Partizan Games in Isabelle/HOLZF

Steven Obua*

Technische Universität München
D-85748 Garching, Boltzmannstr. 3, Germany
e-mail: obua@in.tum.de, url: http://www4.in.tum.de/~obua

Abstract. *Partizan Games* (PGs) have been invented by John H. Conway and are described in his book *On Numbers and Games*. We formalize PGs in Higher Order Logic extended with ZF axioms (HOLZF) using Isabelle, a mechanical proof assistant. We show that PGs can be defined as the unique fixpoint of a function that arises naturally from Conway's original definition. While the construction of PGs in HOLZF relies heavily on the ZF axioms, operations on PGs are defined on a game type that hides its set theoretic origins. A polymorphic type of sets that are not bigger than ZF sets facilitates this. We formalize the induction principle that Conway uses throughout his proofs about games, and prove its correctness. For these purposes we examine how the notions of well-foundedness in HOL and ZF are related in HOLZF. Finally, games (modulo equality) are added to Isabelle's numeric types by showing that they are an instance of the axiomatic type class of partially ordered abelian groups.

1 Introduction

Partizan Games are extensively and beautifully described in Conway's book ONAG [1]. In this paper, we will instead focus on the issues that arise when representing and reasoning about PGs in the mechanical theorem proving assistant Isabelle[1]. For PGs we improve on the methods of Mamane, who has formalized PGs and surreal numbers, which are a special kind of PGs, in Coq [2]. Especially, our constructions and proofs are in direct relation to the ones found in ONAG and therefore short. The proofs in [2] often deviate from the original proofs and are (much) longer. This difference has two major reasons:

1. We use a logic / axiom system, *HOLZF*, that is very suitable to formalize set-theoretic notions but still offers the advantages of Higher Order Logic, so that we can define our own type of games. Mamane uses the Calculus of Inductive Constructions (CIC).
2. We have formalized the induction principle that Conway uses, and proved its correctness once and for all. Actual inductions in the proofs about games are simple instantiations of this general induction principle.

* Supported by the Ph.D. program "Logik in der Informatik" of the "Deutsche Forschungsgemeinschaft."

[1] the theory files can be downloaded from [14]

> Mamane has identified this induction principle and calls it *permuting induction* (and so will we) but has neither proven nor formalized it [2]. We show that also Conway's transitivity proof is perfectly correct according to this principle, contrary to what Mamane [2] states. On the other hand we have discovered a true flaw in one of Conway's proofs.

This paper can be seen from two points of view. One is that this paper is about formalizing PGs, and in order to achieve this, we will introduce Isabelle/HOLZF and some properties of it. The other point of view is that it really is about Isabelle/HOLZF, a logic that extends both Higher Order Logic (HOL) and Zermelo-Fränkel set theory (ZF), and about the first application of it, namely formalizing games, that neither Isabelle/HOL nor Isabelle/ZF is suitable for.

2 Which Axiom System Suits Partizan Games?

Conway defines Partizan Games inductively:

> If L and R are any two sets of games, there is a game $G = \{L \mid R\}$. All games are constructed this way. The *left options* of G are the elements of L, and R is the set of *right options* of G. Two games G and H are called *identical* if they have the same left and right options.

Underlying this definition is the idea that every game has certain positions and is played by two players, Left and Right. Every position is characterized by the moves that either of both players could make if it was his turn. Each such move leads to a new position. In a given position G a move is identified with the position it leads to. The moves of Left are called the left options of G, the moves of Right are called the right options of G. Identifying a game with its starting position we arrive at the above definition of games.

There is a restriction on PGs that Conway demands: There is no infinite sequence of games $(G_i)_{i\in\mathbb{N}}$ such that G_{i+1} is an option (left or right) of G_i. This restriction is contained in the definition of PGs if *set* is understood in the sense of Zermelo-Fränkel set theory.

Isabelle is a generic theorem prover [8]. Its meta logic is intuitionistic higher order logic, on top of which other logics can be built by asserting axioms and declaring types. Two such extensions of pure Isabelle are in more widespread use and we argue that neither of them is suitable for mechanizing Partizan Games:

Isabelle/HOL [8] is the Isabelle implementation of Higher Order Logic. It features packages for defining datatypes and general recursive functions. We might be tempted to use the datatype package to define the *game* type:

$$\textbf{datatype}\ \text{game} = \text{Game (game set) (game set)}. \tag{1}$$

> If admissible, this statement would define a new type *game* that has one constructor *Game* that takes as arguments a *game set*, the left options of the game, and a further *game set*, the right options. But this statement is not

admissible because *Game L* needs to be an injective function for any fixed L but cannot be; as a consequence of the Schroeder-Bernstein theorem one can prove in HOL that no function of type $'a\ set \Rightarrow 'a$ is possibly injective.

