Con(NF)

Con(NF) project contributors

March 21, 2023

Chapter 1

Phase 0: Preliminaries

In this section, we give the background constructions for the model — as much as can be given before we enter into the main recursion.

1.1 Cardinal parameters

Definition 1.1. *Throughout, we fix three parameters:*

- λ *may be any limit ordinal;*
- κ *may be any regular cardinal* $> \lambda$*;*
- μ *may be any strong limit cardinal* $> \kappa$ *of cofinality* $\geq \kappa$*.*

[Note: The assumptions on κ *and* μ *are essential for the construction of the model to work. The assumptions on* λ *are not needed for this model — everything probably works for an arbitrary well-founded strict partial order* λ*, and gives a model of* TTT_λ*. The assumption that* λ *is a limit ordinal is required afterwards, to convert the model of* TTT_λ *into a model of* TST + *typical ambiguity.]*

Definition 1.2. *We refer to sets of size smaller than* κ *as* small *[and all other sets as large].*

Definition 1.3. *We define a* type index *as an element of* $\lambda^\perp := \lambda \cup \{-1\}$*. We define a* proper type index *as an element of* λ*.*

In many ways, -1 *should be viewed as a type index on the same footing as other elements of* λ*; but it frequently needs treating as a special case.*

[In the formalisation, "-1*" is referred to as* $\perp$*. Should we consider following that in the text too?]*

Definition 1.4. *By a* path*, we mean a path in the directed graph (aka quiver)* $(\lambda^\perp, <)$*; that is, a nonempty finite strictly increasing sequence of type indices* $\alpha_0 < \cdots < \alpha_k$*. A path is* proper *if its source (least element) is proper.*

We will write paths as $A : \beta \dashrightarrow \alpha$*. We will write path concatenation and extension as* $\gamma; A$*,* $A; B$*, and similar.*

These may be viewed as the arrows of the free category on $(\lambda^\perp, <)$*. We mostly avoid using this categorical packaging, to keep things elementary and avoid unnecessary dependencies; but we mention it occasionally in the blueprint, as* $\mathrm{Path}(\lambda^\perp)$*.*

[There may be some terminological inconsistency: In places, paths are called extended type indices*, and are considered as sets rather than sequences; also, they are sometimes taken to be decreasing instead of increasing.]*

1.2 Type −1: atoms, litters, and local cardinals

We will define a function taking type indices α to sets τ_α (type α).

Definition 1.5. *Define $\mathcal{L}$ as $(\lambda \cup \{-1\}) \times \lambda \times \mu$ and τ_{-1} as $\mathcal{L} \times \kappa$.*

We refer to elements of τ_{-1} as atoms*.*

By construction, atoms come partitioned into $\mathcal{L}$-many litters *of size κ,*

$$L_{\alpha,\beta,\xi} := \{((\alpha,\beta,\xi),x) \mid x \in \kappa\}.$$

The litters, in turn, come partitioned into λ^2-many classes of size μ:

$$X_{\alpha,\beta} := \{(\alpha,\beta,\xi) \mid \xi \in \mu\}.$$

We generally identify litters $L_{\alpha,\beta,\xi}$ with their indices $(\alpha,\beta,\xi) \in \mathcal{L}$; for instance a quantification over "all litters" is formally a quantification over $\mathcal{L}$.

The "atoms" of τ_{-1} are not quite atoms in a traditional sense — they do not appear as elements of the eventual structure — but they fulfil the same role of getting the set hierarchy off to a sufficiently non-trivial start.

Definition 1.6. *A* near-litter *is a subset N of τ_{-1} with small symmetric difference from some litter. This litter is necessarily unique, and we denote it by N°. We write $\mathcal{NL}$ for the set of all near-litters.*

Definition 1.7. *For $i \in \mathcal{L}$, the i-th* local cardinal *$[L_i]$ is $\{N \subseteq \tau_{-1} : |N \triangle L_i| < \kappa\}$.*

For any near-litter N (including litters), we write $[N]$ for the unique local cardinal that contains N. We introduce the notation $[N]^\circ$ for the litter belonging to the local cardinal of N, with the tacit need to prove that there is only one.

[Note: almost all roles played by local cardinals in the original text should be fulfilled in the formalisation by litter-indices $i \in \mathcal{L}$.]

Lemma 1.8. *Note that $\mathcal{L}$ corresponds to the equivalence classes of an equivalence relation $\sim$ on near-litters, where $X \sim Y$ iff the symmetric difference $X \triangle Y$ is small.*

Proof. This is left as an easy exercise for the reader. □

Lemma 1.9. *Because the cofinality of μ is $\geq \kappa$, there are μ near-litters.*

Proof. First, there are at least μ near-litters. Second, the equivalence induced by the symmetric difference with the i-th litter sends near-litters to sets of size strictly less than κ, hence of size strictly less than the cofinality of μ. But there are at most (exactly, in fact) μ such sets, because by definition of cofinality they are all bounded in μ and $\sum_{\alpha<\mu} 2^\alpha = \mu$ because μ is a strong limit cardinal. □

One might be concerned with the fact that if μ has cofinality κ, it might have more than μ subsets of size κ: but it still has only μ subsets of size $< \kappa$, and that is what matters for counting the near-litters: a near-litter is determined as the symmetric difference of a litter (μ of these) and a small subset (cardinality $< \kappa$) of τ_{-1} (which is of size μ) and there are only μ small subsets of τ_{-1}. If the cofinality of μ were less than κ, the cardinal arithmetic pathology mentioned as of concern could come into play.

1.3 Preliminaries: pretangles, structural permutations

Before embarking on the large recursive construction of the eventual model, we set up some structures in advance which we will make use of during the main recursion.

We first define a big structure, which can be viewed as a model of TTT without extensionality, whose elements we call *pretangles*. In the main recursion, we will carve out our actual model of TTT as a substructure of *tangles* within the pretangles.

Definition 1.10. *Define the sets* $\mathrm{Pretangle}_\alpha$ *of* α-pretangles *inductively.* $\mathrm{Pretangle}_{-1}$ *is defined as* τ_{-1}*. For* α *a proper type index,* $\mathrm{Pretangle}_\alpha$ *is defined as* $\prod_{\beta<\alpha} \mathcal{P}(\mathrm{Pretangle}_\beta)$*.*

We also set up a sequence of groups that act on pretangles; subgroups of these will later act on the tangles.

Definition 1.11. *A* structural (-1)-permutation *[aka a near-litter-perm; we should fix on one name or the other] is a permutation* π *of* τ_{-1}*, along with a permutation* $\bar{\pi}$ *of litters, such that for each litter* L*,* $\pi^{\diamond}(L)$ *is near to* $\bar{\pi}(L)$ *(in the sense that they have small symmetric difference). Denote the group of these by* Str_{-1}*.*

[In fact $\bar{\pi}$ *is uniquely determined by* π *— we could define this just as a subgroup of* $\mathrm{Sym}(\tau_{-1})$*. But it's convenient in formalisation to have the* $\bar{\pi}$ *component explicit.]*

Str_{-1} *has obvious actions on atoms and near-litters.*

Definition 1.12. *For a* proper *type index* α*, the structural* α*-permutations are the group* $\mathrm{Str}_\alpha = \prod_{\beta<\alpha} \mathrm{Str}_\beta$*.*

Definition 1.13. *Any path* $A : \beta \dashrightarrow \alpha$ *gives a group homomorphism* $(-)_A : \mathrm{Str}_\alpha \to \mathrm{Str}_\beta$*.*

Moreover, this is functorial: they make Str *into a functor* $\mathrm{Path}(\lambda^\perp)^{\mathrm{op}} \to \mathrm{Gp}$*.*

[TODO: this should also depend on "paths", but they aren't yet explicitly in the blueprint.]

Definition 1.14. *An* allowable (-1)-permutation *is just a structural* (-1)*-permutation. We denote the group of these by* $\mathrm{All}_{-1} = \mathrm{Str}_{-1}$*. (Later we will define* Str_β *as a proper subgroup of* All_β*, for higher* β*.)*

Definition 1.15. *Let* α *be a proper type index. An* α-support-condition *(or just alpha-*condition*) is a pair* (x, A)*, where* A *is a path from* -1 *to* α *(aka an extended type index) and* x *is either an atom or a near-litter.*

Type-theoretically, $\mathrm{Cond}_\alpha := (\tau_{-1} + \mathcal{NL}) \times \mathrm{Path}(-1, \alpha)$*.*

[It would suffice to use just litters here instead of near-litters; they give an equivalent notion of support, as shown later ("replacing near-litters with litters"). That makes a few things simpler; but using near-litters lets us talk about the action, Def. 1.16, which seems algebraically cleaner.]

