The construction of codes for the atoms and elements of parent sets

M. Randall Holmes

timestamp 6:11 am 2/4/2016; lectured 2/3/2016; typos discovered during the talk corrected; simplicity relation restated, original definition too simple. It still needs repair (8:19 am).
We are working in plain vanilla ZFC.

Let $\lambda$ be a limit ordinal.

If $A$ is a finite subset of $\lambda$, we define $A_1$ as $A \setminus \\{\min(A)\}$. $A_0$ is defined as $A$; $A_{n+1}$ is defined as $(A_n)_1$. We write $A << B$ and say that $A$ downward extends $B$ iff $B \subseteq A$ and all elements of $A \setminus B$ are less than all elements of $B$.

Let $\kappa$ be an uncountable regular cardinal. We call sets of cardinality $< \kappa$ small and all other sets large.

**Motivational remarks:**

Notice that no atoms or clans are mentioned here. These will only be mentioned in remarks like this.
We will recursively define classes $C(A)$, $N(A)$ and $Q(n,A)$ for each finite subset $A$ of $\lambda$ and each natural number $n$ with $0 \leq n \leq |A| + 1$. These classes are in fact sets, which will be proved in due course.

We define an equivalence relation $\sim$ on each of these classes. The equivalence relations can all be denoted by the same symbol because the classes are disjoint.

**Motivational remarks:** Elements of $C(A)$ are codes for atoms in $\text{clan}[A]$. Elements of $N(A)$ are codes for near-litters included in $\text{clan}[A]$. Elements of $Q(n,A)$ are codes for those elements of $\mathcal{P}^n(\text{clan}[A])$ which occur in parent sets, which will turn out to be exactly the elements of $\mathcal{P}^n(\text{clan}[A])$ in the sense of the FM interpretation.
Each class $Q(n, A)$ with $n > 0$ is assigned a complexity which is the minimum element of $A_{n-1}$ if $A_{n-1}$ is nonempty and otherwise $\lambda$.

This is not used until much later.
An element of $C(A)$ will be of the form $(1, a, \alpha, A)$, where $a$ is either an element of $Q(|B| - |A| + 1, B)$ for some $B << A$, or an element of $C(A_1)$ if $A$ is nonempty, or a small ordinal if $A$ is empty.

$(1, a, \alpha, A) \sim (1, b, \beta, A)$ iff $a \sim b$ and $\alpha \equiv \beta$. $\sim$ on small ordinals is defined as equality. This case requires us to state explicitly that the relation $\sim$ will not hold between a small ordinal and anything which is not a small ordinal, or between an element of $C(A)$ and anything not belonging to the same $C(A)$, or between an element of an $N(A)$ and anything not belonging to the same $N(A)$. Under very limited circumstances, elements of distinct classes $Q(n, A)$ may stand in the relation $\sim$ to one another; these are explained below (this is a technical annoyance which can be eliminated in a couple of ways that I am considering).

We say that $a$ is the parent of $(1, a, \alpha, A)$. 
Notice that each $C(A)$ contains a large collection of elements with iterated parent a small ordinal.

We define $L(a, A)$ as $\{(1, a, \alpha, A) : \alpha < \kappa\}$, where $(1, a, \alpha, A) \in C(A)$.

**Motivational Remarks:** Of course $(1, a, \alpha, A)$ is intended to be a code for the atom $\delta(a)^A_\alpha$ where $\delta(a)$ represents the intended referent of $a$. $L(a, A)$ is the code analogue of a litter.
An element of $N(A)$ is of the form $(2, a, D)$ where $(1, a, 0, A) \in C(A)$ and $D$ is a small subset of $C(A)$, and no two distinct elements of $D$ stand in the relation $\sim$ to each other.

We call $a$ the parent of $(2, a, D)$.

$(2, a, D) \sim (2, b, E)$ iff $a \sim b$ and each element of $D$ stands in the relation $\sim$ to some element of $E$ and each element of $E$ stands in the relation $\sim$ to some element of $D$.

An element of $C(A)$ is a formal element of $(2, a, D)$ iff it has parent $a$ and does not stand in the relation $\sim$ to any element of $D$, or if it is an element of $D$ and does not stand in the relation $\sim$ to any element of $L(a, A)$.

