Con(NF)

Con(NF) project contributors

June 24, 2022

Chapter 1

Phase 1: The model description

In this section, we give a complete description of what we claim is a model of tangled type theory. The construction may be supposed carried out in ZFC (or some weak subsystem thereof: we will see how much ZFC is needed).

1.1 Cardinal parameters

Definition 1.1. *Let λ be your favorite limit ordinal.*
Let $\kappa > \lambda$ be your favorite uncountable regular cardinal.
Let μ be your favorite strong limit cardinal $> \kappa$ of cofinality $\geq \kappa$.

Definition 1.2. *We define a* type index *as an element of $\lambda \cup \{-1\}$. We define a* proper type index *as an element of λ.*

I am not convinced that a separate concept of proper type index has merit: I think that -1 is a type index and sometimes is treated differently. But I provide it.

Definition 1.3. *We define an* extended type index *as a nonempty finite decreasing sequence of type indices. We define a* proper extended type index *as a nonempty finite decreasing sequence of proper type indices.*

Definition 1.4. *We refer to sets of size smaller than κ as small [and all other sets as large].*

I waffled about this: but cofinality exactly κ will work.

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 $\lambda^2 \times \mu$ and τ_{-1} as $\mathcal{L} \times \kappa$.*
Note that both sets have size μ.

Holmes comment: I didn't expect that λ^2 but I think I see what it is doing

We may refer to objects of type -1 as "atoms". They are not understood here to be atoms in a conventional sense, but the analogous objects in earlier versions of the construction were atoms and I have this mental habit.

Definition 1.6. *Define L_i for $i \in \mathcal{L}$ as $\{(i,k) : k \in \kappa\}$.*

We will refer to both elements of $\mathcal{L}$ and sets L_i as litters. *To be more precise, we will call L_i the i-*th litter.

Note that the litters make up a partition of type -1 into sets of size κ.

Definition 1.7. *A subset of τ_{-1} is a i-*near-litter *precisely if it has small symmetric difference from a litter.*

We introduce the notation N° for the litter with small symmetric difference from the near-litter N. With the tacit need to prove that there is only one.

Definition 1.8. *Define $X_{\beta,\gamma}$ as $(\beta,\gamma,i) : i \in \mu$. Note that $|X_{\beta,\gamma}| = \mu$ and that the $X_{\beta,\gamma}$ partition $\mathcal{L}$.*

Holmes comment: this is what I expected after my initial startlement at the definition of τ_{-1}.

Definition 1.9. *For $i \in \mathcal{L}$, the i-th* local cardinal *is $\{N \subseteq \tau_{-1} : |N \Delta L| < \kappa\}$.*

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 N, with the tacit need to prove that there is only one.

Lemma 1.10. *Note that $\mathcal{L}$ corresponds to the equivalence classes of an equivalence relation on near-litters, which obtains between two near-litters iff they have small symmetric difference.*

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

Lemma 1.11. *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 1.12. *Define the sets* Pretangle_α *of* α-pretangles *inductively.* Pretangle_{-1} *is defined as τ_{-1}. For α a proper type index,* Pretangle_α *is defined as $\prod_{\beta<\alpha} \mathcal{P}(\text{Pretangle}_\beta)$.*

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

Definition 1.13. *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, $\pi^{\wedge}(L)$ 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 1.14. *For a* proper *type index α, the structural α-permutations are the group* $\mathrm{Str}_\alpha = \prod_{\beta<\alpha} \mathrm{Str}_\beta$.

Definition 1.15. *Any path $A : \beta \to \alpha$ gives a group homomorphism $(-)_A : \mathrm{Str}_\alpha \to \mathrm{Str}_\beta$.*

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

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

Definition 1.16. *An* allowable (-1)-permutation *is just a structural (-1)-permutation. We denote the group of these by* $\mathrm{All}_{-1} = \mathrm{Str}_{-1}$. *(*$\mathrm{Str}_\beta$ *and* All_β *will diverge for higher β later…)*

[Actually this definition should probably go later, in phase 1c or even 2?]