Definition 1.16. *Structural permutations act on support conditions.*

Definition 1.17. *Let* α *be a proper type index. An* α*-support is a small set of* α*-support-conditions. (A little more pedantically, one could call these something like "potential small supports": they become actual supports once they support some element.)*

[NOTES for formalisation: Well-orderings are assumed here in some version of Randall's note, but they almost certainly aren't wanted in the basic definition of supports, as used in phase 1. Strong supports, as used in phase 2, are probably better viewed as a separate notion: a more elaborate data structure that can be used to "present" a support in a particularly good way.]

Definition 1.18. *Suppose $\varphi : H \to \mathrm{Str}_\alpha$ is any group homomorphism, and τ is a set equipped with an H-action.*

Given $x \in \tau$ and S any set of α-support conditions, say S supports *x if every $\pi \in H$ that fixes every element of S also fixes x.*

A support for *x is a support (i.e. a* small *set of conditions) that supports x. [Should we try to always say "small support" to avoid ambiguity?]*

We say x is symmetric *if it has some small support.*

Lemma 1.19. *Because the cofinality of μ is $\geq \kappa$, there are μ potential supports, and therefore at most μ supports for each $x \in \tau$.*

Proof. □

Chapter 2

Phase 1: The model construction

In this chapter, we give almost the entire construction of the structure which will (eventually) be a model of tangled type theory.

This should be read as an attempt at a recursion over the levels $\alpha \in \lambda$ of the structure. We start by describing what data should exist at each level of the structure; then this section consists mostly of showing that given this data at each level $\beta < \alpha$, we can construct *almost* all this data at level α.

Unfortunately, one piece of the data is harder to propagate — essentially just the fact that the set of tangles at level α is of size $< \mu$. Proving this is not only lengthy, but also requires a more complex recursive assumption: not just data at each earlier level, but also information about how they interact between different levels. This difference in the recursive assumptions is the motivation for the phase separation.

2.1 Inductive assumptions

For now, we group the data assumed at each level, according to which constructions it will be needed for.

Definition 2.1. *Fix a proper type index α. Then* tangle data at α *consists of:*

1. *a group* All *(whose elements we call* allowable permutations*), with a map $\varphi : \mathrm{All} \to \mathrm{Str}_\alpha$;*
2. *a set τ (whose elements we call* tangles*), equipped with an* All*-action;*
3. *a map j (*typed near-litters*) from near-litters to τ, equivariant with respect to the* All*-action on near-litters induced by $\mathrm{All} \to \mathrm{Str}_\alpha \to \mathrm{Str}_{-1}$;*
4. *a map k (*typed singletons*) from τ_{-1} to τ, again* All*-equivariant w.r.t. the induced action on τ_{-1};*
5. *for each $x \in \tau$, a small set S_x of α-support-conditions that supports x under the* All*-action (a* designated support *for x);*
6. *an injection $\iota : \tau \to \mu$, satisfying the following conditions:*

(a) *each typed litter $j(L)$ precedes the typed singletons of all its elements $a \in L$ — explicitly, $\iota(j(L)) < \iota(k(a))$;*

(b) *each typed near-litter $k(N)$ which is not a litter comes later than its (typed) litter $j(N^\circ)$, and after (the typed singletons of) all elements of $N \triangle N^\circ$;*

(c) *for each x in τ_α that is not a typed litter or singleton, x comes later than all of its designated support – explicitly, for each (a, A) or (N, A) in S_x, we must have $\iota_\alpha(j(N)), \iota_\alpha(k(a)) < \iota_\alpha(x)$.*

[Note: to see these conditions are not unreasonable, note that each $x \in \tau$ has $< \mu$ many things that it must come after, and that the chains of these constraints are of depth at most 4: litters < atoms < other near-litters < everything else. This is said more carefully in Lemma 3.40 below.]

7. *[Optionally: An injection $\tau \to \text{Pretangle}_\alpha$, equivariant with respect to the induced* All*-action. This may or may not be needed, depending on how the phase 2 data is organised.]*

By core tangle data*, we will mean just* All*, φ, τ, and the* All*-action on τ as above. By* incomplete tangle data*, we will mean all the above data except for the position functions ι and their properties.*

When we assume this data at multiple levels, we will refer to the components τ_β, All_β*, and so on, and call their elements α-tangles, etc.*

There is standard core tangle data at level -1, and also the position function $\iota_{-1} : \tau_{-1} \to \mu$. We will often refer to this uniformly with assumed tangle data at other levels: for instance, if we assume tangle data at all *proper* levels $\beta < \alpha$, we may then refer to τ_β for all levels $\beta < \alpha$: for β proper, this means the tangles of the assumed tangle data, while for $\beta = -1$ it means the fixed set of atoms τ_{-1}.

2.2 Phase 1a: Set codes and alternative extensions

In most of the following few constructions, we will fix a proper type index α and assume we have tangle data given for all proper $\beta < \alpha$. (In fact some results — in particular, the f-maps, A-maps and their properties — only need to refer to the data at certain earlier levels.)

Definition 2.2. *An α-code is a triple (α, γ, G) where $\gamma < \alpha$ and $G \subseteq \tau_\gamma$. [The original set-theoretic implementation includes the α, to keep codes at different levels disjoint. In the type-theoretic implementation, the first component α is unnecessary.]*

Definition 2.3. *We define, for all $\beta, \gamma < \alpha$, with γ proper, a map $f_{\beta,\gamma}$ from τ_β to $X_{\beta,\gamma} \subseteq$* Litter*, as follows.*

For $x \in \tau_\beta$, $f_{\beta,\gamma}(x)$ is the litter N° of the minimal near-litter N (under the ordering induced by $\iota_\beta \circ j_\beta : \mathcal{NL} \to \tau_\beta \to \mu$) such that:

- *$N^\circ \in X_{(\beta,\gamma)}$, i.e. $N^\circ = L_{(\beta,\gamma,i)}$, for some i;*
- *for each nearby near-litter $M \sim N$, $\iota_\gamma(j_\gamma(M)) > \iota_\beta(x)$;*
- *$[N] \neq f_{\beta,\gamma}(y)$, for each $y <_\beta x$.*

[This can be decomposed slightly at both ends. Firstly, since we know that $f_{\beta,\gamma}(x) = (\beta, \gamma, \chi)$ for some χ, we could take the output just to be the component $\chi \in \mu$, and turn it into a litter later. Secondly, $f_{\beta_\gamma}(x)$ depends on x essentially just via its position $\iota_\beta(x) \in \mu$, so we could start by defining a function $g_{\beta,\gamma} : \mu \to \text{Litter}$, and then take $f_{\beta,\gamma}$ as the composite $g_{\beta,\gamma} \circ \iota_\beta$. I don't think these will really make a difference either way, though.]

[Notice that this only depends on a small fragment of the tangle data at β and γ. In particular, it doesn't depend on α nor on the tangle data at other levels. In the formalisation, it would be good to avoid it depending on more than it needs.]

Lemma 2.4. *The maps $f_{\beta,\gamma}$ satisfy:*

1. *each $f_{\beta,\gamma}$ is injective;*
2. *their images are disjoint, for all different pairs (β, γ);*
3. *each $f_{\beta,\gamma}$ is "position-raising": $\iota_\gamma(j_\gamma(N)) > \iota_\beta(x)$, for any near-litter N near to the litter $L_{(\beta,\gamma,f_{\beta,\gamma}(x))}$.*

Proof. □

Definition 2.5. *Let γ be a type index (not necessarily proper) below α.*

For any code (α, γ, G), with δ a proper type index below α and distinct from γ, we define $A_\delta(\alpha, \gamma, G)$ as

$$(\alpha, \delta, \{j_\delta(N) \mid N \in \mathcal{NL},\, N \sim f_{\gamma,\delta}(g),\, g \in G\})$$

if the maps $f_{\gamma,\delta}$ are considered as valued in litters, or

$$(\alpha, \delta, \{j_\delta(N) \mid N \in f_{\gamma,\delta}(g),\, g \in G\})$$

if they are taken as valued directly in local cardinals.

We say that a code c leads to *a code d if d is the image of c under some A-map. We denote this $c \rightsquigarrow d$.*

(Here $j_\delta(N)$ is the "typed near-litters" map for level δ, assumed in the tangle data.)

