is associative. Moreover, $u : x \leftrightsquigarrow^* y$ can be inverted by inverting every single step, and we denote this operation by $u^{-1} : y \leftrightsquigarrow^* x$. Inversion and concatenation interact in the obvious way. Note that $x \rightsquigarrow^* y$ embeds into $x \leftrightsquigarrow^* y$ as a *positive* zig-zag while $y \rightsquigarrow^* x$ embeds into $x \leftrightsquigarrow^* y$ as a *negative* zig-zag, this giving us two distinct embeddings in the case of a closed zig-zag $x \leftrightsquigarrow^* x$.

Since it constitutes a real difference between the type-theoretic development and the set-theoretic on in Section 2, we make the following definition and statements explicit:

Definition 15 (length of sequences or zig-zags; cf. length[t])**.** There is an obvious function

$$\mathsf{length} : \Pi\{x,y:A\}.\,(x \rightsquigarrow^* y) \to \mathbb{N} \tag{23}$$

which calculates the length of a sequence. Starting with $\leftrightsquigarrow$ instead of $\rightsquigarrow$, it calculates the length of a zig-zag. We say that a sequence (zig-zag) α is *empty* if its length is zero, and write

$$(x \overset{0}{\rightsquigarrow} x) \;:\equiv\; \Sigma(u : x \rightsquigarrow^* x).\mathsf{length}(u) = 0 \tag{24}$$

$$(x \overset{0}{\leftrightsquigarrow} x) \;:\equiv\; \Sigma(u : x \leftrightsquigarrow^* x).\mathsf{length}(u) = 0. \tag{25}$$

The trivial closed sequence (or zig-zag) $\varepsilon_x : (x \rightsquigarrow^* x)$ is, of course, empty, but in the case where A is not a set but a higher type, not every empty closed sequence (zig-zag) is equal to ε_x. Instead, an empty closed sequence (zig-zag) at point x corresponds to a path of type $(x = x)$ in A. Since (21) without the second constructor is the usual definition of Martin-Löf's identity type as an inductive family, it is easy to see the following:

Lemma 16. *For any point $x : A$, the three types $(x \rightsquigarrow^0 x)$ and $(x \leftrightsquigarrow^0 x)$ and $(x = x)$ are equivalent.* □

Corollary 17. *If A is a set, then the trivial sequence (or zig-zag) ε_x is the only empty sequence (or zig-zag) at point x.* □

Given a zig-zag $u : x \leftrightsquigarrow^* y$, we can ask whether it is *strictly increasing (decreasing)*, that is, whether each step in the zig-zag comes from the left (right) summand in (22), that is, is of the form $\mathsf{inl}(t)$ with $t : x \rightsquigarrow y$ ($\mathsf{inr}(t)$ with $t : y \rightsquigarrow x$). The terminology is then the obvious one, copying the one from the set-theoretic development: We call a zig-zag a *peak* if it is the concatenation of an increasing and a decreasing zig-zag, a *valley* if it is the concatenation of a decreasing and an increasing zig-zag, and so on.

The next concept which has an interesting equivalent in type theory is the one of wellfoundedness or Noetherianness. As remarked before, the usual (classical) formulation of the form 'no infinite sequence exists' is unsuitable in a constructive setting. The inductive characterisation which we use instead is well-known in type theory and was studied by several authors including Huet (1980). We also refer to Aczel and Rathjen (2001). Given a binary relation $<$ on a type A, one first defines the notion of accessibility as an inductive type family:

Definition 18 ($\Phi_<$ in Aczel 1977; Acc in the cubical library)**.** The family $\mathsf{acc}^< : A \to \mathcal{U}$ is generated inductively by a single constructor,

$$\mathsf{step} : \Pi(a : A).\,(\Pi(x : A).\,(x < a) \to \mathsf{acc}^<(x)) \to \mathsf{acc}^<(a), \tag{26}$$

that is, an element a is *accessible* ($\mathsf{acc}\, a$) if every element smaller than it is. The relation $<$ is *wellfounded* if every element is accessible,

$$\mathsf{isWellFounded}(\,<\,) :\equiv \Pi(a : A).\,\mathsf{acc}^<(a). \tag{27}$$

If $<$ is wellfounded, then $>$, where $(x > y) :\equiv (y < x)$, is called *Noetherian*.

While the definition in The Univalent Foundations Program (2013, Chp. 10.3) is only given for the special case that A is a set and $<$ is valued in propositions, the more general case that we consider works in exactly the same way (cf. our formalisation). In particular, we have the following two results:

Lemma 19 (cf. isPropAcc in the cubical library)**.** *For any x, the type $\mathsf{acc}^<(x)$ is a proposition. Further, the statement that $<$ is wellfounded is a proposition.* □

Proving properties of wellfounded relations is made possible by the following principle:

Lemma 20 (accessibility induction The Univalent Foundations Program 2013, Chp. 10.3; acc-ind)**.** *Assume we are given a family $P : A \to \mathcal{U}$ such that we have*

$$\Pi(a_0 : A).\,\mathsf{acc}^<(a_0) \to (\Pi(a < a_0).\,P(a)) \to P(a_0). \tag{28}$$

In this case, we get:

$$\Pi(a_0 : A).\,\mathsf{acc}^<(a_0) \to P(a_0). \tag{29}$$

If $(\,<\,)$ is wellfounded, the argument $\mathsf{acc}^<(a_0)$ can be omitted and the principle is known as wellfounded induction.

An easy application which demonstrates this induction principle is the following:

Lemma 21. *Let us write $<^+$ for the transitive closure of $<$. If $a : A$ is $<$-accessible, then it is $<^+$-accessible.*

Proof. By $<$-accessibility induction on $P(a) :\equiv (\mathsf{acc}^{<}(a) \to \mathsf{acc}^{<^{+}}(a)$. □

Corollary 22 (of Lemma 21; cf. transitive-wellfounded)**.** *If $<$ is wellfounded, then so is $<^{+}$.* □

Nested induction takes the following form:

Lemma 23 (nested accessibility/wellfounded induction, cf. double-acc-ind)**.** *Assume we are given a relation $<_1$ on a type B, a relation $<_2$ on a type C, and a family $P : B \times C \to \mathcal{U}$. Assume further that we are given*

$$\begin{aligned}\Pi(b : B), (c : C).\ &\mathsf{acc}^{<_1}(b) \to \mathsf{acc}^{<_2}(c) \to \\ &(\Pi(b' <_1 b).\ P(b', c)) \to (\Pi(c' <_2 c).\ P(b, c')) \to \\ &P(b, c).\end{aligned} \tag{30}$$

Then, we get:

$$\Pi(b : B), (c : C).\ \mathsf{acc}^{<_1}(b) \to \mathsf{acc}^{<_2}(c) \to P(b, c). \tag{31}$$

Proof. We carefully apply accessibility induction with the correct motive on the witnesses of both $\mathsf{acc}^{<_1}(b)$ and $\mathsf{acc}^{<_2}(c)$, then apply (30). □

Lemma 23 is needed for the type-theoretic proof of the property (1) in the proof of Lemma 4. The other parts are identical. This means, again, that the list extension of a wellfounded relation is wellfounded. The Agda formalisation of this fact can be found in the file listextension, with the main theorem of that module being isWF⟨>⟩ ⇒isWF⟨>$^{\mathrm{L}}$⟩.

Of course, all statements about wellfounded relations dualise in the obvious way to Noetherian relations; in particular, the list extension of a Noetherian relation is Noetherian. However, proving this in Agda requires a certain amount of work. For example, we show that a monotone function between types with orders reflects accessibility (cf. acc-reflected) and that reversing a list commutes (in a weak sense) with taking its transitive closure (cf. revClosureComm). Only then, we are able to draw the seemingly obvious conclusion from Corollary 22 that the transitive closure of a Noetherian relation is Noetherian (cf. transitive-Noetherian).