Definition 1.17. *Let α be a proper type index. 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.*

Type-theoretically, $\mathrm{Cond} := (\tau_{-1} + \mathrm{NearLit}) \times \mathrm{Path}(-1, \alpha)$. *[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. 1.18, which seems algebraically cleaner.]*

Definition 1.18. *Structural permutations act on support conditions.*

[Hopefully, mathlib has general disjoint unions of group actions, which should supply this automatically.]

Definition 1.19. *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.]

Lemma 1.20. *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. □

Definition 1.21. *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.*

1.4 Phase 1a: Set codes and alternative extensions

Definition 1.22. *Fix a proper type index α.* Phase 1a data at α *consists of:*

- *a set τ (whose elements we call* tangles*);*
- *an injection $\iota : \tau \to \mu$ [alternatively, we could assume "an ordering $<$ on τ of order-type μ"; but we use the orderings mostly in terms of their position-functions, and never need the position-functions to be surjective]*
- *an injection j from the set of near-litters to τ. [These will later be concretely represented as the* typed near-litters *below.]*

In the following few constructions, we will fix a proper type index α and assume we have phase 1a data given for all proper $\beta < \alpha$

We will define τ_α, the implementation of type α of the model. This definition is recursive: when we are defining type α and associated concepts, we are supposing that related concepts have already been defined for all $\beta < \alpha$ [we will now introduce these explicitly as the "phase X data"], and this process will define all types in the structure.

Definition 1.23. *An α-code is a triple (α, γ, G) where $\gamma < \alpha$ and $G \subseteq \tau_\gamma$. [The set-theoretic interpretation includes the α, to keep codes at different levels disjoint. In the type-theoretic implementation, the first component α is almost certainly unnecessary]*

Definition 1.24. *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 $j_\beta \circ \iota_\beta : \text{NearLit} \to \tau_\beta \to \mu$) such that:

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

[This can be decompose 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 $i_\beta(x) \in \mu$, so we could start by defining a function $g_{\beta,\gamma} : \mu \to$ 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 the phase 1a data for α gives us the information already to define $f_{\beta,\gamma}$ for all $\beta, \gamma < \alpha$. This definition will in the end not actually depend on the value of α.

Lemma 1.25. *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 $L_{(\beta,\gamma,f_{\beta,\gamma}(x))}$.*

Definition 1.26. *Let γ be a type index less than α.*

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

$$(\alpha, \delta, \{j_\delta(N) : N \simeq f_{\gamma,\delta}(b),\, b \in B\})$$

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

$$(\alpha, \delta, \{j_\delta(N) : N \in f_{\gamma,\delta}(b),\, b \in B\})$$

if they are taken as valued directly in local cardinals.

(Here $j_\delta(N)$ is the "typed near-litters" map for level δ, assumed in the phase 1a data.)

[For the formalisation, each individual A_δ is probably most easily represented as a function on $\mathcal{P}_{\neq\varnothing}(\tau_(\gamma))$, 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 phase 1a data, rather than assume "α, phase 1a everywhere below α, and $\gamma, \delta < \alpha$".]

Note again that the computation of A_δ does not depend on α, apart from the fact that α must be larger than γ and δ.

Lemma 1.27. *1. Each A_δ is injective (since each $f_{\gamma,\delta}$ is injective, and $f_{\gamma,\delta}$ have disjoint images for different γ).*

2. The ranges of A_δ are disjoint for different δ (since the ranges of $f_{\gamma,\delta}$ are).

NB: injectivity here really depends on having excluded the empty set from the domains of the A_δ!

Definition 1.28. *By the previous lemma, the inverse relation of the union of the A_γ's is a partial function, which we call A^{-1}.*

It can be helpful to view codes as forming a forest *under the A-maps: each code $C = (\alpha, \beta, X)$ has at most one "parent" $A^{-1}(C)$, and many children $A_\gamma(C)$ for each $\gamma \neq \beta$.*

Lemma 1.29. *No code has infinitely many iterated images under A^{-1}.*

Equivalently (this may be easier for the formalisation), the relation $\rightsquigarrow$ on codes — defined by $(\alpha, \gamma, X) \rightsquigarrow A_\gamma, \delta(\alpha, \gamma, X)$ for all codes (α, γ, X) and all $\delta \neq \gamma$ — is well-founded.