[For the formalisation, each individual A_δ is probably best represented as a function $\mathcal{P}(\tau_\gamma) \to \mathcal{P}(\tau_\delta)$, rather than on "codes of the form (α, γ, G)…". Also it could — and perhaps should, for flexibility later — have parameters just "γ, δ, and their tangle data", rather than "α, tangle data everywhere below α, and $\gamma, \delta < \alpha$".]

Lemma 2.6. *1. Each A_δ is injective on nonempty codes.*

2. The ranges of A_δ are disjoint for different δ.

NB: The disjointness of ranges depends on excluding the empty set — either from the domains of the A_δ from the start, or else in the statement of this lemma.

Proof.

Each $f_{\gamma,\delta}$ is injective, and $f_{\gamma,\delta}$ have disjoint images for different γ.

The ranges of $f_{\gamma,\delta}$ are disjoint for different δ. □

Lemma 2.7. *The relation $c \rightsquigarrow d$ on codes ("c leads to d under some A-map") is well-founded.*

Proof. Consider the map m from codes to μ sending (α, γ, X) to $\min \iota_\gamma^{\Subset} X$, where $\iota_\gamma : \tau_\gamma \to \mu$ is the position function assumed in the tangle data.

The definition of the functions $A_{\delta,\gamma}$ and the fact that $\iota f_{\gamma,\delta}(x) > \iota x$ ensure that $j(\alpha, \gamma, X) < j(A_\delta(\alpha, \gamma, X))$, for all suitable γ, δ, X.

In other words, if $c \rightsquigarrow c'$, then $m(c) < m(c')$ in μ. It follows that $\rightsquigarrow$ is well-founded. □

Definition 2.8. *A code is* even *if it only leads to odd codes. A code is* odd *if it leads to some even code.*

Note: This exactly says that even codes are the losing positions of the game whose states are codes and possible moves are taking the preimage under some A-map.

Lemma 2.9. *1. All codes are even or odd.*

2. An odd nonempty code only leads to even codes.

Proof. 1. $\rightsquigarrow$ is well-founded.

2. An odd nonempty code leads to an even code and the A-maps are injective on nonempty codes and have disjoint ranges.

□

Definition 2.10. *We define an equivalence relation $\equiv_\alpha$ on α-codes inductively:*

1. *For every code c, $c \equiv_\alpha c$.*
2. *If (α, γ, G) is even and $\gamma \neq \delta$, then $(\alpha, \gamma, G) \equiv_\alpha A_\delta(\alpha, \gamma, G)$ and $A_\delta(\alpha, \gamma, G) \equiv_\alpha (\alpha, \gamma, G)$.*
3. *If (α, γ, G) is even and $\gamma \neq \delta, \varepsilon$, then $A_\delta(\alpha, \gamma, G) \equiv_\alpha A_\varepsilon(\alpha, \gamma, G)$.*

This is reflexive and symmetric by definition, and transitive by using the properties of code parity.

Lemma 2.11. *Under $\equiv_\alpha$, each code is equivalent to*

1. *exactly one even code*
2. *exactly one code of extension γ for all $\gamma \neq -1$, $\gamma < \alpha$*
3. *at most one code of extension -1*

Proof. □

Definition 2.12. *An* α-semi-tangle *is an element x of $\prod_{\beta<\alpha} \mathcal{P}(\tau_\beta) \times (\alpha + \mathcal{P}(\tau_{-1}))$, whose components we denote x_β, such that:*

- *if x_{-1} is some $\beta < \alpha$, then (α, β, x_β) is a representative code, and for each other γ, $A_{\beta,\gamma}(\alpha, \beta, x_\beta) = (\alpha, \gamma, x_\gamma)$;*
- *if x_{-1} is a set of atoms, then $(\alpha, -1, x_{-1})$ is a representative code, and for each other γ, $A_{-1,\gamma}(\alpha, -1, x_{-1}) = (\alpha, \gamma, x_\gamma)$;*

[Note that these are intermediate between "pretangles" and "tangles". Also note that — as with pretangles — we probably don't need the "preferred extension" component.]

[Actually it's probably simpler to go back to representing these as representative codes for now, and just define here the components x_β, for later embedding them into pretangles. It's only in phase 2 that the embedding into pretangles will really become helpful!]

Definition 2.13. *Membership relations of α-semi-tangles, for the intended model of tangled type theory, can now be defined as follows: for each proper type index $\beta < \alpha$, and $x \in \tau_\beta$, and each α-semi-tangle $y \in \tau_\alpha$, say $x \in_{TTT} y$ just if $x \in y_\beta$.*

This would be enough to enforce extensionality, but something much more radical needs to be done to make all this work, as we are assuming the existence of the maps ι_β, which witness that all the types are of cardinality no greater than μ. There must therefore be a very strong restriction on which sets can appear as components of tangles.

This all cries out for a Theorem which should be here and which I left implicit in the original text.

Theorem 2.14. *For all proper type indices $\beta < \alpha$, β-tangles x, and α-semitangles y, the following (nearly-obviously equivalent) statements hold:*

- *if for all $z \in \tau_\beta$, $(z \in_{\mathrm{TTT}} x \leftrightarrow z \in_{\mathrm{TTT}} y)$, then $x = y$;*
- *if $x_\beta = y_\beta$, then $x = y$.*

Proof. □

2.3 Phase 1b: Actual tangles: the model definition

For the subsequent constructions of this section, we assume we are given tangle data for all $\beta < \alpha$.

Definition 2.15. *A* semi-allowable permutation *at level α is a family of allowable pemutations at all lower levels (including -1), $(\pi_\beta)_{\beta<\alpha} \in \prod_{\beta<\alpha} \mathrm{All}_\beta$.*

Definition 2.16. *Semi-allowable permutations act on α-codes.*

Definition 2.17. *An* allowable permutation *is a semi-allowable permutation which preserves $\equiv_\alpha$: for all α-codes X, Y, $X \equiv_\alpha Y \leftrightarrow \pi(X) \equiv_\alpha \pi(Y)$.*

Lemma 2.18. *A semi-allowable permutation π is allowable just if*

$$f_{\gamma,\delta}(\pi_\gamma(g)) = [(\pi_\delta)^{\wr\!\wr}_{-1} f_{\gamma,\delta}(g)^\circ].$$

holds for all $f_{\gamma,\delta}$ with $\gamma, \delta < \alpha$, and all $g \in \tau_\gamma$.

Proof. This is discussion supporting the preceding lemma. The proof is given more carefully in Randall's more recent version of the note.

The coherence condition can be unpacked.

$$(\beta, \gamma, \{g\}) \equiv_\beta (\beta, \delta, \{(\delta, -1, N) : N \in f_{\gamma,\delta}(g)\})$$

(where $\delta \neq \gamma$). Thus we expect

$$\pi(\beta, \gamma, \{g\}) \equiv_\beta \pi(\beta, \delta, \{(\delta, -1, N) : N \in f_{\gamma,\delta}(g)\})$$

that is,

$$(\beta, \gamma, \{\pi_\gamma(g)\}) \equiv_\beta (\beta, \delta, \{(\delta, -1, (\pi_\delta)^{\wr\!\wr}_{-1} N) : N \in f_{\gamma,\delta}(g)\})$$

so $f_{\gamma,\delta}(\pi_\gamma(g)) = [(\pi_\delta)^{\wr\!\wr}_{-1} L]$, where $f_{\gamma,\delta}(g) = [L]$.

Recalling the notations N° for the litter with small symmetric difference from the near-litter N, we can write this

$$f_{\gamma,\delta}(\pi_\gamma(g)) = [(\pi_\delta)^{\diagdown\!\diagdown}_{-1} f_{\gamma,\delta}(g)^\circ].$$

It is straightforward to show that this condition is equivalent to the coherence condition. Notice that π_γ imposes some restrictions on π_δ, but only on the way it acts on certain typed near-litters (and of course there are reciprocal relations between π_δ and π_γ). □

Lemma 2.19. *The action of allowable permutations on codes preserves parity.*

Proof. This should be straightforward. A target should be the equation $\pi(A_\gamma(X)) = A_\gamma(\pi(X))$ for any allowable permutation π and code X for which π_γ is defined. □

Definition 2.20. *Allowable permutations act on semi-tangles.*

Definition 2.21. *We take the set of α-tangles, τ_α, to be the set of* symmetric *α-semi-tangles under the action of allowable permutations; that is, tangles that are supported by some small α-support S,.*

Lemma 2.22. *If $X \subseteq \tau_\beta$, $Y \subseteq \tau_\gamma$, and $(\alpha, \beta, X) \equiv_\alpha (\alpha, \gamma, Y)$, then (under the action of allowable permutations on codes) a set S of conditions supports β if and only if it supports γ. In particular, (α, β, X) has some small support if and only if (α, γ, Y) does.*

Proof. Immediate from the definition of allowable permutations. □

We state a couple of easy lemmas which are things we assumed in the inductive data, which must be shown to carry forward to τ_α. There are more obligations of this sort which are harder to discharge, which will be provided in phase 2.