Isabelle/ZF [6,7] is based on classical first-order logic and the axioms of ZF set theory. Here we might try to use the provided facilities to define fixpoints; let us define a function h by

$$h\ A = \{(L,\ R) \mid L \subseteq A \wedge R \subseteq A\}. \tag{2}$$

Any set G that fulfills the fixpoint equation $h\ G = G$ would be a good candidate for a set of Partizan Games. But obviously we have $|h\ A| > |A|$ for all sets A and therefore there is no such candidate. Therefore, Partizan Games do not form a set but rather a proper class.

In Isabelle/HOLZF we can solve all these problems; the next section introduces and describes Isabelle/HOLZF.

3 HOLZF = HOL + ZF

We obtain Isabelle/HOLZF by starting from Isabelle/HOL and introducing a new type *ZF* and a relation *Elem* of type $ZF \Rightarrow ZF \Rightarrow bool$ on it; we then make this type into the universe of ZF sets by postulating the ZF axioms.

That something like Isabelle/HOLZF could be possible was suspected by Tjark Weber and the author when they tried to formalize the semantics of the λ-calculus in Isabelle/HOL (and failed). The actual viability and how-to of the approach was brought to the attention of the author by Bob Solovay who outlined HOLZF on the Isabelle mailing list and claimed that "for certain reasons he needed such a monster", opposing Larry Paulson's remark that HOLZF might be "too much of a good thing". Bob Solovay also provided a proof of the consistency of HOLZF relative to the consistency of ZFC + 'there is an inaccessible cardinal'. Mike Gordon has worked on HOLZF already ten years ago [3]. He uses the name HOL-ST instead of HOLZF. Also, Sten Agerholm has used HOL-ST to formalize the inverse limit construction of domain theory to build a model of the λ-calculus [4].

We use the same axioms as Gordon in [3] with one exception; his axiom of separation is superfluous if one is willing to apply the axiom of choice which HOL provides via the Hilbert choice operator.

We first declare the new type *ZF* and introduce six new constants denoting the empty set (*Empty*), the element relation (*Elem*), the sum or union operator (*Sum*), the power set operator (*Power*), the replacement operator (*Repl*) and an infinite set (*Inf*):

typedecl *ZF*

consts
Empty :: *ZF*
Elem :: $ZF \Rightarrow ZF \Rightarrow bool$

$Sum :: ZF \Rightarrow ZF$
$Power :: ZF \Rightarrow ZF$
$Repl :: ZF \Rightarrow (ZF \Rightarrow ZF) \Rightarrow ZF$
$Inf :: ZF$

Standard constructions like unordered pairs (*Upair*), the singleton set (*Singleton*) and the union of two sets (*union*) are defined in terms of the new constants. We also need the function *SucNat* which just encodes the successor of a natural number as a set in the standard way.

Our seven axioms can now be expressed as follows:

axioms
$Empty$: $\neg\ (Elem\ x\ Empty)$
Ext: $(x = y) = (\forall\, z.\ Elem\ z\ x = Elem\ z\ y)$
Sum: $Elem\ z\ (Sum\ x) = (\exists\, y.\ Elem\ z\ y \wedge Elem\ y\ x)$
$Power$: $Elem\ y\ (Power\ x) = (\forall\, z.\ Elem\ z\ y \longrightarrow Elem\ z\ x)$
$Repl$: $Elem\ b\ (Repl\ A\ f) = (\exists\, a.\ Elem\ a\ A \wedge b = f\ a)$
$Regularity$: $A \neq Empty \longrightarrow (\exists\, x.\ Elem\ x\ A \wedge (\forall\, y.\ Elem\ y\ x \longrightarrow \neg\ (Elem\ y\ A)))$
$Infinity$: $Elem\ Empty\ Inf \wedge (\forall\, x.\ Elem\ x\ Inf \longrightarrow Elem\ (SucNat\ x)\ Inf)$

As mentioned, separation (*Sep*) can be defined in terms of replacement and does not need an extra axiom.

constdefs
$Sep :: ZF \Rightarrow (ZF \Rightarrow bool) \Rightarrow ZF$
$Sep\ A\ p \equiv (if\ (\forall\, x.\ Elem\ x\ A \longrightarrow \neg\ (p\ x))\ then\ Empty\ else$
$(let\ z = (\epsilon\, x.\ Elem\ x\ A \wedge p\ x)\ in$
$let\ f = \lambda\ x.\ (if\ p\ x\ then\ x\ else\ z)\ in\ Repl\ A\ f))$

We also define ordered pairs (*Opair*):

constdefs
$Opair :: ZF \Rightarrow ZF \Rightarrow ZF$
$Opair\ a\ b \equiv Upair\ (Upair\ a\ a)\ (Upair\ a\ b)$
$Fst :: ZF \Rightarrow ZF$
$Fst\ q \equiv \epsilon\, x.\ \exists\, y.\ q = Opair\ x\ y$
$Snd :: ZF \Rightarrow ZF$
$Snd\ q \equiv \epsilon\, y.\ \exists\, x.\ q = Opair\ x\ y$

