Nested Multisets, Hereditary Multisets, and Syntactic Ordinals in Isabelle/HOL

Jasmin Christian Blanchette[1,2], Mathias Fleury[2], and Dmitriy Traytel[3]

1 **Vrije Universiteit Amsterdam, The Netherlands**
2 **Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany**
3 **Institute of Information Security, Department of Computer Science, ETH Zürich, Switzerland**

Abstract

We present a collection of formalized results pertaining to (finite) nested multisets, developed using the Isabelle/HOL proof assistant. The nested multiset order is a generalization of the multiset order that can be used to prove termination of processes. Hereditary multisets, a variant of nested multisets, offer a convenient representation of ordinals below ϵ_0. In Isabelle, both nested and hereditary multisets can be comfortably defined as inductive datatypes. Our formal library also provides, somewhat nonstandardly, multisets with negative multiplicities and ordinals with negative coefficients. We present some applications of the library to formalizations of Goodstein's theorem and the decidability of unary programming computable functions (PCF).

Digital Object Identifier 0.0.0.0

1 Introduction

In their seminal article on proving termination using multisets [12], Dershowitz and Manna introduced two orders of increasing strength. The *multiset order* lifts a base partial order on A to (finite) multisets over A. It forms the basis of the multiset path order, which has many applications in term rewriting and automatic theorem proving. The *nested multiset order* is a generalization of the multiset order that makes it possible to nest multisets in arbitrary ways. Nesting can increase the order's strength: If $(A, <)$ has ordinal type $\alpha < \epsilon_0$, the associated multiset order has ordinal type ω^α, whereas the nested order has ordinal type $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$.

In this paper, we present a formal proof of the main properties of the nested multiset order which are useful in applications, namely, preservation of well-foundedness and preservation of totality (linearity). The proof is developed in the Isabelle/HOL proof assistant [24]. To our knowledge, this is the first development of its kind in any proof assistant. Our starting point is the following inductive datatype of nested multisets over a type $'a$ (Section 4):

datatype $'a$ *nmultiset* = Elem $'a$ | MSet (($'a$ *nmultiset*) *multiset*)

This Isabelle command introduces a unary postfix type constructor *nmultiset* equipped with two constructors, Elem : $'a \to 'a$ *nmultiset* and MSet : ($'a$ *nmultiset*) *multiset* $\to 'a$ *nmultiset*, where $'a$ is a type variable and *multiset* is the type constructor of (finite) multisets. It also introduces a recursor that can be used to define primitively recursive functions. In addition, the command provides an induction principle, which allows us to assume, in the MSet M case, that the desired property holds for all nested multisets belonging to M.

The definition of *nmultiset* exhibits recursion through a non-datatype: *multiset*. In recent versions of Isabelle/HOL, recursion is allowed under arbitrary type constructors that are *bounded natural functors* [5, 31], a semantic criterion that is met by bounded sets, multisets,


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and other permutative functors. This very convenient feature is missing in other proof assistants, including Agda, Coq, Lean, and the members of the HOL family.

If we omit the Elem constructor, we obtain the hereditary multisets (Section 5):

datatype *hmultiset* = HMSet (*hmultiset multiset*)

This type is similar to hereditarily finite sets, a model of set theory without the axiom of infinity, but with multisets instead of finite sets. It is easy to embed *hmultiset* in *'a nmultiset*, and using Isabelle's Lifting and Transfer tools [17], we can lift definitions and results from the larger type to the smaller type, such as the definition of the nested multiset order.

Hereditary multisets offer a convenient representation for ordinals below ϵ_0 (Section 6). These are the ordinals that can be expressed syntactically in Cantor normal form:

$$\alpha ::= \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_n} \cdot c_n \qquad \text{where } c_i \in \mathbb{N}^{>0} \text{ and } \alpha_1 > \cdots > \alpha_n$$

The correspondence with hereditary multisets is straightforward:

$$\alpha ::= \{\underbrace{\alpha_1, \ldots, \alpha_1}_{c_1 \text{ occurrences}}, \ldots, \underbrace{\alpha_n, \ldots, \alpha_n}_{c_n \text{ occurrences}}\}$$

The coefficients c_j are represented by multiset multiplicities, and the ω exponents are the multiset's members. Thus: $\{\} = 0$; $\{0\} = \{\{\}\} = \omega^0 = 1$; $\{0,0,0\} = \{\{\},\{\},\{\}\} = \omega^0 \cdot 3 = 3$; $\{1\} = \{\{\{\}\}\} = \omega^{\omega^0} = \omega^1 = \omega$; and $\{\omega\} = \{\{\{\{\}\}\}\} = \omega^\omega$. The standard addition and multiplication operations on ordinals are not commutative—e.g., $1 + \omega = \omega \neq \omega + 1$. Instead, we formalized the Hessenberg (or natural) operations. These are more convenient in many applications and are easier to automate in Isabelle.

When carrying out proofs, we sometimes find ourselves wishing that it would be possible to subtract an ordinal from another. To support this, we define a type of signed multisets, or hybrid multisets [1], and use it to define the signed ordinals (Section 7).

We employ our library to formalize two examples that require ordinals or the nested multiset order: Goodstein's theorem (Section 8) and the decidability of unary programming computable functions (Section 9). Together with colleagues, Blanchette also used the library to formalize a variant of the transfinite Knuth–Bendix order [2]. We gave some thought to proof automation, generalizing existing simplification procedures and exploiting Isabelle's arithmetic type classes. Our work also demonstrates the usefulness of bounded natural functors. The Isabelle theory files are available as part of the *Archive of Formal Proofs* [3].

2 Isabelle/HOL

Isabelle is a generic proof assistant whose metalogic is an intuitionistic fragment of polymorphic higher-order logic (simple type theory). The types are built from type variables *'a*, *'b*, ... and type constructors, written infix or postfix (e.g., $\to$, *multiset*). All types are inhabited. Terms t, u are built from variables x, constants c, abstractions $\lambda x.\ t$, and applications $t\ u$. Constants are higher-order and may be of function type. A formula is a term of type *prop*. The metalogical operators are $\bigwedge$, $\Longrightarrow$, and $\equiv$, for universal quantification, implication, and equality. The notation $\bigwedge x.\ t$ is syntactic sugar for $\bigwedge (\lambda x.\ t)$.

Isabelle/HOL is the instantiation of Isabelle with classical higher-order logic (HOL) extended with type classes as its object logic, complete with a Boolean type *bool*, an equality predicate $=$, the usual connectives ($\neg$, $\wedge$, $\vee$, $\longrightarrow$, $\leftrightarrow$) and quantifiers ($\forall$, $\exists$), and Hilbert choice. HOL formulas, of type *bool*, are embedded in the metalogic. The distinction between *prop* and *bool* is important operationally, but it is not essential to understand this paper.

Isabelle/HOL offers two primitive definitional mechanisms: The **typedef** command introduces a type that is isomorphic to a nonempty subset of an existing type, and the

definition command introduces a constant as equal to an existing term. On top of these, Isabelle provides a rich specification language that includes inductive datatypes and predicates, recursive functions, and their coinductive counterparts, as well as quotient types.

Proofs are expressed either as a sequence of low-level proof methods, *tactics*, that manipulate the proof state directly or in a declarative format called Isar [33], which allows tactics only as terminal procedures. We generally prefer the more readable Isar style. The main proof methods are the *simplifier*, which rewrites terms using conditional oriented equations; the *classical reasoner*, which applies introduction and elimination rules; decision procedures for linear arithmetic; and *metis*, a complete first-order prover based on superposition. In addition, the Sledgehammer tool [26] integrates third-party automatic theorem provers. It can be applied to any proof goal. In case of success, it provides a short Isar proof, often using *metis*.

3 Multisets

Multisets over $'a$ are defined in Isabelle's standard library as isomorphic to the set of multiplicity functions f that are 0 at all but finitely many points x:

typedef $'a\ multiset = \{f : 'a \to nat \mid \mathsf{finite}\ \{x \mid f\ x > 0\}\}$

Multiset values are constructed from the empty multiset $\emptyset$ and the $\mathsf{add}\ x\ A$ operation. For concrete multisets, we use standard set notation. The singleton multiset $\{x\}$ is easy to define in terms of add and $\emptyset$. Multiset union $\uplus$, which adds the multiplicities of its arguments, is an instance of the polymorphic $+ : 'a \to 'a \to 'a$ operator. The relevant $+$ type classes provide a wealth of lemmas and some proof automation. Other operations include $-$, $\cup$, $\cap$, $\subset$, $\subseteq$, $<$, and $\leq$. Given a type $'a$ equipped with a partial order $<$, the $<$ operator on multisets corresponds to the Dershowitz–Manna extension [12]. The extension operator is also available as a function $\mathsf{mult} : ('a \times 'a)\ set \to ('a\ multiset \times 'a\ multiset)\ set$. It is defined as the transitive closure of the one-step Dershowitz–Manna extension $\exists b\ C.\ a \in B \wedge A = (B - \{b\}) + C \wedge (\forall k \in C.\ (k, a) \in R)$.