**Motivational remark:** $(2, a, D)$ is intended to represent $\text{litter}^A(\delta(a)) \Delta \delta^{\text{"D}}$, a general near-litter.
Each element of $Q(n, A)$ is of the form $(3, f, L)$ where $f$ is a function code (to be explained) and $L$ is an argument list (to be explained).

Every argument list is a function from a set of small ordinals to the union of the classes $C(A)$ and $N(A)$ for all $A$. Values at distinct ordinals of an argument list belonging to the same $C(A)$ do not stand in the relation $\sim$. If $\alpha \neq \beta$ and $L(\alpha)$ and $L(\beta)$ both belong to the same $N(A)$, then no formal element of $L(\alpha)$ stands in the relation $\sim$ to any formal element of $L(\beta)$.

**Motivational remark:** The set $\delta ^ \langle \text{rng}(L) \rangle$ of referents of elements of the range of the argument list is intended to be a support for $\delta((3, f, L))$. 
In addition, each argument list belongs to some argument list type. We proceed to define argument list types.

An argument list type $T$ is determined by three components, a domain $D$, the common domain of all of its elements, a function $A$ with domain $D$ and a function $R$ with domain $D$. The triple $(D, A, R)$ is used as a code for the argument list type.

$D$ will be a set of small ordinals. Each element of $T$ has domain $D$.

For each $\alpha \in D$, $A(\alpha)$ will be of the form $(1, A)$, in which case $L(\alpha) \in C(A)$ for each $\alpha \in D$, or the form $(2, A)$, in which case $L(\alpha) \in N(A)$ for each $L \in T$.

**Motivational remark:** The intention is that each argument list type is an orbit in (ordered) support sets under the allowable permutations.
For each $\alpha \in D$, $R(\alpha)$ will be of one of five forms.

(1) $A(\alpha) = (1, A)$ and $R(\alpha) = \beta < \alpha$, in which case $A(\beta) = (2, A)$ and $L(\alpha)$ is a formal element of $L(\beta)$ for each $L \in T$.

(2) $A(\alpha) = (1, A)$ and $R(\alpha) = \kappa$, in which case $L(\alpha)$ does not stand in the relation $\sim$ to any formal element of any $L(\beta) \in N(A)$ for any $\beta \in D, L \in T$.

Motivational remark: The conditions on this slide handle relations of atoms in an ordered support to near-litters appearing earlier in the list. Those on the next slide handle relations of parents of near-litters in an ordered support to items appearing earlier in the list or external to it.
(3) \( A(\alpha) = (2, A) \) and \( A = \emptyset \) and \( R(\alpha) = (1, \beta) \), \( \beta < \kappa \), in which case \( L(\alpha) \) has parent \( \beta \) for each \( L \in T \).

(4) \( A(\alpha) = (2, A) \) and \( A \neq \emptyset \) and \( R(\alpha) = (2, A_1) \), in which case \( L(\alpha) \) is an element of \( C(A_1) \) which does not stand in the relation \( \sim \) to any \( L(\beta) \) for \( \beta \in D \), for any \( L \in T \).

(5) \( A(\alpha) = (2, A) \) and \( R(\alpha) = (4, f, E) \), [4 is used strictly to avoid a pun with \( (3, f, E) \)] where \( E \subseteq D \cap \alpha \) is an argument list and \( f \) is a function code such that \( (3, f, E) \) belongs to a suitable \( Q(m, B) \), and the parent of \( L(\alpha) \) is \( (3, f, E) \in Q(m, B) \) with \( m > 0 \), or, in the case where \( (3, f, E) \in Q(0, A_1) \), the element of \( C(A_1) \) to which \( (3, f, E) \) reduces [see below for definition], for each \( L \in T \).
The argument lists in an argument list type are exactly the ones which satisfy these conditions for the given $D, A, R$ determining the type.