The reasoning about these constants is easy if one can prove an equation for the constant that is characteristic for it and has the form of a simple logical equivalence so that the constant does not appear any more on the right hand side. In that way the actual work can be delegated to the classical reasoner and to the simplifier of Isabelle/HOL. The characteristic equations for *Upair*, *Singleton*, *union*, *Sep*, *Opair*, *Fst* and *Snd* are:

$$\begin{array}{lcl}
Elem\ x\ (Upair\ a\ b) & \equiv & x = a \vee x = b, \\
Elem\ x\ (Singleton\ y) & \equiv & x = y, \\
Elem\ x\ (union\ A\ B) & \equiv & Elem\ x\ A \vee Elem\ x\ B, \\
Elem\ b\ (Sep\ A\ p) & \equiv & Elem\ b\ A \wedge p\ b, \\
Opair\ a\ b = Opair\ c\ d & \equiv & a = c \wedge b = d, \\
Fst\ (Opair\ x\ y) & \equiv & x, \\
Snd\ (Opair\ x\ y) & \equiv & y.
\end{array}$$

The development of set theory proceeds along the same lines as in Isabelle/ZF [6]. We have not developed the whole theory but at least the pieces that are necessary to later construct Partizan Games. Paulson goes on building up an infrastructure for inductive definition and recursion within set theory [7]. This machinery is not sufficient in order to deal with Partizan Games as we have seen before. But this is no problem anyway because we can rely on the machinery of fixpoints, primitive and well-founded recursion that already exists in HOL instead of replicating the mechanisms of ZF! Before delving into that let us first construct Partizan Games in HOLZF.

4 Constructing Partizan Games

Partizan Games do not form a ZF set but, if they exist at all, only a proper class as we have motivated earlier. Intuitively, a proper class is a collection of elements that is too 'big' to form a ZF set but which is rather defined by a property which all elements of the class, and only those, share. In HOL we have the polymorphic type of sets at our disposal which is just an abbreviation for a predicate / property:

$$\alpha\ set = \alpha \Rightarrow bool. \tag{3}$$

Therefore we call an object of type *ZF set* a *class*. When we discuss well-foundedness we will see that this is the right intuition. An object of type *ZF* is referred to as *set*. If we are talking about an object of type α *set* and we do not have necessarily $\alpha = ZF$ then we use the expression *HOL set*.

For each set there is a corresponding class which we obtain by applying the *explode* function to the set.

constdefs
explode :: *ZF* ⇒ *ZF set*
explode z ≡ { *x*. *Elem x z* }
implode :: *ZF set* ⇒ *ZF*
implode ≡ *inv explode*

Obviously we have the characteristic equations

$$(x \in explode\ X) = Elem\ x\ X, \quad implode\ (explode\ x) = x. \tag{4}$$

The empty set corresponds to the empty class $\emptyset$:

$$explode\ Empty = \emptyset. \tag{5}$$

What about the universal class *UNIV*? Russell's Paradox allows us to prove that there is no set which corresponds to the universal class:

$$explode\ z \neq UNIV. \tag{6}$$

Classes without a corresponding set are called *proper*.

We are now able to fix (2) by defining the function whose fixpoint we are interested in not on sets but on classes:

constdefs
fixgames :: *ZF set* ⇒ *ZF set*
fixgames A ≡ { *Opair l r* | *l r*. *explode l* ⊆ *A* ∧ *explode r* ⊆ *A*}

It is easy to see that *fixgames* is a monotone function,

$$mono\ fixgames, \tag{7}$$

where *mono* is defined by

$$mono\ f \equiv \forall A\ B.\ A \leq B \longrightarrow f\ A \leq f\ B. \tag{8}$$

Note that for HOL sets $\leq$ is just $\subseteq$. The Knaster-Tarski theorem is already available in Isabelle/HOL for the complete lattice of HOL sets:

$$mono\ f \Longrightarrow lfp\ f = f\ (lfp\ f), \quad mono\ f \Longrightarrow gfp\ f = f\ (gfp\ f), \tag{9}$$

where *lfp* is the least and *gfp* the greatest fixpoint operator. Therefore we know that *fixgames* has a fixpoint. Every such fixpoint would be acceptable as a sensible definition for Partizan Games so we pick the least and the greatest.

constdefs
games-lfp :: *ZF set*
games-lfp ≡ *lfp fixgames*
games-gfp :: *ZF set*
games-gfp ≡ *gfp fixgames*

From (9) we deduce

$$games\text{-}lfp = fixgames\ games\text{-}lfp, \quad games\text{-}gfp = fixgames\ games\text{-}gfp. \tag{10}$$

Every fixpoint G of *fixgames* is bounded by *games-lfp* and *games-gfp*:

$$G = fixgames\ G \Longrightarrow games\text{-}lfp \subseteq G \wedge G \subseteq games\text{-}gfp. \tag{11}$$

So it seems that we can choose among several, maybe infinitely many definitions for Partizan Games! Fortunately, there is only one fixpoint of *fixgames*:

$$games\text{-}lfp = games\text{-}gfp. \tag{12}$$

To see why, we first define the options of a game.

constdefs
left-option :: *ZF* ⇒ *ZF* ⇒ *bool*
left-option g opt ≡ (*Elem opt* (*Fst g*))
right-option :: *ZF* ⇒ *ZF* ⇒ *bool*
right-option g opt ≡ (*Elem opt* (*Snd g*))
is-option-of :: (*ZF* × *ZF*) *set*
is-option-of ≡ {(*opt*, *g*) . *g* ∈ *games-gfp* ∧ (*left-option g opt* ∨ *right-option g opt*)}

We prove

$$g \in games\text{-}gfp \longrightarrow g \in games\text{-}lfp \tag{13}$$

by induction over *g*.