The operation $\mathsf{image} : ('a \to 'b) \to 'a\ multiset \to 'b\ multiset$ applies a function elementwise to a multiset, and $\mathsf{set} : 'a\ multiset \to 'a\ set$ returns the set of elements in the multiset. Together with the type constructor *multiset* and the cardinal $\aleph_0$, they form a bounded natural functor [5, 31]: image is the functorial action, set is the natural transformation, and $\aleph_0$ is an upper bound on the cardinality of the sets returned by set. Induction and recursion *through* a multiset are expressed in terms of set and image, respectively. The cardinality bound is needed to construct the datatype as the least fixed point of an isomorphism equation.

Isabelle's theory of finite multisets is not as developed as other areas, such as lists and sets. Our first contribution has been to introduce some missing concepts and lemmas, which we added either directly to the Isabelle distribution or collected in a theory file in the *Archive of Formal Proofs* [3]. These include a $\mathsf{replicate}\ n\ x$ operator, which constructs the multiset consisting of n copies of x, and the cartesian product $\times$. We also showed that the Huet–Oppen extension [16] of a partial order coincides with the Dershowitz–Manna extension as well as with the transitive closure of the one-step Dershowitz–Manna extension, allowing users to switch between the three characterizations. Remarkably, the characterizations do not coincide for arbitrary orders [7, Section 3]; each characterization has its advantages.

▶ **Example 1** (McCarthy's 91 Function)**.** Dershowitz and Manna [12] use the multiset order to prove the termination of a tail-recursive formulation of McCarthy's 91 function. Using Isabelle's definitional mechanism for recursive functions [20], the 91 function is specified as

function $\mathsf{g} : nat \to int \to int$ **where**
$\mathsf{g}\ n\ z = (\text{if } n = 0 \text{ then } z \text{ else if } z > 100 \text{ then } \mathsf{g}\ (n-1)\ (z-10) \text{ else } \mathsf{g}\ (n+1)\ (z+11))$

Before it can accept this definition, the command needs a well-founded relation R that includes g's call graph. (Otherwise, we could define a function f such that $\mathsf{f}\ n = \mathsf{f}\ n + 1$ and use this to derive $0 = 1$, a contradiction.) The default automation [9, 21] is not powerful enough to synthesize the relation, so we provide R ourselves. Following Dershowitz and Manna (but correcting $<$ to $\leq$), we define R as $\{((n, z), (n', z')).\ (\tau\ n\ z,\ \tau\ n'\ z') \in \mathsf{mult}\ \{(a, b).\ b < a \wedge a \leq 111\}\}$, where τ is defined as follows, together with an auxiliary function f:

definition $\mathsf{f} : \mathit{int} \to \mathit{int}$ **where** $\mathsf{f}\ x = (\text{if } x > 100 \text{ then } x - 10 \text{ else } 91)$
definition $\tau : \mathit{nat} \to \mathit{int} \to \mathit{int}\ \mathit{multiset}$ **where** $\tau\ n\ z = \mathsf{mset}\ (\mathsf{map}\ (\lambda i.\ \mathsf{f}^i\ z)\ [0\,..\,n-1])$

The main proof obligation is to show that g's call graph is included in R—i.e., that the arguments (n, z) become smaller with each recursive call according to the measure $\tau\ n\ z$ and the multiset order. We followed the original proof, relying on existing lemmas about mult.

The verified SAT solver framework [4] developed mainly by Fleury as part of the IsaFoL (Isabelle Formalization of Logic) effort represents clauses as multisets of literals and problems as multisets of clauses. To improve automation, we developed simplifier plugins, or "simprocs," that cancel terms that appear on both sides of a subtraction, equality, or inequality, rewriting an expression such as $A + \{x, y\} = \mathsf{add}\ x\ A$ to $y = \emptyset$. Since multiset multiplicities are natural numbers, we could start with the cancellation simprocs for *nat*, due to Lawrence Paulson, and generalize them to arbitrary types and operations that enjoy certain properties, including *nat*, *'a multiset*, and the hereditary and signed multiset types introduced in Sections 5 and 7.

The simprocs for *nat* depend on a '$k\times$' operation. We first needed to define its multiset counterpart, $\mathsf{repeat} : \mathit{nat} \to \mathit{'a\ multiset} \to \mathit{'a\ multiset}$, and to prove lemmas about it. The *multiset* instances of our simprocs collectively perform the following steps:

1. Normalize the goal by rewriting $\mathsf{add}\ a\ X$ to $\{a\} + X$ and $\mathsf{replicate}\ n\ a$ to $\mathsf{repeat}\ n\ \{a\}$.
2. Extract A and B from any occurrence of the pattern $A \sim B$ in the goal to normalize, where $\sim$ is among $-$, $=$, $<$, $\leq$, $\subset$, and $\subseteq$.
3. Extract the summands in $A = A_1 + \cdots + A_m$ and $B = B_1 + \cdots + B_n$ to form two lists of multiplicity–term pairs.
4. Find common terms on both sides, subtract the coefficients, and remove the element in the goal using an explicit lemma instantiation.
5. Recombine the simplified terms with $\sim$.
6. Normalize $\{a\}$ back to $\mathsf{add}\ a\ \emptyset$ and simplify add (e.g., replacing $M + \mathsf{add}\ a\ \emptyset + N$ with $\mathsf{add}\ a\ (M + N)$).

In general, the normalization steps are parameterized by rewrite rules, which must be provided for each type instance.

4 Nested Multisets

The type of nested multisets defined in Section 1 is freely generated by the constructors $\mathsf{Elem} : \mathit{'a} \to \mathit{'a\ nmultiset}$ and $\mathsf{MSet} : \mathit{('a\ nmultiset)\ multiset} \to \mathit{'a\ nmultiset}$. The characteristic theorems derived by the **datatype** command [5] include the induction rule

$$(\bigwedge x.\ P\ (\mathsf{Elem}\ x)) \Longrightarrow (\bigwedge N.\ (\bigwedge M.\ M \in N \Longrightarrow P\ M) \Longrightarrow P\ (\mathsf{MSet}\ N)) \Longrightarrow P\ M$$

The command also defines a recursor of type $('a \to 'b) \to (('a\ \mathit{nmultiset} \times 'b)\ \mathit{multiset} \to 'b) \to \mathit{'a\ nmultiset} \to 'b$ and derives its characteristic equations. Using the recursor, we can specify primitively recursive functions on nested multisets. A useful example is the depth of a nested multiset, $\mathsf{depth} : \mathit{'a\ nmultiset} \to \mathit{nat}$, defined as

$$\mathsf{rec}\ (\lambda x.\ 0)\ (\lambda M.\ \text{let}\ X = \mathsf{set}\ (\mathsf{image}\ \mathsf{snd}\ M)\ \text{in if}\ X = \emptyset\ \text{then}\ 0\ \text{else}\ \mathsf{Max}\ X + 1)$$

where snd is the second pair projection and Max returns the maximum of a nonempty finite set of linearly ordered elements. Even for a simple example like this, the recursor-based definition is cryptic. Isabelle's **primrec** command allows users to specify primitively recursive functions by their equations, such as depth Elem $x = 0$ and depth (MSet M) = (let X = set (image depth M) in if $X = \emptyset$ then 0 else Max $X + 1$). Internally, **primrec** defines depth in terms of rec and derives the user's equations from rec's characteristic equations.