Lemma 2.23. *Any code $(\alpha, -1, N)$, where N is a near-litter, gives an element of τ_α.*

Proof. This code is obviously even [though this isn't really necessary, thanks to Lemma 2.22] and it is supported by a singleton condition, a suitable decorated version of N. □

Lemma 2.24. *For any symmetric $b \in \tau_\beta$, $(\alpha, \beta, \{b\}) \in \tau_\alpha$. In particular, this works for each atom $a \in \tau_{-1}$.*

Proof. Take a β-support for b, and extend all the paths in it at the top by the step $\beta < \alpha$. At level -1, any atom is clearly supported by its own singleton.

[This should be easy if we defined paths by induction from the top; if we have paths in reverse, it may need lemmas on how derivatives interact with extension at the top. Or they might follow from functoriality of derivatives?] □

Lemma 2.25. *It should also be evident that (α, β, B) will always be symmetric if $|B| < \kappa$ [take the union of the β-supports of elements of B and add α to all the second components of elements of this union]: all small subsets of a type are realized in each higher type.*

Lemma 2.26. *The action of α-allowable permutations on semi-tangles restricts to an action on tangles. Explicitly, if an α semi-tangle x is symmetric, then so is πx for any α-allowable permutation π.*

Proof. The algebra of group actions should show reasonably easily that if π is an α-allowable permutation and $X \in \tau_\alpha$ has α-support S, then $\pi(X)$ has α-support $\pi^{\diagdown\!\diagdown} S$. □

The obligation to prove that τ_α is of size μ remains outstanding. And of course we want to prove that the entire structure is a model of tangled type theory with τ_γ as type γ for each $\gamma < \lambda$ and $\in_{\text{TTT}}$ as its membership relation.

There is lots to be proven, but that is the entire description.

2.4 Phase 1c: the embedding into pretangles

[Note: this may not eventually be needed, depending on how the "coherence" in phase 2 is assumed.]

For the subsequent constructions of this section, we assume we are given tangle data for all $\beta < \alpha$.

Definition 2.27. *There is an evident injection $\tau_\alpha \to \mathrm{Pretangle}_\alpha$ (where τ_α is constructed from the given tangle data). Moreover, this commutes with the All_α-action.*

2.5 Review

Overall, we have now shown:

Definition 2.28. *For a proper type index $\alpha \in \lambda$, given tangle data at all levels $\beta < \alpha$, then the constructions above provide* most *components of tangle data at level α — everything except for the embedding $\iota_\alpha : \tau_\alpha \to \mu$ and its properties.*

[Note that to treat this as a definition*, in particular of the "designated supports" component, we are implicitly invoking the axiom of choice — a priori we only know that there* exists *some support for each new tangle.]*

Chapter 3

Phase 2: Constraining the number of tangles

In this phase, we give the three hard theorems about the constructions of phase 1: strengthening supports, freedom of action, and constraining the number of tangles.

These meet the major technical difficulty of the recursion: For each topic in phase 2, we need to assume something along the lines of tangle data at all earlier levels, and suitable connections between them". The tricky bit is structing the "…suitable connections…" cleanly.

The greediest approach, which can't fail to work, is to assume some form of "at every level, the tangle data is equal to what is constructed from the data at earlier levels, by the constructions of phase 1". This was used in previous versions, as "coherent tangle data". However, this is somewhat awkward to work with in formalisation. A more cleanly-structured version is given by "freedom of action contexts", Definition 3.1, which axiomatise just as much of the connection between earlier levels as the recursive construction needs to know.

3.1 Freedom of action context

Definition 3.1. *A* freedom of action context *at a proper type index α consists of:*

1. *for every β and path $A : \beta \to \alpha$, core tangle data All_A, τ_A etc at β;*
2. *for each such β, A, when A is a* non-trivial *path (equivalently, when $\beta < \alpha$),* full *tangle data extending the given core data;*
3. *for $A : \beta \to \alpha$ and $\gamma < \beta$, a* derivative *map $(-)_\gamma : \mathrm{All}_A \to \mathrm{All}_{\gamma;A}$, commuting with the maps $\mathrm{All}_\bullet \to \mathrm{Str}_\bullet$ and $(-)_\gamma : \mathrm{Str}_\beta \to \mathrm{Str}_\gamma$.*

Notes:

1. *The name "freedom of action context" is very much a placeholder; it would be good to find a better name.*
2. *This can be repackaged several equivalent ways. For instance, the quantification over paths could be re-grouped as "for each $\beta < \alpha$ and $A : \gamma \to \beta$, full tangle data at γ; and core tangle data at α". This would avoid needing to say "full tangle data extending the given core data", but would make access to the core data less uniform.*

3. *Viewed categorically, the assumed maps make* All *into a functor on the slice category* $(\mathrm{Path}(\lambda, <)/\alpha)^{\mathrm{op}} \to \mathrm{Gp}$*, making the maps* $\mathrm{All} \to \mathrm{Str}$ *a natural transformation.*

4. *The motivation for assuming just "core data" at* α *itself is that for the main constructions, we'll be working in the inductive step of the recursion where we're assuming the data is already available at lower levels, and by the constructions of phase 1, we can assume some but not all of it is available at the "current level".*

5. *The motivation for parametrising the lower data not just by levels but by paths is so that when we synthesise this data at* α *from similar data at each* $\beta < \alpha$*, no coherence between the given data is required.*

Definition 3.2. *Given a freedom of action context at* α *and a path* $A : \beta \to \alpha$*, we can restrict the context along* A *to get a freedom of action context at* β*. Moreover, this is functorial in* A*. Note, in the Lean implementation, instead of 'restricting along* A*', we parametrise many of our functions by a parameter* B*, which is some path from a type index to* α*.*

Definition 3.3. *Given* $\alpha \in \lambda$*, and a freedom of action context at each* $\beta < \alpha$*, together with extensions of their top-level core tangle data to full tangle data at each* $\beta < \alpha$*, there is a* synthesised *freedom of action context at* α*, with its top-level core tangle data given by the constructions of phase 1.*

3.2 Freedom of action [new version]

A key technical lemma, used both for counting tangles and for proving that they satisfy comprehension, is the proof that allowable permutations act freely, in a suitable sense.

Roughly, any consistent and locally small specification of values of derivatives of an allowable permutation at elements of type -1 can be realized.

In order to state the theorem precisely, we must first set up a framework for discussing partial specifications of structural/allowable permutations.

Definition 3.4. *Let* α *be a proper type index. Recall that an* α-condition *is a pair* (x, A)*, where* A *is a path from* -1 *to* α *(aka an extended type index) and* x *is either an atom or a near-litter; the set of these was denoted* $\mathrm{Cond}_\alpha := (\tau_{-1} + \mathcal{NL}) \times \mathrm{Path}(-1, \alpha)$*.*

A binary condition *similarly consists of a pair of atoms or of near-litters, together with a path from* -1 *to* α*:*

$$\mathrm{BiCond}_\alpha := (\tau_{-1}^2 + \mathcal{NL}^2) \times \mathrm{Path}(-1, \alpha).$$

Definition 3.5. *Paths act covariantly on binary conditions. Explicitly, a path* $A : \beta \to \alpha$ *induces a map* $(-)_A : \mathrm{BiCond}_\beta \to \mathrm{BiCond}_\alpha$*, defined by* $(x, B)_A := (x, B; A)$*; and this preserves identities and composition.*

Taking inverse images under these maps, paths act contravariantly on sets of binary conditions: a path $A : \beta \to \alpha$ *induces* $(-)_A : \mathcal{P}(\mathrm{BiCond}_\alpha) \to \mathcal{P}(\mathrm{BiCond}(\beta))$*, given by* $\sigma_A := \{(x, y, B) \mid (x, y, BA) \in \sigma\}$*.*

Definition 3.6. *A binary condition* (x, y, A) *is viewed as representing the condition "*$\pi_A(x) = y$*". Any* α*-structural permutation* π *thus induces a set of binary conditions, its* graph*,* $\{(x, y, A) \in \mathrm{BiCond}_\alpha \mid \pi_A(x) = y\}$*. This gives an evident embedding* $\mathrm{Str}_\alpha \to \mathcal{P}(\mathrm{BiCond}_\alpha)$*; we will notationally identify structural permutations with their graphs.*