Proof. Let us assume that (13) holds for all options of g, that is for all *opt* such that $(opt,\ g) \in$ *is-option-of*. Let us further assume $g \in$ *games-gfp*. Because of *games-gfp* = *fixgames games-gfp* there are L and R such that

$$g = Opair\ L\ R, \quad explode\ L \subseteq games\text{-}gfp, \quad explode\ R \subseteq games\text{-}gfp. \tag{14}$$

All elements of *explode* L and *explode* R are options of g. Applying the induction hypothesis yields *explode* $L \subseteq$ *games-lfp* and *explode* $R \subseteq$ *games-lfp*. Because of *fixgames games-lfp* = *games-lfp* we deduce $g \in$ *games-lfp*. □

Therefore (12) holds.

The alert reader might not be entirely convinced. And rightly so! Above proof only holds up if *is-option-of* is a well-founded relation. So it is time now to turn to well-founded relations in HOLZF.

5 Well-foundedness and Induction in HOLZF

In Isabelle/HOL a relation is called *well-founded* (*wf*) if it comes with an induction principle:

$$wf\ r \equiv \forall P.\ (\forall x.\ (\forall y.\ (y,\ x) \in r \longrightarrow P\ y) \longrightarrow P\ x) \longrightarrow (\forall x.\ P\ x). \tag{15}$$

The predicate *wf* has type $(\alpha \times \alpha)\ set \Rightarrow bool$. Equivalent to (15) is that every non-empty HOL set has a minimal element with respect to the relation:

$$wf\ r = (\forall Q\ x.\ x \in Q \longrightarrow (\exists z{\in}Q.\ \forall y.\ (y,\ z) \in r \longrightarrow y \notin Q)). \tag{16}$$

In order to complete our above proof that *fixgames* has a unique fixpoint we have to show:

$$wf\ is\text{-}option\text{-}of. \tag{17}$$

But *is-option-of* is already a fairly complicated relation; we have to consider both left and right options. Looking around we discover a more basic relation:

constdefs
is-Elem-of :: $(ZF \times ZF)\ set$
is-Elem-of $\equiv \{(a,b)\ .\ Elem\ a\ b\}$

Because of the way we constructed ordered pairs we can derive

$$\exists z.\ Elem\ x\ z \wedge Elem\ y\ z \wedge Elem\ z\ (Opair\ x\ y). \tag{18}$$

Now assume $(opt,\ g) \in$ *is-option-of*. This implies $g \in$ *games-gfp* = *fixgames games-gfp* and therefore we know that there exist sets L and R such that $g =$ *Opair* L R and *Elem opt* $L \vee$ *Elem opt* R. Together with (18) follows

$$(opt,\ g) \in is\text{-}option\text{-}of \Longrightarrow \exists u\ v.\ Elem\ opt\ u \wedge Elem\ u\ v \wedge Elem\ v\ g, \tag{19}$$

and further

$$is\text{-}option\text{-}of \subseteq is\text{-}Elem\text{-}of^{+}. \tag{20}$$

Here r^+ denotes the transitive closure of the relation r which is of course well-founded if r is. Therefore we are left with the proof obligation

$$wf\ is\text{-}Elem\text{-}of. \tag{21}$$

It is not exaggerated to call (21) the main theorem of well-foundedness in HOLZF. Despite that, to the author's knowledge it has neither been considered nor proven in previous work on HOLZF/HOL-ST.

Of course the *Elem* relation is well-founded on every *set P*; for every nonempty subset K of P there is a such that $Elem\ a\ K$ and $\forall\, x.\ Elem\ x\ a \longrightarrow \neg\ Elem\ x\ K$ hold. This is a direct consequence of the axiom of regularity. Considering (16), what (21) says is that *Elem* is also well-founded on every *class* P!

Once this is understood, standard literature on set theory tells us how to proceed [10, ch. 6]. We introduce the notion of ZF-well-foundedness (*wfzf*):

constdefs
$Ext :: ('\alpha \times '\beta)\ set \Rightarrow '\beta \Rightarrow '\alpha\ set$
$Ext\ R\ y \equiv \{x\ .\ (x,\ y) \in R\}$

$regular :: (ZF \times ZF)\ set \Rightarrow bool$
$regular\ R \equiv\ \forall\, A.\ A \neq Empty \longrightarrow$
$\quad (\exists\, x.\ Elem\ x\ A \wedge (\forall\, y.\ (y,\ x) \in R \longrightarrow \neg\ (Elem\ y\ A)))$

$implodeable\text{-}Ext :: (ZF \times ZF)\ set \Rightarrow bool$
$implodeable\text{-}Ext\ R \equiv \forall\, y.\ \exists\, z.\ Ext\ R\ y = explode\ z$

$wfzf :: (ZF \times ZF)\ set \Rightarrow bool$
$wfzf\ R \equiv regular\ R \wedge implodeable\text{-}Ext\ R$

A ZF-well-founded relation is therefore a relation R that is

1. *regular*, that is it is well-founded on every set, and for which
2. for any x the class $Ext\ R\ x$ of all predecessors of x actually is a set.