The next function we define is Dershowitz and Manna's nested multiset order [12]:

function $<$: *'a nmultiset* $\to$ *'a nmultiset* $\to$ *bool* **where**
  Elem $a <$ Elem $b \leftrightarrow a < b$
| Elem $a <$ MSet $M \leftrightarrow$ True
| MSet $M <$ Elem $a \leftrightarrow$ False
| MSet $M <$ MSet $N \leftrightarrow \mathsf{ext_{DM}}\ (<)\ M\ N$

There are several things to note here. First, we use **function** instead of **primrec** because we are nonprimitively recursing on two nested multisets simultaneously. Second, $\mathsf{ext_{DM}}\ R\ A\ B$ is the Dershowitz–Manna multiset order extension of R applied to multisets A and B. It is defined as $\exists X\ Y.\ Y \neq \emptyset \wedge Y \subseteq B \wedge A = (B - Y) + X \wedge (\forall x \in X.\ \exists y \in Y.\ R\ x\ y)$. Third, the **function** command expects a termination proof in the form of a well-founded relation. We provide the well-founded lexicographic product sub $\times_{\mathsf{lex}}$ sub of the immediate subterm relations sub : (*nmultiset* $\times$ *nmultiset*) *set* defined as sub $= \{(X, \mathsf{MSet}\ M) \mid X \in M\}$. The termination proof crucially relies on the fact that $\mathsf{ext_{DM}}$ applies the relation passed to its first argument only to nested multisets contained in its second and third arguments. This is easy to prove for arbitrary relations for $\mathsf{ext_{DM}}$, but would be significantly harder for mult. And since we have not established the transitivity of the function we are defining yet, we cannot use the equivalence of mult and $\mathsf{ext_{DM}}$ on partial orders. This explains our decision to use $\mathsf{ext_{DM}}$ in the definition. Fourth, after obtaining the termination proof, **function** generates an induction principle that matches the recursion schema used in the definition.

Next, we prove several closure properties of the nested multiset order. If $<$: *'a* $\to$ *'a* $\to$ *bool* is a (nonstrict) preorder, then $<$: *'a nmultiset* $\to$ *'a nmultiset* $\to$ *bool* yields a preorder. The same closure property holds for partial orders, total orders, and wellorders (i.e., well-founded total orders). Every closure property corresponds to a type class instantiation, and every type class instantiation gives us a wealth of lemmas and helpful reasoning infrastructure about the nested multiset order.

Only the proofs of transitivity and well-foundedness of the nested multiset order are challenging. For transitivity, we rely on $\mathsf{ext_{DM}}$ being equivalent to mult on transitive relations. We prove transitivity inductively, and the induction hypothesis establishes that $\mathsf{ext_{DM}}$ is only applied to a transitive relation. To prove well-foundedness, we start with an auxiliary lemma: $<$ is well-founded on the set of nested multisets of a fixed depth i. Formally: wf $\{(M, N) \mid \mathsf{depth}\ M = i \wedge \mathsf{depth}\ N = i \wedge M < N\}$. The proof proceeds by induction on i and uses the equivalence of $\mathsf{ext_{DM}}$ and mult (on transitive relations) as well as the proof of preservation of well-foundedness by mult. Finally, we obtain the well-foundedness of the entire relation by observing that depth $M <$ depth N implies $M < N$ and hence $M < N$ can be rewritten to depth $M <$ depth $N \vee$ depth $M =$ depth $N \wedge M < N$. In other words, we lexicographically compare the depths and resort to the nested multiset order to break ties. The claim follows, since with the above lemma both components of the lexicographic comparison are well founded.

5 Hereditary Multisets

Many authors use nested multisets and the order on them in one way or another. However, most of them do not use the Elem constructor, or use it in an easily avoidable way. In this section, we consider nested multisets with no Elem constructor. In set theory, we could simply let $'a$ be the empty set to model this. Since this is not possible in HOL, we define Elem-freedom as an inductive predicate:

inductive no_elem : $'a$ *nmultiset* $\to$ *bool* **where**
$(\bigwedge N.\ N \in M \Longrightarrow \mathsf{no\_elem}\ N) \Longrightarrow \mathsf{no\_elem}\ (\mathsf{MSet}\ M)$

In principle, we could now use **typedef** to carve out a new type consisting of nested multisets satisfying no_elem. However, the resulting type would be isomorphic to the datatype of hereditary multisets as introduced in Section 1, with its single injective constructor HMSet : *hmultiset multiset* $\to$ *hmultiset*. We prefer the datatype definition, since it offers convenient recursion and induction schemas. Nonetheless, the subtype view on hereditary multisets is also useful, as it allows us to lift the infrastructure from nested to hereditary multisets. Formally, we establish this view by providing an isomorphism, as two mutually inverse injections Abs : *unit nmultiset* $\to$ *hmultiset* and Rep : *hmultiset* $\to$ *unit nmultiset* defined by $\mathsf{Abs}\ (\mathsf{MSet}\ M) = \mathsf{HMSet}\ (\mathsf{image}\ \mathsf{Abs}\ M)$ and $\mathsf{Rep}\ (\mathsf{HMSet}\ M) = \mathsf{MSet}\ (\mathsf{image}\ \mathsf{Rep}\ M)$. The isomorphism follows by easy inductions (on *hmultiset* or on the definition of no_elem):

lemma $\bigwedge M$: *hmultiset*. $\mathsf{no\_elem}\ (\mathsf{Rep}\ M)$
$\bigwedge M$: *hmultiset*. $\mathsf{Abs}\ (\mathsf{Rep}\ M) = M$
$\bigwedge N$: *unit nmultiset*. $\mathsf{no\_elem}\ N \Longrightarrow \mathsf{Rep}\ (\mathsf{Abs}\ N) = N$

The Lifting tool [17] exploits these properties to lift constants on nested multisets to hereditary multisets. Here is our definition of the hereditary multiset order:

lift_definition $<$: *hmultiset* $\to$ *hmultiset* $\to$ *bool* **is**
$<$: *unit nmultiset* $\to$ *unit nmultiset* $\to$ *bool*

The **lift_definition** command produces the definition $M < N \leftrightarrow \mathsf{Rep}\ M < \mathsf{Rep}\ N$, which hides the isomorphism. Moreover, the command also provides a setup for the companion Transfer tool [17], which reduces proof goals about the abstract type (*hmultiset*) to goals about the raw type (*unit nmultiset*). Nevertheless, the proof of the expected property $\mathsf{HMSet}\ M < \mathsf{HMSet}\ N \leftrightarrow \mathsf{ext}_{\mathsf{DM}}\ (<)\ M\ N$ is not trivial, because we must ensure that the witnesses for the existential quantifiers in the definition of $\mathsf{ext}_{\mathsf{DM}}$ contain no Elem. With this property in place, it is easy to lift the instantiations of the various order type classes (up to and including wellorders) from nested to hereditary multisets.

Finally, we prove that *hmultiset* is a cancellative commutative monoid by lifting the corresponding structure from the *multiset* type—i.e., by defining $0 = \mathsf{HMSet}\ \emptyset$, $\mathsf{HMSet}\ A + \mathsf{HMSet}\ B = \mathsf{HMSet}\ (A + B)$, and $\mathsf{HMSet}\ A - \mathsf{HMSet}\ B = \mathsf{HMSet}\ (A - B)$. This enables natural-number-like reasoning with multisets, including our generalized cancellation simprocs.

6 Syntactic Ordinals

Hereditary multisets are isomorphic to the ordinals below ϵ_0, which we call the *syntactic ordinals*. Instead of defining a new type, we use *hmultiset* whenever we need such ordinals and provide the main ordinal operations on this type. Our notion of syntactic ordinal is similar to Dershowitz and Moser's "ordinal terms" [13], which they attribute to Takeuti [29, Section 2.11], but unlike them we do not need to treat the ordinal 0 specially.

The empty hereditary multiset 0 corresponds to the ordinal 0. Union + corresponds to Hessenberg addition, which is traditionally denoted by $\oplus$; we write + to exploit the Isabelle type classes for addition. The multiset order $<$ conveniently coincides with its ordinal counterpart. Multiset subtraction also makes sense as a truncated subtraction on ordinals; for example, $1 - 3 = 0$, $\omega - 3 = \omega$, and $\omega^2 + 3 - \omega = \omega^2 + 3$. Notice that if $\alpha < \beta$, then it is not necessarily the case that $\alpha - \beta = 0$. To provide more complete support for ordinals, we define some additional constants and abbreviations on *hmultiset*, starting with the following basic concepts: $\omega^\alpha = \mathsf{HMSet}\,\{\alpha\}$; $1 = \omega^0$; $\omega = \omega^1$. Given 1 and +, Isabelle lets us enter arbitrary numerals and interprets them as $1 + \cdots + 1$.