We have already seen an example of a Noetherian relation in the introduction of this paper, namely the relation (5) that is used in the construction of the free group in Example 1: Given two lists $\ell_1, \ell_2 : \mathsf{List}(M \uplus M)$, we have $\ell_1 \rightsquigarrow \ell_2$ if the first list can be transformed into the second list by removing exactly two elements. The two removed list elements have to be consecutive and 'inverse' to each other, that is, one is of the form $\mathsf{inl}(a)$, the other $\mathsf{inr}(a)$. It is clear that this relation is Noetherian, since each step reduces the length of the list.

Lemma 24 (free groups, continuing Example 1)**.** *The relation $\rightsquigarrow$ on lists defined by* (5) *is Noetherian.* □

4.3 2-polygraphs in homotopy type theory

(files polygraphs, cancelInverses, newman, homotopybasis)**.** With these necessary notions defined, we can now translate generalised 2-polygraphs. Here, it is important that we will not view reduction steps and rewrite steps as plain types but always as parameterised by their source and target, see Remark 6.

Definition 25 (2-polygraph, cf. 2-polygraph)**.** A generalised 2-polygraph is a triple $\Sigma \equiv (A, \rightsquigarrow, \Rightarrow)$ consisting of data of the following types:

- A type $A : \mathcal{U}$ of objects of the 2-polygraph,
- a family $(\rightsquigarrow) : A \to A \to \mathcal{U}$ of reduction steps, and
- a family $(\Rightarrow) : \Pi\{x, y : A\}.\, (x \leftrightsquigarrow^* y) \to (x \leftrightsquigarrow^* y) \to \mathcal{U}$ of rewrite steps of Σ.

We might denote with Σ_1 and Σ_2 the total spaces of the type families of reduction steps and rewrite steps, respectively, in order to recover the content of Section 2.4. Transferring the properties of 2-polygraphs introduced before into the realm of type theory is straightforward. Note that none of these definitions are necessarily propositional, so they carry *data* instead of mere proofs.

- The 1-polygraph (Σ_0, Σ_1) is called *terminating* (cf. isNoetherian) if it comes with transitive Noetherian relation $(>)$ on A and a function
$$(x \rightsquigarrow y) \to (x > y). \tag{32}$$
- Σ is called *closed under congruence* (cf. ⇔*isCongrClosed) if for $u : (x \leftrightsquigarrow^* y)$, $v : (y' \leftrightsquigarrow^* z)$, and $\alpha : (w \overset{*}{\Leftrightarrow} w')$ for some $w, w' : (y \leftrightsquigarrow^* y')$ we have
$$u \cdot \alpha \cdot v : (u \cdot w \cdot v \overset{*}{\Leftrightarrow} u \cdot w' \cdot v). \tag{33}$$
- Σ has a *Winkler-Buchberger structure* (cf. hasWB) if, for each local peak $u : (y \leftsquigarrow x \rightsquigarrow z)$, we have a $u' : (y \leftrightsquigarrow^* z)$ such that $x > x_i$ for any object occurring in the inner part of u', together with an element $\mathsf{WB}(u) : (u \overset{*}{\Leftrightarrow} u')$.
- Σ *cancels inverses* (cf. cancels*RInv) if we supply a function
$$\mathsf{LINV} : \Pi\{x, y : A\}, (u : x \leftrightsquigarrow^* y).\, (u \cdot u^{-1} \overset{*}{\Rightarrow} \varepsilon_x). \tag{34}$$

Remark 26. An important special case, sufficient for the applications in Section 5, is the case where $\rightsquigarrow$ is Noetherian and $>$ can simply be chosen to be the transitive closure of $\rightsquigarrow$ (cf. Theorem 31). Although we do not have an example where it happens, it is plausible that there may be cases where one should take $>$ to be the propositional truncation of $\rightsquigarrow^+$ instead: Everything will work in exactly the same way, but the requirement on the Winkler-Buchberger structure will be weaker.

The other notions of confluence structures can be defined in a straightforward way. By the same construction as in Section 2, a Winkler-Buchberger structure induces a Church-Rosser structure on the 2-polygraph:

Lemma 27 (translation of Lemma 10, cf. module wb2cr)**.** *From a terminating generalised 2-polygraph which is closed under congruence and has a Winkler-Buchberger structure* WB*, we can construct for each* $u : (y \leftrightsquigarrow^* z)$ *a valley* $u' : (y \rightsquigarrow^* x' \,{}^*\!\!\leftsquigarrow z)$ *and a rewrite zig-zag* $\mathsf{CR}(u) : (u \overset{*}{\Leftrightarrow} u')$*. Again, we will call* x' *the* reduct *of* $\mathsf{CR}(u)$*.* □

A homotopy basis in type theory becomes just a single inhabitant of a Π-type:

Definition 28 (cf. hasHomotopyBasis)**.** A *homotopy basis* of Σ is a function
$$\alpha : \Pi\{x\, y : A\}, (u\, v : (x \leftrightsquigarrow^* y)).(u \overset{*}{\Leftrightarrow} v). \tag{35}$$

When collecting all the structures, we need on our 2-polygraph to construct a homotopy basis, the list of assumptions in Theorem 14 is insufficient for the reason explained in Section 3 and made precise in Lemma 16. Recall that, in the proof of Theorem 14, we had to consider the case that u is a closed zig-zag at point y with $s_1(\mathsf{CR}(u))$ an empty zig-zag. In contrast to before, this case is not trivial in the type-theoretic setting with higher equalities. What we need it the following property:

Definition 29 (cf. cancelsEmpty)**.** A 2-polygraph is said to *cancel empty closed zig-zags* if we have a function

$$e : \Pi\{x : A\}, (u : (x \leftrightsquigarrow^* x)).(\mathsf{length}(u) = 0) \to (u \overset{*}{\Leftrightarrow} \varepsilon_x).$$

By Corollary 17, any 2-polygraph Σ, where the type of objects A is a set, cancels empty zig-zags trivially.

With the above definitions at hand, the translation of the construction of the homotopy basis is straightforward:

Theorem 30 (translation of Theorem 14, cf. Noeth×WB×Congr×Cancel⇒Basis)**.** *Let Σ be a 2-polygraph which is terminating, closed under congruence, cancels inverses, has a Winkler-Buchberger structure, and cancels empty closed zig-zags. Then, Σ has a homotopy basis.*

Proof. By and large, the proof proceeds the same way as the one presented for Theorem 14: We apply Noetherian induction (cf. Lemma 20) on the type A with

$$P(x) :\equiv \Pi(u : x \leftrightsquigarrow^* x).(u \overset{*}{\Leftrightarrow} \varepsilon_x).$$

As before, we fix $x : A$ and $u : (x \leftrightsquigarrow^* x)$ and may assume that we are given $\alpha_v := P(y)(v)$ for all $x > y$ and $v : (y \leftrightsquigarrow^* y)$. We again consider $\mathsf{CR}(u) : (u \overset{*}{\Rightarrow} v \cdot w^{-1})$ with $v, w : (x \rightsquigarrow^* y)$ for the reduct y of $\mathsf{CR}(u)$. If v or w (and thus also the other) have length zero, then there is $e_v : (v \overset{*}{\Leftrightarrow} \varepsilon_y)$, allowing us to construct the rewrite chain $u \overset{*}{\Rightarrow} v \cdot w^{-1} \overset{*}{\Leftrightarrow} \varepsilon_x$.

In the case where either v or w have non-zero length, we again can conclude that $x > y$ and that the induction hypothesis can be applied as in the set-theoretic formulation. □

4.4 Noetherian induction for closed zig-zags

(file noethercycle)**.** While so far we considered the rewrite zig-zags of a 2-polygraph mainly as additional data to a 1-polygraph, we can also think of them as a witness of the fact that certain properties of reduction zig-zags carry over to other, rewritten, reduction zig-zags. This separation between structured data and statements is more blurred in a type theoretic setting than in a set-theoretic one, but we can still use the homotopy basis we constructed to say something *about* statements which we can prove about closed reduction zig-zags. In type theory, it is natural to phrase such a tool as a lemma reminiscent of an *induction principle*, that is, a scheme to prove statements about composite structures by recursively proving things about simpler structures.[6]

So far, we have taken the *globular* point of view which considers surfaces between paths with the same source and target. However, our assumptions guarantee that we do not lose generality if we assume that one of the zig-zags is empty, since a rewrite $u \overset{*}{\Leftrightarrow} v$ can equivalently be represented as a rewrite $u \overset{*}{\Leftrightarrow} \varepsilon$. In type theory, it is often more convenient to consider this version and formulate a principle with only one instead of two arguments.

