The Unifying Concept of Subvariance

Johan Gijsbertus Frederik Belinfante

Georgia Institute of Technology, Atlanta, GA 30332-0160 (U.S.A.)
belinfan@math.gatech.edu

Abstract. The work described here is a part of ongoing efforts to construct a set-theoretic framework that is convenient for automated reasoning in mathematics within first order logic. The specific topic in focus here is the theory of invariant and subvariant sets, which permits the development of a unified theory of regular and finite sets. Appendices are included listing theorems involving the axiom of regularity, the classes \textsc{Regular} and \textsc{Finite} of regular sets and finite sets, respectively, as well as general theorems about invariant and subvariant subsets, all proved using McCune's automated reasoning program \textit{Otter}.

1 Introduction

Computer-assisted proofs of elementary theorems of ordinal number theory have been obtained (Belinfante 1999a and 1999b) recently, using McCune's automated reasoning program \textit{Otter}. The axioms used in this work are a minor modification of ones proposed by Art Quaife (1992a and 1992b), which in turn are based on older work by Robert Boyer, et al. (1986) and ultimately on Kurt Gödel's (1940) reformulation of the von Neumann-Bernays theory (Bernays 1991) of classes and sets. To provide an extra challenge, Ishell's (1960) definition of ordinal numbers was used, so that the axiom of regularity does not need to be assumed. We present here an update, as well as describing unanticipated outgrowths of that work.

A prominent feature of Gödel's formalism is the absence of the usual class formation \{x | p(x)\} axiom schema. In its place are individual axioms for nine basic class constructors. Definitions of classes must be expressed directly or indirectly in terms of these primitives, which include two basic classes, the universal class \textit{V} and the membership relation \textit{E}, four unary class constructors \textsc{complement}, \textsc{domain}, \textsc{flip} and \textsc{rotate}, and three binary constructors \textsc{pairset}, \textsc{cart}, and \textsc{intersection}. Kurt Gödel (1940) proved a fundamental Class Existence Metatheorem Schema for class formation, whose proof amounts to a recursive algorithm for converting customary definitions of classes using ordinary class formation to expressions built out of the primitive constructors, together with a proof of termination.

Gödel's algorithm was implemented in Mathematica\textsuperscript{™} (Belinfante 1996 and 2000) to help prepare input files for proofs in set theory using McCune's automated reasoning program \textit{Otter}. To avoid complicated output, the algorithm was modified to avoid use of Kuratowski's definition for ordered pairs. Gödel's ban on self-membership was also removed since the work on ordinal numbers did not assume the axiom of regularity. Because the likelihood of success in proving theorems using programs like \textit{Otter} depends critically on the simplicity of the definitions used and the brevity of the statements of the theorems to be proved, over three thousand simplification rules have been added to produce simple output.
2 Comparing Two Definitions of Ordinal Numbers

A class $x$ is said to be full (or transitive) if every member of $x$ is a subset of $x$. This condition can be restated succinctly in several equivalent ways, for example as $\text{subclass}(U(x),x)$ or as $\text{subclass}(x,P(x))$. Here $U(x)$ is the sum class of $x$, that is, the union of all the members of $x$, and $P(x)$ is the power class of $x$, the class of all subsets of $x$. Isbell defined an ordinal number as a set with the property that any full proper subset of $x$ is a member of $x$.

The quantifiers in Isbell’s definition can be eliminated by introducing the class $\text{FULL}$ of all full sets. Applying Gödel’s algorithm to this class identifies it as the complement of the range of the intersection of the membership relation $E$ and the complement of the subset relation $S$:

$$\text{equal}(\text{complement}(\text{range}(E,\text{complement}(S))),\text{FULL}).$$

The definition of an ordinal number can be reformulated using this class as follows:

$$\text{subclass}(\text{intersection}(\text{FULL},P(x)),\text{succ}(x)).$$

Here $\text{succ}(x)$ is the successor of $x$, defined as the union of $x$ and the singleton of $x$. The class $\Omega$ of all ordinal numbers (according to Isbell’s definition) can also be expressed in terms of $\text{FULL}$ by a simple formula:

$$\text{equal}(\text{complement}(\text{range}(\text{intersection}(\text{complement}(\text{Id}),\text{intersection}(\text{S},\text{complement}(E))),\text{FULL})),\Omega).$$

Here $\text{image}(x,y)$ is the class of all sets $y$ for which there is a member $u$ of $y$ such that $\text{pair}(u,v)$ belongs to $x$. (See Quaife 1992b.)

Historically, von Neumann introduced the axiom of regularity to simplify ordinal number theory. When the axiom of regularity is assumed, one can characterize an ordinal number as a full set whose members are full (Monk 1969). The class of such sets is $\text{intersection}(\text{FULL},P(\text{FULL}))$. The proof that this class contains Isbell’s class $\Omega$ does not use the axiom of regularity. The proof, using Otter, that the two classes are equal in the presence of the axiom of regularity required adding only a few new lemmas to Quaife’s RE group of theorems. (See Appendix 1.) To prove Theorem RE-ON-EQ all one needs is Lemma $\text{RE-FUL}-2$.

Appendix 1 lists clauses for theorems involving the axiom of regularity proved using Otter. These clauses include a flag $\text{AxReg}$ that is true or false depending on whether the axiom of regularity holds, thus turning the axiom of regularity into a definition of this flag. The Skolemized version of this definition is:

```
list(usable).
% axiom D: axiom of regularity
~AxReg =/\ equal(x,0) | member(regular(x),x).
% AX-RG-1
~AxReg =/\ equal(x,0) | equal(intersection(regular(x),x),0).
% AX-RG-2
AxReg =/\ equal(IRREG,0).
% AX-RG-3
AxReg =/\ member(x,IRREG) | equal(intersection(IRREG,x),0).
% AX-RG-4
end_of_list.
```

Introducing the flag $\text{AxReg}$ does affect the performance of Otter, often forcing one to assign a nonzero value to $\text{max_distinct_vars}$.