Proof. Consider the map j 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 phase 1a data.

The definition of the functions $A_{\delta,\gamma}$ and the condition that $\iota f_{\gamma,\delta}()$ 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 $j(C) < j(C')$ in μ. It follows that $\rightsquigarrow$ is well-founded. □

Definition 1.30. *We define an equivalence relation $\equiv_\alpha$ on α-codes.*

We have $(\alpha, \gamma, \varnothing) \equiv_\alpha (\alpha, \delta, D)$ iff $D = \varnothing$.

Assume for the rest of the definition that (α, γ, G) is a code.

Define the height *of a code as its number of iterated images under A^{-1}. Formally this can be given by induction over the well-founded relation $\rightsquigarrow$ defined in Lemma 1.29: If C has no predecessors under $\rightsquigarrow$, then $h(C) = 0$; if C has a predecessor $C' \rightsquigarrow C$, this is unique by injectivity of A^{-1}, and we set $h(C) = h(C') + 1$.*

If (α, γ, G) has even height (including 0 as an important case) then $(\alpha, \delta, D) \equiv_\alpha (\alpha, \gamma, G)$ (for δ a type index distinct from γ) iff $(\alpha, \delta, D) = A_\delta(\alpha, \gamma, G)$.

If (α, γ, G) has odd height then $(\alpha, \delta, D) \equiv_\alpha (\alpha, \gamma, G)$ iff $(\alpha, \delta, D) \equiv_\alpha A^{-1}(\alpha, \gamma, G)$ (which reduces to the previous case).

Definition 1.31. *Note that each equivalence class under $\equiv_\alpha$ with nonempty third component contains exactly one element with even height. We call this as the* representative *code of its equivalence class, and we refer to such codes as representative codes generally, with the additional stipulation that $(\alpha, -1, \varnothing)$ is a representative code as well.*

Note also that any equivalence class contains at most one code of the form $(\alpha, -1, X)$, and contains exactly one code of the form (α, γ, G) for each $\gamma < \alpha$.

Definition 1.32. *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 1.33. *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 1.34. *For all proper type indices $\beta < \alpha$, β-tangles x, and α-semitangles y, the following (nearly-obviously equivalent) statements hold:*

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

1.5 Phase 1b: Actual tangles: the model definition

Definition 1.35. Phase 1b data *at a proper type index α consists of:*

- *phase 1a data at α;*
- *a group* All *(whose elements we call "allowable permutations"), with a map $\varphi : \text{All} \to \text{Str}_\alpha$;*
- *a group action of* All *on the tangles τ (that is, on the set τ assumed in the 1a-data).*

[Note: happily, "group actions" are already provided in mathlib.]

For the subsequent constructions of this section, we assume we are given phase 1b data for all $\beta < \alpha$.

Definition 1.36. *An* 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 1.37. *Semi-allowable permutations act on α-codes.*

Definition 1.38. *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 1.39. *A semi-allowable permutation π is allowable just if*

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

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,

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

so $f_{\gamma,\delta}(\pi_\gamma(g)) = [(\pi_\delta)^{\diagdown\!\diagdown}_{-1} L]$, 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 1.40. *The image under an allowable permutation of a representative code is a representative code.*

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.

[Note: This will probably need the phase 1b data to include an assumption that the typed-near-litter embeddings commute with the group action.] □

Lemma 1.41. *Allowable permutations act on semi-tangles.*

Definition 1.42. *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 1.43. *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 α if and only if it supports β. 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 1.44. *Any code $(\alpha, -1, N)$, where N is a near-litter, gives an element of τ_α.*

Proof. This code is obviously representative [though this isn't really necessary, thanks to Lemma 1.43] and it is supported by a singleton condition, a suitable decorated version of N. □

Lemma 1.45. *For any symmetric $b \in \tau_\beta$, $(\alpha, \beta, \{b\}) \in \tau_\alpha$. In particular, this works for each atom $a \in \tau_{-a}$.*

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 1.46. *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 1.47. *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 $\pi^{\leftarrow} S$. □

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 $\in_{\mathrm{TTT}}$ as its membership relation.