The Hessenberg product is defined by taking the cartesian product of the operands' multisets and applying addition on the resulting pairs to obtain a multiset of ordinals, i.e., an ordinal: $\alpha \cdot \beta = \mathsf{HMSet}\ (\mathsf{image}\ (+)\ (\mathsf{hmsetmset}\ \alpha \times \mathsf{hmsetmset}\ \beta))$, where hmsetmset is HMSet's inverse. The expected properties are easy to prove: Multiplication is associative, commutative, and distributive over addition; 0 is absorbent; 1 is neutral; $0 \neq 1$; etc.

It is interesting to compare our definition of the Hessenberg product with the literature. The following definition, by Ludwig and Waldmann [23], illustrates how suboptimal abstractions can lead to convoluted formulations:

- For $\alpha \in \mathbf{O} \setminus \{0\}$ we define: $\quad 0 \odot 0 = 0 \quad 0 \odot \alpha = 0 \quad \alpha \odot 0 = 0$
- Let for $m, m' \in \mathbb{N}^{>0}$, $n_1, \ldots, n_m$, $n'_1, \ldots, n'_{m'} \in \mathbb{N}^{>0}$, $b_1, \ldots, b_m$, $b'_1, \ldots, b'_{m'} \in \mathbf{O}$ such that $b_1 > b_2 > \cdots > b_m$ and $b'_1 > b'_2 > \cdots > b'_m$,

  $$a = \textstyle\sum_{i=1}^{m} (\omega^{b_i} \cdot n_i),\ b = \sum_{i=1}^{m'} (\omega^{b'_i} \cdot n'_i)$$

  We define then

  $$\alpha \odot \beta = \textstyle\bigoplus_{i=1}^{m} \bigoplus_{j=1}^{m'} \left(\omega^{b_i \oplus b'_j} \cdot \left(\mathrm{coeff}(\alpha, b_i) + \mathrm{coeff}(\beta, b'_j)\right)\right)$$

Subtraction is so ill behaved that multiplication does not distribute over it: For $\alpha = \omega^2 + \omega$, $\beta = 1$, and $\gamma = \omega$, we have $\alpha \cdot (\beta - \gamma) = \omega^2 + \omega \neq \omega = \alpha \cdot \beta - \alpha \cdot \gamma$. As a result, some Isabelle type classes for subtraction cannot be instantiated for the ordinals.

Given an ordinal $\sum_{i=1}^{n} (\omega^{\alpha_i} \cdot c_i)$ in Cantor normal form, its degree corresponds to the largest exponent, α_1. Unfortunately, this definition does not gracefully handle the case where $n = 0$. We could extend the ordinal type with a special value (e.g., $-\infty$), but this would require a distinct type and operations on that type. Instead, we introduce the concept of a *head ω-factor*. For 0, the head ω-factor is 0; for nonzero ordinals of degree α, it is ω^α, which is always nonzero. The degree is the maximum element in the multiset that corresponds to the ordinal. Formally: $\mathsf{head}_\omega\ \alpha = \big(\text{if } \alpha = 0 \text{ then } 0 \text{ else } \omega^{\mathsf{Max}\ (\mathsf{set}\ (\mathsf{hmsetmset}\ \alpha))}\big)$.

The following decomposition lemma, which was brought to our attention by Uwe Waldmann, is useful when comparing ordinals. Given two ordinals α_1, α_2 such that $\alpha_1 < \alpha_2$, we can always express them as sums of the form $\alpha_i = \gamma + \beta_i$, where the head ω-factor of β_1 is smaller than that of β_2:

lemma *hmset_pair_decompose_less*:
$\alpha_1 < \alpha_2 \Longrightarrow \exists \gamma\ \beta_1\ \beta_2.\ \alpha_1 = \gamma + \beta_1 \wedge \alpha_2 = \gamma + \beta_2 \wedge \mathsf{head}_\omega\ \beta_1 < \mathsf{head}_\omega\ \beta_2$

▶ **Example 2** (Hydra Battle)**.** The hydra battle [19] is a classic process whose termination cannot be proved by induction on the natural numbers. Following Dershowitz and Moser's reformulation [13], we use Lisp-style lists, or unlabeled binary trees, to represent hydras: **datatype** *lisp* = Nil | Cons *lisp lisp*. The functions car and cdr are defined to extract the first and second arguments of Cons, respectively; they return Nil when invoked on a Nil node.

A hydra consists of a list of heads. In a departure from standard Greek mythology, each head is recursively a list of heads. The battle also involve a hero, Hercules, who gets to chop

off one of the hydra's "leaf" heads at each round. If the head has no grandparent node, the head is lost and there is no regrowth; otherwise, the branch issuing from the grandparent node is copied n times, where n starts at 1 and is increased by 1 in each round. We refer to the literature for more details (and some nice drawings of polycephalic monsters).

Formally, the battle is captured by the h function below, which depends on an auxiliary d:

function $\mathsf{d} : \mathit{nat} \to \mathit{lisp} \to \mathit{lisp}$ **where**
$\mathsf{d}\ n\ x = (\text{if } \mathsf{car}\ x = \mathsf{Nil} \text{ then } \mathsf{cdr}\ x$
$\quad \text{else if } \mathsf{car}\ (\mathsf{car}\ x) = \mathsf{Nil} \text{ then } (\mathsf{Cons}\ (\mathsf{cdr}\ (\mathsf{car}\ x)))^n\ (\mathsf{cdr}\ x)$
$\quad \text{else } \mathsf{Cons}\ (\mathsf{d}\ n\ (\mathsf{car}\ x))\ (\mathsf{cdr}\ x))$

function $\mathsf{h} : \mathit{nat} \to \mathit{lisp} \to \mathit{lisp}$ **where** $\mathsf{h}\ n\ x = (\text{if } x = \mathsf{Nil} \text{ then } \mathsf{Nil} \text{ else } \mathsf{h}\ (n{+}1)\ (\mathsf{d}\ n\ x))$

For d, termination is easy: The left subtree of x becomes smaller with each iteration, so we can take $\{((n', x'), (n, x)).\ |x'| < |x|\}$ as the relation, where $|x|$ returns the number of nodes of a list x. For h, instead of $|x|$, we use $\mathsf{encode}\ x$ as the measure, where encode is defined primitively recursively by $\mathsf{encode}\ \mathsf{Nil} = 0$ and $\mathsf{encode}\ (\mathsf{Cons}\ l\ r) = \omega^{\mathsf{encode}\ l} + \mathsf{encode}\ r$.

Thanks to well-foundedness of $<$, it suffices to show that the ordinal decreases in each recursive call to h—i.e., if $x \neq \mathsf{Nil}$, then $\mathsf{encode}\ (\mathsf{d}\ n\ x) < \mathsf{encode}\ x$. The proof is by structural induction on x. The Nil case is trivial. In the $\mathsf{Cons}\ l\ r$ case, if $l = \mathsf{Nil}$, we have $0 < \omega^0 + \mathsf{encode}\ r$. Otherwise, we distinguish two subcases. If $\mathsf{car}\ l = \mathsf{Nil}$, we must prove $\mathsf{encode}\ (\mathsf{f}\ n\ (\mathsf{cdr}\ l)\ r) < \omega^{\mathsf{encode}\ l} + \mathsf{encode}\ r$, which amounts to $n \cdot \omega^{\mathsf{encode}\ (\mathsf{cdr}\ l)} + \mathsf{encode}\ r < \omega^{\omega^{\mathsf{encode}\ (\mathsf{car}\ l)} + \mathsf{encode}\ (\mathsf{cdr}\ l)} + \mathsf{encode}\ r$. The right-hand side is greater because the exponent to ω has an additional nonzero term, $\omega^{\mathsf{encode}\ (\mathsf{car}\ l)}$, which dwarfs the n factor on the left. In the remaining case, we have $\mathsf{car}\ l \neq \mathsf{Nil}$. The proof obligation amounts to $\mathsf{encode}\ (\mathsf{d}\ n\ l) < \mathsf{encode}\ l$, corresponding to the induction hypothesis for the left subtree.

The termination proof is about 30 lines long in Isabelle. As is often the case, the formalization revealed some inaccuracies in the informal argument. Although this is not mentioned by Dershowitz and Moser, the termination proof depends on $\mathsf{car}\ \mathsf{Nil} = \mathsf{Nil}$. They also claim that $\mathsf{Cons}\ \mathsf{Nil}\ \mathsf{Nil}$ represents a "leaf of a hydra," but the $\mathsf{car}\ x = \mathsf{Nil}$ test in the definition of d can only mean that a simple Nil node is used for leaves.