Moreover, our Theorem 30 is more general than what is needed for the applications in type theory that we will present in Section 5. For the sake of simplicity, we specialise it slightly and formulate the following principle:

Theorem 31 (Noetherian induction for closed zig-zags, cf. the theorem induction-for-closed-zigzags)**.** *Let $(A, \rightsquigarrow)$ be a 1-polygraph such that $\rightsquigarrow$ is Noetherian. Assume further that $\rightsquigarrow$ is locally confluent, that is, that for each local peak $u : (y \leftsquigarrow x \rightsquigarrow z)$, there is a valley $\overline{u} : (y \rightsquigarrow^* w \,{}^*\!\!\leftsquigarrow z)$.*

Let P be a type family of the form

$$P : \Pi\{x : A\}. (x \leftrightsquigarrow^* x) \to \mathcal{U},$$

with the following properties:

(1) *$P(e)$ holds for all $e : (x \leftrightsquigarrow^* x)$ with* $\mathsf{length}(e) = 0$.
(2) *For all rewrite zig-zags $u : (x \leftrightsquigarrow^* y)$, we have $P(u \cdot u^{-1})$.*
(3) *P is closed under* rotation*: For $u : (x \leftrightsquigarrow^* y)$ and $v : (y \leftrightsquigarrow^* x)$, we have*

$$P(u \cdot v) \to P(v \cdot u).$$

(4) *P is closed under* pasting *of closed zig-zags: If $u, v, w : (x \leftrightsquigarrow^* y)$ are parallel zig-zags, we have*

$$P(u \cdot v^{-1}) \to P(v \cdot w^{-1}) \to P(u \cdot w^{-1}).$$

(5) *P is closed under* inversion *of closed zig-zags: For $u : (x \leftrightsquigarrow^* x)$ we have*

$$P(u) \to P(u^{-1}).$$

(6) *P holds for the 'outlines' of the chosen local confluence diamonds, that is, for every local peak u, we have $P(u \cdot \overline{u}^{-1})$.*

Then, P holds for all closed reduction zig-zags:

$$\Pi\{x : A\}, (u : x \leftrightsquigarrow^* x). P(u).$$

Proof. The given data give rise to the 2-polygraph $(A, \rightsquigarrow, \Rightarrow)$, where the family of reduction steps between $u, v : (x \leftrightsquigarrow^* y)$ is defined by

$$(u \Rightarrow v) :\equiv P\left(u \cdot v^{-1}\right). \tag{36}$$

First, we observe that the definition, together with the conditions on P, imply that we have

$$(u \overset{*}{\Leftrightarrow} v) \to P\left(u \cdot v^{-1}\right). \tag{37}$$

To see this, we apply induction on the length of reduction a zig-zag $\alpha : (u \overset{*}{\Leftrightarrow} v)$:

- If α has length 0, we are done by (2).
- If α starts with a reduction $\beta : (u \Rightarrow w)$, we obtain $P(w \cdot v^{-1})$ by induction from the remaining reduction zig-zag and $P(u \cdot w^{-1})$ from β by definition, from which (4) lets us deduce $P(u \cdot v^{-1})$.
- Lastly, if α instead starts with a reversed reduction step $\beta : (w \Rightarrow u)$, we use (5) and are again in the situation of the previous case.

Therefore, a homotopy basis for our 2-polygraph $(A, \rightsquigarrow, \Rightarrow)$ suffices to complete the proof. We check the conditions of Theorem 30:

- **Cancellation of inverses:** Note that the requirement $s \cdot s^{-1} \overset{*}{\Rightarrow} \varepsilon$ follows from $s \cdot s^{-1} \Rightarrow \varepsilon$, which is the same as $s \Rightarrow s$, which (by the above observation) is implied by $s \overset{*}{\Rightarrow} s$, which is trivial. Alternatively, cancellation of inverses is also a direct consequence of (2).
- **Closure under congruence:** To show that $(\Rightarrow)$ is closed under congruence, let $u : (x \leftrightsquigarrow^* y)$ and $v : (y' \leftrightsquigarrow^* z)$ be given together with $w, w' : (y \leftrightsquigarrow^* y')$, for which we further assume $\alpha : (w \Rightarrow w')$. We need to show that

$$P\left((u \cdot w \cdot v) \cdot (u \cdot w' \cdot v)^{-1}\right),$$

which by rotation follows from $P\left(u^{-1} \cdot u \cdot w \cdot v \cdot v^{-1} \cdot w'^{-1}\right)$. Note that the special case of (4) with $v \equiv \varepsilon$ allows us to concatenate closed zig-zags and, in particular, it suffices to prove

$P\left(u^{-1}\cdot u\right)$ and $P\left(w\cdot v\cdot v^{-1}\cdot w'^{-1}\right)$. The former holds by the discussed cancellation of inverses. For the latter, we use the same trick again to write it as concatenation of $P\left(v\cdot v^{-1}\right)$ and $P\left(w'^{-1}\cdot w\right)$. This time, the second part holds by rotation of the assumption α.

- **Winkler-Buchberger structure:** The local confluence structure is also a Winkler-Buchberger structure.[7]
- **Cancellation of empty zig-zags:** This is almost directly given by (1).

As pointed out above, the thereby obtained homotopy basis completes the construction of $P(u)$ for every $u : (x \leftrightsquigarrow^* x)$. □

Remark 32. We could of course derive an induction principle that is more general than Theorem 31, where we have given up some generality to obtain simplicity. On the other hand, Theorem 31 holds even if we remove the assumptions (2) and (5), that is, $P(u\cdot u^{-1})$ and $P(u)\to P(u^{-1})$, which we have proved in our conference paper (Kraus and von Raumer 2020). Unfortunately, if we want to present Theorem 31 as a special case of Theorem 30, the assumptions (2) and (5) seem to be unavoidable. This slight loss in generality stems from the symmetric definition of a homotopy basis in Theorem 14, but poses no meaningful restriction in our applications.

5. Applications in Homotopy Type Theory

From now on, we are working in 'full' homotopy type theory; that is, in addition to the components detailed in Section 4.1, we assume that the theory has higher inductive types and families, and that the universe $\mathcal{U}$ is univalent: By assuming that for $A, B : \mathcal{U}$ the canonical function

$$(A = B) \to (A \simeq B)$$

is an equivalence itself, we allow ourselves to treat equivalences and equalities between types as the same.

Besides quotients and pushouts, which will be introduced as we go along, we need the truncation $\|A\|_n$, which for a type $A : \mathcal{U}$ and $n \geq -1$ represents a copy of A which is made an n-type by equating all (higher) equalities above the stated level. Truncation is defined as a higher inductive type with constructors

$$\iota : A \to \|A\|_n \text{ and}$$
$$\mathsf{trunc} : \mathsf{is}\text{-}n\text{-}\mathsf{type}\|A\|_n.$$

Note that the induction principle for $\|A\|_n$ states that the type can only eliminate into (families of) n-types.

After establishing some preliminary results for our applications in Section 5.1, we will, in Section 5.2, use the induction principle for closed zig-zags to characterise functions that map from a quotient into a 1-type (groupoid). This characterisation will then be applied to show that the fundamental group of the free group is trivial (Section 5.3) and that the pushout of 1-types over a set has no non-trivial second homotopy groups (Section 5.4). We will then, in Section 5.5 compare the two results with each other and put them in a series of six different applications which are all approximations of open problems in the field of synthetic homotopy theory.

5.1 On quotients, coequalisers, and truncation

As explained in the introduction, the *set-quotient* $A/\rightsquigarrow$ is the higher inductive type with constructors ι, glue, and trunc, see (3). The construction can be split into two steps. Recall from Section 1.4 that we write $A/\!\!/\rightsquigarrow$ for the *untruncated quotient* or *coequaliser* which has only the constructors ι and glue, and $\|-\|_0$ for the *set-truncation* as described above.