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

1.6 Phase 1c: the embedding into pretangles

Definition 1.48. Phase 1c data *at a proper type index α consists of:*

- *phase 1b data (including phase 1a data) at α;*
- *an injection $\tau \to \mathrm{Pretangle}_\alpha$;*
- *commuting with the* All*-action (where $\varphi : \mathrm{All} \to \mathrm{Str}_\alpha$ and its action on τ is the group action assumed in the phase 1b data, and its action on pretangles is given by composing the Str_α-action with φ).*

For the subsequent constructions of this section, we assume we are given phase 1b data for all $\beta < \alpha$.

Definition 1.49. *There is an injection $\tau_\alpha \to \mathrm{Pretangle}_\alpha$ (where τ_α is constructed from the given phase 1 data). Moreover, this commutes with the All_α-action.*

Chapter 2

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: We need (it seems) to assume not just that we have sets of tangles etc at all earlier levels, but also that they fit together correctly, in the sense of being inductively defined at all levels by the constructions so far. This is the reason why we need the major phase separation in the components of the recursion.

2.1 Assumptions for phase 2: Coherent data at all lower levels

Definition 2.1. (Full) phase 1 data *at α consists of phase 1c data (including 1a and 1b components), together with:*

1. *a "typed singleton" injection $k : \tau_{-1} \to \tau$;*
2. *a "designated" small support $S(x)$ for each tangle $x \in \tau$*

and such that the position functions $\iota_\beta : \tau_\beta \to \mu$ (or equivalently, their induced orderings) satisfy:

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 $N \Delta N^\circ$;*
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 $\iota_\alpha(j(N)), \iota_\alpha(k(a)) < \iota_\alpha(x)$.*

Note: to see these conditions are not unreasonable, note that each element 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. We make this precise in Lemma 2.27 below.

Note: we could have called this "phase 1d data", and should perhaps state it already in phase 1. But it's not needed before phase 2.

Definition 2.2. *For a type index α,* concrete phase 1 data *at level α consists of a subset $\tau \subseteq \mathrm{Pretangle}_\alpha$ and a subgroup $\mathrm{All} \subseteq G_\alpha$, together with full phase 1 data on these.*

Notes for formalisation:

- *The reason we use* subsets *from here on is because we will soon impose equalities between these components, and subset equalities are much less nasty to deal with than type equalities.*
- *To define this, we almost certainly want to factor out the type τ as a parameter of the phase 1a data from the start, and likewise the group* All *in the phase 1b data.*
- *We could avoid that "factoring out as a parameter" by assuming this concreteness already in the phase 1 data, i.e. positing τ as a subset of pretangles and* All *as a subgroup of Str_α from the start. At least for τ, that seems a bit unnatural — it adds an entirely irrelevant extra assumption throughout phase 1. For* All *it perhaps wouldn't be so unnatural, since* All *does need to be assumed with at least a homomorphism to Str_α anyway.*

The standard *concrete phase 1 data at level -1 are the defined $\tau_{-1} = \mathrm{Pretangle}_{-1}$ and $\mathrm{All}_{-1} = \mathrm{Str}_{-1}$.*

Definition 2.3. *For a proper type index $\alpha \in \lambda$, if we have concrete phase 1 data at all levels $\beta < \alpha$, then the constructions in phase 1 provide* most *components of concrete phase 1 data at level α — everything except for the embedding $\iota_\alpha : \tau_\alpha \to \mu$!*

In particular, the new set of α-tangles $\tau_\alpha \subseteq \mathrm{Pretangle}_\alpha$ is the the image of the α-tangles defined in phase 1, under their embedding into pretangles; and similarly for the allowable permutations. (We could call these "concrete α-tangles" and "abstract α-tangles", if we need to for disambiguation.)

[To give the "designated supports" here, we will need to invoke the axiom of choice. This is not entirely necessary, but not too extravagant either: if I (PLL) am not mistaken, what the overall proof really requires here is something like DC_λ, but I can't see how to get by with just that without complicating the recursion structure further, and we are making use of AC elsewhere anyway.]