The Skolem function $\text{regular}(x)$ occurs also in Quaife’s work, but he did not deal with the possibility that the axiom of regularity might fail to hold. Reasons for considering such a possibility have been discussed by Peter Aczel (1988). If $\text{AxReg}$ is false, then there is some class, represented by the Skolem constant IRREG, that fails to satisfy the $\text{AxReg}$ axiom. Theorems about IRREG are only of interest if the
axiom of regularity does not hold. Theorem RE-AX-6D says that the class OMEGA is disjoint from the class IRREG, and Theorem RE-AX-6E says that if \( x \) is disjoint from IRREG, then so is the power class \( P(x) \). So also \( P(\Omega) \), \( P(P(\Omega)) \), and so on are also disjoint from IRREG. The upshot is that there is no obvious way to construct any elements of IRREG.

In a theory with proper classes as well as sets one can formulate weaker or stronger axioms depending on whether the quantifiers are chosen to range over sets or classes; the flag \texttt{AxReg} refers to the strong axiom of regularity. For the axiom of regularity, the weak and strong versions are in fact equivalent. (See for example, Rubin 1967.) Because of this, one can eliminate the Skolem constant \texttt{IRREG}. (See theorem \texttt{RE-REG-2} in Appendix 1.)

3 Other Formulations of the Axiom of Regularity

Gödel's algorithm can be used to eliminate quantifiers over sets; any statement with quantifiers over sets can be converted to a logically equivalent equation without quantifiers. (Belinfante 2000) What happens is that the quantifiers are neatly built into equivalent set-theoretic constructs like \texttt{domain} and \texttt{composite}. For example, the axiom of regularity is usually formulated using quantifiers as:

\[ \texttt{AxReg} \leftrightarrow (\forall x \ (\text{equal}(x,0) \lor (\exists u \ (\text{member}(u,x) \& \text{disjoint}(u,x))))). \]

Since the quantifier over \( u \) is restricted to sets, this statement can be converted into an equivalent statement without quantifiers:

\[ \texttt{AxReg} \leftrightarrow (\forall x \ (\text{equal}(0,x) \lor \neg \text{disjoint}(x,P(\text{complement}(x)))). \]

In this case the quantifier is hidden in the introduced power class functor. Replacing \( x \) by its complement, one obtains the following reformulation of the axiom of regularity:

\[ \texttt{AxReg} \leftrightarrow (\forall x \ (\text{subclass}(P(x),x) \rightarrow \text{equal}(x,V))). \]

That is, the axiom of regularity says that the universal class \( V \) is the only class containing its own power class. Both of these reformulations of the axiom of regularity have the advantage over the original one in that their classifications do not introduce the Skolem function \texttt{regular}(\( x \)). Yet another equivalent version of the axiom of regularity with this virtue is:

\[ \texttt{AxReg} \leftrightarrow (\forall x \ (\text{subclass}(x,\text{image}(E,x)) \rightarrow \text{equal}(x,0))). \]

This formulation is equivalent to the statement that the axiom of regularity forbids an infinitely descending chain of membership: \( \cdots \in x_2 \in x_1 \in x_0 \). I personally like this one best because it makes clear the intention of the axiom.

4 Proving the Consistency of the Axiom of Regularity

One of the earliest relative consistency proofs in set theory was the proof that the axiom of regularity is consistent with the other axioms of set theory. One can prove this by constructing an inner model of set theory; what is needed is a class \texttt{REGULAR} of sets closed under all the basic operations of set theory, such that the axiom of regularity holds when restricted to members of this subclass. Only the broad outlines of the construction of this model will be described here. (See Appendix 2 for a list of theorems proved about the class \texttt{REGULAR}.)

The basic idea for this construction occurs already in a paper by Mirimanoff (1917). One begins by introducing the class \texttt{DESCENDING} of all infinitely descending
sets \( x \), that is, sets with the property that every member of \( x \) in turn has a member also belonging to \( x \). Using Gödel's algorithm, one finds that this condition can be formulated without quantifiers as \( \text{subclass}(x, \text{image}(E, x)) \), and that the class of all descending sets can be defined by

\[
\text{equal}(\text{complement}(\text{fix}(\text{composite}(E, \text{DISJOINT}))), \text{DESCENDING}).
\]

Here \( \text{fix}(x) \) is the class of fixed points of \( x \), and \text{DISJOINT} is the disjointness relation, defined as

\[
\text{equal}(\text{composite}(\text{id}, \text{complement}(\text{composite}(E, \text{inverse}(E)))), \text{DISJOINT}).
\]

An ordinary or regular set is defined as a set which holds no members that belong to an infinitely descending set. For the class \text{REGULAR} of regular sets, Gödel's algorithm yields the definition

\[
\text{equal}(\text{complement}(\text{U}(\text{DESCENDING})), \text{REGULAR}).
\]

The \text{GOEDEL} program transforms the weak form of the axiom of regularity to the statement \( \text{subclass}(\text{DESCENDING}), \text{singleton}(0) \). It is easy to show that this is equivalent to \( \text{equal}(\text{REGULAR}, V) \), which asserts that all sets are regular.

