New Foundations is consistent

Sky Wilshaw

July 2023

Underlying theory

All of the definitions and theorems that follow have been machine-checked by Lean.

The construction described in this paper takes place in a dependent type theory with:

- a proof-irrelevant impredicative universe of propositions called Prop;
- predicative universes indexed by ω, called $\mathsf{Type} = \mathsf{Type}\ 0 : \mathsf{Type}\ 1 : \ldots$;
- dependent function types $\prod_{(x:\alpha)} \beta$ for all types α, β, where we denote function application by juxtaposition;
- inductive types at each universe;
- quotient types, where we denote the quotient of a type α by the relation $\sim$ by $\alpha/\sim$, and denote quotient introduction $\alpha \to \alpha/\sim$ by $x \mapsto [x]$;
- a *definitional* reduction rule that if $f : \alpha \to \beta$ lifts to $g : \alpha/\sim \to \beta$, then $g\ [x] = f\ x$.

We write $\mathsf{Type}\ u = \mathsf{Sort}\ (u+1)$ and $\mathsf{Prop} = \mathsf{Sort}\ 0$ for conciseness. We stipulate the following axioms.

- propositional extensionality: that if $p \Leftrightarrow q$ then we have $p = q$;
- a form of the axiom of choice: a function for each type α that maps a proof that α is nonempty to some $x : \alpha$.

Lean's dependent type theory satisfies these constraints. It is known that such a type theory can be modelled in $\mathsf{ZFC}+\{\text{there are } n \text{ inaccessible cardinals} \mid n < \omega\}$ (see `https://github.com/digama0/lean-type-theory/releases`).

We model cardinals and ordinals as quotients over a universe of types. However, apart from this, we make no direct use of higher universes, so the proof can be expected to work with no inaccessible cardinal assumptions.

Note that we use the notation π_k for the kth projection of a tuple or structure with at least k entries.

Chapter 1

Definitions and results from mathlib

In this section, we state a number of well-known definitions and results from the community repository mathlib. The definitions are included so that the representations of types we use are clear.

1.1 Sets, groups, and supports

Definition 1.1. A *set* of a type α is a function $\alpha \to \mathsf{Prop}$. The type of sets of α is denoted $\mathsf{Set}\ \alpha$.

Definition 1.2. The *pointwise image* of a set $s : \mathsf{Set}\ \alpha$ under a function $f : \alpha \to \beta$ is denoted $f''s = \{y : \beta \mid \exists x \in s, y = f\ x\}$. The *preimage* of a set $t : \mathsf{Set}\ \beta$ under f is denoted $f^{-1}{}'t = \{x : \alpha \mid f\ x \in t\}$.

Definition 1.3. The *symmetric difference* of two sets $s, t : \mathsf{Set}\ \alpha$ is defined by $s \triangle t = (s \setminus t) \cup (t \setminus s)$.

Definition 1.4. A *group action* of G on α is a function $G \to \alpha \to \alpha$ denoted by $\cdot$ such that for all $x, y : G, a : \alpha$, we have $(x \cdot y) \cdot a = x \cdot (y \cdot a)$.

Definition 1.5. Let G be a group that acts on α and β. Let s be a set of α, and let $b : \beta$. We say that s *supports* b if for all $x \in G$, we have $x \cdot b = b$ whenever $x \cdot a = a$ for all $a \in s$.

Lemma 1.6. Let $s : \mathsf{Set}\ \alpha$ support $b : \beta$ under actions of G. Then for $x, y \in G$, $x \cdot b = y \cdot b$ whenever $x \cdot a = y \cdot a$ for all $a \in s$.

Proof. Apply the definition of a support to $y^{-1} \cdot x$. □

1.2 Cardinals and ordinals

Definition 1.7. An *equivalence* between two types α and β, denoted $e : \alpha \simeq \beta$, is a pair of functions $f : \alpha \to \beta, g : \beta \to \alpha$ that are inverses of each other. Equivalences $e : \alpha \simeq \beta$ naturally coerce to their underlying function $f : \alpha \to \beta$. We use the syntax e^{-1} to denote the inverse equivalence $\beta \simeq \alpha$ constructed from g and f.

Remark. $(e^{-1})^{-1} = e$ holds definitionally.

Definition 1.8. The type of *permutations* of a type α is $\alpha \simeq \alpha$, denoted $\mathsf{Perm}\ \alpha$.

Definition 1.9. The type of *cardinals* is the quotient of Type by the equivalence relation $\sim$, where $\alpha \sim \beta$ if $\alpha \simeq \beta$ is nonempty. We denote the cardinal of a type by $\#\alpha = [\alpha]$.

Definition 1.10. Let $r : \alpha \to \alpha \to \mathsf{Prop}$ be a relation on α. We say that $x : \alpha$ is *r-accessible* if for all y with $r\ y\ x$, we have that y is r-accessible. A relation $r : \alpha \to \alpha \to \mathsf{Prop}$ is *well-founded* if every element is accessible.

Remark. This is a constructive form of well-foundedness that behaves very nicely in Lean's type system.

Theorem 1.11 (well-founded recursion)**.** Let r be a well-founded relation on α. Let $C : \alpha \to \mathsf{Sort}\ u$ be a motive for the recursion. Let h have type

$$\prod_{(x:\alpha)} \left(\prod_{(y:\alpha)} r\ y\ x \to C\ y \right) \to C\ x$$

Then we can construct $C\ x$ for each $x : \alpha$.

Remark. More rigorously, well-founded recursion over r is a function of type

$$\prod_{(C:\alpha\to\mathsf{Sort}\ u)} \left[\left(\prod_{(x:\alpha)} \left(\prod_{(y:\alpha)} r\ y\ x \to C\ y \right) \to C\ x \right) \to \prod_{(x:\alpha)} C\ x \right]$$

Setting $u = 0$ gives well-founded induction. This result is obtained by recursion over accessibility, which is an inductive type.

Definition 1.12. A relation is a *well-order* if it is trichotomous, transitive, and well-founded.

Definition 1.13. Let α, β be endowed with relations r, s. An equivalence $e : \alpha \simeq \beta$ is an *order isomorphism* if for each $x, y : \alpha$, we have $s\ (e\ x)\ (e\ y) \Leftrightarrow r\ x\ y$.

Definition 1.14. The type of *ordinals* is the quotient of the type of well-ordered elements of Type by the equivalence relation $\sim$, where $\alpha \sim \beta$ if the type of order isomorphisms of α and β is nonempty.

Standard properties of cardinals and ordinals are assumed.

Definition 1.15. A *partial value* of a type α is a proposition p and a function $p \to \alpha$. That is, if $h : p$ is a proof of p, then we can acquire a value $x : \alpha$. The type of such values is denoted $\mathsf{Part}\ \alpha$.

Definition 1.16. A *partial function* from α to β is a function from α to partial values of type β. The type of such values is denoted $\alpha \dashrightarrow \beta$.

We use standard function notation on partial functions.

Remark. By propositional extensionality, all empty partial values are equal, and all inhabited partial values with equal values are equal.

1.3 Quivers and paths

Definition 1.17. A *quiver* on a type α of vertices assigns to every pair $x, y : \alpha$ of vertices a type $\mathrm{Hom}(x, y)$ of arrows from x to y.

Definition 1.18. A *path* in a quiver between two vertices $x, y : \alpha$ is a finite list of vertices beginning with x and ending with y, connecting each pair of adjacent vertices a, b with an element of $\mathrm{Hom}(a, b)$. The type of such paths is written $x \rightsquigarrow y$. The empty path is written $\varnothing : x \rightsquigarrow x$. The *composition* of paths $p : x \rightsquigarrow y, q : y \rightsquigarrow z$ is denoted by $p \gg q : x \rightsquigarrow z$.