Definition 2.4. *For $\alpha \in \lambda + 1$,* Phase 1 data below α *consists of concrete phase 1 data at every level $\beta < \alpha$. Say this is* coherent *if it is standard at level -1, and at each proper type index $\beta < \gamma$, all components at level β that* could *have been induced from the data at lower levels $\gamma < \beta$ are, in fact, equal to what would have been induced that way.*

- *Note we want to provide this definiton also for $\alpha = \lambda$! I think this should be available in mathlib as* `with_top`*.*
- *We could unify "standard at level -1" with induced at proper type indices by generalising the definition of each component of the "induced concrete phase 1 data at α" to include the case $\alpha = -1$, defining the induced data as the standard one in that case. Is this more natural? Will it make life easier?*

For $\alpha < \beta$, "phase 2 data below α" can be restricted to "phase 2 data below β".

Most constructions in phase 2 will assume coherent phase 1 data below α, for some fixed α.

[Terminology: perhaps we should call this "phase 2 data at α"? That seems more consistent for the overall naming scheme, but "coherent phase 1 data below α" seems clearer and more transparent in itself. Maybe we should reconsider the naming scheme, so that it's not in tension with clarity!]

2.2 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 coherent phase 1 data below α. [Actually, perhaps this section doesn't need the coherence and could be done in phase 1, i.e. working purely in a single level, assuming full phase 1 data is available at lower levels? This should become clearer as it is formalised.]

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 2.5. *[Note: all this stuff — raising and lowering indices, and the results about repeatedly applying them to structural permutations, etc — is MUCH more clearly handled in algebraic language, using the category of paths, the functoriality of* Str_α *and* All_α *in paths, and maps between group actions, etc.]*

For any α-support S and an extended type index C with minimum element greater than α, we define S^C as $\{((x, A \cup C, \gamma), (y, B \cup C, \delta)) : ((x, A, \gamma), (y, B, \delta)) \in S\}$.

If S is an α-support and $\beta < \alpha$, S_β is defined as the largest support U such that $U^{\{\alpha\}} \subseteq S$ and U is a β-support.

Definition 2.6. *A strong support is a support S with certain additional properties.*

1. *If $((\beta, -1, x), A, \gamma) \in S$ then x is a singleton or a litter.*

2. *If* $((\beta,-1,\{x\}),A,\gamma) \in S$*, then* $((\beta,-1,L),A,\gamma) <_S ((\beta,-1,\{x\}),A,\gamma)$*, where* L *is the litter containing* x*.*

3. *If* $((\beta,-1,L),A,\delta) \in S$ *and* $[L] = f_{\gamma,\beta}(y)$*, where* $\gamma < \delta$*, then there is a* γ*-support* T *of* y *such that* $T^{A\,\{\beta\}} \subseteq S$ *and each element of the domain of* $T^{A\,\{\beta\}}$ *is* $\leq_S ((\beta,-1,L),A)$*. Note that these conditions imply that there is an index-raised version of a strong* γ*-support of* y *included in* T*.*

Lemma 2.7. *Any support can be extended to a strong support.*

Proof. It should be straightforward to see that any X with support S has a support S° which satisfies the first condition. Replace each element $((\beta,-1,x),A)$ of the domain of S for which x is a near-litter and not a litter with $((\beta,-1,x^\circ),A)$ and $((\beta,-1,\{y\}),A)$ for each y in the symmetric difference of x and x°.

We describe the process of extending a support to a strong support, assuming that it already satisfies the first condition. Before each typed singleton element of the domain, insert the appropriate typed litter (removing an extra copy of it if it occurred later in the order). This will only need to be done once for each typed singleton element.