The class \text{REGULAR} is a proper class with many remarkable properties such as being its own power class. Proofs of 75 theorems about \text{REGULAR} listed in Appendix 2 were obtained using \text{Otter}. Theorem \text{RE-REG-1} says that the strong form of the axiom of regularity implies the weak form. Theorem \text{RE-REG-2} is a step toward the converse. The full proof would require knowing that the transitive closure of any set is a set; the latter assertion is equivalent to the statement \( \text{equal}(\text{U}(\text{FULL}), V) \) for which an \text{Otter} proof is as yet lacking (Belinfante 1999b.)

## 5 Subvariant and Invariant Subsets

A class \( y \) is invariant under \( x \) if \( \text{subclass}(\text{image}(x, y), y) \) holds. We say that a class \( y \) is subvariant under \( x \) when the reverse inclusion \( \text{subclass}(y, \text{image}(x, y)) \) holds. Using this terminology, a descending set is one that is subvariant under the membership relation \( E \).

The condition of subvariance is useful in many applications as a tool for constructing invariant sets. For example, in the recursion theorem, one constructs a function as a union of approximations that satisfy a subvariance condition. For this reason alone it seems a good idea to embark on a general study of subvariance and its relation to invariance. Over a hundred theorems about invariant and subvariant subsets proved using \text{Otter} are listed in Appendix 3.

The definitions of the classes of sets that are invariant and subvariant under a given relation \( x \) both involve the constructor \text{fix}, but differ crucially:

\[
\text{equal}(\text{fix}(\text{composite}(S, \text{IMAGE}(x))), \text{invar}(x)).
\]

\[
\text{equal}(\text{complement}(\text{fix}(\text{composite}(E, \text{complement}(\text{composite}(x, \text{inverse}(E)))))), \text{subvar}(x)).
\]

The function \text{IMAGE}(x) maps a set \( y \) to \text{image}(x, y) whenever the latter is a set. (Belinfante 2000) The explanation for the difference between these two equations has to do with the distinction between sets and proper classes. The axiom of replacement implies that a subclass of a set is a set. When \( y \) is a set, the invariance condition implies that \text{image}(x, y) is also a set, and so the ordered pair of \( x \) and \text{image}(x, y) is a point of the graph of \text{IMAGE}(x). But a set \( y \) can satisfy the subvariance condition without \text{image}(x, y) necessarily being a set. One can get a formula for \text{subvar} analogous to the definition of \text{invar} if an extra condition holds, for example if all vertical sections of \( x \) are sets. (See Theorem \text{SBV-IMG3} in Appendix 3.) The construction of the class of regular sets can be formulated succinctly using \text{subvar} as

\[
\text{equal}(\text{complement}(\text{U}(\text{subvar}(E))), \text{REGULAR}).
\]
6 The Class of Finite Sets

The customary definition of a finite set as one that can be put into one-to-one correspondence with a natural number makes the concept of finiteness appear to depend on a specific construction of natural numbers. One can however define finiteness without explicit reference to natural numbers. One such formulation is discussed by Hrbacek and Jech (1999). Formisano and Omoleo (1998) give a slightly different formulation, which is adopted here. The idea is to define a set to be finite if it does not belong to an infinitely descending chain of proper subsets. This yields a definition for the class of finite sets formally resembling that of the class of regular sets, but with the proper subset relation $PS$ replacing the membership relation $E$:

$$\text{equal}(\text{complement}(U(\text{subvar}(PS))), \text{FINITE}).$$

Using this definition of the class $\text{FINITE}$, one can prove all the usual theorems about finiteness. Among the theorems about finite sets proved using $\text{Otter}$ are that all natural numbers are finite, that all other ordinals are infinite, and a key property of $\text{FINITE}$, that it is the smallest class which holds the empty set, and is invariant under the operation of adjoining singletons. This property is formally analogous to ordinary induction, so a good name for it would be $\text{FINITE}$ induction. Applications of $\text{FINITE}$ induction include the $\text{Otter}$ proofs obtained for the theorem that the binary union of finite sets is finite, that the power set of a finite set is finite, and that a set if finite if and only if it can be put in one-to-one correspondence with a natural number. A more refined version $\text{FIN-IND-2}$ of $\text{FINITE}$ induction was used in the $\text{Otter}$ proof that the sum class of a finite set of finite sets is finite.

7 Conclusion

For most applications of automated reasoning in modern mathematics the availability of a substantial amount of set theory is essential. Much progress has been made recently toward mechanizing set theory, especially by Larry Paulson and his coworkers (Noël 1993, Paulson and Grąbczewski 1996), the Mizar group (Rudnicki and Trybulec 1999), Megill (1997), among others. Some of these efforts have been modestly described as proof checking rather than proof finding, but the distinction between the two activities is not sharp, and both are challenging at the present state of the art. I regard it as a healthy sign that each of these groups uses slightly different axioms for set theory, and that the methods employed are generally quite different. Despite all this progress in using computers to find and check the correctness of proofs of theorems in set theory, the process is still far from being routine.

For the work described here, a primary obstacle has been to find succinct and useful definitions of the classes one needs. Starting with the primitive constructors provided by the axioms, the challenge is to introduce just enough additional constructors to help reduce the complexity of the statements of theorems, but not so many that the proliferation of new concepts itself causes an unnecessary explosion in the clause lists. Introducing the notion of subvariance makes possible a unified treatment of regular sets and finite sets, and lays the groundwork for a proof of the recursion theorems needed to develop arithmetic.

Appendix 1. Theorems involving the Axiom of Regularity.