▶ **Example 3** (Ludwig and Waldmann)**.** The following Isar proof outline closely follows the pen-and-paper proof of a property Ludwig and Waldmann needed to reason about their transfinite Knuth–Bendix order [23]. The informal proof was communicated to us privately by Waldmann. A more restrictive formulation is claimed as Lemma 10 in their paper. The steps are justified by short proofs, omitted below, most of which were generated by Sledgehammer. Using Isabelle's built-in automation and the cancellation simprocs, it would be possible to eliminate some of the intermediate steps.

lemma
  assumes $\alpha_2 + \beta_2 \cdot \gamma < \alpha_1 + \beta_1 \cdot \gamma$ **and** $\beta_2 \leq \beta_1$ **and** $\gamma < \delta$
  shows $\alpha_2 + \beta_2 \cdot \delta < \alpha_1 + \beta_1 \cdot \delta$
proof
  obtain $\eta\ \gamma'\ \delta'$ **where** $\gamma = \eta + \gamma'$ **and** $\delta = \eta + \delta'$ **and** $\mathsf{head}_\omega\ \gamma' < \mathsf{head}_\omega\ \delta'$
  obtain $\beta_0\ \beta_1'\ \beta_2'$ **where** $\beta_1 = \beta_0 + \beta_1'$ **and** $\beta_2 = \beta_0 + \beta_2'$ **and**
    $\mathsf{head}_\omega\ \beta_2' < \mathsf{head}_\omega\ \beta_1' \vee \beta_2' = 0 \wedge \beta_1' = 0$ {by *hmset_pair_decompose_less*}
  have $\alpha_2 + \beta_0 \cdot \gamma + \beta_2' \cdot \gamma = \alpha_2 + \beta_2 \cdot \gamma$ {by $\beta_2 = \beta_0 + \beta_2'$}
  also have $< \alpha_1 + \beta_1 \cdot \gamma$ {by $\alpha_2 + \beta_2 \cdot \gamma < \alpha_1 + \beta_1 \cdot \gamma$}
  also have $= \alpha_1 + \beta_0 \cdot \gamma + \beta_1' \cdot \gamma$ {by $\beta_1 = \beta_0 + \beta_1'$}
  finally have $\alpha_2 + \beta_2' \cdot \gamma < \alpha_1 + \beta_1' \cdot \gamma$ {by cancellation}

have $\alpha_2 + \beta_2 \cdot \delta = \alpha_2 + \beta_0 \cdot \delta + \beta_2' \cdot \delta$ {by $\beta_2 = \beta_0 + \beta_2'$}
also have $= \alpha_2 + \beta_0 \cdot \delta + \beta_2' \cdot \eta + \beta_2' \cdot \delta'$ {by $\delta = \eta + \delta'$}
also have $\leq \alpha_2 + \beta_0 \cdot \delta + \beta_2' \cdot \eta + \beta_2' \cdot \delta' + \beta_2' \cdot \gamma'$ {by monotonicity}
also have $= \alpha_2 + \beta_2' \cdot \gamma + \beta_0 \cdot \delta + \beta_2' \cdot \delta'$ {by $\gamma = \eta + \gamma'$}
also have $< \alpha_1 + \beta_1' \cdot \gamma + \beta_0 \cdot \delta + \beta_2' \cdot \delta'$ {by $\alpha_2 + \beta_2' \cdot \gamma < \alpha_1 + \beta_1' \cdot \gamma$}
also have $= \alpha_1 + \beta_1' \cdot \eta + \beta_1' \cdot \gamma' + \beta_0 \cdot \eta + \beta_0 \cdot \delta' + \beta_2' \cdot \delta'$ {by $\gamma = \eta + \gamma'$, $\delta = \eta + \delta'$}
also have $\leq \alpha_1 + \beta_1' \cdot \eta + \beta_0 \cdot \eta + \beta_0 \cdot \delta' + \beta_1' \cdot \delta'$
{by $\mathsf{head}_\omega\,(\beta_1' \cdot \gamma' + \beta_2' \cdot \delta') < \mathsf{head}_\omega\,(\beta_1' \cdot \delta') \vee \beta_1' \cdot \gamma' = \beta_2' \cdot \delta' = \beta_1' \cdot \delta' = 0$}
finally show $\alpha_2 + \beta_2 \cdot \delta < \alpha_1 + \beta_1 \cdot \delta$ {by $\beta_1 = \beta_0 + \beta_1'$, $\delta = \eta + \delta'$}
qed

7 Signed Variants of Multisets and Hereditary Multisets

Syntactic ordinals do not enjoy the property that Ludwig and Waldmann refer to as "continuity" [23]: Given syntactic ordinals α, β such that $\alpha < \beta$, there does not necessarily exist an ordinal γ such that $\alpha + \gamma = \beta$. Yet, syntactic ordinals correspond formally to polynomials over the indeterminate ω. Why not allow negative coefficients (e.g., $\omega^2 - 2\omega + 1$) and take $\gamma = \beta - \alpha$? The practical motivation arose in the context of the formalization of a transfinite Knuth–Bendix order [2]. Although it is a simple idea, we could not find it in the literature.

We start by defining the *signed multisets*, also called the hybrid multisets [1]. In principle, we could define them in a similar way as the plain multisets (Section 3), by substituting *int* for *nat* in the **typedef** command. However, we prefer to perform a quotient construction, so as to be able to lift operations and lemmas about plain multisets:

quotient_type *'a zmultiset* = *'a multiset* × *'a multiset* / R

where $R = \lambda(P_1, N_1)\ (P_2, N_2).\ P_1 + N_2 = P_2 + N_1$. The **quotient_type** command [18] introduces a type that is isomorphic to the set of R-equivalence classes. In the same way as a pair (m, n) of natural numbers can represent the integer $m - n$, we use a pair (P, N) of plain multisets to capture the signed multiset $P - N$, whose multiplicities can be negative. For example, both $(\emptyset, \{7\})$ and $(\{3\}, \{3, 7\})$ correspond to the signed multiset that contains 7 with multiplicity -1 (whatever that means) and no other element. Notice that $R\ (\emptyset, \{7\})\ (\{3\}, \{3, 7\})$, since $\emptyset + \{3, 7\} = \{7\} + \{3\}$. The pos and neg functions, of type *'a zmultiset* → *'a multiset*, return normalized P and N components, with $P \cap N = \emptyset$.

The main signed multiset operations are defined by specifying them on the raw level of multiset pairs and lifting them to the abstract level of equivalence classes. The Lifting tool emits a proof obligation stating that equivalence classes are respected. For example, for unary functions f on multisets, this means that when f is invoked on R-equivalent arguments x and y, the results $f\ x$ and $f\ y$ are R-equivalent. A few definitions are given below:

lift_definition 0 : *'a zmultiset* **is** $(\emptyset, \emptyset)$

lift_definition $-$: *'a zmultiset* → *'a zmultiset* → *'a zmultiset* **is**
$\lambda(P_1, N_1)\ (P_2, N_2).\ (P_1 + N_2,\ N_1 + P_2)$

lift_definition zcount : *'a zmultiset* → *'a* → *int* **is**
$\lambda(P, N)\ x.$ int (count $P\ x$) − int (count $N\ x$)

Many lemmas about such definitions can be lifted from the raw types using the Transfer tool, which exploits parametricity to transfer results across types. Most HOL functions arising in practice are parametric.

The type *zhmultiset* of signed hereditary multisets is defined using the **typedef** command as isomorphic to *hmultiset zmultiset*. Notice that the multisets contained in such a signed

multiset are not signed. We leave the study of "hereditarily signed multisets," and of ordinals in which the exponents of ω can recursively contain negative coefficients, to other researchers.