The condition $((\beta,-1,L),A,\delta) \in S$ and $[L] = f_{\gamma,\beta}(y)$ for $\gamma < \delta$ forces insertion of a γ-support for y. Of course this might cause further insertions. Notice that the third components of all inserted items will be $\leq \gamma < \delta$. So it is not possible for an infinite regress of insertions to occur which would cause the extended support to fail to be a well-ordering. (Of course, if an item is inserted which occurs later in the order, remove later occurrences). □

The third components in the elements of support domains seem actually to be necessary to ensure that strong supports can always be produced. What the third component is doing is providing a comment on the index of the sub-support the element is required for. There is another approach: indexing the f maps with α,β,γ instead of just β,γ and requiring that ranges of f maps with distinct indices be disjoint, which would require very straightforward modifications of the paper above, would ensure that we could tell the difference between insertions into an α support and insertions into a β-support for $\beta < \alpha$. It seems that the added complexity is about the same with both approaches, and I have left it this way. The approach used here also has the interesting effect that the extension over type γ of a type α set is determined by its extension over type β in a way which does not depend on α; it is interesting that this can be done.

Note for the formal verification project: This should be ready to go.

2.3 Freedom of action of allowable permutations

The practical application of strong supports is to the proof that allowable permutations act freely in a suitable sense, and in guiding applications of this theorem.

We claim that any locally small specification of values of derivatives of an allowable permutation at elements of type -1 can be realized.

We give an exact statement of what is meant, then we prove it.

Definition 2.8. *An α-local bijection consists of: for each α-extended type index (i.e. path $A : -1 \to \alpha$), a permutation π^0_A of some subset $\mathrm{dom}(\pi^0_A) \subseteq \tau{-}1$, such that the intersection of $\mathrm{dom}(\pi^0_A)$ with any litter is small (including possibly empty).*

Definition 2.9. *We say that x is an* exception *of a near-litter permutation π iff (L being the litter containing x) either $\pi(x) \notin (\pi^{\nwarrow} L)^\circ$ or $\pi^{-1}(x) \notin (\pi^{-1\,\nwarrow} L)^\circ$.*

Stipulation: For each litter L we specify a well-ordering $\leq_L$ of order type κ with coordinated strict well-ordering $<_L$. [In fact this is provided trivially by the current implementation of atoms.]

Definition 2.10. *Given coherent part 1 data below α, a path $A : \beta \to \alpha$, and a litter L, say L is A-*flexible *(or that the pair (L, A) is* flexible*) if L is not in the range of $f_{\beta,\gamma}$, for any $\gamma < \alpha$.*

(In particular, if $\beta = -1$, every such pair is flexible, since $f_{\gamma,\beta}$-maps is only defined for β proper.)

Definition 2.11. *Given coherent part 1 data below α, say that* freedom of action holds (at α) *if: Given an α-local bijection π^0 and a specification for each path $A : \beta \to \alpha$ a bijection of the β-flexible litters, there is a unique allowable permutation $\pi \in \mathrm{All}_\alpha$ such that:*

1. *πA extends π^0_A for each A;*
2. *for each path $A : \beta \to \alpha$ and A-flexible litter L, $\pi_A(j_\beta(L)) = j_\beta(N)$ for some N near to $\chi_A(L)$;*
3. *The only exceptions of π are π^0, and everywhere else, π is order-preserving between litters. That is: for each path $A : -1 \to \alpha$ and litter L, and writing N for the near-litter $\pi^{\nwarrow}_A L$, the restriction of π_A to L $\mathrm{dom}(\pi^0_A)$ is the unique bijection from L $\mathrm{dom}(\pi^0_A)$ to N° $\mathrm{rge}(\pi^0_A)$ which is strictly increasing with respect to the (given/canonical) orders on L and N°.*

Note: Earlier versions of the note/blueprint stated the 3rd condition in terms of exceptions; recent versions of the note spell it out in more detail and add the "increasing" condition, which is necessary for the uniqueness in the theorem conclusion. If there are any doubts, the current version of Randall's note should be consulted!

Lemma 2.12. *Given coherent part 1 data below α, if freedom of action holds at each level $\beta < \alpha$, then it holds at level α.*

Note: I (Peter) have an alternative approach to freedom of action (slightly different in both statement and proof) which might possibly be easier to formalise, avoiding the dependence on strong supports in the proof. Iff Randall approves it then I will add it in the blueprint as an alternative. But it doesn't change the overall structure of the main argument/recursion, so this doesn't need to be worried about until the point of formalising the FoA theorem itself.