Lemma 33. *For a relation* $(\rightsquigarrow)$ *on* A*, we have*

$$(A/\rightsquigarrow) \simeq \|A/\!/\!\rightsquigarrow\|_0. \tag{38}$$

Proof. The direct approach of constructing functions back and forth works without difficulties. □

For a given type X, there is a canonical map from the function type $(A/\!/\!\rightsquigarrow) \to X$ to the Σ-type of pairs (f, h), where

$$f : A \to X, \tag{39}$$
$$h : \Pi\{x, y : A\}.(x \rightsquigarrow y) \to f(x) = f(y). \tag{40}$$

This map is given by:

$$g \mapsto (g \circ [-], \mathsf{ap}_g \circ \mathsf{glue}). \tag{41}$$

The universal property of the higher inductive type $A/\!/\!\rightsquigarrow$ tells us that this function is an equivalence (one can of course also show this with the dependent elimination principle of $A/\!/\!\rightsquigarrow$, if that is assumed instead as primitive).

We will need to prove statements about equalities in coequalisers. For this, we use the following result which characterises the path spaces of $(A/\!/\!\rightsquigarrow)$:

Theorem 34 (induction for coequaliser paths, Kraus and von Raumer 2019)**.** *Let a relation* $(\rightsquigarrow) : A \to A \to \mathcal{U}$ *as before and a point* $x_0 : A$ *be given. Assume we further have a type family*

$$P : \Pi\{y : A\}.(\iota(x_0) =_{A/\!/\rightsquigarrow} \iota(y)) \to \mathcal{U} \tag{42}$$

together with terms

$$r : P(\mathsf{refl}_{\iota(x_0)}), \tag{43}$$
$$e : \Pi\{y, z : A\}, (q : \iota(x_0) = \iota(y)), (s : y \rightsquigarrow z).P(q) \simeq P(q \cdot \mathsf{glue}(s)). \tag{44}$$

Then, we can construct a term

$$\mathsf{ind}_{r,e} : \Pi\{y : A\}, (q : \iota(x_0) = \iota(y)).P(q) \tag{45}$$

with the following β*-rules:*

$$\mathsf{ind}_{r,e}(\mathsf{refl}_{\iota(x_0)}) = r, \tag{46}$$
$$\mathsf{ind}_{r,e}(q \cdot \mathsf{glue}(s)) = e(q, s, \mathsf{ind}_{r,e}(q)). \tag{47}$$

□

To have all prerequisites we need in order to characterise the functions out of a quotient, we need one additional characterisation of maps: The type of functions from $\|A\|_0$ type into a 1-type consists of those functions which map any loop p to the reflexivity witness in B: For types A and B, we have a canonical function

$$(\|A\|_0 \to B) \to (A \to B) \tag{48}$$

which is given by precomposition with $|-|_0$. Any such function $g \circ |-|_0$ is moreover constant on loop spaces in the sense that

$$\mathsf{ap}_{g\circ|-|_0} : (x = x) \to (g(x) = g(x)) \tag{49}$$

satisfies $\mathsf{ap}_{g\circ|-|_0}(p) = \mathsf{refl}$, for all x and p. For a 1-truncated type B, the following known result by Capriotti, Kraus, and Vezzosi states that this property is all one needs to reverse (48):

Theorem 35 (Capriotti et al. 2015)**.** *Let A be a type and B be a 1-truncated type. The canonical function from $(\|A\|_0 \to B)$ to the type*

$$\Sigma(f : A \to B).\Pi(a : A), (p : a = a).\mathsf{ap}_f(p) = \mathsf{refl} \tag{50}$$

is an equivalence. □

5.2 A characterisation of functions on quotients

As before, let $(\rightsquigarrow)$ be a relation on A. Assume further that we are given a function $f : A \to X$ and a proof h that f sends related points to equal points, as in (39) and (40). There is an obvious function

$$h^* : \Pi\{x, y : A\}.(x \leftrightsquigarrow^* y) \to f(x) = f(y), \tag{51}$$

defined by recursion on $x \leftrightsquigarrow^* y$ which in each step composes with a path given by h or the inverse of such a path. Given (f, h) and a third map $k : X \to Y$, it is easy to prove by induction on $x \leftrightsquigarrow^* y$ that we have

$$\mathsf{ap}_k \circ h^* = (\mathsf{ap}_k \circ h)^*. \tag{52}$$

We also note that, for chains u, v,

$$h^*(u \cdot v) = h^*(u) \cdot h^*(v) \text{ and} \tag{53}$$

$$h^*(u^{-1}) = (h^*(u))^{-1}. \tag{54}$$

Of particular interest is the function $\mathsf{glue}^* : \Pi\{x, y : A\}.(x \leftrightsquigarrow^* y) \to \iota(x) = \iota(y)$. It is in general not an equivalence: For example, for $t : x \rightsquigarrow x$, the chain $t \cdot t^{-1}$ and the empty chain both get mapped to refl. Thus, glue^* does not preserve inequality (but see Corollary 17). However, we have the following result:

Lemma 36. *The function* $\mathsf{glue}^* : (x \leftrightsquigarrow^* y) \to \iota(x) = \iota(y)$ *is surjective.*

Proof. Fixing one endpoint $x_0 : A$ and setting

$$P : \Pi\{y : A\}.(\iota(x_0) = \iota(y)) \to \mathcal{U} \tag{55}$$

$$P(q) :\equiv \left\| \Sigma(u : x_0 \leftrightsquigarrow^* y).\mathsf{glue}^*(u) = q \right\|_{-1} \tag{56}$$

we need to show that, for all q, we have $P(q)$. We use Theorem 34, where r is given by the trivial chain. To construct e, we need to prove $P(q) \simeq P(q \cdot \mathsf{glue}(s))$ for any $s : y \rightsquigarrow z$. This amounts to constructing functions in both directions between the types $\Sigma(u : x_0 \leftrightsquigarrow^* y).\mathsf{glue}^*(u) = q$ and $\Sigma(u : x_0 \leftrightsquigarrow^* y).\mathsf{glue}^*(u) = q \cdot \mathsf{glue}(s)$, where extending a chain with s or with s^{-1} is sufficient. □

The following is a 'derived induction principle' for equalities in coequalisers:

Lemma 37. *For a family $P : \Pi\{x : A /\!\!/ \rightsquigarrow\}.x = x \to \mathcal{U}$ such that each $P(q)$ is a proposition, the two types*

$$\Pi\{x : A\}, (u : x \leftrightsquigarrow^* x).\, P(\mathsf{glue}^*(u)). \tag{57}$$

and

$$\Pi(c : A /\!\!/ \rightsquigarrow), (q : a = a).\, P(q) \tag{58}$$

are equivalent.

Proof. Both types are propositions, and the second clearly implies the first. For the other direction, induction on $c : A /\!\!/ \rightsquigarrow$ lets us assume that c is of the form $\iota(x)$ for some $x : A$; the case for the constructor glue is automatic. The statement then follows from the surjectivity of glue^*. □

Theorem 38. *Let $A : \mathcal{U}$ be a type, $(\rightsquigarrow) : A \to A \to \mathcal{U}$ be a relation, and $X : \mathcal{U}$ be a 1-type. Then, the type of functions $(A/\rightsquigarrow \to X)$ is equivalent to the type of triples (f, h, c) (a nested Σ-type), where*

$$f : A \to X \tag{59}$$
$$h : \Pi\{x, y : A\}.(x \rightsquigarrow y) \to f(x) = f(y) \tag{60}$$
$$c : \Pi\{x : A\}(u : x \leftrightsquigarrow^* x).h^*(u) = \mathsf{refl}. \tag{61}$$