A few theorems listed here do not include the flag $\text{AxReg}$, and are valid whether or not the axiom of regularity holds.
list(wuable).
% Revised versions of Quaife's Theorems.
-AsEq | equal(intersection(regular(x),x),0). % RE-0
-AsEq | disjoint(regular(x),x). % RE-0'
-AsEq | member(x,y) | member(regular(y),y). % RE-0''
-AsEq | member(x,V) |
  member(regular(union(y,singleton(x))),union(y,singleton(x))). % RE-0'''
-AsEq | member(x,y). % RE-1
-AsEq | member(x,y) | subclass(y,x). % RE-1-SU
-AsEq | equal(RUSSELL,V). % RE-RUS
-AsEq | equal(fix(E),0). % RE-E-FP
-AsEq | disjoint(E,inverse(S)). % RE-E-SP
-AsEq | equal(singleton(x),x). % RE-2
-AsEq | equal(singleton(x),x) | equal(singleton(singleton(x)),x). % RE-3
-AsEq | equal(U(x),x) | equal(singleton(U(x)),x). % RE-5
-AsEq | member(x,singleton(x)) | equal(U(x),x). % RE-5'
-AsEq | member(x,y) | member(y,x). % RE-4
-AsEq | disjoint(E,inverse(E)). % RE-E-IN
-AsEq | member(x,U(x)). % RE-4-A
-AsEq | disjoint(E,BIOUP). % RE-E-BC
-AsEq | member(x,V) | equal(regular(union(y,singleton(x))),x) | disjoint(y,x). % RE-4-B1
-AsEq | member(x,V) | member(regular(union(y,singleton(x))),y) | disjoint(y,x). % RE-4-B3
-AsEq | member(x,y) | member(y,x) | member(x,z). % RE-4-B6
-AsEq | member(x,U(U(x))). % RE-4-B9
-AsEq | member(P(x),U(x)). % RE-PC-SC
-AsEq | equal(pair(x,y),x). % RE-5A
-AsEq | equal(pair(x,y),y). % RE-5B
-AsEq | equal(first(x),x) | member(x,cart(V,V)). % RE-6A
-AsEq | equal(second(x),x) | member(x,cart(V,V)). % RE-6B
-AsEq | member(x,V) | member(complement(x),V). % RE-7
-AsEq | equal(first(x),x) | equal(second(x),x). % RE-6A'
-AsEq | equal(first(x),second(x),x) | member(first(x),V). % RE-6A'
-AsEq | equal(first(x),second(x),x) | member(second(x),V). % RE-6B'
-AsEq | equal(first(x),second(x),x) | equal(first(x),x). % RE-6B'
-AsEq | equal(first(x),second(x),x) | member(x,cart(V,V)). % RE-6C'
-AsEq | equal(pair(x,y),second(pair(x,y)),pair(x,y)) | member(x,y). % RE-10A
-AsEq | equal(pair(x,y),second(pair(x,y)),pair(x,y)) | member(y,x). % RE-10B
% Some new theorems
-AsEq | subclass(U(x),x) | equal(x,0) | member(0,x). % RE-FUL-1
-AsEq | member(P(x),x). % RE-FC
-AsEq | equal(x,0) | member(regular(x),P(complement(x))). % RE-A1
-AsEq | equal(x,V) | member(regular(complement(x)),P(x)). % RE-A2
-AsEq | disjoint(P(x),complement(x)) | equal(x,V). % RE-A3
-AsEq | subclass(P(x),x) | equal(x,V). % RE-A4
-AsEq | full(x) | subclass(intersection(x,P(y),y)) | subclass(x,y). % RE-FUL-2
% Equivalence of two definitions of ordinal numbers.
-AsEq | equal(intersection(PULL,P(PULL)),OMEGA). % RE-0H-EQ
% Other consequences of the axiom of regularity.
-AsEq | equal(REGULAR,V). % RE-REG-1
-AsEq | subclass(x,regular(x)) | equal(x,0). % RE-1M
-AsEq | subclass(x,complement(E,x)) | equal(x,0). % RE-0M
-AsEq | equal(composite(x,inverse(SINGLETON)),composite(E,x)). % RE-C0
-AsEq | equal(fix(composite(E,DISJUNCTION),complement(singleton(0)))) | equal(x,0). % RE-DJT-E
% The lemmas RE-E-LEM does not require AsEq.
-AsEq | equal(D(x),0) | subclass(x,E) | equal(x,0). % RE-E-LEM
-AsEq | subclass(x,E) | subclass(D(x),E) | equal(x,0). % RE-E
% Theorems about IRREG
AsEq | disjoint(P(complement(IRREG)),IRREG). % RE-A2
AsEq | equal(complement(IRREG),V). % RE-A2-6A
AsEq | subclass(P(complement(IRREG)),complement(IRREG)). % RE-A2-6B
-AsEq | regular(V) | equal(U(FULL),V) | AsEq. % RE-REG-2
-AsEq | member(0,IRREG) | AsEq. % RE-A2-6C
AsEq | disjoint(OMEGA,IRREG). % RE-A2-6D
-AsEq | disjoint(x,IRREG) | AsEq | disjoint(P(x),IRREG). % RE-A2-6E
Appendix 2. Theorems about the class of regular sets.