Proof. [This proof is is substantially clarified in more recent versions of Randall's note.]

We prove this by exhibiting a recursive procedure for computing π and its derivatives along a strong support; this succeeds because all objects have strong supports, and because computing all values of derivatives of π on type -1 allows computation of all derivatives of π at all types.

For each extended type index, we specify a permutation χ_A of local cardinals $[L]$ is not of the form $f_{\gamma,\beta}(y)$ for any $\gamma < \mathtt{min}(A)$ and $y \in \tau_\gamma$.

We consider an item $((\beta, -1, \{x\}), A, \gamma)$ and our aim is to compute $\pi_{A\cup\{-1\}}(x)$. By hypothesis of the recursion, we have already computed π_A at $((\beta, -1, L), A)$, where L is the litter which contains x.

There are two cases. If (A, x) is in the domain of π^0, we compute $\pi_{A\cup\{-1\}}(x) = \pi^0_A(x)$ and we are done.

Otherwise we use the hypothesis of the recursion: we compute $\pi_{A\cup\{-1\}}(x)$ for any x in L with (A, x) not in the domain of π_0 using the fact that we have already computed $\pi_A(\beta, -1, L) = (\beta, -1, N)$: we define $\pi_{A\cup\{-1\}}$ to agree with the unique bijective map from the elements of L not in the domain of π^0_A to the elements of N° not in the domain of π^0_A which is strictly increasing in the sense that it sends larger objects in the sense of $<_L$ to larger objects in the sense of $<_{N^\circ}$.

Now we consider items of the form $((\beta, -1, L), A, \delta)$ in the strong support where L is a litter.

If $[L]$ is not of the form $f_{\gamma,\beta}(y)$ for a $\gamma < \mathtt{min}(A)$ and $y \in \tau_\gamma$, we compute $\pi_A((\beta, -1, L))$ as

$$(\beta, -1, \pi^0_A{}^{\diagdown\!\diagdown} L \cup (\chi_A([L])^\circ \;\; \pi^0_A{}^{\diagdown\!\diagdown}(\tau_{-1} \;\; L))) :$$

we map L to the near-litter in $\chi_A([L])$ with the exact modifications required by the local bijection.

If $[L]$ is of the form $f_{\gamma,\beta}(y)$ for a $\gamma < \mathtt{min}(A)$ and $y \in \tau_\gamma$ then we proceed just as above but we take the action on $[L]$ from a different source: the coherence condition tells us that $[L]$ should be mapped to $f_{\gamma,\beta}(\pi_{A\ \{\beta\}\cup\{\gamma\}}(y))$, so we compute $\pi_A((\beta, -1, L))$ as

$$(\beta, -1, \pi^0_A{}^{\diagdown\!\diagdown} L \cup (f_{\gamma,\beta}(\pi_{A\ \{\beta\}\cup\{\gamma\}}(y))^\circ \;\; \pi^0_A{}^{\diagdown\!\diagdown}(\tau_{-1} \;\; L))),$$

which is essentially the same idea but a bit more complex.

The nasty recursive idea here is that we already know how to compute $\pi_{A\ \{\beta\}\cup\{\gamma\}}(y)$ from the embedded strong γ support of y because we assume as a hypothesis of the recursion that we already know how to carry out this computation for γ-supports, $\gamma < \alpha$. The local information we need about the γ allowable permutation (the local bijection to be used and the relevant χ maps) is included in the information we are given initially about π (though this data will have indices modified when used on the γ-support, of course). Notice that we certainly do know how to do it for 0-supports, because the recursive clause will never be invoked if $\alpha = 0$: the rest of the procedure tells us what to do.

Once we know how to carry out this calculation along any α-strong support, we can compute the derivatives of π on elements of type -1 along all type paths, and so compute the value of π and all of its derivatives on all types. The method of calculation clearly gives an allowable permutation without exceptions other than those dictated by the local bijection. □

Note for the formal verification project: This section is vitally important and should be ready to work on (once the model and the definition of strong support are handled). Setting up the recursive definition of the computation may be nasty.