Because of the axiom of regularity we have *regular is-Elem-of*. Obviously

$$Ext\ is\text{-}Elem\text{-}of\ =\ explode \tag{22}$$

holds, therefore we also know *implodeable-Ext is-Elem-of*. Finally we deduce

$$wfzf\ is\text{-}Elem\text{-}of. \tag{23}$$

Thus the main theorem (21) can be reformulated as

$$wfzf\ R \Longrightarrow wf\ R. \tag{24}$$

The proof of (24) is quite involved, at least if one needs to build all the tools from scratch. We refer the reader who is interested in the details to the set theory literature [10, ch. 6] or the Isabelle theory files themselves [14].

6 The Type of Games

So there is a unique class of Partizan Games. If we carefully listen to Conway's 'cry for a Mathematician's liberation movement', also known as Appendix to Part Zero of ONAG, we might hear that it is desirable to package this class as a type so that we can forget about its set-theoretic origin. Software engineers call this approach data abstraction.

Defining the type is easy enough:

typedef *game* = *games-lfp*

The next item on our wish list is to have a function *Game* that takes the left and right options of a game as arguments and constructs a game out of them. But what type should the left and right options have, maybe *game set*? We have seen earlier that this does not work; we would want *Game* to be injective, but there is no injective function of type

$$Game :: (game\ set) \Rightarrow (game\ set) \Rightarrow game. \tag{25}$$

A solution to this problem would be not only to introduce the type of games, but also the type *gameset* of sets of games.

But this would entail the dreary definition of element relation, union operator and so on just for this one type.

There is a better solution; we still introduce a new type and we still have to define a third suit of operators on 'sets', but we do it in a general way. This new type is a natural addition to the types already available in HOL which is definable only in HOLZF:

typedef $'\alpha\ zet = \{A :: '\alpha\ set\ .\ \exists f\ z.\ inj\text{-}on\ f\ A \wedge f\ `\ A \subseteq explode\ z\}$

We define a polymorphic type *zet* of 'sets' that are 'not bigger' than some set of type *ZF*. We need a new name for them in order to not confuse them with our other notions class, set and HOL set. We will call them zets. A zet of type $\alpha\ zet$ corresponds to an HOL set A of type $\alpha\ set$ such that there is a set z and a mapping f from α to ZF such that the image of A under f is contained in the class that corresponds to z. We ensure that f preserves the size of A by requiring f to be injective on A.

We then define operators on zets that mimic those available on sets. We will not bother the reader with details here; we just state the names of the functions that are used frequently and also their characteristic property:

name :: type	**characteristic property**
$zin :: \alpha \Rightarrow \alpha\ zet \Rightarrow bool$	
$zempty :: \alpha\ zet$	$\neg\ zin\ x\ zempty$
$zimage$ $:: (\alpha \Rightarrow \beta) \Rightarrow \alpha\ zet \Rightarrow \beta\ zet$	$zin\ y\ (zimage\ f\ A) = (\exists x.\ zin\ x\ A \wedge y = f\ x)$
$zunion :: \alpha\ zet \Rightarrow \alpha\ zet \Rightarrow \alpha\ zet$	$zin\ x\ (zunion\ a\ b) = (zin\ x\ a \vee zin\ x\ b)$

Now we are equipped with the tools to introduce the left and right options of a game; we also introduce the *Game* constructor we wanted all along.

consts
left-options :: *game* ⇒ *game zet*
right-options :: *game* ⇒ *game zet*
options :: *game* ⇒ *game zet*
option-of :: (*game* × *game*) *set*
Game :: *game zet* ⇒ *game zet* ⇒ *game*

Again, we do not give definitions here; the curious reader is referred to the Isabelle theory file. All that matters are the characteristic properties which have been proven from the definitions:

$$(Game\ L1\ R1 = Game\ L2\ R2) = (L1 = L2 \wedge R1 = R2), \tag{26}$$

$$g = Game\ (left\text{-}options\ g)\ (right\text{-}options\ g), \tag{27}$$

$$zin\ opt\ (left\text{-}options\ g) \Longrightarrow zin\ opt\ (options\ g), \tag{28}$$

$$zin\ opt\ (right\text{-}options\ g) \Longrightarrow zin\ opt\ (options\ g), \tag{29}$$

$$((opt,\ g) \in option\text{-}of) = zin\ opt\ (options\ g). \tag{30}$$

The construction of our *game* type is completed by proving

$$wf\ option\text{-}of. \tag{31}$$

This is an immediate consequence of the well-foundedness of *is-option-of* and gives us an induction principle for *game*s.