In this appendix are listed clauses for the definitions of the classes of descending and regular sets, and theorems about these proved using Otter. Clauses flagged with an asterisk are equations which are (usually) also placed on the demodulator list in Otter input files.

list(usable).
% Definition of the class of descending sets
equal(complement (fix(composite(E,DISJUNCT))) , DESCENDING).  %+DF-DESC
% Theorems about the class DESCENDING
-member(x , DESCENDING) | subclass(x , image(E,x)).  % DESC-1
-member(x , V) | -subclass(x , image(E,x)) | member(x , DESCENDING).  % DESC-2
member(0 , DESCENDING).  % DESC-3
equal(A , (DESCENDING) , 0).  % DESC-4
-member(x , P (DESCENDING)) | member(U , (x , DESCENDING)).  % DESC-5
-member(x , DESCENDING) | member(union(x , singleton(y)) , DESCENDING) | disjoint(x , y).  % DESC-6
-member(x , y) | -member(x , U DESCENDING) | -member(y , V) | member(y , U DESCENDING).  % DESC-7
subclass(image(E , U DESCENDING)) , (U , DESCENDING)).  % DESC-8
-member(x , y) | member(singleton(x) , DESCENDING).  % DESC-9
-member(x , complement(x)) | -member(singleton(x) , DESCENDING).  % DESC-10
-member(x , y) | -member(y , x) | member(pairs(x,y) , DESCENDING).  % DESC-11
% Definition of the class of regular sets
equal(complement(U DESCENDING) , REGULAR).  %+DF-REG
% Theorems about the class REGULAR
equal(complement(REGULAR) , U DESCENDING).  % REG-C
equal(P REGULAR) , REGULAR).  % REG-RG
equal(image(E , U DESCENDING) , U DESCENDING)).  % REG-SC
-subclass(x , REGULAR) | subclass(U , x , REGULAR).  % REG2-SU
-subclass(x , U(x) , REGULAR) | subclass(x , REGULAR).  % REG2-SU2
-subclass(x , REGULAR) | subclass(P(x) , REGULAR).  % REG2-SU3
equal(image(inverse(S) , U DESCENDING) , image(V , U DESCENDING))).  % REG-C-SR
equal(U , (U DESCENDING) , image(V , U DESCENDING))).  % REG-C-SC
-member(DESCENDING , V) | equal(REGULAR , V).  % REG-C-V
equal(image(Di , U DESCENDING) , image(V , U DESCENDING))).  % REG-C-EX
equal(A , ((U DESCENDING) , complement (image(V , U DESCENDING)))).  % REG-C-A
equal(image(V , REGULAR) , V).  % REG-MV
equal(U REGULAR) , REGULAR).  % REG-3
-subclass(P(x) , REGULAR) | subclass(x , REGULAR).  % REG3-SU
-subclass(x , REGULAR) , inverse(S) , REGULAR).  % REG-HER
-member(x , REGULAR) | subclass(x , REGULAR).  % REG-4
-member(x , y) | -member(y , REGULAR) | member(x , REGULAR).  % REG-5
-member(x , REGULAR) | member(P(x) , REGULAR).  % REG-6
-member(x , REGULAR) | member(U , REGULAR).  % REG-7
-member(x , REGULAR) | -subclass(y , x) | member(y , REGULAR).  % REG-8
-member(0 , REGULAR).  % REG-9
equal(intersection(REGULAR , P (U DESCENDING))) , singleton(0)).  % REG-C-PC
equal(image(S , REGULAR) , V).  % REG-IMS
equal(image(DISJOINT , REGULAR) , V).  % REG-DJT
equal(A , REGULAR) , 0).  % REG-A
-member(x , REGULAR) | -member(y , REGULAR) | member(union(x , y) , REGULAR).  % REG-10
-member(x , REGULAR) | member(intersection(x , y) , REGULAR).  % REG-11
-member(x , REGULAR) | member(singleton(x) , REGULAR).  % REG-12
-member(x , REGULAR) | member(y , REGULAR) | member(pairs(x , y) , REGULAR).  % REG-13
-member(P(x) , REGULAR) | member(x , REGULAR).  % REG-14
-member(U , REGULAR) | member(x , REGULAR).  % REG-15
-member(pairs(x , y) , cart(REGULAR , REGULAR) | member(cart(x , y) , REGULAR).  % REG-OP
-member(x , DESCENDING) | disjoint(x , REGULAR).  % REG-16
-member(x , DESCENDING) | disjoint(y , REGULAR) | disjoint(x , y).  % REG-17
equal(intersection(DESCENDING , REGULAR) , singleton(0)).  % REG-19
-disjoint(DESCENDING , image(E , x)) | subclass(x , REGULAR).  % REG-20
- subclass(x, REGULAR) | disjoint(DISTINCT, image(R, x)). % REG-21
- member(x, REGULAR) | member(D(x), REGULAR). % REG-22
- member(x, REGULAR) | member(R(x), REGULAR). % REG-23
- member(x, REGULAR) | member(rotate(x), REGULAR). % REG-80
- subclass(x, REGULAR) | subclass(D(x), REGULAR). % REG-24
- subclass(x, REGULAR) | subclass(R(x), REGULAR). % REG-25
  subclass(cart(x, REGULAR, REGULAR), REGULAR). % REG-26
- member(x, REGULAR) | subclass(x, image(E, x)) | equal(x, 0). % REG-27
- member(REGULAR, x) % REG-28
  equal(image(Di, REGULAR), V). % REG-DI
  equal(singleton(REGULAR), 0). % REG-S1
- member(cart(REGULAR, REGULAR), x). % REG-CF-V
  equal(D(REGULAR), REGULAR). %*REG-30
  equal(R(REGULAR), REGULAR). %*REG-31
- member(x, REGULAR) | member(first(x), REGULAR). % REG-PST
- member(x, REGULAR) | member(second(x), REGULAR). % REG-SEC
  equal(intersection(REGULAR, cart(V, x)), cart(REGULAR, REGULAR, REGULAR)).% REG-32
  equal(composite(x, REGULAR), cart(REGULAR, image(x, REGULAR))). %*REG-O1
  equal(composite(REGULAR, x), cart(image(inverse(x), REGULAR), REGULAR)). %*REG-O2
  equal(image(REGULAR, x), intersection(REGULAR, image(V, intersection(REGULAR, x))). %*REG-3M
  equal(inverse(REGULAR), cart(REGULAR, REGULAR)). %*REG-34
- member(x, REGULAR) | member(inverse(x), REGULAR). % REG-FN
- member(x, REGULAR) | member(flip(x), REGULAR). % REG-FL
  equal(fix(REGULAR), REGULAR). %*REG-3E
  equal(intersection(id, REGULAR), id(REGULAR)). %*REG-DEX
  subclass(REGULAR, RUSSELL). % REG-35
end_of_list.