Proof. We have the following chain of equivalences:

$$\begin{aligned}
& A/\rightsquigarrow \to X \\
\text{by Lemma 33} \quad \simeq \quad & \left\| A /\!\!/ \rightsquigarrow \right\|_0 \to X \\
\text{by Theorem 35} \quad \simeq \quad & \Sigma g : (A /\!\!/ \rightsquigarrow) \to X. \\
& \quad c : \Pi\{x : A /\!\!/ \rightsquigarrow\}, (q : x = x).\mathsf{ap}_g(q) = \mathsf{refl} \\
\text{by Lemma 37} \quad \simeq \quad & \Sigma g : (A /\!\!/ \rightsquigarrow) \to X. \\
& \quad c : \Pi\{x : A\}, (u : x \leftrightsquigarrow^* x).\mathsf{ap}_g(\mathsf{glue}^*(\gamma)) = \mathsf{refl} \\
\text{by (52)} \quad \simeq \quad & \Sigma g : (A /\!\!/ \rightsquigarrow) \to X. \\
& \quad c : \Pi\{x : A\}, (u : x \leftrightsquigarrow^* x).(\mathsf{ap}_g \circ \mathsf{glue})^*(\gamma) = \mathsf{refl} \\
\text{by (41)} \quad \simeq \quad & \Sigma f : A \to X. \\
& \Sigma h : \Pi\{x, y : A\}.(x \rightsquigarrow y) \to f(x) = f(y). \\
& \quad c : \Pi\{x : A\}, (u : x \leftrightsquigarrow^* x).h^*(u) = \mathsf{refl}
\end{aligned}$$

□

Remark 39. It is an easy exercise to show that the component c in the statement of Theorem 38 can be equivalently replaced by

$$c' : \Pi\{x, y : A\}(u, v : x \leftrightsquigarrow x).h^*(u) = h^*(v)$$

to obtain a binary version of the requirement.

We are now ready to combine the theory developed in this section with the construction of the homotopy basis to obtain a full characterisation of maps from a set-quotient into a one-type.

Theorem 40. *Let $A : \mathcal{U}$ be a type, $(\rightsquigarrow) : A \to A \to \mathcal{U}$ a Noetherian and locally confluent relation, with the local confluence valley of u denoted by $\overline{u}$ as in Theorem 31. Further, let $X : \mathcal{U}$ be a 1-type. Then, the type of functions $(A/\rightsquigarrow) \to X$ is equivalent to the type of tuples (f, h, d_1, d_2), where*

$$f : A \to X \tag{62}$$
$$h : \Pi\{x, y : A\}.(x \rightsquigarrow y) \to f(x) = f(y) \tag{63}$$
$$d_1 : \Pi\{x : A\}.\Pi(p : x = x).\mathsf{ap}_f(p) = \mathsf{refl} \tag{64}$$
$$d_2 : \Pi\{x, y, z : A\}(u : y \leftsquigarrow x \rightsquigarrow z).h^*\left(u \cdot \overline{u}^{-1}\right) = \mathsf{refl}. \tag{65}$$

Further, if A is a set, the type $A/\rightsquigarrow \to X$ is equivalent to the type of triples (f, h, d_2).

Proof. The case for A being a set follows immediately from the main statement, since the type of d_1 becomes contractible.

For the main statement, we want to apply Theorem 38. We need to show that the type of c in (61) is equivalent to the type of pairs (d_1, d_2) above. Note that they are all propositions. From c, we immediately derive (d_1, d_2) from Corollary 17.

Let us assume we are given (d_1, d_2). We need to derive c. We want to apply the induction principle given by Theorem 31 with

$$P(u) :\equiv (h^*(u) = \mathsf{refl}).$$

Now, we need to show the six closure properties of P to complete the proof:

P is true for empty closed zig-zags By Lemma 16, each empty closed zig-zag is the image of a loop in A under the equivalence between empty closed zig-zags, which by d_1 is mapped to refl.

P is true for concatenation of inverses For $s : (x \rightsquigarrow y)$ the type

$$h^*(s \cdot s^{-1}) \equiv h(s) \cdot h(s)^{-1} = \mathsf{refl}$$

is inhabited. For longer zig-zags, the statement follows by induction.

P is closed under rotation For u and v, we have

$$\begin{aligned} P(u \cdot v) &\equiv (h^*(u \cdot v) = \mathsf{refl}) \\ &\simeq (h^*(u) = h^*(v)^{-1}) \\ &\simeq (h^*(v) = h^*(u)^{-1}) \\ &\equiv P(v \cdot u). \end{aligned}$$

P is closed under pasting We can calculate

$$\begin{aligned} h^*(u \cdot w^{-1}) &= h^*(u) \bullet h^*(w^{-1}) \\ &= h^*(u) \bullet h^*(v^{-1}) \bullet h^*(v) \bullet h^*(w^{-1}) \\ &= h^*(u \cdot v^{-1}) \bullet h^*(v \cdot w^{-1}) \\ &= \mathsf{refl} \cdot \mathsf{refl}. \end{aligned}$$

if $h^*(u \cdot w^{-1}) = h^*(w \cdot v^{-1}) = \mathsf{refl}$.

P is closed under inversion By $h^*(u^{-1}) = h^*(u)^{-1} = \mathsf{refl}^{-1} \equiv \mathsf{refl}$ whenever we have $h^*(u) = \mathsf{refl}$.

P holds for the outlines of local confluence diagrams This is given directly by d_2.

□

5.3 Free ∞-groups

We want to use Theorem 40 to show that the free higher group $\mathsf{F}(M)$ has trivial fundamental groups. Recall that this is the example discussed in the introduction, with $\mathsf{F}(M)$ defined in equation (7).

Theorem 41. *The fundamental groups of the free higher group on a set are trivial. In other words, for a set M and any $x : \mathsf{F}(M)$, we have*

$$\pi\ (\mathsf{F}(M), x)\ =\ \mathbf{1}. \tag{66}$$

We split the proof into several small lemmas. We keep using the relation $\rightsquigarrow$ of Example 1 and Lemma 24. Further, recall the functions ω_1 (9) and ω_2 (10) from the introduction, as well as the map ω (11).

Lemma 42 (free group; continuing Example 1 and Lemma 24)**.** *For the relation $\rightsquigarrow$ of Example 1, we can construct the outlines of a local confluence structure consisting for each local peak $u : (\ell_x \leftsquigarrow \ell \rightsquigarrow \ell_y)$ of a valley*

$$\overline{u} : (\ell_x \rightsquigarrow^* \ell' \,{}^*\!\!\leftsquigarrow \ell_y),$$

which furthermore can be proven to be coherent by the presence of a 2-path

$$d_2(u) : \omega_2^* \left(u \cdot \overline{u}^{-1}\right) = \mathsf{refl}.$$

Proof. We perform a standard critical pair analysis on the span and assume that ℓ_x is obtained from ℓ by removing a redex (x, x^{-1}), and likewise that ℓ_y is obtained from ℓ by removing a redex (y, y^{-1}). Taking in consideration the symmetry of the assumptions, we end up with only three cases:

(1) The two redexes are at the same position of ℓ (they 'fully overlap'), implying $x = y$ and $\ell_x = \ell_y$.
(2) The two redexes partially overlap, in the sense that $x^{-1} = y$ (or $y^{-1} = x$, which is equivalent). In this case, we again have $\ell_x = \ell_y$.
(3) There is no overlap between the two redexes ('Peiffer branching').

The case (1) is trivial because we can set $\mathsf{wb}(u) = \epsilon_{\ell_x}$ and

$$\omega_2^*(u) = \omega_2(s) \cdot \omega_2(s)^{-1} = \mathsf{refl}$$

for $s : (\ell \rightsquigarrow \ell_x)$. For the remaining two cases, we need to recall the definition of ω_1 and ω_2 and observe the following: The function $\omega_1 : \mathsf{List}(M \uplus M) \to \mathsf{F}(M)$, cf. (9), factors as

$$\mathsf{List}(M \uplus M) \longrightarrow \mathsf{List}(\mathsf{base} = \mathsf{base}) \longrightarrow \mathsf{base} = \mathsf{base} \tag{67}$$

where the first map applies loop on every list element, while the second concatenates; note that $\mathsf{F}(M) :\equiv (\mathsf{base} = \mathsf{base})$. The function ω_2 (10) can then be factored similarly.