Remark. In mathlib, paths are defined as an inductive type. If there is exactly one morphism in a given hom-set $\mathrm{Hom}(a, b)$, it is denoted $a \to b$. We will implicitly convert morphisms $e : \mathrm{Hom}(a, b)$ to their *corresponding paths* $e : a \rightsquigarrow b$.

Definition 1.19. The *length* of a path is the number of arrows in that path, or exactly one less than the number of vertices in the list.

Chapter 2

The base type (`ConNF.BaseType`)

We describe the base level of our construction, as well as all of the other objects that can be described outside the main induction.

2.1 Model parameters

Definition 2.1. A set of *model parameters* is

- a type λ endowed with a well-order;
- a type κ;
- a type μ endowed with a well-order,

such that

(i) the order type of λ is a nonzero limit ordinal;

(ii) the order type of μ is the initial ordinal corresponding to the cardinal $\#\mu$;

(iii) $\#\mu$ is a strong limit cardinal;

(iv) $\#\lambda < \#\kappa < \#\mu$;

(v) the cofinality of the initial ordinal corresponding to $\#\mu$ is at least $\#\kappa$.

Lemma 2.2. There exists a set of model parameters.

Proof. Take $\lambda = \aleph_0, \kappa = \aleph_1, \mu = \beth_{\omega_1}$. These form a set of model parameters by standard properties of cardinals. ☐

Every definition and theorem following this will implicitly assume a set of model parameters as an additional argument.

Lemma 2.3. (i) λ, κ, μ are infinite.

(ii) λ and μ have no maximal element.

Proof. Part (i). λ is a nonzero limit, hence is infinite; condition (iv) then guarantees the result for κ, μ. *Part (ii).* Initial ordinals have no maximal element. □

Definition 2.4. The type of *type indices*, denoted $\lambda^\bot$, is λ together with a symbol denoted $\bot$. The order on $\lambda^\bot$ places $\bot$ below all elements of λ.

Lemma 2.5. $\#\lambda^\bot = \#\lambda$.

Proof. $\#\lambda^\bot = \#\lambda + 1$, and λ is infinite by lemma 2.3(i). □

Lemma 2.6. The type indices are well-ordered.

Proof. They are clearly linearly ordered, and the relation $<$ is well-founded. □

Lemma 2.7. For $x : \mu$, $\#\{y \mid y < x\} < \#\mu$ and $\#\{y \mid y \leq x\} < \#\mu$.

Proof. Definition 2.1 requires that the order type of μ is an initial ordinal, so we have $\#\{y \mid y < x\} < \#\mu$. Then $\#\{y \mid y \leq x\} = \#\{y \mid y < x\} + \#\{x\} < \#\mu$ as $\#\mu$ is infinite by lemma 2.3(i). □

2.2 Smallness

Definition 2.8. A set s of any type α is called *small* if $\#s < \#\kappa$.

Remark. Note that cardinals are defined on types and not sets: technically we mean that the cardinality of the subtype $\{x : \alpha \mid x \in s\}$ is less than $\#\kappa$.

Lemma 2.9. Let $f : \alpha \to \beta$ and $s, t : \text{Set}\ \alpha$. Then,

(i) the empty set is small;

(ii) singletons are small;

(iii) if $s \subseteq t$ and t is small then s is small;

(iv) if s, t are small then $s \cup t$ is small;

(v) if s, t are small then $s \triangle t$ is small;

(vi) if s is small then $s \triangle t$ is small if and only if t is small;

(vii) if ι is a type with $\#\iota < \#\kappa$ and $g : \iota \to \text{Set}\ \alpha$ with $g\ i$ small for each $i \in \iota$, then $\bigcup_{i:\iota} g\ i$ is small;

(viii) if s is small then $f''s$ is small;

(ix) if $s : \text{Set}\ \beta$ is small and f is injective then $f^{-1\prime}s$ is small;

(x) if $t : \text{Set}\ \beta$ is small, f is injective, and $f''s \subseteq t$, then s is small;

(xi) if f is a partial function and s is small then $f''s$ is small.

Proof. (i) $\#\{\} = 0 < \aleph_0 \leq \#\kappa$ by lemma 2.3.

(ii) $\#\{x\} = 1 < \aleph_0 \leq \#\kappa$ by lemma 2.3.

(iii) Follows from transitivity.

(iv) $\aleph_0 \leq \#\kappa$ so $\#\kappa$ is additively closed.

(v) $s \mathbin{\triangle} t \subseteq s \cup t$ so done by (iii).

(vi) $s \mathbin{\triangle} t \mathbin{\triangle} s = t$ so done by applying (iv) twice.

(vii) Follows since κ is regular by definition 2.1.

(viii) The set $f''s$ injects into s so $\#(f''s) \leq \#s$.

(ix) The set $f^{-1}{}'s$ injects into s if f is injective.

(x) Follows from (iii) and (ix), as $f^{-1}{}'(f''s) = s$ for injective f.

(xi) By (viii), the set of partial values of type β in the range of f is small, by treating f as a total function $\alpha \to \mathsf{Part}\ \beta$. The result then holds by applying (x) to the natural injection $\iota : \beta \to \mathsf{Part}\ \beta$.

□

Definition 2.10. Sets are *near* if their symmetric difference is small.

Lemma 2.11. Let $f : \alpha \to \beta$ and $s, t, u : \mathsf{Set}\ \alpha$.

(i) s is near s;

(ii) if s is near t then t is near s;

(iii) if s is near t and t is near u then s is near u;

(iv) if s is near t then $f''s$ is near $f''t$;

(v) if s is small, then s is near t if and only if t is small;

(vi) if s is near t and $\#\kappa \leq \#s$, then $\#\kappa \leq \#t$;

(vii) if s is near t and $\#\kappa \leq \#s$, then $\#\kappa \leq \#(s \cap t)$.

Proof. (i) Follows from lemma 2.9(i).

(ii) The symmetric difference is commutative.

(iii) Follows from lemma 2.9(iii, iv) and the fact that $s \mathbin{\triangle} u \subseteq (s \mathbin{\triangle} t) \cup (t \mathbin{\triangle} u)$.

(iv) Follows from lemma 2.9(iii, viii) and the fact that $(f''s) \mathbin{\triangle} (f''t) \subseteq f''(s \mathbin{\triangle} t)$.

(v) Follows from lemma 2.9(vi).

(vi) Suppose not, so $\#t < \#\kappa$. Then as s is near t, s is small, contradicting the assumption.

(vii) Suppose not, so $\#(s \cap t) < \#\kappa$. As s is near t, the set $(s \cup t) \setminus (s \cap t)$ is small. But

$$\#(s \cup t) \leq \#((s \cup t) \setminus (s \cap t)) + \#(s \cap t)$$

Both summands on the right-hand side are less than $\#\kappa$, so $s \cup t$ must be small. But this contradicts the assumption that $\#\kappa \leq \#s$.

□

2.3 Litters

Definition 2.12. A *litter* is a triple $L = \langle \nu, \beta, \gamma \rangle$ with $\nu : \mu, \beta : \lambda^\perp, \gamma : \lambda$, such that $\beta \neq \gamma$. The type of litters is denoted $\mathcal{L}$.

Lemma 2.13. $\#\mathcal{L} = \#\mu$.

Proof. Note that $\#(\mu \times \lambda^\perp \times \lambda) = \#\mu$ so $\#\mathcal{L} \leq \#\mu$. But $\#\mu \leq \#\mathcal{L}$ by considering the injection $\nu \mapsto \langle \nu, \perp, 0 \rangle$, so the result follows by antisymmetry. □

2.4 Atoms

Definition 2.14. The type of *atoms* is $\mathcal{A} = \mathcal{L} \times \kappa$. We denote the first projection of an atom a by $\pi_1\, a = a^\circ$.

Lemma 2.15. $\#\mathcal{A} = \#\mu$.