Appendix 3. Theorems about invariant and subvariant sets.

In this appendix are listed clauses for the definitions of the classes of invariant and
subvariant sets, and theorems about these that were proved using Otter.

list(usable).
% definition of the class of invariant subsets
equal(fix(composite(S, IMAGE(x))), invar(x)). %*DEF-IVR
% theorems about invar(x).
- member(x, invar(y)) | subclass(image(y, x, x), x). % TVR-1
- member(x, y) | -member(y, invar(BIGCUP)) | member(U(x, y). % TVR-BC1
- member(x, invar(BIGCUP)) | member(U(x), FULL). % TVR-BC1A
- member(x, invar(BIGCUP)) | member(image(inverse(S), x), FULL). % TVR-BC1B
- member(D(x), invar(x)) | subclass(R(x), D(x)). % TVR-D01
- member(x, V) | -subclass(image(y, x, x), x) | member(x, invar(y)). % TVR-2
- equal(composite(x, y), composite(x, y)) | -member(R(y), V) | member(fix(IMAGE(y), invar(IMAGE(x))). % TVR-PP
- member(x, omega) | member(intersection(omega, complement(z), invar(SUC)C)). % TVR-DM-C
- member(x, invar(BIGCUP)) | member(image(inverse(S), x), invar(BIGCUP)). % TVR-BC-S
- member(D(x), V) | subclass(R(x), D(x)) | member(D(x), invar(x)). % TVR-D02
- member(x, FULL) | member(P(x), invar(BIGCUP)). % TVR-BC2
  equal(U(invar(BIGCUP)) \ U(FULL)). % TVR-BC2A
  equal(image(inverse(S), invar(BIGCUP)), U(FULL)). % TVR-BC2C
  subclass(intersection(FULL), invar(BIGCUP)). % TVR-BC3
- member(x, invar(y)) | member(image(y, x, x), invar(y)). % TVR-3M
- member(D(x), invar(x)) | member(R(x), x, invar(x)). % TVR-R4
- subclass(composite(x, y), composite(x, y)) | subclass(image(IMAGEx, invar(x), x), invar(x)). % TVR-DMG
  subclass(image(IMAGEx, invar(x), x), invar(x)). % TVR-DMG2
- equal(D(IMAGE(x), y), V) | subclass(image(BIGCUP, invar(IMAGEx)), invar(x)). % TVR-DMG3
- subclass(x, y) | subclass(image(y, x), invar(y), x)). % TVR-SU
  equal(invar(union(x, y)), intersection(invar(x), invar(y))). % TVR-U
  subclass(P(complement(D(x))), invar(x)). % TVR-C-D0
- member(D(x), V) | -member(invar(x), V). % TVR-V
- member(0, invar(x)) | member(A(invar(x)), 0). % TVR-HEMD
- equal(A(invar(x)), 0). % TVR-O-A
- subclass(invar(x), D(IMAGE(x))). % TVR-5
  equal(fix(composite(complement(E)), composite(x, inverse(E)))), complement(invar(x))). %*TVR-5
- member(x, P(invar(y)))) | member(U(x), invar(y)).
- equal(image(BIGCUP, P(invar(x))), invar(x)).
- subclass(x, invar(y)) | subclass(image(y, U(x)), U(x)).
- subclass(x, invar(y)) | equal(x, 0) | member(A(x), invar(y)).
- equal(image(BIGCUP, P(invar(x))), invar(x)).

% examples
- equal(invar(0), V).
- equal(invar(id(x)), V).
- equal(image(id(x)), V).
- equal(invar(E), singleton(0)).
- equal(invar(S), singleton(0)).
- equal(invar(inverse(S)), fix(image(inverse(S)))).
- subclass(x, image(inverse(S)), invar(inverse(S))).
- subclass(image(inverse(S)), FULL).
- subclass(image(BIGCUP, image(BIGCUP)), FULL).
- equal(image(BIGCUP, image(BIGCUP)), FULL).
- equal(image(BIGCUP, P(FULL)), FULL).
- equal(image(BIGCUP, P(FULL)), FULL).

% applications to ordinal number theory
- member(a, invar(SUC)).
- subclass(Omega, invar(BIGCUP)).
- equal(intersection(P(o), image(BIGCUP)), succ(o)).
- subclass(intersection(Omega, image(BIGCUP)), invar(SUC)).
- equal(intersection(Omega, image(BIGCUP)), intersected(Omega, image(BIGCUP))).
- subclass(o, invar(SUC)).
- disjoint(intersection(R(SUC), image(BIGCUP)), image(BIGCUP), R))).
- equal(intersection(image(SUC, Omega), invar(SUC))).
- equal(image(BIGCUP, image(SUC, Omega))).

% definition of the class of invariant subsets
- equal(complement(fix(composite(E, complement(composite(x, inverse(E))))), subvar(x))).