For case (3), we can remove the redex (y, y^{-1}) from ℓ_x and have constructed a list ℓ' equal to the one we get if we remove (x, x^{-1}) from ℓ_y. We combine these reductions to obtain $\mathsf{wb}(u) : (\ell_x \rightsquigarrow \ell' \leftsquigarrow \ell_y)$. To provide the coherence $d_2(u)$ in this case, we can, by (67), assume that we are given a list of loops around base instead of a list of elements of $M \uplus M$. We first repeatedly use that associativity of path composition is coherent (we have 'MacLane's pentagon' by trivial path induction). Then, we have to show that the two canonical ways of simplifying $e_1 \bullet (p \bullet p^{-1}) \bullet e_2 \bullet (q \bullet q^{-1}) \bullet e_3$ to $e_1 \bullet e_2 \bullet e_3$ are equal.

This can be achieved in two ways: A common pattern in homotopy type theory is now to generalise to the case that p and q are equalities with arbitrary endpoints rather than loops, and then do path induction. If p and q are both refl, then both simplifications become refl as well. Instead of applying path induction directly, it is possible to prove this lemma only using naturality and the Eckmann-Hilton theorem (The Univalent Foundations Program 2013, Thm 2.1.6): The choice of whether to first reduce on the left and then on the right or vice versa corresponds to the two ways (in the reference called $\star$ and $\star'$) of defining horizontal composition of 2-paths by first whiskering on the left or on the right, respectively. As the proof of the theorem states, these two ways coincide.

In case (2), we can set $\ell' = \ell_x = \ell_y$ and $\mathsf{wb}(u) = \epsilon_{\ell'}$. Analogously to case (3), we can construct $d_2(u)$ by showing that the two ways of reducing $e_1 \bullet p \bullet p^{-1} \bullet p \bullet e_2$ to $e_1 \bullet p \bullet e_2$ are equal. This time, we have to generalise not only the endpoint of p but both endpoints of p and the respective endpoints of e_1 and e_2 to reduce the problem to the case where p is refl, and the equalities are definitionally the same. □

Lemma 43. *The free higher group* $\mathsf{F}(M)$ *is a retract of* $\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow$, *in the sense that there is a map*

$$\varphi : \mathsf{F}(M) \to \mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow \tag{68}$$

such that $\omega \circ \varphi$ *is the identity on* $\mathsf{F}(M)$.

Proof. For any $x : M \uplus M$, the operation 'adding x to a list'

$$(x \cdot \_) \ : \ \mathsf{List}(M \uplus M) \to \mathsf{List}(M \uplus M) \tag{69}$$

can be lifted to a function of type

$$(\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow) \to (\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow). \tag{70}$$

Moreover, the function (70) is inverse to $(x^{-1} \cdot \_)$ and thus an equivalence.

Let $\star$ be the unique element of the unit type $\mathbf{1}$. We define the relation $\sim$ on the unit type by $(\star \rightsquigarrow \star) :\equiv M$. Then, $\mathsf{hcolim}(M \rightrightarrows \mathbf{1})$ is by definition the coequaliser $(\mathbf{1}/\!\!/\!\sim)$, and $\mathsf{F}(M)$ is given by $(\iota(\star) = \iota(\star))$. This allows us to define φ using Theorem 34 with the constant family $P :\equiv (\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow)$, with the equivalence of the component e given by (70).

A further application of Theorem 34 shows that $\omega \circ \varphi$ is pointwise equal to the identity. □

Proof of Theorem 41. By The Univalent Foundations Program (2013, Thms 7.2.9 and 7.3.12), the statement of the theorem is equivalent to the claim that $\|\mathsf{F}(M)\|_1$ is a set.

We now consider the following diagram:

$$\begin{tikzcd}
\mathsf{F}(M) \arrow[d, "\varphi"] \arrow[rrd, "|-|_1"] & & \\
\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow \arrow[r, "\omega"] \arrow[d, "|-|_0"] & \mathsf{F}(M) \arrow[r, "|-|_1"] & \|\mathsf{F}(M)\|_1 \\
\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow \arrow[rru, dashed] & &
\end{tikzcd} \tag{71}$$

The dashed map exists by the combination of Theorem 40 (note that we are in the simplified case where the type to be quotiented is a set) together with Lemma 42 (and Lemma 33). By construction, the bottom triangle commutes. The top triangle commutes by Lemma 43.

Therefore, the map $|-|_1$ factors through a set (namely $\mathsf{List}(M \uplus M)/\!\!/\!\rightsquigarrow$). This means that $\|\mathsf{F}(M)\|_1$ is a retract of a set, and therefore itself a set. □

5.4 Pushouts of 1-types

In this section, we will prove another theorem using the characterisation of maps out of quotients Theorem 40. At first glance it might look completely distinct from the application of free ∞-groups, but, as we will see in Section 5.5, it is a more general formulation of the same phenomenon.

The subject of study of this section will be *pushouts* of types. We will always assume that we are given a span of $B \xleftarrow{f} A \xrightarrow{g} C$ of types and functions, of which we will take the pushout:

$$\begin{tikzcd} A \arrow[r, "g"] \arrow[d, "f"'] & C \arrow[d, "i_1"'] \\ B \arrow[r, "i_0"] & B \sqcup_A C \end{tikzcd} \tag{72}$$

The pushout is, as common in homotopy type theory, defined as a higher inductive type with point constructors $i_0 : B \to B \sqcup_A C$ and $i_1 : C \to B \sqcup_A C$ as well as a path constructor

$$\mathsf{glue} : \Pi(a : A).\, i_0(f(a)) = i_1(g(a)),$$

which makes the diagram (72) commute.

In contrast to Section 5.3, we will not prove statement about first but about *second* homotopy groups, but again, we do not make any statements about the homotopy levels above that.

Theorem 44. *Given a pushout as in (72), if A is a set and B, C are 1-types, then all second homotopy groups of $B \sqcup_A C$ are trivial. In other words, $\|B \sqcup_A C\|_2$ is a 1-type.*

Proof sketch. The argument is almost completely analogous to the proof of Theorem 41. The main difference is that the type $\mathsf{List}(M \uplus M)$ is not sufficient any more. Instead, we need to be slightly more subtle when we encode the equalities in the pushout. The following construction is due to Favonia and Shulman (Hou (Favonia) and Shulman 2016), who use it in their formulation of the *Seifert-van Kampen Theorem.*

Given the square in (72) and $b, b' : B$, we consider the type $L_{b,b'}$ of lists of the form

$$[b, p_0, x_1, q_1, y_1, p_1, x_2, q_2, y_2, \dots, y_n, p_n, b'] \tag{73}$$

where[8]

$$\begin{aligned} & x_i : A && y_i : A && (74) \\ & p_0 : b = f(x_1) && p_n : f(y_n) = b' && (75) \\ & p_i : f(y_i) = f(x_{i+1}) && q_i : g(x_i) = g(y_i) && (76) \end{aligned}$$

The corresponding relation is generated by

$$\begin{aligned} [\dots, q_k, y_k, \mathsf{refl}, y_k, q_{k+1}, \dots] &\rightsquigarrow [\dots, q_k \cdot q_{k+1}, \dots] \\ [\dots, p_k, x_k, \mathsf{refl}, x_k, p_{k+1}, \dots] &\rightsquigarrow [\dots, p_k \cdot p_{k+1}, \dots] \end{aligned} \tag{77}$$

The statement of the Seifert-van Kampen Theorem is that the set-quotient $L_{b,b'}/\rightsquigarrow$ is equivalent to the set-truncated type $\left\| i_0(b) = i_0(b') \right\|_0$ of equalities in the pushout. Similarly to $L_{b,b'}$, there are three further types of lists where one or both of the endpoints are in C instead of B. In general, we can define a type of lists $L_{x,x'}$ for $x, x' : B \uplus C$, and the Seifert-van Kampen Theorem states that $L_{x,x'}/\rightsquigarrow$ is equivalent to $\left\| i(x) = i(x') \right\|_0$, with $i : B \uplus C \to B \sqcup_A C$ given by (i_0, i_1).