Proof. $\#\mathcal{L} = \#\mu$ by lemma 2.13, and $\#\aleph_0 \leq \#\kappa < \#\mu$ by definition 2.1. □

Definition 2.16. The *litter set* of a litter L is the set of atoms with first projection equal to L, denoted $\mathcal{A}_L$.

Lemma 2.17. (i) $\#\mathcal{A}_L = \#\kappa$;

(ii) $a \in \mathcal{A}_L \Leftrightarrow a^\circ = L$;

(iii) the litter sets are pairwise disjoint.

Proof. (i) Each litter set is naturally in bijection with κ.

(ii) If an atom a is in $\mathcal{A}_L$ and $\mathcal{A}_{L'}$, then $a^\circ = L$ and $a^\circ = L'$ so $L = L'$.

□

2.5 Near-litters

Definition 2.18. A set of atoms *is a near-litter* to a given litter L if it is near the litter set of L.

Lemma 2.19. (i) $\mathcal{A}_L$ is a near-litter to L;

(ii) if s, t are near-litters to L then s is near t;

(iii) if s is a near-litter to L, $\#s = \#\kappa$;

(iv) a set cannot be a near-litter to two different litters;

(v) there are μ near-litters to a given litter.

Proof. (i) Direct from lemma 2.11(i).

(ii) Follows from lemma 2.11(iii).

(iii) We have

$$\#s \leq \#(s \setminus \mathcal{A}_L) + \#(\mathcal{A}_L)$$

The first term is less than $\#\kappa$ by lemma 2.9(iii); the second is exactly $\#\kappa$ by lemma 2.17(i). Thus $\#s \leq \#\kappa$. Suppose $\#s < \#\kappa$. Note that

$$\#\kappa = \#\mathcal{A}_L \leq \#(\mathcal{A}_L \setminus s) + \#s$$

But $\#s < \#\kappa$ by assumption, and $\#(\mathcal{A}_L \setminus s) < \#\kappa$ by lemma 2.17(i). This gives a contradiction.

(iv) First note that if $\mathcal{A}_L$ is a near-litter to L', then $L = L'$. Suppose $L \neq L'$. Then $\mathcal{A}_L \subseteq \mathcal{A}_L \mathbin{\triangle} \mathcal{A}_{L'}$. Hence the cardinality of $\mathcal{A}_L \mathbin{\triangle} \mathcal{A}_{L'}$ is at least $\#\kappa$, contradicting nearness. For general sets, if s is a near-litter to L and L', we must have that $\mathcal{A}_L$ is a near-litter to L', reducing to the original case.

(v) We argue by antisymmetry. First, we show that the number of near-litters to L is at most $\#\mu$. Note that as $\#\mu$ is a strong limit cardinal, the type of sets (of atoms, say) of size less than the cofinality of $\#\mu$ also has cardinality $\#\mu$. But as the cofinality of $\#\mu$ is at least $\#\kappa$, it suffices to show an injection from the type of near-litters to L to the type of sets of atoms of size at most $\#\kappa$, which can be done by the natural coercion.

Conversely, we need an injection from $\mathcal{A}$ to the type of near-litters to L. The map $a \mapsto \mathcal{A}_L \mathbin{\triangle} \{a\}$ suffices.

□

Definition 2.20. A *near-litter* is a dependent pair $\langle L, s\rangle$, where L is a litter and s is a set of atoms that is a near-litter to L. We denote the type of near-litters by $\mathcal{N}$. We define a natural injective coercion from a near-litter to its second component; this is often used in extensionality arguments.

Remark. Retaining the data of which litter a given near-litter is near to allows us to get better definitional properties.

Definition 2.21. The first projection $\pi_1 : \mathcal{N} \to \mathcal{L}$ is written with a superscript circle: $N \mapsto N^\circ$. The injection $\mathsf{NL} : \mathcal{L} \to \mathcal{N}$ is defined by $\mathsf{NL}\, L = \langle L, \mathcal{A}_L\rangle$, sending a litter to its *associated near-litter*.

Lemma 2.22. Let $N : \mathcal{N}$. Then $N \mathbin{\triangle} \mathcal{A}_{N^\circ}$ is small.

Proof. Suppose $N = \langle L, s\rangle$. Then s is near to $\mathcal{A}_L$ as required. □

Lemma 2.23. $\#\mathcal{N} = \#\mu$.

Proof.

$$\begin{aligned}
\#\mathcal{N} &= \# \sum_{(L:\mathcal{L})} \{s : \mathsf{Set}\,\mathcal{A} \mid s \text{ near } \mathcal{A}_L\} \\
&= \sum_{(L:\mathcal{L})} \#\{s : \mathsf{Set}\,\mathcal{A} \mid s \text{ near } \mathcal{A}_L\} \\
\text{(lemma 2.19(v))} \quad &= \sum_{(L:\mathcal{L})} \#\mu \\
&= \#\mathcal{L} \cdot \#\mu \\
\text{(lemma 2.13)} \quad &= \#\mu \cdot \#\mu \\
\text{(lemma 2.3)} \quad &= \#\mu
\end{aligned}$$

□

Lemma 2.24. Let $N : \mathcal{N}$. Then $\#N = \#\kappa$.

Proof. We argue by antisymmetry that

$$\#(N \mathbin{\triangle} \mathcal{A}_{N^\circ} \mathbin{\triangle} \mathcal{A}_{N^\circ}) = \#\kappa$$

First, we show that this is at most $\#\kappa$. By monotonicity it suffices to show that

$$\#((N \mathbin{\triangle} \mathcal{A}_{N^\circ}) \cup \mathcal{A}_{N^\circ}) \leq \#\kappa$$

By lemma 2.22 and lemma 2.17(i), this holds.

Conversely, suppose $N \mathbin{\triangle} \mathcal{A}_{N^\circ} \mathbin{\triangle} \mathcal{A}_{N^\circ}$ is small. Then as $N \mathbin{\triangle} \mathcal{A}_{N^\circ}$ is small, by lemma 2.9(vi) we must have that $\mathcal{A}_{N^\circ}$ is small, which is a contradiction. □

Lemma 2.25. Let $N_1, N_2 : \mathcal{N}$. Then if $N_1^\circ = N_2^\circ$, their intersection $N_1 \cap N_2$ is nonempty.

Proof. First, note that N_1 is near N_2, so $N_2 \setminus N_1$ is small. Suppose the intersection is empty, then $N_2 \setminus N_1 = N_2$. But then N_2 would be small, contradicting lemma 2.24. □

2.6 Near-litter permutations

Definition 2.26. A *near-litter permutation* is a pair $\pi = \langle \pi^A, \pi^L \rangle$ where $\pi^A : \mathsf{Perm}\,\mathcal{A}$ and $\pi^L : \mathsf{Perm}\,\mathcal{L}$, such that if s is a near-litter to L, $\pi^{A\,\prime\prime} s$ is a near-litter to $\pi^L\, L$. Thus a near-litter permutation induces a permutation of near-litters. The type of near-litter permutations is denoted $\mathcal{P}$.

We suppress the superscripts on near-litter permutations and use function application syntax for the action of a near-litter permutation on atoms, litters, and near-litters: for example, $\pi^A\ a = \pi\ a$. Note that the action on litters is 'rough': we map litters to litters and not near-litters. If the precise image of a litter L under a permutation π is desired, it can be obtained using $\pi\ (\mathsf{NL}\ L)$.

Lemma 2.27. If the atom permutations of two near-litter permutations agree, then the permutations are equal.

Proof. Let $L : \mathcal{L}$ and π, π' be near-litter permutations. The values of $\pi\ (\mathsf{NL}\ L)$ and $\pi'\ (\mathsf{NL}\ L)$ depend only on the atom maps in question. The result then follows from lemma 2.19(iv). □

Lemma 2.28. The near-litter permutations form a group with identity id and operation $\circ$.