The ι function embeds *hmultiset* into *zhmultiset*. Operations such as $+$, $-$, and $<$ are lifted from the underlying multisets. Ordinal multiplication is by far the most problematic operation. It can be defined in terms of the cartesian product on signed multisets:

lift_definition $\cdot$: *zhmultiset* $\to$ *zhmultiset* $\to$ *zhmultiset* **is**
$\lambda M\ N$: *hmultiset zmultiset*.
$\quad \iota$ (hmsetmset (HMSet (pos M) $\cdot$ HMSet (pos N)))
$- \iota$ (hmsetmset (HMSet (pos M) $\cdot$ HMSet (neg N)))
$+ \iota$ (hmsetmset (HMSet (neg M) $\cdot$ HMSet (neg N)))
$- \iota$ (hmsetmset (HMSet (neg M) $\cdot$ HMSet (pos N)))

It is difficult to prove the associativity of this multiplication operator. Using the Transfer tool, we can quickly reduce the proof obligation to the following property on unsigned ordinals:

$$\begin{aligned}
&\alpha_2 \cdot (\beta_2 \cdot \gamma_2 + \beta_1 \cdot \gamma_1 - (\beta_2 \cdot \gamma_1 + \gamma_2 \cdot \beta_1)) + \gamma_2 \cdot (\alpha_2 \cdot \beta_1 + \beta_2 \cdot \alpha_1 - (\alpha_2 \cdot \beta_2 + \alpha_1 \cdot \beta_1)) + \\
&\alpha_1 \cdot (\beta_2 \cdot \gamma_1 + \gamma_2 \cdot \beta_1 - (\beta_2 \cdot \gamma_2 + \beta_1 \cdot \gamma_1)) + \gamma_1 \cdot (\alpha_2 \cdot \beta_2 + \alpha_1 \cdot \beta_1 - (\alpha_2 \cdot \beta_1 + \beta_2 \cdot \alpha_1)) \\
= {}& \alpha_2 \cdot (\beta_2 \cdot \gamma_1 + \gamma_2 \cdot \beta_1 - (\beta_2 \cdot \gamma_2 + \beta_1 \cdot \gamma_1)) + \gamma_2 \cdot (\alpha_2 \cdot \beta_2 + \alpha_1 \cdot \beta_1 - (\alpha_2 \cdot \beta_1 + \beta_2 \cdot \alpha_1)) + \\
&\alpha_1 \cdot (\beta_2 \cdot \gamma_2 + \beta_1 \cdot \gamma_1 - (\beta_2 \cdot \gamma_1 + \gamma_2 \cdot \beta_1)) + \gamma_1 \cdot (\alpha_2 \cdot \beta_1 + \beta_2 \cdot \alpha_1 - (\alpha_2 \cdot \beta_2 + \alpha_1 \cdot \beta_1))
\end{aligned}$$

After staring at this goal for a few hours, we discovered the lemma $\alpha \cdot (\gamma - \beta) + \alpha \cdot \beta = \alpha \cdot (\beta - \gamma) + \alpha \cdot \gamma$ about truncated subtraction, which must be applied four times.

We instantiated our generalized cancellation simprocs for *zmultiset* and *zhmultiset*. As preprocessing steps, we normalize unary minuses so that $-A+B+A$ is rewritten to $(B+A)-A$; the subtraction simproc then cancels the two A's, yielding B.

▶ **Example 4** (Ludwig and Waldmann, Continued)**.** By exploiting signed ordinals, we obtain a much simpler proof of Ludwig and Waldmann's property from Example 3:

have $\iota\,\alpha_2 + \iota\,\beta_2 \cdot \iota\,\delta = \iota\,\alpha_2 + \iota\,\beta_2 \cdot \iota\,\gamma + \iota\,\beta_2 \cdot (\iota\,\delta - \iota\,\gamma)$	{by algebraic manipulations}
also have $< \iota\,\alpha_1 + \iota\,\beta_1 \cdot \iota\,\gamma + \iota\,\beta_2 \cdot (\iota\,\delta - \iota\,\gamma)$	{by $\alpha_2 + \beta_2 \cdot \gamma < \alpha_1 + \beta_1 \cdot \gamma$}
also have $\leq \iota\,\alpha_1 + \iota\,\beta_1 \cdot \iota\,\gamma + \iota\,\beta_1 \cdot (\iota\,\delta - \iota\,\gamma)$	{by $\beta_2 \leq \beta_1$, $\gamma < \delta$}
also have $= \iota\,\alpha_1 + \iota\,\beta_1 \cdot \iota\,\delta$	{by algebraic manipulations}
finally show $\alpha_2 + \beta_2 \cdot \delta < \alpha_1 + \beta_1 \cdot \delta$	{by algebraic manipulations}

The next-to-last step eliminates the subtraction $\iota\,\delta - \iota\,\gamma$, paving the way for the final step, which eliminates the ι casts. Waldmann privately confirmed that he was aware of this shortcut but did not dare take it without a solid theoretical foundation for signed ordinals.

8 Application to Goodstein's Theorem

Goodstein's theorem [14] states that every Goodstein sequence eventually terminates at 0. Before we can define these sequences, we must first introduce an auxiliary notion. A natural number is in *hereditary base* n if it is expressed as a product $c_1 n^k + c_2 n^{k-1} + \cdots + c_{k-1} n + c_k$, where $0 \leq c_i < n$ for each i, $c_1 \neq 0$, and the exponents $k, k-1, \ldots, 1$ are recursively expressed in hereditary base n. For example, 500 is written as $2 \cdot 3^{2 \cdot 3 + 2} + 3^2 + 3 + 2$ in hereditary base 3.

The Goodstein sequence $\mathcal{G}_s$ of natural numbers is defined as follows. The starting value is given by s: $\mathcal{G}_s(0) = s$. The remaining values $\mathcal{G}_s(i+1)$ are obtained by expressing $\mathcal{G}_s(i)$ in base $i+2$, replacing all occurrences of $i+2$ by $i+3$, and subtracting 1. For example, we have $\mathcal{G}_4(0) = 4 = 2^2$, $\mathcal{G}_4(1) = 3^3 - 1 = 26 = 2 \cdot 3^2 + 2 \cdot 3 + 2$, and $\mathcal{G}_4(2) = 2 \cdot 4^2 + 2 \cdot 4 + 2 - 1 = 41$. Somewhat counterintuitively, the sequence eventually converges to 0: $\mathcal{G}_4(3 \cdot 2^{402\,653\,211} - 1) = 0$.

Our formal proof relies on two functions encode and decode that convert between natural numbers and hereditary base n. The adjective 'hereditary' suggests that hereditary multisets, or syntactic ordinals, are a suitable data structure. Following this idea, $2 \cdot \omega^2 + 2 \cdot \omega + 2$ represents $2 \cdot 3^2 + 2 \cdot 3 + 2$ in hereditary base 3, or $2 \cdot 5^2 + 2 \cdot 5 + 2$ in hereditary base 5.

The encode and decode functions are defined in a local context that fixes a constant $\mathsf{base} \geq 2$. Beyond the end of the local context, we must supply the base explicitly as additional argument to encode and decode, which we indicate as a subscript.

function encode : $nat \to nat \to hmultiset$ **where**
  $\mathsf{encode}\ e\ n = (\text{if } n = 0 \text{ then } 0 \text{ else } (n \bmod \mathsf{base}) \cdot \omega^{\mathsf{encode}\ 0\ e} + \mathsf{encode}\ (e+1)\ (n/\mathsf{base}))$

primrec decode : $nat \to hmultiset \to nat$ **where**
  $\mathsf{decode}\ e\ (\mathsf{HMSet}\ M) = \big(\sum_{\alpha \in M} \mathsf{base}^{\mathsf{decode}\ 0\ \alpha}\big) / \mathsf{base}^e$

The argument e gives the exponent of the current base, starting from 0. The recursion scheme for encode is somewhat unusual. Termination is established using the measure $\lambda(e, n).\ n \cdot (\mathsf{base}^e + 1)$. The Goodstein sequence is defined as follows, where start is fixed:

primrec goodstein : $nat \to nat$ **where**
  $\mathsf{goodstein}\ 0 = \mathsf{start}$
| $\mathsf{goodstein}\ (\mathsf{Suc}\ i) = \mathsf{decode}_{i+3}\ 0\ (\mathsf{encode}_{i+2}\ 0\ (\mathsf{goodstein}\ i)) - 1$

If $\mathsf{goodstein}\ i > 0$, then the ordinal associated with $\mathsf{goodstein}\ (i+1)$ is smaller than the one for $\mathsf{goodstein}\ i$. Let $\mathcal{E}_i = \mathsf{encode}_{i+2}\ 0\ (\mathsf{goodstein}\ i)$. The proof sketch is as follows:

lemma *goodstein_step*:
  assumes $\mathsf{goodstein}\ i > 0$
  shows $\mathcal{E}_i > \mathcal{E}_{i+1}$
proof
  have $\mathsf{decode}_{i+3}\ 0\ \mathcal{E}_i > 0$ {by *decode_0_iff*}
  have $\mathcal{E}_i = \mathsf{encode}_{i+3}\ 0\ (\mathsf{decode}_{i+3}\ 0\ \mathcal{E}_i)$ {by *encode_decode_exp_0*}
  also have $> \mathsf{encode}_{i+3}\ 0\ (\mathsf{decode}_{i+3}\ 0\ \mathcal{E}_i - 1)$
    {by *less_imp_encode_less*, $\mathsf{decode}_{i+3}\ 0\ \mathcal{E}_i > 0$}
  finally show $\mathcal{E}_i > \mathcal{E}_{i+1}$ {by definition of $\mathcal{E}_{i+1}$}
qed