7 The Partially Ordered Group Pg

In this section we formalize comparison, equality, addition and negation of games and show that they form a partially ordered group when considered modulo equality. We will not give any intuition behind these operations; ONAG provides plenty of intuition.

Easiest to define is the negation of games:

consts
neg-game :: *game* ⇒ *game*
recdef *neg-game option-of*
neg-game g = *Game* (*zimage neg-game* (*right-options g*))
(*zimage neg-game* (*left-options g*))

The above statements define *neg-game* via well-founded recursion over *option-of*.

lemma *neg-game* (*neg-game g*) = *g*
apply (*induct g rule*: *neg-game.induct*)
...

The short proof (8 lines) of above lemma is by induction over g using the automatically generated and pre-proven induction rule *neg-game.induct*:

$$\begin{aligned}(\bigwedge g.\ & (\forall x.\ zin\ x\ (left\text{-}options\ g) \longrightarrow P\ x) \\ & \Longrightarrow (\forall x.\ zin\ x\ (right\text{-}options\ g) \longrightarrow P\ x) \\ & \Longrightarrow P\ g) \Longrightarrow P\ x\end{aligned} \tag{32}$$

Next comes comparison of games:

consts
  ge-game :: (*game* × *game*) ⇒ *bool*
recdef *ge-game* (*gprod-2-1 option-of*)
  ge-game (*G*, *H*) =
    (∀ *x*. *if zin x* (*right-options G*) *then* (
        if zin x (*left-options H*) *then* ¬ (*ge-game* (*H*, *x*) ∨ (*ge-game* (*x*, *G*)))
                                else ¬ (*ge-game* (*H*, *x*)))
        else (*if zin x* (*left-options H*) *then* ¬ (*ge-game* (*x*, *G*)) *else True*))

The above definition uses the *if*-operator to give recdef the necessary hints for proving termination. A better definition can easily be derived:

$$\textit{ge-game}\ (G,\ H) = (\forall\, x.\ (\textit{zin}\ x\ (\textit{right-options}\ G) \longrightarrow \neg\ \textit{ge-game}\ (H,\ x)) \\ \wedge\ (\textit{zin}\ x\ (\textit{left-options}\ H) \longrightarrow \neg\ \textit{ge-game}\ (x,\ G))).$$

Because *ge-game* is essentially a function of two arguments which swaps the order of its arguments when calling itself recursively it is important to provide the right termination relation, *gprod-2-1 option-of*, where

$$\textit{gprod-2-1}\ R \equiv \{((a,\ b),\ (c,\ d)) \mid a = d \wedge (b,\ c) \in R \vee b = c \wedge (a,\ d) \in R.$$

It seems clear that *wf* (*gprod-2-1 R*) should follow from *wf R*; we need to prove this, otherwise recdef will reject above definition of *ge-game*. Actually, *gprod-2-1* is only a special case of a more general well-founded relation that crops up in most definitions and proofs dealing with games. Mamane calls the induction principle that this relation induces *permuting induction* [2, p. 41, p. 95]. He was not able to formalize the principle in CIC but only instances of it like *gprod-2-1*. We have a problem at this point, too; our general relation should not only deal with pairs of games, but with n-tuples of games for arbitrary n. So for $n = 2$ our relation should have type

$$((\textit{game} \times \textit{game}) \times (\textit{game} \times \textit{game})\ \textit{set} \tag{33}$$

but for $n = 3$ it must have type

$$((\textit{game} \times \textit{game} \times \textit{game}) \times (\textit{game} \times \textit{game} \times \textit{game})\ \textit{set} \tag{34}$$

and so on. We have no dependent types available in HOL, but there is a solution; we define our relation not on tuples, but inductively on lists.

consts
  lprod :: (′α × ′α) *set* ⇒ (′α *list* × ′α *list*) *set*
inductive *lprod R*
intros
  (*a*, *b*) ∈ *R* ⟹ ([*a*], [*b*]) ∈ *lprod R*
  (*ah*@*at*, *bh*@*bt*) ∈ *lprod R* ⟹ (*a*,*b*) ∈ *R* ∨ *a* = *b*
    ⟹ (*ah*@*a*#*at*, *bh*@*b*#*bt*) ∈ *lprod R*

Here $xs@ys$ denotes the concatenation of two lists xs and ys, $x\#xs$ denotes the consing of x to the list xs; *lprod R* is really a generalized version of *gprod-2-1 R*:

$$gprod\text{-}2\text{-}1\ R \subseteq inv\text{-}image\ (lprod\ R)\ (\lambda(a,\ b).\ [a,\ b]). \tag{35}$$