Lemma 2.29. Let π be a near-litter permutation and let N be a near-litter. Then, the following equality of sets holds.

$$\pi\ N = (\pi\ (\mathsf{NL}\ N^\circ)) \mathbin{\triangle} (\pi''(\mathcal{A}_{N^\circ} \mathbin{\triangle} N))$$

Proof. After applying set extensionality, this proof becomes simple case checking. □

Chapter 3

Tangled structure

We now describe how the different levels of our structure are to be tangled together.

3.1 Extended type indices

Definition 3.1. We define a quiver structure on type indices. For α, β type indices, $\mathrm{Hom}(\alpha, \beta)$ is the type $\beta < \alpha$. Thus, there is a morphism $\alpha \to \beta$ if and only if $\beta < \alpha$, and all such morphisms are equal by proof irrelevance.

Definition 3.2. A path from a type index to $\bot$ is called an *extended (type) index*.

Lemma 3.3. (i) If $A : \alpha \rightsquigarrow \beta$ is a path of type indices, $\beta \leq \alpha$.

(ii) If $A : \alpha \rightsquigarrow \alpha$, then A is the empty path.

(iii) If $\alpha : \lambda$, then the extended index $A : \alpha \rightsquigarrow \bot$ has nonzero length.

Proof. (i) Induction on A.

(ii) If A were nonempty, it would be of the form $B \gg h$ where $B : \alpha \rightsquigarrow \beta$ and $h : \beta \to \alpha$. By (i), $\beta \leq \alpha$, but h is the fact that $\alpha < \beta$, giving a contradiction.

(iii) $\alpha \neq \bot$ so A is not the empty path.

□

Lemma 3.4. $0 \neq \#(\alpha \rightsquigarrow \bot) \leq \#\lambda$.

Proof. There is at least one extended index for each α: the nil path for $\alpha = \bot$ or the one-arrow path otherwise. For the other inequality, there is an injection from paths $\alpha \rightsquigarrow \bot$ to lists, so it suffices to show that the type of lists of type indices has cardinality at most $\#\lambda$. But $\aleph_0 \leq \#\lambda$ by 2.3(i), so it suffices to show that $\#\lambda^\bot \leq \#\lambda$, which is 2.5. □

3.2 Pretangles

Omitted; currently unused.

3.3 Trees

Definition 3.5. Let α be a type index and τ be a type. Then the type of *α-trees of τ* is

$$\mathsf{Tree}_\tau\ \alpha = (\alpha \rightsquigarrow \bot) \to \tau$$

Definition 3.6. There is a natural equivalence $\mathsf{Tree}_\tau\ \bot \simeq \tau$ given by $a \mapsto a\ \varnothing$ and $a \mapsto (A \mapsto a)$.

Definition 3.7. Let τ have a group structure. Then we endow $\mathsf{Tree}_\tau\ \alpha$ with a group structure by defining

$$(a_1 \cdot a_2)\ A = a_1\ A \cdot a_2\ A$$

This makes the equivalence $\mathsf{Tree}_\tau\ \bot \simeq \tau$ into an isomorphism of groups.

Definition 3.8. The *derivative* functor maps paths of type indices $\alpha \rightsquigarrow \beta$ to functions $\mathsf{Tree}_\tau\ \alpha \to \mathsf{Tree}_\tau\ \beta$. Applying it to a path A and tree a gives the tree $B \mapsto a\ (A \gg B)$. The application of this functor to a path A and tree a is denoted using a subscript, so

$$a_A = (B \mapsto a\ (A \gg B))$$

Remark. This is a functor from the category of type indices where the morphisms are the decreasing paths (i.e. the category where morphisms are elements of $\alpha \rightsquigarrow \beta$ for $\alpha, \beta : \lambda^\bot$) to the category of all trees of a fixed type τ, where the morphisms are functions. The map of objects is simply $\alpha \mapsto \mathsf{Tree}_\tau\ \alpha$, or more concisely, just Tree_τ. If τ has a group structure, this map preserves multiplication. This means that we can treat this as a functor to the category of all trees on τ where the morphisms are group homomorphisms.

Lemma 3.9. The derivative map is a functor in the sense described above:

(i) $a_\varnothing = a$;

(ii) $(a_A)_B = a_{A \gg B}$;

(iii) $(a_1 \cdot a_2)_A = {a_1}_A \cdot {a_2}_A$.

In addition, if $A : \alpha \rightsquigarrow \bot$, then a_A and $a\ A$ are equal up to the equivalence in definition 3.6.

Proof. All of these results follow from the basic laws of quivers. $\square$

Definition 3.10. If τ has a group action on some type σ, we pull it back under the equivalence given in definition 3.6 to give $\mathsf{Tree}_\tau\ \bot$ the same action on σ.

3.4 Structural permutations

Definition 3.11. For α a type index, an *α-structural permutation* is an α-tree of near-litter permutations. The type of α-structural permutations is denoted Str_α, so

$$\mathsf{Str}_\alpha = \mathsf{Tree}_{\mathcal{P}}\ \alpha = (\alpha \rightsquigarrow \bot) \to \mathcal{P}$$

3.5 Supports and support conditions

Definition 3.12. For α a type index, the type of *α-support conditions* is

$$(\alpha \rightsquigarrow \bot) \times (\mathcal{A} \oplus \mathcal{N})$$

That is, an α-support condition is an α-extended type index, together with an atom or near-litter.

Lemma 3.13. For each α, there are $\#\mu$ α-support conditions.

Proof. By lemma 2.15 and lemma 2.23, we must show that

$$\#(\alpha \rightsquigarrow \bot) \cdot (\#\mu + \#\mu) = \#\mu$$

This follows from standard properties of cardinals and lemma 3.4. □

Definition 3.14. α-structural permutations π act on α-support conditions by mapping

$$\langle A, x \rangle \mapsto \langle A, \pi\ A\ x \rangle$$

where the action of a near-litter permutation on an element of $\mathcal{A} \oplus \mathcal{N}$ is defined in the natural way.

Definition 3.15. Let α be a type index, τ be a type, $x : \tau$, and G be a group that acts on τ. A *support* for x under this action is a small set of α-support conditions that support x (in the sense of definition 1.5). An object is said to be *supported* if its type of supports is nonempty.

Chapter 4

f-maps

We now describe the mechanism for creating the f-maps, and begin the main recursion.

4.1 Position functions

Definition 4.1. Let α, β be types. A *position function* on α taking values in β is an injection $\alpha \to \beta$. We denote all position functions by $\iota : \alpha \to \beta$.

Let α have a position function taking values in β, and suppose β has a relation $<$.

Definition 4.2. We then define the relation $<$ on α by $x < y \Leftrightarrow \iota\, x < \iota\, y$.

Lemma 4.3. If β is well-ordered, then α is well-ordered.

Proof. Trichotomy and transitivity are clear. The inverse image of a well-founded relation is well-founded by induction on accessibility, completing the proof. □

4.2 Hypotheses

Definition 4.4. Let α be a type index. *Tangle data* at level α is

- a type τ_α of *tangles*;
- a type All_α of *allowable permutations*;
- a group structure on All_α;
- a group homomorphism $\mathsf{str}_\alpha : \mathsf{All}_\alpha \to \mathsf{Str}_\alpha$;
- a group action of All_α on τ_α written by juxtaposition; and
- a function assigning to each $t : \tau_\alpha$ a support for it under the action of All_α, called its *designated support*, denoted DS_α.

Definition 4.5. Let α be a type index with tangle data. We say that level α has *positioned tangles* if there is a position function on τ_α taking values in μ. The existence of this position function proves that there are at most $\#\mu$ tangles at level α.