The main lemmas that are directly or indirectly needed are listed below:

lemma *less_imp_encode_less*: $n < p \Longrightarrow \mathsf{encode}\ e\ n < \mathsf{encode}\ e\ p$
lemma *less_imp_decode_less*:
  $\mathsf{well\_base}\ \alpha \Longrightarrow \mathsf{aligned}\ e\ \alpha \Longrightarrow \mathsf{aligned}\ e\ \beta \Longrightarrow \alpha < \beta \Longrightarrow \mathsf{decode}\ e\ \alpha < \mathsf{decode}\ e\ \beta$
lemma *decode_0_iff*: $\mathsf{well\_base}\ \alpha \Longrightarrow \mathsf{aligned}\ e\ \alpha \Longrightarrow (\mathsf{decode}\ e\ \alpha = 0 \longleftrightarrow \alpha = 0)$
lemma *decode_encode*: $\mathsf{decode}\ e\ (\mathsf{encode}\ e\ n) = n$
lemma *encode_decode_exp_0*: $\mathsf{well\_base}\ \alpha \Longrightarrow \mathsf{encode}\ 0\ (\mathsf{decode}\ 0\ \alpha) = \alpha$

The first and second properties are the most difficult ones. The first one is proved by well-founded induction, following the recursion of encode. This induction principle is derived automatically by the **function** command and is available as *encode.induct*. The second property is proved by strong induction on α. The assumptions ensure that the coefficients stored in α are smaller than the base (well_base) and that the last e digits are all 0s (aligned).

The entire formalization is about 580 lines long. The main difficulty was to come up with the right lemmas and inductions, especially given that we did not consult the literature. Reasoning about ordinals was fairly comfortable. Nevertheless, we are very impressed by

Winkler, Zankl, and Middeldorp's automatic proof of the termination of a term rewriting system that computes Goodstein's sequence [34]. Possibly the key to their success is that they avoid converting back and forth between the natural numbers and hereditary base notation.

9 Application to Decidability of Unary PCF

Plotkin's programming computable functions (PCF) language [27] is a simply typed λ-calculus that has natural numbers $\mathbb{N}$ as a base type and permits recursion on them. Types are interpreted as Scott domains, i.e., $[\![\mathbb{N}]\!] = \mathbb{N} \cup \{\bot\}$. For unary PCF—a fragment of PCF which has only the base type o with the single value $\top$ (in addition to the domain's $\bot$)—behavioral equivalence is decidable [22, 28]. Schmidt-Schauß's elegant proof [28] is based on an inductive enumeration of representative terms. The termination of the enumeration is ensured by abstracting types into hereditary multisets. In our ongoing formalization effort of this decidability result (which poses many challenges unrelated to multisets), we proved Schmidt-Schauß's key Lemma 11 about hereditary multisets.

Types of unary PCF are defined as **datatype** $\mathit{type} = \mathsf{o} \mid \mathit{type} \Rightarrow \mathit{type}$, where $\Rightarrow$ is a right-associative infix datatype constructor. For a type $T = U_0 \Rightarrow \cdots \Rightarrow U_{n-1} \Rightarrow \mathsf{o}$, we write $\mathsf{ar}\ T$ for its arity n and T_i for the type of its $(i+1)$st argument U_i. We measure a type using a primitively recursive function $\delta : \mathit{type} \to \mathit{hmultiset}$ defined by $\delta\ \mathsf{o} = 0$ and $\delta\ (T \Rightarrow U) = \omega^{\delta\ T} + \delta\ U$. For a type T, Schmidt-Schauß constructs a set of representative closed terms of behavioral equivalence classes. The construction is recursive and relies on a decrease in the involved types' measures for termination. More precisely, given T, the construction recursively computes sets of representative terms for types $\pi_i^j\ T$ for all $i < \mathsf{ar}\ T$ and $j \le \mathsf{ar}\ T_i$, where the operator π is defined recursively as follows:

fun $\pi : \mathit{nat} \to \mathit{nat} \to \mathit{type} \to \mathit{type}$ **where**
$\pi_i^0\ T = \text{if } i < \mathsf{ar}\ T \text{ then } T_0 \Rightarrow \cdots \Rightarrow T_{i-1} \Rightarrow T_{i+1} \Rightarrow \cdots \Rightarrow T_{\mathsf{ar}\ T-1} \Rightarrow \mathsf{o} \text{ else } \mathsf{o}$
$\pi_i^{j+1}\ T = \text{if } i < \mathsf{ar}\ T \wedge j < \mathsf{ar}\ T_i \text{ then } \pi_j^0\ T_i \Rightarrow \cdots \Rightarrow \pi_j^{\mathsf{ar}\ (T_i)_j}\ T_i \Rightarrow \pi_i^0\ T \text{ else } \mathsf{o}$

Finally, we prove that π indeed decreases the measure of types using the induction principle that follows the structure of π's definition and is provided by Isabelle's **fun** command [20].

lemma δ_π:
  assumes $i < \mathsf{ar}\ T$ **and** $j \le \mathsf{ar}\ T_i$
  shows $\delta\ (\pi_i^j\ T) < \delta\ T$
proof ($\mathit{induct\ rule} : \pi.\mathit{induct}$)
  fix $T\ i$
  assume $i < \mathsf{ar}\ T$
  show $\delta\ (\pi_i^0\ T) < \delta\ T$ {by definition of δ and π and simple multiset reasoning}
next
  fix $T\ i\ j$
  assume $i < \mathsf{ar}\ T$ **and** $j < \mathsf{ar}\ T_i$ **and**
    IH_1: $\delta\ (\pi_j^0\ T_i) < \delta\ T_i$ **and** IH_2: $\forall k < \mathsf{ar}\ (T_i)_j.\ \delta\ (\pi_j^{k+1}\ T_i) < \delta\ T_i$
  define $\mathsf{X} = \{\delta\ (\pi_j^0\ T_i)\} + \mathsf{image}\ (\lambda k.\ \delta\ (\pi_j^{k+1}\ T_i))\ \{0, \ldots, \mathsf{ar}\ (T_i)_j - 1\}$ **and**
    $\mathsf{Y} = \{\delta\ T_i\}$ **and** $\mathsf{Z} = \mathsf{image}\ \delta\ \{T_0, \ldots, T_{i-1}, T_{i+1}, \ldots, T_{\mathsf{ar}\ T-1}\}$
  have $\delta\ (\pi_i^{j+1}\ T) = \mathsf{HMSet}\ (\mathsf{X} + \mathsf{Z})$ {by definitions of δ and π}
  also have $\mathsf{X} + \mathsf{Z} < \mathsf{Y} + \mathsf{Z}$ {by Dershowitz–Manna characterization of $<$, IH_1, IH_2}
  also have $\mathsf{HMSet}\ (\mathsf{Y} + \mathsf{Z}) = \delta\ T$ {by definition of δ}
  finally show $\delta\ (\pi_i^{j+1}\ T) < \delta\ T$ {by above calculation}
qed

The key step to help automation is to define X, Y, and Z (all of type *hmultiset multiset*) such that after unfolding the Dershowitz–Manna definition of the multiset order, the inequality is easily fulfilled (given the induction hypotheses).

Our formal proof is very close to Schmidt-Schauß's informal argument. It is rare to be able to formalize a technical proof so closely. This can be due to the care taken by the informal proof writer, due to good formal library support, or due to a combination of both.

10 Related Work

The nonnested multiset order has been formalized in several proof assistants, including Coq [8, 11], HOL4, and Isabelle/HOL [30].

Norrish and Huffman [25] present two formalizations of ordinals, in HOL4 and Isabelle/HOL. The HOL4 formalization models ordinals as quotients of wellorders with respect to wellorder isomorphism. The Isabelle/HOL development also relies on a quotient construction, but from a more syntactic notion of raw ordinals, defined as **datatype** *preordinal* = Zero | StrictLim (*nat* → *preordinal*). Independently, Blanchette, Popescu, and Traytel [6] formalized ordinals and cardinals in Isabelle/HOL, representing ordinals by well-ordered relations dispersed over arbitrary types. Thereby they avoided fixing an a priori bound on the ordinals that can be constructed. All these formalizations go beyond ϵ_0 but are ultimately limited by the expressiveness of HOL, which is strictly weaker than set theory. Another difference with our current work is that they use the standard addition and product and not Hessenberg's.