Definition 3.7. *We say that a structural permutation* π extends *a set of binary conditions* σ *if* $\sigma \subseteq \pi$*, viewing* π *as its graph. More generally, given any group with a map* $\varphi : G \to \mathrm{Str}$ *(e.g. the allowable permutations of some tangle data), we say* $\pi \in G$ *extends a set of binary conditions* σ *if* $\sigma \subseteq \varphi(\pi)$*.*

Proposition 3.8. *This identification of permutations respects the actions of paths. That is, given a structural permutation* π*, the graph of the derivative* π_A *is precisely the restriction along* A *of the graph of* π *(as our notational identification suggests).*

Proof. □

Definition 3.9. *Previously, we viewed* α*-conditions* $(x, A) \in \mathrm{Cond}_\alpha$ *as representing the conditions* $\pi_A(x) = x$*, as used in supports (and under this view we could identify them with the binary conditions* (x, y, A)*). In this section, we will more often view them as elements of the domain or range of a set of binary conditions. Precisely, for a set* σ *of binary conditions, we take* $\mathrm{dom}\sigma := \{(x, A) \mid (x, y, A) \in \sigma\}$*, and similarly* $\mathrm{rge}\sigma := \{(y, A) \mid (x, y, a) \in \sigma\}$*.*

By a slight abuse of terminology, when working with sets of conditions, we will often say "litter" to mean "litter-with-path", i.e. "condition of the form (L, A)*", and similarly for near-litter" and "atom".*

Definition 3.10. *We extend standard properties of binary relations to sets of binary conditions:*

Say a set of binary conditions $\sigma \subseteq \mathrm{BiCond}_\alpha$ *is* total *if its domain is* Cond_α*; co-total if its range is; and one-to-one if whenever* (x, y, A) *and* (x', y', A') *are in* σ*,* $x = x'$ *if and only if* $y = y'$*.*

The dual σ^{-1} *of a set of binary conditions is* $\{(y, x, A) \mid (x, y, A) \in \sigma\}$*.*

Proposition 3.11. *A set of binary conditions is the graph of a structural permutation precisely if it is one-to-one and total.*

(NOTE: this may not be explicitly needed, but it is helpful to know, and should be a good warmup and definition-test ahead of the main freedom of action theorem, in particular Proposition 3.24.)

Proposition 3.12. *If a set of* α*-conditions* σ *is total (resp. co-total, 1-1) then so is its restriction* σ_A *for any path* $A : \beta \to \alpha$*.*

Definition 3.13. *Given a freedom of action context at a proper type index* α*, a* flexible litter *is a condition* (L, A)*, with* L *a litter, such that* L *is not in the image of any of the* f*-maps* $f^A_{\gamma,\delta}$ *(for any* $\gamma, \delta < \beta$ $\dashrightarrow^A \alpha$*).*

Definition 3.14. *Assume a freedom of action context at a proper type index* α*.*

An allowable partial permutation *is a set* σ *of binary* α*-conditions satisfying the following properties:*

1. *It is one-to-one.*
2. *(Flexible litters:) Either* $\mathrm{dom}\sigma$ *and* $\mathrm{rge}\sigma$ *both include all flexible litters, or else they both include* co-μ-many*, i.e. there are at least* μ*-many flexible litters not in* $\mathrm{dom}\sigma$*, and dually for* $\mathrm{rge}\sigma$*.*
3. *(Atoms:) For each litter* (L, A) *not in* $\mathrm{dom}\sigma$*, then* $\mathrm{dom}\sigma$ *contains* $< \kappa$*-many atoms* (x, A) *with* $x \in L$*; and dually for* $\mathrm{rge}(\sigma)$*. On the other hand, for each litter* $(L, A) \in \mathrm{dom}\sigma$ *(say* $\sigma_A(L)$ *is the unique near-litter such that* $(L, \sigma_A(L), A) \in \sigma$*), then either*

domσ *contains* $< \kappa$*-many atoms of* (L, A) *and* rgeσ *contains* $< \kappa$*-many atoms of* (M, A)*, or else* domσ *contains all atoms of* (L, A) *and their setwise image* $\sigma''_A(L)$ *is precisely* $\sigma_A(L)$*; and dually for each litter* $(L, A) \in$ rgeσ*.*

4. *(Near-litters.) For each near-litter* $(N, A) \in$ domσ*, its litter* (N°, A) *and all atoms* (x, A) *from the symmetric difference* $N \triangle N^\circ$ *are also in* A*, and* $\sigma_A(N) = \sigma_A(N^\circ) \triangle \sigma''_A(N \triangle N^\circ)$*; and dually for each near-litter in* rgeσ*.*

5. *(Non-flexible litters.) Whenever* σ *contains some triple* $(f^A_{\gamma,\delta}(x), N, (-1; \delta; A))$ *(where* $\gamma, \delta < \beta \dashrightarrow^A \alpha$*,* δ *is proper, and* $x \in \tau_{\gamma;A}$*), then* $(\text{dom}\sigma)_{\gamma;A}$ *supports* x*, and every allowable permutation* $\rho \in \text{All}_A$ *extending* σ_A *has* $N = f^A_{\gamma,\delta}(\rho_\gamma x)$*. Dually, whenever* σ *contains some* $(N, f^A_{\gamma,\delta}(x), (-1; \delta; A))$*, then* $(\text{rge}\sigma)_{\gamma;A}$ *supports* x*, and any allowable* ρ *extending* σ_A *has* $N = f^A_{\gamma,\delta}(\rho^{-1}_\gamma x)$ *(note the inverse in this case).*

We will often refer to these conditions individually, as "the allowability condition for atoms", etc.

The motivation for allowable permutations is precisely to state the freedom of action property:

Definition 3.15. *Given a freedom of action context at a proper type index* α*, say that* freedom of action holds *if for every allowable permutation* σ*, there is some allowable permutation* $\pi \in \text{All}_\alpha$ *extending* σ*.*

Theorem 3.16. *Suppose given, for each* $\beta < \alpha$*, a freedom of action context together with full tangle data at the top level; and such that for all their restrictions along any path* $A : \gamma \to \beta$ *(including identity paths), freedom of action holds.*

Then freedom of action holds in the synthesised freedom of action context at α *(i.e. with the tangles and allowable permutations at* α *defined by the constructions of phase 1).*

Proof. The proof is quite lengthy, so we break it down into various lemmas.

In outline: We first define an ordering on allowable partial permutations (Definition 3.17), show that it has unions of chains (Proposition 3.18), and so apply Zorn's lemma to conclude that it has a maximal element above any element. This is the easy part. The main work then consists of showing that any maximal partial permutation is total and co-total (Lemma 3.23), and hence is an allowable permutation (Proposition 3.24).

Most lemmas involved can (and should) be given in an arbitrary freedom-of-action context, not required to be the synthesised one. □

Definition 3.17. *Work in a freedom of action context at a proper type index* α*.*

Given allowable partial permutations σ, ρ*, say that* $\sigma \sqsubseteq \rho$ *("*σ carefully extends ρ*") if:*

1. $\sigma \subseteq \rho$*;*
2. *if* ρ *has any new flexible litter, then it has all of them (in both domain and range);*
3. *within each litter, if* domρ *has any new atom, then it must have all atoms in that litter (and hence must also have the litter); and dually.*

Proposition 3.18. $\sqsubseteq$ *has upper bounds for chains, given by taking unions. That is: given a set of allowable partial permutations that is a chain under* $\sqsubseteq$*, its union is again an allowable partial permutation, and carefully extends each element of the chain. (Indeed, this gives a supremum for the chain, and so shows that* $\sqsubseteq$ *is chain-complete; but "upper bounds" is all we need in order to apply Zorn's lemma.)*

To prove that maximal allowable partial permutations are total, we will make use of a well-ordering on (unary) conditions:

Definition 3.19. *Fix a freedom of action context at α. Then the relation $\prec$ on Cond_α, read as* "constrains"*, is defined by:*

- $(L, A) \prec (x, A)$*, whenever L is a litter and $x \in L$; (an atom is constrained by the litter it belongs to);*
- $(N^\circ, A) \prec (N, A)$ *when N is a near-litter not equal to its corresponding litter N°;*
- $(x, A) \prec (N, A)$ *for N as above and all $x \in N \,\Delta\, N^\circ$;*
- $(y, B : (\gamma < \beta) : A) \prec (L, A)$ *for all paths $A \colon \beta \to \alpha$, and $\gamma, \delta < \beta$, and $L = f^A_{\gamma,\delta}(x), x \in \tau_{\gamma:A}$, and $(y, B) \in S_x$, where $S_x \subseteq \mathrm{Cond}_\gamma$ is the designated support of x.*

Note: "x constrains y" is effectively defined separately in the three cases that y is an atom, a non-litter near-litter, or a non-flexible litter. Flexible litters are not constrained by anything.