The construction of ω and φ is essentially the same as before, using the version of Theorem 34 for pushouts available in Kraus and von Raumer (2019). For the relation (77), we can show the analogous to Lemmas 24 and 42. The analogous to (71) is

$$\begin{tikzcd}
i(x) = i(x') \arrow[d, "\varphi"] \arrow[rrd, "|-|_1"] & & \\
L_{x,x'} /\!/ \rightsquigarrow \arrow[r, "\omega"] \arrow[d, "|-|_0"] & i(x) = i(x') \arrow[r, "|-|_1"] & \left\| i(x) = i(x') \right\|_1 \\
L_{x,x'} / \rightsquigarrow \arrow[rru, dashed] & &
\end{tikzcd} \tag{78}$$

There is a small subtlety: Since A is a set and B, C are 1-types, the type of lists $L_{x,x'}$ is a set. This is important since it allows us (as before) to use the simpler version of Theorem 40. The above diagram shows that $\left\| i(x) = i(x') \right\|_1$ is a set. Choosing x and x' to be identical, this means that $\| \Omega(B \sqcup_A C, i(x)) \|_1$ is a set, which is equivalent to the statement that $\left\| \Omega^2(B \sqcup_A C, i(x)) \right\|_0$ (the second homotopy group) is trivial. It follows by the usual induction principle of the pushout that $\left\| \Omega^2(B \sqcup_A C, z) \right\|_0$ for arbitrary $z : B \sqcup_A C$ is trivial. □

5.5 Relation between the applications

Let us compare the lists used in the proofs of Theorems 41 and 44. We can observe that the former ones are a specialisation of the latter once with the following restrictions: The types B and C are each set to be the unit type $\mathbf{1}$. A in (72) is set to be $A' \uplus \mathbf{1}$ when we want to consider the free group on A'. Then, the elements b, b', p_i, and q_i in (73) carry no information, and a list entry of the right summand of $A' \uplus \mathbf{1}$ indicates a switch from the 'left to right' or vice versa in $\mathsf{List}(A' \uplus A')$. Under this transformation, the relations (5) and (77) correspond to each other.

Indeed, all of the following questions can be reduced to the question, whether the pushout $B \sqcup_A C$ of 1-types B and C over a set A is a 1-type. The first one is the problem which we approximated in Section 5.3:

(i) Is the free higher group on a set again a set?
(ii) Is the suspension of a set a 1-type (open problem recorded in The Univalent Foundations Program 2013, Ex 8.2)?
(iii) Given a 1-type B with a base point $b_0 : B$. If we add a single loop around b_0, is the type still a 1-type?
(iv) Given B and b_0 as above, imagine we add M-many loops around b_0 for some given set M. Is the resulting type still a 1-type?
(v) If we add a path (not necessarily a loop) to a 1-type B, is the result still a 1-type?
(vi) If we add an M-indexed family of paths to a 1-type B (for some set M), is the resulting type still a 1-type?

All questions are of the form:

'Can a change at level 1 induce a change at level 2 or higher?'

Only (i) seems to be about level 0 and 1, but this is simply because we have taken a loop space. With our Theorem 40, we can show an approximation for each of these questions analogously to Theorem 41. This means that we show:

'A change at level 1 does not induce a change at level 2 (but we don't know about higher levels)'.

We can obtain all of these approximations by setting in Theorem 44, respectively:

(i) B, C to both be the unit type $\mathbf{1}$ and A is $A' \uplus \mathbf{1}$, where A' is the set on which we want the free higher group (this is the usual translation from coequalisers to pushouts);
(ii) B and C both to be $\mathbf{1}$;
(iii) $A :\equiv \mathbf{1}$ and C be the circle S^1;
(iv) $A :\equiv \mathbf{1}$ and $C :\equiv M \times \mathsf{S}^1$;
(v) A to be the 2-element type $\mathbf{2}$ and $C :\equiv \mathbf{1}$;
(vi) $A :\equiv M \times \mathbf{2}$ and $C :\equiv M$.

6. Concluding Remarks

Our work has shown that methods from higher-dimensional rewriting can be used to tackle some of the coherence problems appearing in homotopy type theory. One limitation of our results so far is that they only make statements about one specific dimension of the spaces which we consider. It may very well be possible to generalise our method to show 'higher' versions of the same coherences, or, in other words, better approximations of the same open problems. For example, one could try to relax the condition of 1-truncatedness in Theorem 40 to 2-truncatedness. For this generalisation, we expect that the proofs of 2-dimensional coherence would have to be coherent as well. It remains to see whether 3-dimensional rewriting theory, as proposed by Mimram (2014), could be a useful vantage point to guide proofs about these 3-dimensional coherence theorems.

The 'fully untruncated' version of Theorem 44 would state that the pushout of $B \leftarrow A \rightarrow C$ (with A a set and B, C groupoids) is a groupoid, with the special case being that the free higher group on a set is a set. We do not expect that this is provable in homotopy type theory. Even if a workable version of ∞-rewriting theory is formulated, this looks like one of the problems requiring an infinite tower of coherences that, akin to semisimplicial types (Herbelin 2015; Kraus 2018), are not expected to be expressible in homotopy type theory. However, we conjecture that this can be done in *two-level type theory* (Altenkirch et al. 2016; Annenkov et al. 2019; Voevodsky 2013) in the style of Kraus (2021). At the workshop *Logique et Structures Supérieures* at the Centre International de Rencontres Mathématiques (CIRM) in February 2022, Christian Sattler outlined an argument to generalise the statement to any externally chosen truncation level (Sattler 2022).

Another line of research about potential generalisation is the question of whether it is possible to weaken the assumption that the polygraph is terminating. Instead, it could be enough to assume the *decreasingness* of the relation, which Vincent van Oostrom suggests as an alternative (van Oostrom 2008).

Supplementary Material. To view supplementary material for this article, please visit http://doi.org/10.1017/S0960129523000026.

Acknowledgements. We would first of all like to thank Vincent van Oostrom for the interest he took in reading our LICS paper (Kraus and von Raumer 2020) and for the detailed explanation on how our approach compared to the terminology and the results about abstract rewriting systems. The more modular presentation of our proofs, the generalisation of the results, and the clearer connection to higher-dimensional rewriting would not have happened without his input.

We would also like to thank Carlo Angiuli who, like Vincent van Oostrom, pointed us towards the work of Craig C. Squier on rewriting systems.

Furthermore, we would like to express our gratitude to many people whose comments helped us to improve the presentation of both our LICS and the current paper. We thank the Theory of Computation group at the University of Birmingham, the Type Theory group in Budapest, the Functional Programming lab in Nottingham, the participants and the organiser (Chuangjie Xu) of the *FAUM* meeting (Herrsching 2019) and *Types in Munich* (online 2020), the LICS'20 organisers, as well as the participants and the organisers (Dan Christensen and Chris Kapulkin) of *HoTTEST* (Homotopy Type Theory Electronic Seminar Talks). We especially acknowledge the remarks by Steve Awodey, Ulrik Buchholtz, Thierry Coquand, Eric Finster, Martín Hötzel Escardó, Egbert Rijke, Anders Mörtberg, Chuangjie Xu, and, in particular, Christian Sattler.

Funding. This work was supported by The Royal Society, grant reference URF\R1\191055.

Notes

1 We thank Vincent van Oostrom for pointing this out and for a variety of further helpful remarks; we also refer to the acknowledgements at the end of the paper.
2 To avoid the problem that closedness poses here, one may try to instead consider *two* parallel but not necessarily closed zig-zags. However, the type of such pairs is not inductively generated either, and the situation is essentially the same. This is reminiscent of the well-known fact in type theory that the principles `UIP` (parallel equalities are equal) and `Axiom K` (loops are equal to reflexivity) are interderivable.
3 The more general statement is *nested wellfounded induction*, discussed in the type-theoretic setting below, cf. Lemma 23.
4 In the terminology of bicategories, the expression $u \cdot \alpha \cdot v$ corresponds to the *horizontal composition* of the identities on u and v with α. We do not require *vertical composition*, which would correspond to an operation which turns a rewrite sequence in $(u \overset{*}{\Rrightarrow} v)$ into a step in $(u \Rrightarrow v)$.
5 It is known that this simple form already implies the seemingly stronger formulations of function extensionality.
6 Depending on the terminology one uses, one may wish to reserve the term *induction principle* for the native elimination principles that inductive types are equipped with. Our result (Theorem 31) is of course not such an elimination principle, but can be stated in a similar form. Being slightly less strict about the usage of the term, it thus seems reasonable to call it an induction principle.
7 While it seems fair to consider this connection between a local confluence structure and a Winkler-Buchberger structure obvious, we found it rather cumbersome to formalise is. The corresponding Agda statement is more than 70 lines of code long, and this number does not even take into account that we had to write a range of auxiliary lemmas solely for this particular statement.
8 We remove the 0-truncations around the path spaces. These are without effect here since B, C are 1-types.