The inverse image *inv-image R f* of a relation R under a map f is well-founded if R is. Therefore all we have to show is the well-foundedness of *lprod R* which then proves the well-foundedness of *gprod-2-1 R* and similar relations. Using induction one shows

$$lprod\ R \subseteq inv\text{-}image\ (mult\ (R^+))\ multiset\text{-}of, \tag{36}$$

that is we reduce the well-foundedness of *lprod R* to the well-foundedness of the multiset order *mult* (R^+). The function *multiset-of* takes a list as its argument and returns the corresponding multiset. See [9, ch. 2.5] for more information on multisets and the multiset order. Luckily, multisets have already been formalized in Isabelle and the well-foundedness of *mult* (R^+) is available as a lemma. Therefore we show easily

$$wf\ R \Longrightarrow wf\ (lprod\ R). \tag{37}$$

When showing that *ge-game* is a partial order, one has to show transitivity:

$$ge\text{-}game\ x\ y \Longrightarrow ge\text{-}game\ y\ z \Longrightarrow ge\text{-}game\ x\ z. \tag{38}$$

The proof of (38) that Conway gives in ONAG is particularly short and elegant. Mamane gives a much longer CIC proof [2, pp. 49-53]. We have a short proof (44 lines in Isar) that is in direct correspondence to the proof of Conway. The trick is to convert a statement of the form $P\ x\ y\ z$ where in this case we set

$$P\ x\ y\ z \equiv ge\text{-}game\ (x,\ y) \wedge ge\text{-}game\ (y,\ z) \longrightarrow ge\text{-}game\ (x,\ z), \tag{39}$$

into a statement of the form $\forall x\ y\ z.\ gs = [x,\ y,\ z] \longrightarrow P\ x\ y\ z$ and prove this by well-founded induction over *gs* with respect to *lprod option-of*. This ensures that when trying to prove $P\ x\ y\ z$ we can use the induction hypothesis $P\ a\ b\ c$ for all a, b and c that fulfill

$$([a,\ b,\ c],\ [x,\ y,\ z]) \in lprod\ option\text{-}of. \tag{40}$$

Currently one has to show manually that (40) holds for particular instances, typically involving proofs of two or three lines using the introduction rules for *lprod*. Of course this could be automated.

Equality (*eq-game*) is defined in terms of *ge-game*. Addition (*plus-game*) is defined recursively; there are no technical differences to the definition of *ge-game*. We also introduce the zero game (*zero-game*).

constdefs
eq-game :: *game* ⇒ *game* ⇒ *bool*
eq-game G H ≡ *ge-game* (*G*, *H*) ∧ *ge-game* (*H*, *G*)

zero-game :: *game*
zero-game ≡ *Game zempty zempty*

consts
plus-game :: *game* × *game* ⇒ *game*
recdef *plus-game gprod-2-2 option-of*
plus-game (*G*, *H*) = *Game*
(*zunion* (*zimage* (λ *g*. *plus-game* (*g*, *H*)) (*left-options G*))
(*zimage* (λ *h*. *plus-game* (*G*, *h*)) (*left-options H*)))
(*zunion* (*zimage* (λ *g*. *plus-game* (*g*, *H*)) (*right-options G*))
(*zimage* (λ *h*. *plus-game* (*G*, *h*)) (*right-options H*)))

Most properties of addition and comparison are straightforward to prove; just copy the proofs that Conway gives and apply above reformulation technique. There is one proof though where this does not work because the proof that Conway gives is flawed. The proof is supposed to verify the theorem

$$\textit{ge-game}\ (y,\ z) = \textit{ge-game}\ (\textit{plus-game}\ (x,\ y),\ \textit{plus-game}\ (x,\ z)). \tag{41}$$

The error is on page 18 of ONAG in the proof of theorem 5. Conway claims that the truth of

$$x^R + y \le x + z\ \vee\ x + y^R \le x + z\ \vee\ x + y \le x^L + z\ \vee\ x + y \le x + z^L, \tag{42}$$

assuming furthermore $y \ge z$, implies the truth of

$$x^R + y \le x + y\ \vee\ x + y^R \le x + y\ \vee\ x + z \le x^L + z\ \vee\ x + z \le x + z^L, \tag{43}$$

obviously taking for granted

$$y \ge z \Longrightarrow x + z \le x + y. \tag{44}$$

But this is just what he is trying to prove!

We can fix this error quickly; two of the assumptions in (42) lead immediately to a contradiction by applying the induction hypothesis:

$$x + y^R \le x + z \Rightarrow y^R \le z \le y, \quad x + y \le x + z^L \Rightarrow z \le y \le z^L. \tag{45}$$

The other two assumptions yield a contradiction by first unfolding the definition of $\le$ and then applying the induction hypothesis:

$$x^R + y \le x + z \Rightarrow \neg(x^R + z \le x^R + y) \Rightarrow z \not\le y, \tag{46}$$

$$x + y \le x^L + z \Rightarrow \neg(x^L + z \le x^L + y) \Rightarrow z \not\le y. \tag{47}$$

Note that we were able to apply the induction hypothesis in several different disguises because all of $[x, z, y^R]$, $[x, z^L, y]$, $[x^R, y, z]$ and $[x^L, y, z]$ are predecessors of $[x, y, z]$ with respect to *lprod option-of*.