Definition 4.6. Let $\alpha : \lambda$ be a proper type index with tangle data. We say that we have *typed objects* at level α if we have

- an injection $\mathsf{typed}^a_\alpha : \mathcal{A} \to \tau_\alpha$ called the *typed atom* map; and
- an injection $\mathsf{typed}^N_\alpha : \mathcal{N} \to \tau_\alpha$ called the *typed near-litter* map, that commutes with allowable permutations in the sense that for all $\rho : \mathsf{All}_\alpha, N : \mathcal{N}$, we have

$$\rho\,(\mathsf{typed}^N_\alpha\,N) = \mathsf{typed}^N_\alpha\,(\rho\,(\alpha \to \bot)\,N)$$

Definition 4.7. An assignment of *base positions* is a pair of position functions on $\mathcal{A}$ and $\mathcal{N}$ both taking values in μ, such that

- $a \in \mathcal{A}_L \implies \iota\,(\mathsf{NL}\,L) < \iota\,a$;
- $\iota\,(\mathsf{NL}\,N^\circ) \leq \iota\,N$;
- $a \in N \mathbin{\triangle} \mathcal{A}_{N^\circ} \implies \iota\,a < \iota\,N$;
- $\iota\,a \neq \iota\,N$.

Remark. At the moment, we define no coherence conditions between the position function, the typed objects, and the base positions data. Later, they will be tied together.

Definition 4.8. Tangle data at level $\alpha = \bot$ is defined as follows.

- $\tau_\bot = \mathcal{A}$;
- $\mathsf{All}_\bot = \mathcal{P}$;
- the homomorphism $\mathsf{All}_\bot \to \mathsf{Str}_\bot$ is given by definition 3.6;
- the designated support of an atom $a : \mathcal{A}$ is $\{\langle a, \varnothing\rangle\}$.

4.3 Construction

Lemma 4.9. Let α and β be types, and let α be well-ordered. Let $d : \alpha \to \mathsf{Set}\,\beta$ assign to each $x : \alpha$ a set of *denied sets*. Suppose that for each $x : \alpha$, we have

$$\#\{y : \alpha \mid y < x\} + \#(d\,x) < \#\beta$$

Then there is an injective function $f : \alpha \to \beta$ with the property that for each $x : \alpha$, $f\,x \notin d\,x$.

Proof. For a given $x : \alpha$, if we have already constructed $f\,y$ for $y < x$, we can pick a value for $f\,x$ not in $d\,x$ or equal to any $f\,y$, as

$$\begin{aligned} \#(f''\{y : \alpha \mid y < x\} \cup d\,x) &\leq \#(f''\{y : \alpha \mid y < x\}) + \#(d\,x) \\ &\leq \#\{y : \alpha \mid y < x\} + \#(d\,x) \\ &< \#\beta \end{aligned}$$

We have thus constructed $f : \alpha \to \beta$ satisfying the property that $f\,x \notin d\,x$. For injectivity, suppose $x \neq y : \alpha$. Then either $x < y$ or $y < x$; assume the latter without loss of generality. The construction of $f\,x$ was done under the constraint $f\,x \neq f\,z$ for each $z < x$, giving the result as required. ☐

Let $\beta : \lambda^\perp$ and $\gamma : \lambda$ with $\beta \neq \gamma$. Let β and γ have tangle data and positioned tangles. Let γ have typed objects.

Definition 4.10. Construct the function $d : \tau_\beta \to \mathsf{Set}\,\mu$ by

$$\begin{aligned}\nu \in d\,t \Leftrightarrow (\exists N : \mathcal{N},\, N^\circ = \langle \nu, \beta, \gamma \rangle \wedge \iota\,(\mathsf{typed}^N_\gamma\; N) \leq \iota\,t)\\ \vee\,(\beta = \bot \wedge \iota\,(\mathsf{typed}^N_\gamma\;(\mathsf{NL}\,\langle \nu, \bot, \gamma \rangle)) \leq \iota\,t)\end{aligned}$$

Lemma 4.11. Let $t : \tau_\beta$. Then

(i) $\#\{t' : \tau_\beta \mid t' < t\} < \#\mu$;

(ii) $\#\{t' : \tau_\gamma \mid \iota\,t' \leq \iota\,t\} < \#\mu$.

Proof. Both proofs follow the same strategy. First, we use lemma 2.7 to reduce to showing that

$$\#\{t' : \tau_\beta \mid t' < t\} \leq \#\{\nu : \mu \mid \nu < \iota\,t\}$$

and

$$\#\{t' : \tau_\gamma \mid \iota\,t' < \iota\,t\} \leq \#\{\nu : \mu \mid \nu \leq \iota\,t\}$$

These inequalities of cardinals can be easily shown by proving that the injection n has the correct codomain in each case. □

Lemma 4.12. Let $t : \tau_\beta$. Then $\#\{t' : \tau_\beta \mid t' < t\} + \#(d\,t) < \#\mu$.

Proof. By lemma 4.11(i), it suffices to show $\#(d\,t) < \#\mu$. We show that there are less than μ positions that satisfy each of the two conditions in definition 4.10.

For the first condition, we must show that

$$\#\{\nu : \mu \mid \exists N : \mathcal{N},\, N^\circ = \langle \nu, \beta, \gamma \rangle \wedge \iota\,(\mathsf{typed}^N_\gamma\; N) \leq \iota\,t\} < \#\mu$$

By lemma 4.11(ii) it suffices to produce an injection

$$\{\nu : \mu \mid \exists N : \mathcal{N},\, N^\circ = \langle \nu, \beta, \gamma \rangle \wedge \iota\,(\mathsf{typed}^N_\gamma\; N) \leq \iota\,t\} \to \{t' : \tau_\gamma \mid \iota\,t' \leq \iota\,t\}$$

This injection is given by mapping ν to $\mathsf{typed}^N_\gamma\; N$ where N is chosen such that $N^\circ = \langle \nu, \beta, \gamma \rangle$ and $\iota\,(\mathsf{typed}^N_\gamma\; N) \leq \iota\,t$. It can be seen that this is an injection as the typed near-litter map is injective.

Now suppose $\beta = \bot$. For the second condition, it suffices by lemma 2.7 to produce an injection

$$\{\nu : \mu \mid \iota\,(\mathsf{typed}^N_\gamma\;(\mathsf{NL}\,\langle \nu, \bot, \gamma \rangle)) \leq \iota\,t\} \to \{\nu : \mu \mid \nu \leq \iota\,t\}$$

In this case, we map ν to $\iota\,(\mathsf{typed}^N_\gamma\;(\mathsf{NL}\,\langle \nu, \bot, \gamma \rangle))$. This is also injective, as required. □

Definition 4.13. The *f-map* from β to γ is the function $f_{\beta,\gamma} : \tau_\beta \to \mathcal{L}$ defined by

$$f_{\beta,\gamma}\;t = \langle g\,t, \beta, \gamma \rangle$$

where g is chosen by applying lemma 4.9 to definition 4.10 and lemma 4.12.

Lemma 4.14. Suppose $f_{\beta,\gamma}\;t = f_{\beta',\gamma'}\;t'$. Then $\beta = \beta'$ and $\gamma = \gamma'$.

Proof. Apply the second and third projections to $f_{\beta,\gamma}\ t$ and $f_{\beta',\gamma'}\ t'$. $\square$

Lemma 4.15. $f_{\beta,\gamma}$ is injective.

Proof. Follows from lemma 4.9. $\square$

Lemma 4.16. Let N be a near-litter with $N^\circ = f_{\beta,\gamma}\ t$. Then $\iota\ t < \iota\,(\mathsf{typed}^N_\gamma N)$.

Proof. Follows from lemma 4.9 and definition 4.10. $\square$

Lemma 4.17. Let a be an atom. Then $\iota\ a < \iota\,(\mathsf{typed}^N_\gamma(\mathsf{NL}\,(f_{\bot,\gamma}\ a)))$.

Proof. Follows from lemma 4.9 and definition 4.10. $\square$

Chapter 5

Construction of new tangles

To do.

Chapter 6

Freedom of action

We prove the freedom of action theorem.