References

Aczel, P. (1977). An introduction to inductive definitions. In: *Studies in Logic and the Foundations of Mathematics*, vol. 90, Elsevier, 739–782.

Aczel, P. and Rathjen, M. (2001). Notes on constructive set theory. Available online at http://aleteya.cs.buap.mx/~jlavalle/papers/Constructive%20Set%20and%20Type%20Theories/notesOnConstructiveSetTheory.pdf.

Alleaume, C. and Malbos, P. (2016). Coherence of quasi-terminating decreasing 2-polygraphs. In: *5th International Workshop on Confluence*, 46.

Allioux, A., Finster, E. and Sozeau, M. Types are internal ∞-groupoids. Draft paper. Available online at https://ericfinster.github.io/files/type-int-grpds.pdf.

Altenkirch, T., Capriotti, P. and Kraus, N. (2016). Extending homotopy type theory with strict equality. In: Talbot, J.-M. and Regnier, L. (eds.) *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, vol. 62, 21:1–21:17.

Annenkov, D., Capriotti, P., Kraus, N. and Sattler, C. (2019). Two-level type theory and applications. *ArXiv*. Available online at https://arxiv.org/abs/1705.03307.

Awodey, S. and Warren, M. A. (2009). Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society* **146** 45–55.

Barendregt, H., Abramsky, S., Gabbay, D. M. and Maibaum, T. S. E. (1992). Handbook of logic in computer science. In: *Lambda Calculi with Types*, vol. 2, 117–309.

Bezem, M., Buchholtz, U., Cagne, P., Dundas, B. I. and Grayson, D. R. Symmetry. Book in progress.

Buchberger, B. and Winkler, F. (1983). A criterion for eliminating unnecessary reductions in the knuth–bendix algorithm. *Algebra, Combinatorics and Logic in Computer Science* **42** 849–869.

Burroni, A. (1993). Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Science* **115** (1) 43–62.

Capriotti, P., Kraus, N. and Vezzosi, A. (2015). Functions out of higher truncations. In Kreutzer, S. (ed.) *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 41, Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 359–373.

Cohen, C., Coquand, T., Huber, S. and Mörtberg, A. (2017). Cubical type theory: a constructive interpretation of the univalence axiom. *IfCoLog Journal of Logics and their Applications* **4** (10) 3127–3169.

Dershowitz, N. and Manna, Z. (1979). Proving termination with multiset orderings. In: Maurer, H. A. (ed.) *ICALP'79: Automata, Languages and Programming*, Berlin, Heidelberg, Springer Berlin Heidelberg, 188–202.

Forest, S. and Mimram, S. (2018). Coherence of gray categories via rewriting. In: Kirchner, H. (ed.) *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 108, Dagstuhl, Germany, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 15:1–15:16.

Gabriel, P. (1972). Unzerlegbare Darstellungen I. *Manuscripta Mathematica* **6** (1) 71–103.

Gaussent, S., Guiraud, Y. and Malbos, P. (2015). Coherent presentations of artin monoids. *Compositio Mathematica* **151** (5) 957–998.

Guiraud, Y. and Malbos, P. (2012). Coherence in monoidal track categories. *Mathematical Structures in Computer Science* **22** (6) 931–969.

Guiraud, Y., Malbos, P. and Mimram, S. (2013). A homotopical completion procedure with applications to coherence of monoids. In: van Raamsdonk, F. (ed.) *24th International Conference on Rewriting Techniques and Applications (RTA 2013)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 21, Dagstuhl, Germany, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 223–238.

Hage, N. and Malbos, P. (2017). Knuth's coherent presentations of plactic monoids of type A. *Algebras and Representation Theory* **20** (5) 1259–1288.

Herbelin, H. (2015). A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science* **25** (5) 1116–1131.

Hou (Favonia), K.-B. and Shulman, M. (2016). The Seifert-van Kampen theorem in homotopy type theory. In: *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 62, Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 22:1–22:16.

Huet, G. P. (1980). Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM* **27** (4) 797–821.

Kapulkin, C. and Lumsdaine, P. L. (2018). The simplicial model of Univalent Foundations (after Voevodsky). ArXiv e-prints. To appear in the Journal of the European Mathematical Society.

Kraus, N. (2015a). The general universal property of the propositional truncation. In: Herbelin, H., Letouzey, P. and Sozeau, M. (eds.) *20th International Conference on Types for Proofs and Programs (TYPES 2014)*, vol. 39, Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 111–145.

Kraus, N. (2015). *Truncation Levels in Homotopy Type Theory*. Phd thesis, School of Computer Science, University of Nottingham, Nottingham, UK. Available online at http://eprints.nottingham.ac.uk/28986/.

Kraus, N. (2018). On the role of semisimplicial types. Abstract, presented at TYPES'18.

Kraus, N. (2021). Internal ∞-categorical models of dependent type theory: Towards 2LTT eating HoTT. In: *Symposium on Logic in Computer Science (LICS 2021)*, 1–14.

Kraus, N. and Altenkirch, T. (2018). Free higher groups in homotopy type theory. In: *Symposium on Logic in Computer Science (LICS 2018)*, 599–608, ACM.

Kraus, N. and von Raumer, J. (2019). Path spaces of higher inductive types in homotopy type theory. In: *Symposium on Logic in Computer Science (LICS 2019)*.

Kraus, N. and von Raumer, J. (2020). Coherence via well-foundedness: Taming set-quotients in homotopy type theory. In: *Symposium on Logic in Computer Science (LICS 2020)*, New York, NY, USA, Association for Computing Machinery, 662–675.

Lumsdaine, P. L. (2009). Weak omega-categories from intensional type theory. In: *Typed Lambda Calculi and Applications (TLCA)*, Springer-Verlag, 172–187.

Martin-Löf, P. (1984). *Intuitionistic Type Theory*, Studies in Proof Theory, vol. 1, Bibliopolis.

Mimram, S. (2014). Towards 3-dimensional rewriting theory. *Logical Methods in Computer Science* **10** (1) 1–47.

Newman, M. H. A. (1942). On theories with a combinatorial definition of "equivalence". *Annals of Mathematics* 223–243.

Nipkow, T. (1998). An inductive proof of the wellfoundedness of the multiset order. Unpublished note.

Sattler, C. (2022). Filtered colimits and free groups on sets. Available online at https://www.cirm-math.fr/RepOrga/2689/Abstracts/sattler.pdf.

Squier, C. and Otto, F. (1987). The word problem for finitely presented monoids and finite canonical rewriting systems. In: *International Conference on Rewriting Techniques and Applications*, Springer, 74–82.

Squier, C. C. (1987). Word problems and a homological finiteness condition for monoids. *Journal of Pure and Applied Algebra* **49** (1–2) 201–217.

Squier, C. C., Otto, F. and Kobayashi, Y. (1994). A finiteness condition for rewriting systems. *Theoretical Computer Science* **131** (2) 271–294.

Street, R. (1987). The algebra of oriented simplexes. *Journal of Pure and Applied Algebra* **49** (3) 283–335.

The Univalent Foundations Program. (2013). *Homotopy Type Theory: Univalent Foundations of Mathematics*. Available online at http://homotopytypetheory.org/book/.

van den Berg, B. and Garner, R. (2011). Types are weak ω-groupoids. *Proceedings of the London Mathematical Society* **102** (2) 370–394.

van Oostrom, V. (2008). Confluence by decreasing diagrams. In: *Rewriting Techniques and Applications (RTA 2008)*, 306–320.

Voevodsky, V. (2013). A simple type system with two identity types. Unpublished note.

Cite this article: Kraus N and von Raumer J (2022). A rewriting coherence theorem with applications in homotopy type theory. *Mathematical Structures in Computer Science* **32**, 982–1014. https://doi.org/10.1017/S0960129523000026