It is important to reiterate that we will reference an argument list type using its “argument list type code” $(D, A, R)$, which is a much simpler object.
A function code $f$ has an input type, which is an argument list type as just defined, and an output type, which is a $Q(n, A)$. $(3, f, L) \in Q(n, A)$ iff $L$ belongs to the input type of $f$. Further, if $L$ belongs to the input type of $f$, all range elements of $L$ must belong to some $C(B)$ or $N(B)$ with $B << A_{n-1}$ if $n > 0$, or simply to $C(A)$ if $n = 0$. This imposes a strong restriction on what input types go with what output types.
We say that an argument list \( L \) extends an argument list \( M \) iff they agree on the common part of their domain and all domain elements of \( M \) which are not domain elements of \( L \) exceed all domain elements of \( L \). It is important to notice that the domain of \( M \) does not have to include the domain of \( L \).

**Motivational Remark:** The odd definition of extension given here has to do with the fact that we will want to represent elements of a set represented by a code in \( Q(n, A) \) using codes whose argument lists are extensions of the code for that set, but there are items which may appear in the argument list of a code in \( Q(n, A) \) which cannot appear in the argument list of a code in \( Q(n – 1, A) \). We are bounding the complexity of items in the supports we use.
If $n = 0$, all function codes with output type $Q(0, A)$ are of the form $(1, \alpha, D, A)$, where $D$ is a set of small ordinals and $\alpha \in D$. The input type for this code is inhabited by all argument lists with domain $D$ and range $C(A)$. We say that $(3, (1, \alpha, D, A), L)$ reduces to $L(\alpha)$ [this notion of reduction is the one used above] and $(3, (1, \alpha, D, A), L) \sim (3, (1, \beta, E, A), M)$ iff $L(\alpha) \sim M(\beta)$.

We have thus defined all elements of the classes $Q(0, A)$.

**Motivational remark:** Notice that all the functions represented by these function codes are simply projection operators.
If $n > 0$, all function codes with output type $Q(n, A)$ are of the form $(2, U, T, n, A)$, where $T$ is the code for an argument list type and $U$ is a set of function codes with output type $Q(n - 1, A)$ [with some restrictions on input type of elements of $U$ to be explained]. If $L$ belongs to the input type coded by $T$, all range elements of $L$ must belong to some $C(B)$ or $N(B)$ with $B << A_{n-1}$.

$(3, (2, U, T, n, A), L)$ is an element of $Q(n-1, A)$ iff $L$ belongs to the argument list type coded by $T$. 
(3, g, M) ∈ Q(n − 1, A) is a formal element of (3, (2, U, T, n, A), L) iff g ∈ U and M extends L in the formal sense defined above. The formal requirement on U is that the input type of each g ∈ U must be inhabited by argument lists which extend argument lists of the type coded by T. Notice that arguments in L of excessively complex type cannot appear in M. This is part of what makes the recursion work (and enforces sethood of the various classes mentioned).

**Motivational remark:** The intention here is that

\[ \delta(f^*(\delta \circ L)) = \{g^*(\delta \circ M) : g ∈ U \land L ≤ M\} \]

where \( f^* \) denotes the function represented by the function code \( f \) in the natural sense and \( ≤ \) represents our extension relation on argument lists. Notice that sets defined in this way will be highly symmetric.
We now define the relation $\sim$ on $Q(n,A)$. We define an item in an argument list $L$ as an element of a $C(A)$ which appears either as a range element of $L$, or as the second component $a$ or an element of the third component $D$ of a $(2,a,D)$ in the range of $L$.