6.1 Basic definitions and results

6.1.1 Local permutations

Definition 6.1. Let α be a type. A *local permutation* on α is a domain $s : \mathrm{Set}\ \alpha$ and two functions $f, g : \alpha \to \alpha$ that map s inside itself and are inverse to each other on s. Such local permutations are denoted $\pi = \langle s, f, g \rangle$. We write

$$\operatorname{dom} \pi = s; \quad \pi\ x = f\ x$$

The inverse local permutation is $\pi^{-1} = \langle s, g, f \rangle$.

Remark. The maps f, g are defined on all of α, but only have nice properties on s. Outside their domain, the values of the functions are unimportant.

Lemma 6.2. Let π, π' be local permutations on α with disjoint domains. Then there is a local permutation defined on $\operatorname{dom} \pi \cup \operatorname{dom} \pi'$, whose action on $\operatorname{dom} \pi$ coincides with that of π, and correspondingly for π'.

Proof. Define

$$s = \operatorname{dom} \pi \cup \operatorname{dom} \pi'; \quad f\ x = \begin{cases} \pi\ x & \text{if } x \in \operatorname{dom} \pi \\ \pi'\ x & \text{otherwise} \end{cases}; \quad g\ x = \begin{cases} \pi^{-1}\ x & \text{if } x \in \operatorname{dom} \pi \\ \pi'^{-1}\ x & \text{otherwise} \end{cases}$$

□

Suppose we have a function $f : \alpha \to \alpha$, and a set s on which f is injective. We will construct a pair of functions g and h that agree with f and its inverse respectively on s, in such a way that forms a local permutation of α. In particular, consider the diagram

$$\cdots \to L\,2 \to L\,1 \to L\,0 \to s \setminus f''s \xrightarrow{f} \cdots \xrightarrow{f} f''s \setminus s \to R\,0 \to R\,1 \to R\,2 \to \cdots$$

To fill in the orbits of f, we construct a sequence of disjoint subsets of α called $L : \mathbb{N} \to \mathsf{Set}\ \alpha$ and $R : \mathbb{N} \to \mathsf{Set}\ \alpha$, where for each $i : \mathbb{N}$,

$$\#(L\ i) = \#(s \setminus f''s); \quad \#(R\ i) = \#(f''s \setminus s)$$

There are natural bijections along this diagram, mapping $L\ (n+1)$ to $L\ n$ and $R\ n$ to $R\ (n+1)$. There are also bijections $f''s \setminus s \to R\ 0$ and $L\ 0 \to s \setminus f''s$. This yields a local permutation defined on

$$s \cup f''s \cup \left(\bigcup_{i:\mathbb{N}} L\ i\right) \cup \left(\bigcup_{i:\mathbb{N}} R\ i\right)$$

We now prove this claim using a number of intermediate lemmas. In this subsection, suppose that

(i) α is a type;

(ii) $f : \alpha \to \alpha$ is a function;

(iii) $s, t : \mathsf{Set}\ \alpha$;

(iv) $\#(s \triangle (f''s)) \le \#t$ and $\aleph_0 \le \#t$;

(v) the sets $s \cup f''s$ and t are disjoint;

(vi) f is injective on s.

We will contain all of the orbits of f in $s \cup f''s \cup t$.

Lemma 6.3. There exists a subset of t with an equivalence e to the type

$$(\mathbb{N} \times (s \setminus f''s)) \oplus (\mathbb{N} \times (f''s \setminus s))$$

We denote this type σ.

Proof. By assumption (iv),

$$\#(s \setminus f''s) + \#(f''s \setminus s) \le \#t$$

As $\aleph_0 \le \#t$,

$$\#\mathbb{N} \cdot (\#(s \setminus f''s) + \#(f''s \setminus s)) \le \#t$$

giving the result. □

For the purposes of this proof, we call this subset the *sandbox* u, so $e : u \simeq \sigma$. Then the set described above as $L\ i$ is the image of the map $x \mapsto \mathsf{inl}\ \langle i, x\rangle$ under the equivalence (where inl denotes the left injection into a sum type), and $R\ i$ similarly with the right injection inr.

Definition 6.4. Define the *shift right* function $r : \sigma \to \alpha$ by

$$\begin{aligned} r\ (\mathsf{inl}\ \langle 0, x\rangle) &= x \\ r\ (\mathsf{inl}\ \langle n+1, x\rangle) &= e^{-1}\ (\mathsf{inl}\ \langle n, x\rangle) \\ r\ (\mathsf{inr}\ \langle n, x\rangle) &= e^{-1}\ (\mathsf{inr}\ \langle n+1, x\rangle) \end{aligned}$$

Similarly define the *shift left* function $l : \sigma \to \alpha$ by

$$\begin{aligned} l\ (\mathsf{inl}\ \langle n, x\rangle) &= e^{-1}\ (\mathsf{inl}\ \langle n+1, x\rangle) \\ l\ (\mathsf{inr}\ \langle 0, x\rangle) &= x \\ l\ (\mathsf{inr}\ \langle n+1, x\rangle) &= e^{-1}\ (\mathsf{inr}\ \langle n, x\rangle) \end{aligned}$$

Definition 6.5. Define $g : \alpha \to \alpha$ by

$$g\,x = \begin{cases} r\,(e\,x) & \text{if } x \in u \\ e^{-1}\,(\mathsf{inr}\,\langle 0, x\rangle) & \text{if } x \in f''s \setminus s \\ f\,x & \text{otherwise} \end{cases}$$

Define $h : \alpha \to \alpha$ by

$$h\,x = \begin{cases} l\,(e\,x) & \text{if } x \in u \\ e^{-1}\,(\mathsf{inl}\,\langle 0, x\rangle) & \text{if } x \in s \setminus f''s \\ f'\,x & \text{otherwise} \end{cases}$$

where $f' : \alpha \to \alpha$ is an inverse to f on s, which exists by injectivity.

We can now prove the result.

Lemma 6.6. Assuming assumptions (i)–(vi), there exists a local permutation π defined on a subset of t whose action agrees with f on s.

Proof. The local permutation in question is $\langle s \cup f''s \cup u, g, h\rangle$. The required properties of a local permutation follow almost immediately from the definitions, although require a lot of case-checking. □

6.1.2 Sublitters

Definition 6.7. A *sublitter* is a litter L and a set of atoms s such that $s \subseteq \mathcal{A}_L$, and $\mathcal{A}_L \setminus s$ is small.

A sublitter has a natural coercion into a set of atoms, and into a near-litter.

Lemma 6.8. The cardinality of any sublitter is $\#\kappa$.

Proof. Follows directly from lemma 2.24. □

Lemma 6.9. Let S, T be sublitters. Then there is an equivalence $S \simeq T$. For each such pair we make a choice of equivalence denoted $\pi_{S,T} : S \simeq T$.

Proof. Both have cardinality $\#\kappa$ by lemma 6.8. □

6.1.3 Hypotheses

We assume an assignment of base positions.

Definition 6.10. Let $\alpha : \lambda$ be a type index with tangle data, typed objects, and positioned tangles. We say that α has *positioned typed objects* if

(i) for all atoms a, $\iota\,a = \iota\,(\mathsf{typed}^a_\alpha\,a)$;

(ii) for all near-litters N, $\iota\,N = \iota\,(\mathsf{typed}^N_\alpha\,N)$;

(iii) for all tangles $t : \tau_\alpha$, α-extended indices A, and atoms a,

$$\langle A, a\rangle \in \mathsf{DS}_\alpha\,t \Rightarrow \iota\,a \leq \iota\,t$$

(iv) for all tangles $t : \tau_\alpha$, α-extended indices A, and near-litters N,

$$\langle A, N \rangle \in \mathsf{DS}_\alpha\, t \Rightarrow \iota\, N \leq \iota\, t$$

Definition 6.11. Let $\alpha : \lambda$. Then *freedom of action data* at level α is

(i) tangle data for all $\beta : \lambda$ with $\beta \leq \alpha$;

(ii) positioned tangles for all $\beta : \lambda$ with $\beta < \alpha$;

(iii) typed objects for all $\beta : \lambda$ with $\beta \leq \alpha$;

(iv) positioned typed objects for all $\beta : \alpha$ with $\beta < \alpha$.