Theorem 2.13. *Given coherent phase 1 data below α, it satisfies freedom of action at every level.*

Proof. By induction over $\beta < \alpha$, using Lemma 2.12. □

2.4 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 2.14. *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 $\pi^{\leftthreetimes}_\beta B = B$) 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 $S \;\; S_\beta$ 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 $N \Delta N^\circ$ 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 $X = \varnothing$. 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 $((\beta, -1, \varnothing), A, \delta)$ 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 $((\beta, -1, \varnothing), \chi_{y,T}, \delta)$ 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 2.15. *Fix coherent phase 1 data below α. A* support-specification *is …[this is quite long to define; we should get the text from more recent versions of Randall's note]*

[Perhaps the coherence is not needed?]

Lemma 2.16. *There are $< \mu$-many support specifications.*

Definition 2.17. *Any support specification can be realised (non-uniquely) to give a strong support.*

Lemma 2.18. *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 $\pi S = S'$.*

[Note: we don't need to explicitly assume freedom of action here: we could appeal to the Theorem 2.13 to know that it holds. But the explicit assumption keeps things more modular, and avoids unnecessary dependency, which will be helpful in case it turns out the argument needs restructuring at any point.]

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

Lemma 2.19. *Every strong support can be obtained as the realisation of some specification*

Corollary 2.20. *There are $< \mu$-many orbits of strong supports, under the group action of allowable permutations.*

Proof. Immediate from Lemma 2.16 and Lemma 2.19. □

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

Definition 2.21. *A* coding function specification *is …[again, see recent versions of Randall's note]*

[Perhaps the coherence is not needed?]

Lemma 2.22. *There are $< \mu$-many coding function specifications.*

Definition 2.23. *Any coding function specification can be realised as a coding function.*

Lemma 2.24. *Every coding function can be obtained as the realisation of some specification*

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

Corollary 2.25. *There are $< \mu$-many coding functions.*

Proof. Immediate from Lemma 2.22 and Lemma 2.24. □

Corollary 2.26. *There are μ-many α-tangles.*

Proof. Every tangle can be obtained as a coding function (of which there are $< \mu$) applied to a 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 2.27. *Given coherent phase 1 data below 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 $N \Delta N^\circ$;*
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.

2.5 Completing the recursion

We can now wrap up the recursion. It is a little subtle: to know that the recursion fits together right as we build it up, we need to pin it down some more, to make it uniquely determined.

[A probably cleaner approach would be to state and prove a general principle of λ-*dependent-choice*, (it might be already in mathlib), and apply that to choices of coherent data, avoiding the need for "very coherent". The current approach is effectively unwinding what a general proof of λ-DC would give, applied to this case.]

Definition 2.28. *For $\alpha \in \lambda + 1$, a choice of coherent phase 1 data below α is* very coherent *if its ordering functions and designated supports at each level are the ones provided by Lemma 2.27 applied to the earlier levels, using Theorem 2.13 to supply the assumption of freedom of action.*

[This is sweeping a little under the rug. Both those theorems are stated as existentials, but they are used proof-relevantly here; so before this lemma, we must define "functions" that are just those theorems, wrung through the axiom of choice. See also discusson above replacing this with an explicit principle of "dependent choice".]

Lemma 2.29. *For any α, very coherent data below α is unique (if it exists).*

[Actually, we probably don't need this: it effectively gets inlined in the main recursion.]

Proof. By induction on $\beta < \alpha$: all components of the data at level β are either propositions, or the coherence condition fixes them in terms of earlier data. □

Theorem 2.30. *For every $\alpha \in \lambda + 1$, there is a unique choice of very coherent data below α.*

Proof. By induction on $\beta < \alpha$: take the unique choice below each $\beta < \alpha$; extend each one up to give very coherent data up-to-and-including each $\beta < \alpha$; and then by uniqueness, these fit together to give very coherent data below α. □

Definition 2.31. *The (genuine, definitive!) tangles are the substructure* Tangle *of* Pretangle *given by the unique choice of very coherent data up to λ, provided by Theorem 2.30.*

Chapter 3

Tangles model tangled type theory

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

3.1 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.

3.2 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.