We say that $(1,a,\alpha,A) \in C(A)$ is remote from an argument list $L$ if it has an iterated parent which is a small ordinal and it has no iterated parent which is an item in $L$ and it is not an item in $L$. Notice that computations of $\sim$ on such objects are trivial.
We say that a notation \((1, a, \alpha, A)\) is a novel item in an argument list \(L\) iff it is a range element which is not a formal element of any element of \(N(A)\) in the range of \(L\), or it is the parent of a range element in \(L\) and is not itself an element of the range of \(L\), or it is an element of the third component \(D\) of an element \((2, a, D)\) of \(N(A)\) in the range of \(L\). We may briefly refer to a range element of an argument list in an \(N(A)\) as novel if its parent is novel.
Note that a non-novel element $L(\alpha)$ of the range of an argument list $L$, if in a $C(A)$, either has parent a small ordinal or is a formal element of an $L(\beta)$ for $\beta < \alpha$ and so either has parent the second component of $L(\beta)$ or is an element of the third component of $L(\beta)$, or, if in an $N(A)$, has parent either a small ordinal, or an $L(\beta) \in C(A_1)$ for $\beta < \alpha$, or parent $g(M)$ where $g$ is a function code appearing as a component of the code for the argument list type of $L$ and $M$ is a sublist of $L \cap \alpha$. The novel items are the only ones which introduce new material not found earlier in the argument list or in the code for its type.
We say that \((3, g, P)\) is an \(M\)-bounded formal element of \((3, (2, U, T, n, A), L)\) iff \(M\) is an argument list, \((3, g, P)\) is a formal element of \((3, (2, U, T, n, A), L)\) and each novel item in \(P\) (which we know extends \(L\)) is either an item in \(L\), an item in \(M\), or remote from both \(L\) and \(M\).

(added after the talk, important)

Each range element \(P(\alpha)\) of \(P\) is either an element of the third component of an element of an \(N(B)\) appearing as \(P(\beta)\) for \(\beta < \alpha\), or an element of a \(C(B)\) with iterated parent either an item in \(L\), an item in \(M\), or a small ordinal, or an element of an \(N(B)\) with parent a small ordinal, or an element of an \(N(B)\) with parent an element of \(C(B_1)\) with an iterated parent either an item in \(L\), an item in \(M\), or a small ordinal, or a \((3, h, Q)\) with \(h\) a function code in the transitive closure of \(g\) and so of \((3, (2, U, T, n, A), L)\) and \(Q\) a sublist of \(P \cap \alpha\) – range elements of \(Q\) thus being similarly bounded in complexity.
We say that \( x = (3, (2, U, T, n, A), L) \in Q(n, A) \) is linked to \( y = (3, (2, U', T', n', A'), L') \in Q(n', A') \) iff each \( L \)-bounded formal element of \( y \) stands in the relation \( \sim \) to some \( L' \)-bounded formal element of \( x \) and each \( L' \)-bounded formal element of \( x \) stands in the relation \( \sim \) to some \( L \)-bounded formal element of \( y \). The device of boundedness is a method of avoiding having to consider equivalences between notations in \( C(A) \)'s with parents of unbounded complexity. Notice that formal elements of an element of \( Q(n, A) \) are in \( Q(n - 1, A) \), and this will eventually bottom out at \( Q(0, A) \), reducing to \( C(A) \).

Note that we state this in a way which makes it possible for the relation \( \sim \) to hold between elements of classes \( Q(n, A) \neq Q(n', A') \): this can happen only if neither object has any iterated formal element in \( C(A) \) or \( C(A') \). The need for this could be avoided by stipulating that the set \( U \) in any \( (2, U, T, n, A) \in Q(n, A) \) be nonempty.
The danger here is that formal elements involve extended argument lists: but the bounding ensures that nothing really new can be added. If we know how to compute $\sim$ on all simpler objects (in a suitable sense to be stated exactly below), we do know how to compute it on everything added in the extensions of argument lists here.
We say that \( x \sim y \) for \( x, y \in Q(n, A) \), \( n > 0 \) iff there is a finite sequence \( z \) with \( z(0) = x \), \( z(m) = y \), \( z(i) \) linked to \( z(i + 1) \) for each \( i \). [the linkage relation is actually transitive (I believe) but one cannot prove this before the \( \sim \) relation is defined, so I state things this way to ensure that it is clear that \( \sim \) is an equivalence relation].