Note: This relation is not transitive — that's OK, it's not intended to be!

Proposition 3.20. *The relation $\prec$ is well-founded.*

Proof. By the conditions on orderings assumed in full tangle data, and the properties of f-maps, whenever some constraint $(x, A) \prec (y, B)$ holds, we have $\iota_A(x) < \iota_B(y)$ in μ. □

In order to make use of the inductive assumption that freedom of action holds at earlier levels, we will need a few notes about consquences of freedom of action, and about how the definitions of this section interact with restriction of the context.

Proposition 3.21. *If σ is an allowable permutation in a freedom of action context at α, then for any $A : \beta \to \alpha$, the restriction σ_A is again allowable in the context restricted along A.*

Proof. Hopefully straightforward, but probably a bit lengthy, using Proposition 3.12, Proposition 3.8. □

Proposition 3.22. *Given a freedom of action context at a proper type index α, an allowable permuation σ, and an α-tangle x such that* domσ *supports x:*

1. *if $\pi, \pi' \in \mathrm{All}_\alpha$ are allowable permutations extending σ, then $\pi x = \pi' x$;*
2. *if freedom of action holds, there is a unique $y \in \tau_\alpha$ such that every allowable permutation π extending σ has $\pi x = y$; we denote this unique value by σx.*

Proof. For the first part: given such π, π', $(\pi^{-1})\pi$ acts trivially on domσ, and hence fixes x. The second part follows from the first together with the freedom of action property. □

Lemma 3.23. *Any allowable partial permutation that is maximal under $\sqsubseteq$ is total.*

Proof. This is the main hard work of the proof of Theorem 3.16, and should probably be broken down into separate lemmas.

It suffices to show the claim: For any allowable partial σ, and any x, if everything that constrains x is in domσ, then there is some $\sigma' \sqsupseteq \sigma$ with $x \in$ domσ. (Given this claim and a maximal element σ, it follows by induction over "constrains" that domσ is all of Cond_α. Co-totality follows dually.)

Proof of claim: Treat the 4 cases for x separately.

1. A flexible litter (L, A). TO GIVE.
2. An atom (x, A). TO GIVE.
3. A non-litter near-litter (N, A). TO GIVE.
4. A non-flexible litter $(f^A_{\gamma,\delta}(x), -1; \delta; A)$. TO GIVE.

Note: The "straightforward" parts of checking allowability are fairly repetitive between the different cases here. This is the sort of thing that — if just tackled greedily — could take a lot of time and create a lot of code duplication. Probably good to write a few of these cases directly, but then see if they can be abstracted out as lemmas that can be re-used across cases. For the near-litter and non-flexible litter conditions, the point is that these are of the form "if [thing x] is present, then [condition P holds]", where condition P is something which is clearly preserved by subset-expansion (i.e. if $\sigma \subseteq \rho$ and P holds for σ, then it holds for ρ). So when we expand σ to ρ, these allowability conditions will still hold for all things that were already in σ — we only need to check for any new things that were added in ρ.

We should also avoid code-duplication of checking dual conditions separately, probably by lemmas explicitly connecting each condition to its dual, or by explicitly making use of duals in the definition of the allowability conditions. □

Proposition 3.24. *In the synthesised freedom of action context at α, any partial allowable permutation (at α) that is total and co-total extends to an actual allowable permutation at α.*

Proof. Take σ a total and co-total allowable partial permutation. For each $\beta < \alpha$, σ_β is again allowable, so by the assumption of FoA at earlier levels, σ_β extends to an allowable permutation $\pi_\beta \in \mathrm{All}_\beta$. [If we assumed All $\to$ Str is injective, then π_β is uniquely determined, since σ_β is total and co-total.] Together these π_β comprise a semi-allowable permutation π extending σ; the allowability condition for non-flexible litters ensures that the unpacked coherence condition holds, and hence by Lemma 2.18, π is allowable. □

3.3 Strong supports defined

NOTE: much of this section has not been updated since early in the formalisation, so may not match the current implementation and abstractions very closely.

Throughout this section, fix some $\alpha \in \lambda + 1$, and assume a freedom of action context at α.

Treating supports as sets suffices for the model description, but we will need to analyze supports and orbits with more care, so it is better for purposes of the proof to equip supports with a well-orderings.

We may write $x \leq_S y$ for $(x, y) \in S$, and $x <_S y$ when we also want to indicate that x, y are distinct.

If π is an α-allowable permutation and S is an α-support, we define $\pi[S]$ as

$$\{((\pi_A(x), A, \gamma), (\pi_B(y), B, \delta)) : ((x, A, \gamma), (y, B, \delta)) \in S\}.$$

If S is an α-support, we define S^+ as $\{(x, A, \alpha) : (x, A, \alpha) \in S\}$.

We can then say that S is a support of X if X is an α-code, S is an α-support, and for any α-allowable permutation π, if $\pi[S^+] = S^+$ then $\pi(x) = x$. In some sense the items in the support with third components less than α are fluff, but they *are* important as we will see.

Remark on definitions of support and symmetry: It should be clear that the supports we have defined here do exactly the same work as the set supports in the model description (since the additional order structure and the third components of support domain elements actually do no work at this point).

This probably represents a chunk of formal verification work, as what is obvious to people is not always obvious to theorem provers.

Definition 3.25. *A* strong support *is a support S equipped with a well-ordering $\prec_S$, such that whenever $c \in S$ and d constrains c, then $d \in S$ and $d \prec_S c$.*

Lemma 3.26. *Any support is a subset of some strong support (and hence any tangle is supported by some strong support).*

Proof. Given a support S, take its down-closure of $\downarrow S$ under the "constrains" relation.

First we need to check this has size $< \kappa$. The "constrains" relation is well-founded, and each condition has $< \kappa$ (immediate) predecessors, so by induction, the down-closure of any singleton is of size $<$ *kappa*. Then $\downarrow S$ is the union of the down-closures of its singletons (down-closure under any relation preserves arbitrary unions); so it's a union of $< \kappa$-many sets of size $< \kappa$.

Now "constrains" on $\downarrow S$ is a well-founded relation; and any well-founded relation can be extended to a well-ordering. [This is a standard general theorem which is hopefully already in mathlib. If not, it can be proven probably most easily by Zorn's lemma.] □

3.4 Types are of size μ (so the construction actually succeeds)

[Note: this section is treated MUCH more clearly in Randall's more recent versions of the note — we should certainly pull in from that newer version before attempting to formalise the section.]

Now we argue that (given that everything worked out correctly already at lower types) each type α is of size μ, which ensures that the construction actually succeeds at every type.

Definition 3.27. *For any support S and tangle x, we can define a function $\chi_{x,S}$ which sends $T = \pi(S)$ to $\pi(x)$ for every T in the orbit of S under the action of allowable permutations. We call such functions* coding functions*. Note that if $\pi[S] = \pi'[S]$ then $(\pi^{-1} \circ \pi')[S] = S$, so $(\pi^{-1} \circ \pi')(x) = x$, so $\pi(x) = \pi'(x)$, ensuring that the map $\chi_{x,S}$ for which we gave an implicit definition is well defined.*

The strategy of our argument for the size of the types is to show that that there are $< \mu$ coding functions for each type whose domain includes a strong support, which implies that there are no more than μ (and so exactly μ) elements of each type, since every element of a type is obtainable by applying a coding function (of which there are $< \mu$) to a support (of which there are μ), and every element of a type has a strong support.

We describe all coding functions for type 0 (without concerning ourselves about whether supports are strong). The orbit of a 0-support in the allowable permutations is determined by the positions in the support order occupied by near-litters, and for each position in the support order occupied by a singleton, the position, if any, of the near-litter in the support order which includes it. There are no more than 2^κ ways to specify an orbit. Now for each such equivalence class, there is a natural partition of type -1 into near-litters, singletons,

and a large complement set. Notice that near-litters in the partition will be obtained by removing any singletons in the domain of the support which are included in them. The partition has $\nu < \kappa$ elements, and there will be $2^\nu \leq 2^\kappa$ coding functions for that orbit in the supports, determined by specifying for each compartment in the partition whether it is to be included or excluded from the set computed from a support in that orbit. So there are no more than $2^\kappa < \mu$ coding functions over type 0.