Definition 6.12. Let $\alpha : \lambda$. Then the *freedom of action assumptions* at level α are

(i) freedom of action data at level α;

(ii) for each $\beta, \gamma : \lambda^\perp$ with $\gamma < \beta \leq \alpha$, a one-step derivative homomorphism $\mathsf{All}_\beta \to \mathsf{All}_\gamma$;

(iii) a proof that the above homomorphism commutes with the map $\mathsf{str} : \mathsf{All} \to \mathsf{Str}$;

(iv) a proof for each $\beta \leq \alpha$ that the designated support map commutes with allowable permutations, i.e. for all $t : \tau_\beta$ and $\rho : \mathsf{All}_\beta$, we have

$$\rho\,(\mathsf{DS}_\beta\, t) = \mathsf{DS}_\beta\,(\rho\, t)$$

(v) a proof for each $\beta, \gamma, \delta \leq \alpha$ with $\gamma < \beta$, $\delta < \beta$, $\gamma \neq \delta$, that for all $\rho : \mathsf{All}_\beta$ and $t : \tau_\gamma$,

$$(\rho_\delta)_\perp\,(f_{\gamma,\delta}\, t) = f_{\gamma,\delta}\,(\rho_\gamma\, t)$$

where subscripts denote applications of the derivative map defined in (ii);

(vi) a function that combines a family of allowable permutations $\rho : \prod_{(\perp \leq \gamma < \beta)} \mathsf{All}_\gamma$ into a single β-allowable permutation ρ', given that for each $t : \tau_\gamma$ we have

$$(\rho\;\delta)_\perp\,(f_{\gamma,\delta}\, t) = f_{\gamma,\delta}\,(\rho\;\gamma\; t)$$

in such a way that $\rho'_\gamma = \rho\;\gamma$.

In this chapter, we let $\alpha : \lambda$ and assume freedom of action assumptions at level α. All other type indices mentioned are assumed to be at most α.

Definition 6.13. We can define the derivative functor on allowable permutations by recursion on paths. From now, subscripts on allowable permutations will refer to this map, not the one-step derivative homomorphism in definition 6.12(ii).

Lemma 6.14. Let $A : \beta \rightsquigarrow \gamma$ and $\rho : \mathsf{All}_\beta$. Then $\mathsf{str}_\gamma\, \rho_A = (\mathsf{str}_\beta\, \rho)_A$.

Proof. A simple induction on paths using definition 6.12(iii). □

6.1.4 Constraints

Support conditions can be said to constrain each other in a number of ways. The 'constrains' relation is well-founded.

Definition 6.15. Define a well-order on $\mathcal{A} \oplus \mathcal{L}$ by the pullback of the two base position functions, which were stipulated to be injective and disjoint in definition 4.7.

Definition 6.16. The type of *α-condition positions* is

$$(\mathcal{A} \oplus \mathcal{N}) \times (\alpha \rightsquigarrow \bot)$$

It is well-ordered by the lexicographic order, where the type $\alpha \rightsquigarrow \bot$ is well-ordered arbitrarily.

Definition 6.17. We define a position function that assigns an α-condition position to each α-support condition, by $\langle A, x\rangle \mapsto \langle x, A\rangle$. In this case, we do *not* use the notation $<$ to denote the pullback well-order on α-support conditions.

Definition 6.18. We inductively define the *constrains* relation on β-support conditions, denoted $\prec$, by the following constructors.

(i) Let $A : \beta \rightsquigarrow \bot$ and $a : \mathcal{A}$. Then
$$\langle A, a^\circ\rangle \prec \langle A, a\rangle$$

(ii) Let $A : \beta \rightsquigarrow \bot$ and $N : \mathcal{N}$ with $\mathcal{A}_{N^\circ} \neq N$. Then
$$\langle A, \mathsf{NL}\, N^\circ\rangle \prec \langle A, N\rangle$$

(iii) Let $A : \beta \rightsquigarrow \bot$, $a : \mathcal{A}$, and $N : \mathcal{N}$, such that $a \in \mathcal{A}_{N^\circ} \mathbin{\triangle} N$. Then
$$\langle A, a\rangle \prec \langle A, N\rangle$$

(iv) Let $\gamma, \delta, \varepsilon : \lambda$ with $\delta, \varepsilon < \gamma$ and $\delta \neq \varepsilon$. Let $A : \beta \rightsquigarrow \gamma$, $t : \tau_\delta$, and $c \in \mathsf{DS}_\delta\, t$. Then
$$\langle A \gg (\gamma \to \delta) \gg \pi_1\, c, \pi_2\, c\rangle \prec \langle A \gg (\gamma \to \varepsilon) \gg (\varepsilon \to \bot), f_{\delta,\varepsilon}\, t\rangle$$

(v) Let $\gamma, \varepsilon : \lambda$ with $\varepsilon < \gamma$. Let $A : \beta \rightsquigarrow \gamma$ and $a : \mathcal{A}$. Then
$$\langle A \gg (\gamma \to \bot), a\rangle \prec \langle A \gg (\gamma \to \varepsilon) \gg (\varepsilon \to \bot), f_{\bot,\varepsilon}\, a\rangle$$

Lemma 6.19. Let c, d be β-support conditions. Then $c \prec d \Rightarrow \iota\, c < \iota\, d$.

Proof. It suffices to show that $\iota\,(\pi_1\, c) < \iota\,(\pi_1\, d)$. By analysing each case, this holds by definitions 4.7 and 6.10 and lemmas 4.16 and 4.17. □

Lemma 6.20. The relation $\prec$ is well-founded.

Proof. It is a subrelation of the inverse image of a well-founded relation ($<$ on μ). □

Lemma 6.21. Let $A : \beta \rightsquigarrow \gamma$ and $c \prec d$ be γ-support conditions. Then $\langle A \gg \pi_1\, c, \pi_2\, c\rangle \prec \langle A \gg \pi_1\, d, \pi_2\, d\rangle$.

Proof. Follows from simple case analysis. □

Definition 6.22. We define the relation $<$ on β-support conditions by the reflexive closure of $\prec$. We define $\leq$ by the transitive closure of $<$.

Remark. This relation $<$ is a subrelation of the pullback of $<$ under the position function.

Lemma 6.23. The relation $<$ on β-support conditions is well-founded.

Proof. The transitive closure of a well-founded relation is well-founded. □

Lemma 6.24. Let c be a β-support condition. Then the set $\{d \mid d \prec c\}$ is small.

Proof. First suppose $c = \langle A, a\rangle$ for an atom a. Then $\{d \mid d \prec c\} = \{\langle A, \pi_1\ a\rangle\}$, which is a singleton and thus small by lemma 2.9(ii).

Now suppose $c = \langle A, N\rangle$ for a near-litter N. By lemma 2.9(iv), it suffices to show that the amount of predecessors under each constructor is small, then their union will be small.

Constructor (i) cannot occur. Constructor (ii) yields a singleton, which is small. As $\mathcal{A}_{N^\circ} \triangle N$ is small, the set of predecessors under constructor (iii) is small. As designated supports are small, (iv) yields a small set. Finally, constructor (v) yields a singleton. □

6.1.5 Reductions of supports

Definition 6.25. A support condition is *reduced* if its second component is an atom or a litter.

Definition 6.26. Let S be a set of β-support conditions. The *reflexive transitive closure* of S is

$$\{c \mid \exists d \in S,\ c \leq d\}$$

The *transitive closure* of S is

$$\{c \mid \exists d \in S,\ c < d\}$$

Definition 6.27. Let S be a set of support conditions. The *reduction* of S is the reflexive transitive closure of S, but we only keep reduced conditions. That is,

$$\{c \mid (\exists d \in S,\ c \leq d) \wedge c \text{ reduced}\}$$

We now prove that the reduction of a small set is small.