In Coq, Castéran and Contejean [10] formalized ordinal notations up to Γ_0. As case studies, they considered the hydra battle and Goodstein's theorem. Grimm [15] ported and extended this work, covering three alternative notions of ordinals. Vermaat [32] formalized tree ordinals, a syntactic representation similar to Norrish and Huffman's.

11 Conclusion

We presented a formalization in Isabelle/HOL of nested multisets, hereditary multisets, and ordinals below ϵ_0. Their datatype definitions emphasize the close connections between the three notions. The signed generalizations of these types, with potentially negative multiplicities, offer a subtraction operator that enjoys nice algebraic properties. The signed syntactic ordinals do not appear to have been studied before.

We found the Lifting and Transfer tools invaluable for carrying definitions across the different related multiset types. We relied heavily on Sledgehammer; it sometimes generated complex Isar proofs, which we occasionally inserted in our development. The cancellation simprocs also played a role, after we had adapted them so that they work on multisets. Well-founded recursion using **function** was vital, as it often is. But perhaps the most noteworthy aspect of our work is that all the necessary types could be introduced easily, either as inductive datatypes, as subsets of existing types, or as quotients. The support for recursion through bounded natural functors, a distinguishing feature of Isabelle, was crucial to define nested and hereditary multisets.

Acknowledgment

Lawrence Paulson gave us the idea to formalize the nested multiset order. Aart Middeldorp pointed us to related work. Uwe Waldmann shared his insights about syntactic ordinals with us, including the lemma and the proof of Example 3. Robert Lewis, Mark Summerfield, and Sophie Tourret suggested many textual improvements.

Blanchette has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka).

References

1 Jean-Pierre Banâtre, Pascal Fradet, and Yann Radenac. Generalised multisets for chemical programming. *Math. Struct. Comput. Sci.*, 16(4):557–580, 2006.

2 Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. Formalization of Knuth–Bendix orders for lambda-free higher-order terms. *Archive of Formal Proofs*, 2016. URL: `https://devel.isa-afp.org/entries/Lambda_Free_KBOs.shtml`.

3 Jasmin Christian Blanchette, Mathias Fleury, and Dmitriy Traytel. Formalization of nested multisets, hereditary multisets, and syntactic ordinals. *Archive of Formal Proofs*, 2016. URL: `https://devel.isa-afp.org/entries/Nested_Multisets_Ordinals.shtml`.

4 Jasmin Christian Blanchette, Mathias Fleury, and Christoph Weidenbach. A verified SAT solver framework with learn, forget, restart, and incrementality. In Nicola Olivetti and Ashish Tiwari, editors, *IJCAR 2016*, volume 9706 of *LNCS*, pages 25–44. Springer, 2016.

5 Jasmin Christian Blanchette, Johannes Hölzl, Andreas Lochbihler, Lorenz Panny, Andrei Popescu, and Dmitriy Traytel. Truly modular (co)datatypes for Isabelle/HOL. In Gerwin Klein and Ruben Gamboa, editors, *ITP 2014*, volume 8558 of *LNCS*, pages 93–110. Springer, 2014.

6 Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Cardinals in Isabelle/HOL. In Gerwin Klein and Ruben Gamboa, editors, *ITP 2014*, volume 8558 of *LNCS*, pages 111–127. Springer, 2014.

7 Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A lambda-free higher-order recursive path order. In Javier Esparza and Andrzej Murawski, editors, *FoSSaCS 2017*, LNCS. Springer, 2017.

8 Frédéric Blanqui and Adam Koprowski. CoLoR: A Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Math. Struct. Comput. Sci.*, 21(4):827–859, 2011.

9 Lukas Bulwahn, Alexander Krauss, and Tobias Nipkow. Finding lexicographic orders for termination proofs in Isabelle/HOL. In Klaus Schneider and Jens Brandt, editors, *TPHOLs 2007*, volume 4732 of *LNCS*, pages 38–53. Springer, 2007.

10 Pierre Castéran and Évelyne Contejean. On ordinal notations. *Coq User Contributions*, 2006. URL: `http://www.lix.polytechnique.fr/coq/V8.2pl1/contribs/Cantor.html`.

11 Evelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. Certification of automated termination proofs. In Boris Konev and Frank Wolter, editors, *FroCoS 2007*, volume 4720 of *LNCS*, pages 148–162. Springer, 2007.

12 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.

13 Nachum Dershowitz and Georg Moser. The Hydra Battle revisited. In Hubert Comon-Lundh, Claude Kirchner, and Hélène Kirchner, editors, *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *LNCS*, pages 1–27. Springer, 2007.

14 R. L. Goodstein. On the restricted ordinal theorem. *J. Symb. Logic*, 9(2):33–41, 1944.

15 José Grimm. Implementation of three types of ordinals in Coq. Technical Report RR-8407, Inria, 2013. URL: `https://hal.inria.fr/hal-00911710/document`.

16 Gerard Huet and Derek C. Oppen. Equations and rewrite rules: A survey. In Ronald V. Book, editor, *Formal Language Theory: Perspectives and Open Problems*, pages 349–405. Academic Press, 1980.

17 Brian Huffman and Ondřej Kunčar. Lifting and Transfer: A modular design for quotients in Isabelle/HOL. In Georges Gonthier and Michael Norrish, editors, *CPP 2013*, volume 8307 of *LNCS*, pages 131–146. Springer, 2013.

18 Cezary Kaliszyk and Christian Urban. Quotients revisited for Isabelle/HOL. In William C. Chu, W. Eric Wong, Mathew J. Palakal, and Chih-Cheng Hung, editors, *SAC 2011*, pages 1639–1644. ACM, 2011.

19 Laurie Kirby and Jeff Paris. Accessible independence results for Peano arithmetic. *Bull. London Math. Soc.*, 4:285–293, 1982.

20 Alexander Krauss. Partial recursive functions in higher-order logic. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR 2006*, volume 4130 of *LNCS*, pages 589–603. Springer, 2006.

21 Alexander Krauss. Certified size-change termination. In Frank Pfenning, editor, *CADE-21*, volume 4603 of *LNCS*, pages 460–475. Springer, 2007.

22 Ralph Loader. Unary PCF is decidable. *Theor. Comput. Sci.*, 206(1-2):317–329, 1998.

23 Michel Ludwig and Uwe Waldmann. An extension of the Knuth-Bendix ordering with LPO-like properties. In Nachum Dershowitz and Andrei Voronkov, editors, *LPAR 2007*, volume 4790 of *LNCS*, pages 348–362. Springer, 2007.

24 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

25 Michael Norrish and Brian Huffman. Ordinals in HOL: Transfinite arithmetic up to (and beyond) ω_1. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP 2013*, volume 7998 of *LNCS*, pages 133–146. Springer, 2013.

26 Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *IWIL-2010*, volume 2 of *EPiC*, pages 1–11. EasyChair, 2012.

27 Gordon D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):223–255, 1977.

28 Manfred Schmidt-Schauß. Decidability of behavioural equivalence in unary PCF. *Theor. Comput. Sci.*, 216(1-2):363–373, 1999.

29 Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 2nd edition, 1987.

30 René Thiemann, Guillaume Allais, and Julian Nagele. On the formalization of termination techniques based on multiset orderings. In Ashish Tiwari, editor, *RTA 2012*, volume 15 of *LIPIcs*, pages 339–354. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2012.

31 Dmitriy Traytel, Andrei Popescu, and Jasmin Christian Blanchette. Foundational, compositional (co)datatypes for higher-order logic: Category theory applied to theorem proving. In *LICS 2012*, pages 596–605. IEEE Computer Society, 2012.

32 Martijn Vermaat. *Infinitary Rewriting in Coq*. M.Sc. thesis, Vrije Universiteit Amsterdam, 2010. URL: `http://www.cs.vu.nl/~tcs/mt/vermaat.pdf`.

33 Makarius Wenzel. Isabelle/Isar—A generic framework for human-readable proof documents. In Roman Matuszewski and Anna Zalewska, editors, *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*, volume 10(23) of *Studies in Logic, Grammar, and Rhetoric*. Uniwersytet w Białymstoku, 2007.

34 Sarah Winkler, Harald Zankl, and Aart Middeldorp. Beyond Peano arithmetic—Automatically proving termination of the Goodstein sequence. In Femke van Raamsdonk, editor, *RTA 2013*, volume 21 of *LIPIcs*, pages 335–351. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2013.