We specify an object X and a strong support S for X, and develop a recipe for the coding function $\chi_{X,S}$ which can be used to see that there are $< \mu$ coding functions.

$X = (\alpha, \beta, B)$, where B is a subset of τ_β. By inductive hypothesis, each element b of B can be expressed as $\chi_{b,T_b}(T_b)$, where T_b is a strong support for b end extending S_β (which is defined as the largest β-support U such that $U^{\{\alpha\}} \subseteq S$).

We claim that $\chi_{X,S}$ can be defined in terms of the orbit of S in the allowable permutations and the set of coding functions χ_{b,T_b}. There are $< \mu$ sets of type β coding functions by inductive hypothesis, and we will argue that there are $< \mu$ orbits in the α-strong supports under allowable permutations, so this will imply that there are $\leq \mu$ elements of type α (it is obvious that there are $\geq \mu$ elements of each type). Of course we get $\leq \mu$ codes for each $\beta < \alpha$, but we know that $\lambda < \kappa < \mu$.

The definition that we claim works is that $\chi_{X,S}(U) = (\alpha, \beta, B')$, where B' is the set of all $\chi_{b,T_b}(U')$ for $b \in B$ and U' end extending U_β. Clearly this definition depends only on the orbit of S and the set of coding functions derived from B.

The function we have defined is certainly a coding function, in the sense that $\chi_{X,S}(\pi(S)) = \pi(\chi_{X,S}(S))$. What requires work is to show that $\chi_{X,S}(S) = S$, from which it follows that it is in fact the intended function.

Clearly each $b \in B$ belongs to $\chi_{X,S}(S)$ as defined, because $b = \chi_{b,T_b}(T_b)$, and T_b end extends S_β.

An arbitrary $c \in \chi_{X,S}(S)$ is of the form $\chi_{b,T_b}(U)$, where U end extends S_β and of course must be in the orbit of T_b under allowable permutations.

Our strategy is to show that there is an allowable permutation π which fixes X (so that $\pi^{\diagdown\!\diagdown}_\beta B = B$) such that $\pi_\beta[T_b] = U$, so that $\pi_\beta(b) = c$, so $c \in B$, whence $\chi_{X,S}(S)$ as defined is equal to X as required.

We build a support $S + T_b^{\{\alpha\}}$ and a support $S + U^{\{\alpha\}}$ with parallel structure by appending T_b (respectively U) to S then removing all but the first occurrence of each repeated item. The parallelism of structure is enforced by the identity of items taken from $S \setminus S_\beta$ in both supports and the fact that U is the image of T_b under some allowable permutation.

We construct an α-allowable permutation whose action takes one of these supports to the other, which will complete the plan given above. For this we use the freedom of action theorem. We define a local bijection which sends (A, x) to y just in case a $(\beta, A, \{x\})$ in the first support corresponds to a $(\beta, A, \{y\})$ in the other, and further enforces agreement of derivatives of the permutation to be constructed with derivatives of the known permutation π' sending T_b to U at exceptions of derivatives of π' which lie in litters in T_b. This causes singleton items in the first support to be mapped to the corresponding singleton items in the other support. We have to argue that litters in the domain of $S + T_b^{\{\alpha\}}$ (which is a strong support) are mapped to the correct near-litters in the domain of $S + U^{\{\alpha\}}$. If there is a failure, there is a first one. The local cardinal of the first failure is treated correctly (because a support of it is treated correctly), so the failure must consist in the map constructed having an exception which is moved by the permutation into or out of the litter in question (if a litter L in T_b is mapped to a near-litter N in U, all elements of $N \triangle N^\circ$ are treated correctly because they are values at exceptions of the known permutation), so a failure implies an exception of the constructed permutation lying in L which is not an exception of the known

map and whose singleton is not an item in T_b, and there are no such exceptions.

The constructed map fixes X because of its identity action on S, and it sends b to c because its action sends T_b to U, which is what we claimed,

The final move is to argue that there are $< \mu$ orbits in the α-allowable permutations. The idea is that the orbit in which a permutation lies is completely determined by a certain amount of combinatorial information, similarly to what happened in type 0 but a bit more complex. The orbit is specified if we know the second and third components of each item, taken from λ items in each case, the first and second components of the first item, and whether the third component is a singleton or a near-litter. If this is a singleton, we want to know the position in the support of a near-litter containing it (which will be present). If this is a near-litter and its local cardinal is an image under an f map, we can extract from information about the preceding part of the support order a subsupport which is an index-raised version of a strong support for the near-litter and so for its inverse image under the f map: as part of our specification, we take the coding function which generates that inverse image.

We give exact details. If S is a strong support (or an image of a strong support under an allowable permutation), we define its specification S^* as a well-ordering of the same length in which an item $((\beta, -1, x), A, \gamma)$ will be replaced by an item $((\beta, -1, X), A, \gamma)$ in a way that we describe. If $x = \{y\}$, we replace x with $\{\delta\}$, where δ is the position of a typed near-litter containing y in the obvious sense. If x is a near litter which does not belong to any $f_{\gamma,\beta}(y)$ with $\gamma < \mathtt{min}(A)$, then $X = \emptyset$. If $x \in f_{\gamma,\beta}(y)$ with $\gamma < \mathtt{min}(A)$, then we extract the maximal strong support T of y such that $T^{\{\alpha\}} \subseteq S$, and set $X = \chi_{y,T}$, a coding function.

There is a straightforward argument by induction on the structure of strong supports that if we have two items with strong supports which have the same specification in the sense we have just described, there is an allowable permutation (by freedom of action) whose action sends the one support to the other, and so the one item to the other.

Suppose $S^* = T^*$: we discuss the construction of an allowable permutation π such that $\pi[S^*] = T^*$. It should not be a surprise that we construct the desired π as an extension (as in the freedom of action theorem) of a local bijection defined by consulting the parallel structures of S and T. If $((\beta, -1, \{x\}), A, \delta)$ and $((\beta, -1, \{y\}), A, \delta)$ appear at corresponding positions in S and T, we have π^0 send (A, x) to y. If $((\beta, -1, M), A, \delta)$ and $((\beta, -1, N), A, \delta)$ appear at corresponding positions in S and T and $((\beta, -1, \emptyset), A, \delta)$ appears at the corresponding position in $S^* = T^*$, we can provide that the map χ_A used in the freedom of action construction maps $[M]$ to $[N]$. If $((\beta, -1, M), A, \delta)$ and $((\beta, -1, N), A, \delta)$ appear at corresponding positions in S and T and $((\beta, -1, \emptyset), \chi_{y,T}, \delta)$ appears at the corresponding position in $S^* = T^*$, then we know by the inductive hypothesis that everything works before this item in the support that the action of the permutation π_A constructed so far will send $[M]$ to $[N]$. In both near-litter cases, we need to do a little more work to ensure that M is mapped exactly to N without exceptions. The idea is to extend π^0_A so as to map each element of M which is not in M° to something in $N^\circ \cap N$, and each element of M° which is not in M to something not in N, and do the analogous things for $(\pi^0_A)^{-1}$, and then fill in orbits, which only requires countably many atoms for each orbit [this is why we take κ to be uncountable], with the rule that images and preimages chosen in the filling out process are chosen so as not to create exceptions (their images and preimages will agree with expected actions on near-litters in the supports, which is really action on their local cardinals, because all elements of the symmetric differences of near-litters with their corresponding litters are treated individually). The extension of this local bijection will have exactly the desired effect.

There are clearly $< \mu$ specifications since these are small structures built with components

taken from sets of size $< \mu$. Notice the recursive dependency on the coding functions for items of lower types being taken from sets of size $< \mu$.

Pulling out the main items from the discussion above, for formalisation targets:

Definition 3.28. *Fix a freedom of action context for α. A* support-specification *is …[this is quite long to define; we should get the text from more recent versions of Randall's note]*

Lemma 3.29. *There are $< \mu$-many support specifications.*

Definition 3.30. *Any support specification can be realised (non-uniquely) to give a strong support.*

Lemma 3.31. *If freedom of action holds at α, then realisations of a support specification are unique modulo the group action. That is, for any two realisations S, S' of a support specification S, there is some allowable permutation π such that $\pi S = S'$.*

Proof. Hopefully quite direct, given heavy use of freedom of action. May need some lemmas about realisations. □

Lemma 3.32. *Every strong support can be obtained as the realisation of some specification*

Corollary 3.33. *There are $< \mu$-many orbits of strong supports, under the group action of allowable permutations.*

Proof. Immediate from Lemma 3.29 and Lemma 3.32. □

Now, everything that we did with supports, we repeat with coding functions.