**Motivational remark:** It is in fact the case that elements of \( Q(n, A) \) are equivalent iff they have the same formal elements up to equivalence. It will take a fair amount of work after this definition is complete to verify this. It is necessary to restrict what formal elements are considered in computing \( \sim \), because general formal elements may be of unbounded complexity; there is a converse danger [which I fell into in one version] of restricting the formal elements used in the definition of \( \sim \) too much and getting the wrong relation!
I claim that I have fully defined the classes $C(A)$, $N(A)$, $Q(n, A)$, and the relations $\sim$ on each class. Moreover $\sim$ is an equivalence relation. All computations of $\sim$-equivalence involve further computations of $\sim$-equivalence on simpler objects, eventually bottoming out in equality of small ordinals or of hereditarily finite sets.
(corrected after the talk, the needed simplicity is less simple than I said originally)

The precise measure of simplicity of objects and instances of the equivalence relation \( \sim \) which drives the recursion needs to be stated. A small ordinal is simpler than any element of any of our classes. An object \( x \) of any of our classes is simpler than an object \( y \) of one of our classes if \( x \) is in the transitive closure of \( y \). An object \((3, f, L)\) in a class \( Q(n, A) \) is simpler than \( x \) if \( f \) is in the transitive closure of \( x \) and all items in \( L \) are simpler than \( x \) or have iterated parents simpler than \( x \).
The simplicity relation we use is the intersection of all transitive relations satisfying the stated conditions on simplicity; it is a well-founded relation. We then observe that the evaluation of the truth value of $x \sim y$ reduces in all cases to evaluation of the truth values of instances $z \sim w$ in which $z$ is simpler than $x$ and $w$ is simpler than $y$, and that the verification of membership of an object $x$ in any of our classes will hinge on equivalences $z \sim w$ only in cases where $z$ and $w$ are simpler than $x$. The device of boundedness of formal elements ensures that this holds in computations of $x \sim y$ when $x$ and $y$ belong to classes $Q(n, A)$; in other cases the soundness of the recursion is obvious.
Now we argue that these classes are sets.

We recall that we have assigned each set $Q(n, A)$, $n > 0$, a complexity, equal to the minimum element of $A_{n-1}$ if $A_{n-1}$ is nonempty, and otherwise to $\lambda$. Observe that for each $(3, f, L)$ in $Q(n, A)$, we have each element of $A$ belonging to a $C(B)$ or $N(B)$ with $B << A_{n-1}$. This means that each function code in a $Q(m, D)$, $m > 0$, involved in the argument list type of $f$, contains the parent of an element of $N(B)$, and so we have $D << B << A_{n-1}$ and $m = |D| - |B| + 1$, which tells us that the complexity of $Q(m, D)$ is the minimum element of $D(|D| - |B| + 1) - 1 = B$, and so is less than or equal to the complexity of $A_{n-1}$. 
Clearly the function codes with output type \( Q(0, A) \) make up a set.

We define the complexity of a function code with output type \( Q(n, A) \), \( n > 0 \), as the complexity of \( Q(n, A) \).
Fix a function code of a given complexity and assume that the function codes of lower complexity make up a set. A function code with output type a given class \( Q(n, A) \) with \( n \) positive is determined by a set \( U \) of function codes with output type in \( Q(n-1, A) \) and so of lower complexity, and an argument list type code \( T \) containing as components a small collection of function codes of the same or lower complexity.
Since we suppose as inductive hypothesis that the function codes of lower complexity than the given function code make up a set, we see that the set theoretical rank (the rank in the usual cumulative hierarchy) of the given function code exceeds the supremum of the set theoretical ranks of a small collection of function codes of the same complexity by no more than a fixed ordinal (determined by the structure of argument list codes, the hypothesized bound on the rank of codes of lower complexity, and known bounds on the ranks of small ordinals and finite subsets of $\lambda$). This cannot be true unless the function codes of the complexity of the given function code make up a set. Since there is only a set of complexities, there is only a set of function codes.
Once we know that the function codes make up a set, it is clear that all the classes $C(A)$, $N(A)$, $Q(n,A)$ make up sets. Each element of each of these classes has set theoretical rank exceeding the supremum of the ranks of a small collection of components by no more than a fixed ordinal. In the case of $C(A)$, we are talking about only one component on which we cannot immediately place a bound, its parent. In the case of $N(A)$, there is the parent and a small collection of elements of $C(A)$. In the case of $Q(n,A)$, there is the function code [we know now that these are bounded in rank] and the small range of the argument list, taken from sets $C(B)$ and $N(B)$. 