Definition 6.28. Define the nth closure recursively by

$$\begin{aligned} \mathsf{nthClosure}\ S\ 0 &= S \\ \mathsf{nthClosure}\ S\ (n+1) &= \{c \mid \exists d \in \mathsf{nthClosure}\ S\ n,\ c \prec d\} \end{aligned}$$

Lemma 6.29. Let S be small. Then $\mathsf{nthClosure}\ S\ n$ is small.

Proof. Induction on n using lemma 2.9(vii) and lemma 6.24. □

Lemma 6.30. The reflexive transitive closure of S is the union of the nth closures.

Proof. Use the fact that any element c of the reflexive transitive closure of S gives rise to a finite list of elements starting with c and ending with an element of S, and if c_1 and c_2 are neighbours in the list, then $c_1 \prec c_2$. □

Lemma 6.31. Let S be small. Then the reflexive transitive closure of S is small. Hence the transitive closure and the reduction of S are small as they are subsets.

Proof. Use lemmas 6.29 and 6.30. □

Lemma 6.32. Let $c \in S$. Then the reduction of S supports c under the action of structural permutations.

Proof. Let π be a β-structural permutation, and suppose π fixes every element of the reduction of S. If $c = \langle A, a \rangle$ where a is an atom, c is reduced so is in the reduction of S. If $c = \langle A, \mathsf{NL}\ L \rangle$ where L is a litter, c is again reduced. So consider the case where $c = \langle A, N \rangle$ and $\mathcal{A}_{N^\circ} \neq N$. Applying lemma 2.29, we must show that

$$(\pi_A\ (\mathsf{NL}\ N^\circ)) \triangle (\pi_A{}''(\mathcal{A}_{N^\circ} \triangle N)) = \mathcal{A}_{N^\circ} \triangle (\mathcal{A}_{N^\circ} \triangle N)$$

As $\langle A, \mathsf{NL}\ N^\circ \rangle$ is reduced and $\langle A, a \rangle$ is reduced for each atom in $\mathcal{A}_{N^\circ} \triangle N$, the result holds. □

Definition 6.33. The *reduced support* for a tangle $t : \tau_\beta$ is the reduction of its designated support, which supports it under the action of allowable permutations by lemma 6.32.

6.1.6 Flexibility of litters

Definition 6.34. Let A be a β-extended type index and L a litter. We say that L is *A-inflexible* if either

(i) there exist $\gamma, \delta, \varepsilon : \lambda$ with $\delta, \varepsilon < \gamma$ and $\delta \neq \varepsilon$, and a path $B : \beta \rightsquigarrow \gamma$ and tangle $t : \tau_\delta$, such that

$$A = B \gg (\gamma \to \varepsilon) \gg (\varepsilon \to \bot); \quad L = f_{\delta,\varepsilon}\ t$$

or

(ii) there exist $\gamma, \varepsilon : \lambda$ with $\varepsilon < \gamma$, and a path $B : \beta \rightsquigarrow \gamma$ and atom a, such that

$$A = B \gg (\gamma \to \varepsilon) \gg (\varepsilon \to \bot); \quad L = f_{\bot,\varepsilon}\ a$$

We call (i) the *proper* case and (ii) the *base* case. A litter which is not A-inflexible is called *A-flexible*.

Lemma 6.35. If L is A-inflexible, then L is $B \gg A$-inflexible. Conversely, if L is $B \gg A$-flexible, L is A-flexible.

Proof. Case checking. □

Definition 6.36. *Proper A-inflexible path data* is a tuple $\langle \gamma, \delta, \varepsilon, B \rangle$ where

$$\delta, \varepsilon < \gamma; \quad \delta \neq \varepsilon; \quad A = B \gg (\gamma \to \varepsilon) \gg (\varepsilon \to \bot)$$

Base A-inflexible path data is a tuple $\langle \gamma, \varepsilon, B \rangle$ where

$$\varepsilon < \gamma; \quad A = B \gg (\gamma \to \varepsilon) \gg (\varepsilon \to \bot)$$

Definition 6.37. *Proper $\langle A, L \rangle$-inflexible data* is a pair $\langle D, t \rangle$ where D is proper A-inflexible path data and $L = f_{\delta,\varepsilon}\ t$. *Base $\langle A, L \rangle$-inflexible data* is a pair $\langle D, a \rangle$ where D is base A-inflexible path data and $L = f_{\bot,\varepsilon}\ a$.

Lemma 6.38. A litter can be A-inflexible in precisely one way; that is, the types of proper and base $\langle A, L\rangle$-inflexible data are jointly a subsingleton.

Proof. Use lemmas 4.14 and 4.15. □

6.2 Approximations

6.2.1 Near-litter approximations

Definition 6.39. A *near-litter approximation* is a pair $\varphi = \langle \varphi^A, \varphi^L\rangle$ where φ^A is a local permutation of atoms and φ^L is a local permutation of litters, such that for each litter L, the set $\mathcal{A}_L \cap \operatorname{dom} \varphi^A$ is small.

As with near-litter permutations, we often suppress the superscripts, and allow this type to act on atoms and litters, even those not in the domain of the local permutations in question. Near-litter approximations have inverses $\varphi^{-1} = \langle (\varphi^A)^{-1}, (\varphi^A)^{-1}\rangle$.

Lemma 6.40. Let N be a near-litter. Then $N \cap \operatorname{dom} \varphi^A$ is small.

Proof. It suffices to show that the following set is small.

$$(\mathcal{A}_{N^\circ} \cap \operatorname{dom} \varphi^A) \mathbin{\triangle} ((\mathcal{A}_{N^\circ} \mathbin{\triangle} N) \cap \operatorname{dom} \varphi^A)$$

The left and right components are both small, giving the result by lemma 2.9(v). □

Definition 6.41. Let L be a litter and φ be a near-litter approximation. Define the *φ-largest sublitter* of L to be $L \setminus \operatorname{dom} \varphi^A$; it is a sublitter by the definition of a near-litter approximation. This is the largest sublitter of L disjoint from the domain of φ^A.

Lemma 6.42. Let a be an atom and φ be a near-litter approximation. Then a is in the φ-largest sublitter of a° if and only if $a \notin \operatorname{dom} \varphi$.

Proof. Almost by definition, using lemma 2.17(ii). □

Definition 6.43. An atom a is an *exception* to a near-litter permutation π if

$$(\pi\ a)^\circ \neq \pi\ a^\circ \text{ or } (\pi^{-1}\ a)^\circ \neq \pi^{-1}\ a^\circ$$

Definition 6.44. A near-litter approximation φ *approximates* a near-litter permutation π if for all $a \in \operatorname{dom} \varphi^A$ and $L \in \operatorname{dom} \varphi^L$, we have

$$\pi\ a = \varphi\ a; \quad \pi\ L = \varphi\ L$$

We say that φ *exactly approximates* π if in addition every exception to π lies in $\operatorname{dom} \varphi^A$.

Definition 6.45. Let A be a β-extended type index. We say that a near-litter approximation φ is *A-free* if every litter in $\operatorname{dom} \varphi^L$ is A-flexible.

6.2.2 Structural approximations

Definition 6.46. Let β be a type index. A *β-structural approximation* is a β-tree of near-litter approximations.

Definition 6.47. Let φ be a β-structural approximation, and π be a β-structural permutation. We say that φ *(exactly) approximates* π if for each β-extended index A, $\varphi\ A$ (exactly) approximates $\pi\ A$. We say φ is *free* if for each β-extended index A, $\varphi\ A$ is A-free.

Lemma 6.48. Let φ be a β-structural approximation, π be a β-structural permutation, and $A : \beta \rightsquigarrow \gamma$. Suppose φ (exactly) approximates π. Then φ_A (exactly) approximates π_A.

Proof. Almost by definition. □