Definition 3.34. *A* coding function specification *is …[again, see recent versions of Randall's note]*

Lemma 3.35. *There are $< \mu$-many coding function specifications.*

Definition 3.36. *Any coding function specification can be realised as a coding function.*

Lemma 3.37. *Every coding function can be obtained as the realisation of some specification*

Proof. See Randall's more recent versions of the note. □

Corollary 3.38. *There are $< \mu$-many coding functions.*

Proof. Immediate from Lemma 3.35 and Lemma 3.37. □

Corollary 3.39. *There are μ-many α-tangles.*

Proof. Every tangle can be obtained as a coding function (of which there are $< \mu$) applied to a strong support (of which there are μ). □

This completes the proof that each type is of size μ, which finally closes the loop on the recursion — or very nearly so: we still have to show that the orderings can be chosen to satisfy the extra technical conditions required.

Lemma 3.40. *Given a freedom of action context for α satisfying freedom of action at α, we can choose supports $S(x)$ for all α-tangles x and a position function $\iota_\alpha : \tau_\alpha \to \mu$ satisfying the conditions demanded in Definition 2.1:*

1. *for each litter L, the typed litter of L precedes the typed singletons of all its elements $a \in L$ — explicitly, $\iota(j(L)) < \iota(k(a))$;*

2. *for each near-litter N which is not a litter, $j(N)$ comes after its (typed) litter $j(N^\circ)$, and after (the typed singletons of) all elements of $N \triangle N^\circ$;*

3. *for each x in τ_α that is not a typed litter or singleton, x comes later than all of its designated support – explicitly, for each (a, A) or (N, A) in the in $S(x)$, we must have $\iota_\alpha(j(N)), \iota_\alpha(k(a)) < \iota_\alpha(x)$.*

Proof. The constraints can be seen as defining a relation $\prec$ on τ_α; we want to extend $\prec$ to a well-ordering of order-type $\leq \mu$, or equivalently, to give an injection $\iota : \tau_\alpha \to \mu$ sending $\prec$ to $<$. We know:

- $\prec$ has depth 4: it is of the form "litters $\prec$ atoms $\prec$ other near-litters $\prec$ everything else";
- each tangle has $< \kappa$-many predecessors under $\prec$;
- $|\tau_\mu| \leq \mu$.

These conditions should suffice to imply the existence of the desired ordering/functions. □

Note for the formal verification project: I think everything is here, but filling in details to the satisfaction of a theorem prover will be work.

3.5 Completing the recursion

We can now wrap up the recursion.

Definition 3.41. *For each $\alpha \in \lambda$, we have the following data:*

1. *a full freedom of action context C_α (whose components we write as All^α_A, τ^α_A, etc.),*
2. *such that each restruction $(C_\alpha)_A$ satisfies freedom of action,*
3. *and with $< \mu$-many coding functions at the top level.*

This data is defined by recursion on α. Given such data for each $\beta < \alpha$,

1. *the core freedom of action context at α is synthesised by Definition 3.3 (i.e. using the constructions of phase 1 to give the core tangle data at the top level, and assembling the context at proper paths from the given lower contexts);*
2. *freedom of action at the top level follows by Theorem 3.16;*
3. *the top-level coding functions are bounded by Corollary 3.38;*
4. *this allows us to complete the full tangle data at the top level, by Lemma 3.40.*

Definition 3.42. *The (actual, genuine, definitive!) tangles are the structure* Tangle *extracted from the freedom of action contexts defined in Definition 3.41 as follows:*

1. *τ_α is the top-level tangles of the freedom of action context at α;*

2. *"by construction" (i.e. by the defining clause of Definition 3.41),* τ_α *is constructed by Definition 2.21 from the tangles* τ_β *for* $\beta < \alpha$*; thus Definition 2.13 gives a membership relation* $\in_{\beta,\alpha}$ *from* τ_β *to* τ_α*, for each* $\beta < \alpha$*.*

[Note: The definition of $\in_{\beta,\alpha}$ *uses a type equality on* τ_α*, so may not be very nice to work with directly in subsequenet lemmas. However, it's hard to see how any approach to this inductively constructed structure could avoid such issues. When we get to this stage, we can think about how to deal with this issue.]*

Chapter 4

Tangles model tangled type theory

Now we can prove that the structure constructed in the previous chapters is a model of tangled type theory.

4.1 The structure is a model of predicative TTT

There is then a very direct proof that the structure presented is a model of predicative TTT (in which the definition of a set at a particular type may not mention any higher type). Use E for the membership relation of the structure. It should be evident that $xEy \leftrightarrow \pi_\beta(x)E\pi(y)$, where x is of type β, y is of type α, and π is an α-allowable permutation.

Suppose that we are considering the existence of $\{x : \phi^s\}$, where ϕ is a formula of the language of TST with $\in$ translated as E, and s is a strictly increasing sequence of types. The truth value of each subformula of ϕ will be preserved if we replace each x of type $s(i)$ with $\pi_{A_{s,i}}(x)$, where $x = s(j)$ and $A_{s,i}$ is the set of all s_k for $i \leq k \leq j+1$. The formula ϕ will contain various parameters a_i of types $s(n_i)$ and it is then evident that the set $\{x : \phi^s\}$ will be fixed by any $s(j+1)$-allowable permutation π such that π_{As,n_i} fixes a_i for each i. But this means that $(s(j+1), s(j), \{x : \phi^s\})$ is symmetric and belongs to type $s(j+1)$.

This procedure will certainly work if the set definition is predicative (all bound variables are of type no higher than that of x, parameters at the type of the set being defined are allowed).

There are easier proofs of the consistency of predicative tangled type theory; there is a reason of course that we have pursued this one. **Note for the formal verification project:** We note that in order to avoid metamathematics, we actually suggest proving finitely many instances of comprehension with typed parameters from which the full comprehension scheme can be deduced. That there are such finite schemes (mod the infinite sequence of types) is well-known.

4.2 Impredicativity: verifying the axiom of union

What remains to complete the proof is that typed versions of the axiom of set union hold. That this is sufficient is a fact about predicative type theory. If we have predicative comprehension and union, we note that for any formula ϕ, $\{\iota^k(x) : \phi(x)\}$ will be predicative if k is taken to be large enough, then application of union k times to this set will give $\{x : \phi(x)\}$. $\iota(x)$ here denotes $\{x\}$. It is evidently sufficient to prove that unions of sets of singletons exist.

So what we need to show is that if $(\alpha, \beta, \{(\beta, \gamma, \{g\}) : g \in G\})$ is symmetric, then (β, γ, G) is symmetric.

Suppose that $(\alpha, \beta, \{(\beta, \gamma, \{g\}) : g \in G\})$ is symmetric. It then has a strong support S. We claim that S_β (same notion defined above) is a β-support for (β, γ, G).

Suppose that $\pi[S_\beta] = S_\beta$.

Any $g \in G$ has a strong γ-support T which extends $(S_\beta)_\gamma$.

Construct using freedom of action technology a permutation π^* which acts as the identity on $S \setminus S_\beta$, such that π^*_β agrees with π on S_β [so in fact π^* will fix $(\alpha, \beta, \{(\beta, \gamma, \{g\}) : g \in G\})$] and $(\pi^*_\beta)_\gamma$ agrees with π_γ on T, both on the orbits under π of items in T and on the orbits under π of exceptions of π which are in litters in T. It will follow that π^* fixes $(\alpha, \beta, \{(\beta, \gamma, \{g\}) : g \in G\})$ and that $(\pi^*_\beta)_\gamma$ has the same value as π_γ at g, which means that $\pi_\gamma(g) \in G$ (and the same things follow for the inverse of π) which verifies that that S_β (same notion defined above) is a β-support for (β, γ, G), so the axiom of union holds in the interpreted TTT.

The application of the freedom of action theorem works because no movement of typed atoms of type γ stipulated by the behaviour of π_γ can force movement of elements of $S \setminus S_\beta$, because this would have to be mediated by the action of π on S_β, which fixes all elements of S_β. **Note for formal verification project:** This is a high level description which will probably acquire more detailed text if we get to it. The whole idea is here, I'm not saying there is a gap. But I have a strong suspicion that unwinding the details will induce more text.