% some simple examples
- equal(subvar(composite(id(x)), subvar(x)).
- equal(subvar(composite(id(x), y)), intersection(P(x), subvar(y))).
- equal(subvar(0), singleton(0)).
- equal(subvar(V), V).
- equal(subvar(id(x)), P(x)).
- equal(subvar(Di), complement(R(SINGLET))).
- equal(subvar(cart(V, x), P(x)).
- equal(subvar(E), DESCENDING).
- equal(subvar(image(BIGCUP)), invar(BIGCUP)).

% basic theorems about subvar
- member(x, subvar(y)) | subclass(x, image(y, x)).
- member(x, y) | subclass(x, image(y, x)) | member(x, subvar(y)).

% technical lemmas needed for the theory of finite sets
- member(x, y) | member(y, subvar(PS)) | member(intersection(y, P(x)), subvar(PS)).
- member(x, y) | member(image(y, PS), P(y)) | member(intersection(y, P(x)), subvar(PS)).
- member(x, subvar(PS)) | subclass(image(id(complement(y))), x), subvar(PS)).
- member(x, y) | member(0, y) | subclass(image(CUP, cart(y, R(SINGLET))), y) | member(subvar(id(complement(y))), x), subvar(PS)).
- member(x, y) | member(0, y) | subclass(image(CUP, cart(y, image(SINGLET))), y) | member(subvar(id(complement(y))), x), subvar(PS)).
- member(y, x) | subclass(image(CUP, cart(x, image(SINGLET))), x) | member(subvar(id(complement(y))), x), subvar(PS)).

% other basic theorems about subvar
- member(x, P(subvar(y))) | member(U(x), subvar(y)).
- member(pair(x, y), cart(subvar(x), subvar(y))).
- member(cart(x, y), subvar(cross(x, y))).
- subclass(intersection(subvar(x), subvar(y)), subvar(composite(x, y))).
% subclass(x,y) | subclass(subvar(x),subvar(y)).  % SBV-SU1
 subclass(subvar(x),P(R(x))).  % SBV-SU2
-member(x,V) | member(subvar(x),V).  % SBV-MEM
 subclass(P(fix(x)),subvar(x)).  % SBV-SU3
 equal(subvar(x),V).  % SBV-SR1
 equal(subvar(x),V).  % SBV-SR2
 equal(subvar(x),V).  % SBV-Q
 member(0,subvar(x)).  % SBV-MEM0
 member(0,image(image(x),subvar(y))).  % SBV-MEM0
 equal(A(subvar(x)),0).  % SBV-A
-member(x,subvar(y)) | -member(x,image(x),y) | member(union(x,singleton(z)),subvar(y)).  % SBV-SS
 equal(image(x,U(subvar(x))),U(subvar(x))).  % SBV-SC2
 % using subvar to construct invariant subsets
-member(x,P(pod)) | member(U(subvar(inverse(x))),inv(x)).  % SBV-SC3
 subclass(inv(x),P(pod),subvar(inverse(x))).  % SBV-E-IN
 subclass(intersection(P(D(x)),inv(x)),subvar(inverse(x))).  % SBV-IVB1
 subclass(intersection(P(D(x)),inv(x)),subvar(x)).  % SBV-IVB2
 equal(intersection(inv(x),subvar(x)),fix(image(x))).  % SBV-TV84
 equal(intersection(FULL,subvar(inverse(x)),fix(image(x))).  % SBV-FUL
 -FUNCTION(x) | subclass(subvar(inverse(x)),inv(x)).  % SBV-FU1
 -FUNCTION(x) | equal(subvar(inverse(x)),intersection(P(D(x)),inv(x))).  % SBV-FU2
 equal(subvar(image(x)),fix(image(x))).  % SBV-SW
 equal(image(D(x),image(x)),intersection(D(image(x)),subvar(x))).  % SBV-IMG2
 equal(image(D(x),image(x)),intersection(D(image(x)),subvar(x))).  % SBV-IMG2
 equal(image(D(x),image(x)),intersection(D(image(x)),subvar(x))).  % SBV-IMG2
 subclass(image(image(x),subvar(x))).  % SBV-IMG5
 subclass(image(image(x),subvar(x))).  % SBV-IMG5
 subclass(image(x,U(intersection(D(image(x)),subvar(x)))),U(subvar(x))).  % SBV-SC4
 % building in initial conditions
-member(x,subvar(union(id(y),x))) | subclass(x,union(y,image(x))).  % SBV-U-1
-member(x,V) | subclass(x,union(y,image(x))).  % SBV-U-2
-member(x,intersection[D(image(x)),subvar(union(id(y),x)))] | member(union(x,image(x)),subvar(union(id(y),x))).  % SBV-U-3
-member(x,V) | member(subvar(x),V).  % SBV-RA1
-member(x,V) | member(U(subvar(x)),inv(x)).  % SBV-RB2
-member(x,V) | member(U(subvar(x)),fix(image(x))).  % SBV-RB3
 subclass(subvar(SUCC),DESCENDING).  % SBV-SUC1
 equal(intersection(REGULAR,subvar(SUCC)),singleton(0)).  % SBV-SUC2
 equal(intersection(P(OMEGA),subvar(SUCC)),singleton(0)).  % SBV-SUC3
 end_of_list.

Appendix 4. Theorems about the class of finite sets.