Does the type *game* form a group with respect to the defined operations? No, it does not! We only have the theorem

$$\textit{eq-game}\ (\textit{plus-game}\ (x,\ \textit{neg-game}\ x))\ \textit{zero-game} \tag{48}$$

not the stronger, but false statement

$$\textit{plus-game}\ (x,\ \textit{neg-game}\ x) = \textit{zero-game}. \tag{49}$$

The equality relation *eq-game* is compatible with the other operations. Furthermore *eq-game* is an equivalence relation, that is transitive, reflexive and symmetric. Therefore we can define a new type *Pg* of Partizan Games that consists of the equivalence classes of *game* with respect to *eq-game*.

typedef $Pg = UNIV\ //\ \{\ (p,\ q)\ .\ \textit{eq-game}\ p\ q\ \}$

Using the techniques described in [12] we then lift the theorems we have shown about *game*s to theorems about *Pg*s. The icing on the cake is the isabelle metatheorem

instance Pg :: pordered-ab-group-add

that states that *Pg* is an instance of the axiomatic type class of partially ordered groups *pordered-ab-group-add*.

8 Conclusion

We have presented a formalization of Conway's Partizan Games. Our work can be split into two parts.

One part consists of the development of HOLZF in Isabelle and provides infrastructure for this logic. The main result in this context is to identify a notion of well-foundedness particularly suited to the ZF part of HOLZF and to connect this notion with the common notion of well-foundedness in HOL via (24). This allows us to use all of the HOL machinery when dealing with recursion. Furthermore we argue that HOLZF is not only theoretically stronger than both ZFC and HOL but that this difference is also of practical importance, as the example of Partizan Games shows. Interesting is that we have now available a new type of 'set' called zet which might be a valuable addition to the datatype package of Isabelle/HOL. For example, it might then be possible to define Partizan Games directly by

datatype $\textit{game} = \textit{Game}\ (\textit{game zet})\ (\textit{game zet})$

Also part of the developed infrastructure is the *lprod*-relation that allows defining of and reasoning about recursive functions of several arguments of the same type.

The second part of our work can be seen as an application of Isabelle/HOLZF. Knowing *wf is-Elem-of* it was easy to show that there is a unique fixpoint of Partizan Games. We have shown that Conway's proofs withstand uttermost scrutiny with the exception of the slip in the proof of (41).

Altogether we have written about 2200 lines of theory text[2]. About 60% is infrastructure development, about 40% specific to Partizan Games. This is not too much text; actually the total time of proving that Partizan Games form a

[2] which can be downloaded from [14]

partially ordered group was not more than a couple of days after the type *game* had been constructed and an induction principle for it had been established.

Acknowledgments. Stefan Berghofer told the author why (1) cannot work and caused him to look into the multiset order. Clemens Ballarin helped to prove the properties of the *zunion* operator. Norbert Schirmer taught the author how to feed congruence rules to the recdef-package. Thanks to Bob Solovay for providing the initial idea and consistency proof of HOLZF, and to Tobias Nipkow for providing references to Mike Gordon's work.

References

1. John H. Conway. *On Numbers And Games*, 2nd ed., A K Peters Ltd., 2001.
2. Lionel E. Mamane. Surreal Numbers in Coq. *TYPES 2004*, LNCS 3839, Springer 2005, pp. 170-185.
3. Mike J.C. Gordon. Set Theory, Higher Order Logic or Both. *Theorem Proving in Higher Order Logics, 9th International Conference, TPHOLs'96*, LNCS 1125, Springer 1996, pp. 190-201.
4. Sten Agerholm. Formalising a Model of the λ-Calculus in HOL-ST. Technical Report 354, University of Cambridge Computer Laboratory, 1994.
5. Sten Agerholm, Mike J.C. Gordon. Experiments with ZF Set Theory in HOL and Isabelle. Technical Report RS-95-37, BRICS 1995.
6. Lawrence C. Paulson. Set theory for verification: I. From foundations to functions. *J. Automated Reasoning* 11 (1993), 353-389.
7. Lawrence C. Paulson. Set theory for verification: II. Induction and Recursion. *J. Automated Reasoning* 15 (1995), 167-215.
8. Tobias Nipkow, Lawrence C. Paulson, Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, Springer 2002.
9. F. Baader, T. Nipkow. *Term Rewriting and All That,* Cambridge U.P. 1998.
10. Thomas Jech. *Set Theory*, 3rd rev. ed., Springer 2003.
11. Lawrence C. Paulson. Organizing Numerical Theories Using Axiomatic Type Classes. *Journal of Automated Reasoning*, 2004, Vol. 33, No. 1, pages 29-49.
12. Lawrence C. Paulson. Defining Functions on Equivalence Classes. *ACM Transactions on Computational Logic*, in press.
13. Steven Obua. Proving Bounds for Real Linear Programs in Isabelle/HOL. *TPHOLs 2005*, LNCS 3683, Springer 2005, pp. 227-244.
14. Steven Obua. *Partizan Games in Isabelle/HOLZF.* http://www4.in.tum.de/~obua/partizan.