In this appendix are listed clauses for the definition of the class of finite sets, and
theorems about finite sets proved using Otter. The relation Q is the equiopollence
relation, and the relation K is the cover relation: pair(x,y) belongs to K if y has
eactly one more element than x. The function CUP takes pair(x,y) to union(x,y).

list(unable).
% definition of the class of finite sets
equal(complement(U(subvar(P(x)))),FINITE).  % DBF-FIN
% theorems about the class FINITE
equal(U(subvar(P(x))),complement(FINITE)).  % FIN-C-INF
equal(image(P(x),complement(FINITE)),complement(FINITE)).  % FIN-P-INF
% three versions of the FINITE induction theorem
-member(0,x) | subclass(image(CUP,cart(x,R(singleton(x)))),x) | subclass(FINITE,x).  % FIN-IND3
-member(0,x) | subclass(image(K(x),x) | subclass(FINITE,x).  % FIN-K-1
-member(0,x) | subclass(image(CUP,cart(x,image(singleton(x)))),x) | subclass(FINITE,x).  % FIN-IND2
-member(x,subvar(P(z))) | disjoint(x,FINITE).  % FIN-DJ1
member(0,FINITE).  % FIN-O
equal(A(FINITE),0).  % FIN-A
member(singleton(x),FINITE).  % FIN-SI
equal(image(BIGCUP,FINITE),V). \% F1H-BC1
subclass(R(SINGLETON),FINITE). \% F1H-BAG
equal(image(BIGCUP,P(FINITE)),V). \% F1H-BC2
equal(U(FINITE),V). \% F1H-SC1
~member(FINITE,x). \% F1H-MEM
~member(image(FINITE),x). \% F1H-C-OM
~member(x,complement(FINITE)) | member(x,disjoint(FINITE,P(x),singleton(0))) . \% F1H-PP1
~member(x,FINITE) | ~member(D(x),FINITE) | member(R(x),FINITE). \% F1H-PS
~equal(intersection(P(x)),singleton(0)) | ~member(x,V) | member(x,FINITE). \% F1H-PP2
~member(x,FINITE) | ~subclass(y,x) | member(y,FINITE). \% F1H-SU
\% images of finite sets under functions are finite
~FUNCTION(x) | ~member(y,FINITE) | member(image(x,y),FINITE). \% F1H-FU1
FUNCTION(x) \& subclass(image(IMAG(x),FINITE),FINITE). \% F1H-FU2
~member(P(x),FINITE) \& member(x,FINITE). \% F1H-PC1
~member(x,FINITE) \& member(D(x),FINITE). \% F1H-D01
~member(x,FINITE) \& member(image(x),FINITE). \% F1H-RA1
~member(x,FINITE) \& member(image(x),FINITE). \% F1H-RA1
~member(x,FINITE) \& member(image(x),FINITE). \% F1H-RA1
~member(x,FINITE) \& member(image(x),FINITE). \% F1H-RA1
~member(x,FINITE) \& member(image(x),FINITE). \% F1H-RA1
\% images of finite sets under functions are finite
~FUNCTION(x) | ~member(y,FINITE) | member(composite(x,y),FINITE). \% F1H-CN1
~FUNCTION(x) | ~member(y,FINITE) | member(composite(x,y),FINITE). \% F1H-CN2
~member(x,FINITE) \& member(y,disjoint(x),FINITE). \% F1H-DJ2
~member(x,FINITE) \& member(y,disjoint(x),FINITE). \% F1H-DJ3
~subclass(image(FINITE,x),disjoint(x,FINITE)). \% F1H-SC2
~member(x,FINITE) \& member(image(FINITE,y),FINITE). \% F1H-PP3
~member(x,FINITE) \& member(image(FINITE,y),FINITE). \% F1H-PP3
~member(x,FINITE) \& member(image(FINITE,y),FINITE). \% F1H-PP3
\% images of finite sets under functions are finite
~member(x,FINITE) \& member(image(FINITE,y),FINITE). \% F1H-UP
member(pair(x,y),FINITE). \% F1H-UP
\% various domain laws
equal(image(FINITE,x),cart(D(x),y)). \% F1H-C03
equal(image(FINITE,FINITE),cart_D(x,y)). \% F1H-C04
equal(image(FINITE,x),cart_D(x,y)). \% F1H-B2
\% finite unions of finite sets
~member(x,FINITE) | ~subclass(x,FINITE) \& member(U(x),FINITE). \% F1H-SC4
equal(image(BIGCUP,image(FINITE),FINITE),FINITE). \% F1H-SC5
~member(x,FINITE) \& member(x,FINITE) \& member(x,FINITE). \% F1H-SC6
~member(x,FINITE) \& member(x,FINITE) \& member(x,FINITE). \% F1H-SC7
\% cartesian products of finite sets are finite
~member(pair(x,y),cart(FINITE,FINITE)) | member(cart(D(x),FINITE)). \% F1H-CP2
FUNCTION(x) | member(D(x),FINITE). \% F1H-PS3
equal(image(FINITE),FINITE). \% F1H-Q-1
subclass(image(FINITE),FINITE). \% F1H-Q-2
\% the finite sets are those equipollent to a natural number
equal(image(0,FINITE),FINITE). \% F1H-Q-0M
\% a characterization of finite sets of natural numbers
equal(image(0,FINITE),FINITE). \% F1H-Q-0M
equal(image(BIGCUP,complement(FINITE)),complement(FINITE)). \% F1H-BC4
equal(image(BIGCUP,complement(FINITE),complement(FINITE)). \% F1H-BC4
equal(image(BIGCUP,complement(FINITE)),complement(FINITE)). \% F1H-BC4

equal(image(BIGCUP,complement(FINITE)),complement(FINITE)). \% F1H-BC4

equal(image(BIGCUP,complement(FINITE)),complement(FINITE)). \% F1H-BC4
References