Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers* **

Gerhard Schellhorn and Wolfgang Reif

Abt. Programmiermethodik, Universität Ulm, D-89069 Ulm, Germany

Table of Contents

1 Introduction

This paper describes a problem set for automated theorem provers taken from a KIV case study in software verification. The goal is to prove 45 consequences of the axioms specifying the data type of finite enumerations. We present

- a structured algebraic specification of finite enumerations with 102 axioms.
- 52 theorems, 45 of which can be proved without induction (some of them rely on the 7 inductive consequences)
- results for the 5 provers Otter, Protein, Setheo, Spass and ${}_3T^AP$ in two versions, with and without an axiom reduction preprocessing step ([RS97])

* This research was partly sponsored by the German Research Foundation (DFG) under grant Re 828/2-2.

** Technical Report Nr. 97-12, Department of Computer Science, University of Ulm

Test files are available for Otter, Protein, Setheo, Spass ($_3TAP$ can be called directly from KIV) and also in the common syntax of the DFG-Schwerpunkt "Deduktion" [RH96]. Section 2 describes the specification of the datatype. Section 3 gives a listing of the available axioms and Section 4 contains the theorems to prove. Section 5 describes the test scenario, and Section 6 gives our experimental results.

2 The Datatype of Finite Enumerations

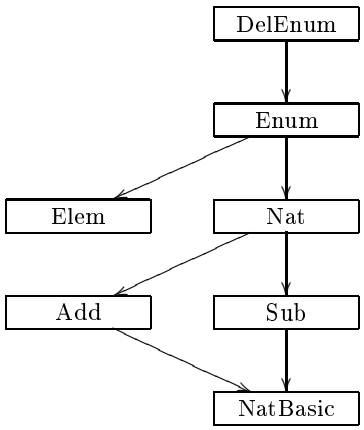
The set of theorems deals with the datatype of finite enumerations. These are bijections from a finite subset of data elements to an initial segment [0, ..., n] of the natural numbers. Actually, the specification of enumerations was part of a larger KIV case study on the verification of depth-first search on graphs. There the data type was used to enumerate a set of nodes.

Enumerations can be constructed by $\emptyset$ (the empty bijection), and by en $\oplus$ d adding a data element d (with fresh code number) to the enumeration en. Adding an element twice has no effect. The size function # en returns the number of elements recorded in the enumeration, and the predicate d $\in$ en tests for membership. The selector num_of(d, en) returns the number of the element d in en, and el_of(n, en) gives back the element number n in en. Both operations are unspecified if d $\notin$ en or if n $\geq$ # en, respectively. Finally, the operation en $\ominus$ d removes the element d and its code number from the enumeration (if there is such an element). In addition all code numbers greater than num_of(d, en) are decremented by one in order to avoid gaps in the range of en $\ominus$ d.

E.g. for en = $\emptyset \oplus$ a $\oplus$ b $\oplus$ c we have num_of(a,en) = 0, num_of(b,en) = 1, num_of(c,en) = 2, num_of(a,en $\ominus$ b) = 0 and num_of(c,en $\ominus$ b) = 1.

Specifications in KIV ([RSS95],[Rei95],[RSS97]) are *structured* algebraic specifications. They are built up from elementary first-order theories with the usual operations known in algebraic specification: union, enrichment, parameterization, actualization and renaming. Their semantics is the class of all models (loose semantics). Reachability constraints like "nat generated by 0, +1" or "list generated by nil, cons" restrict the semantics to term-generated models. The constraints are reflected by induction principles in the calculus for theorem proving used in KIV. The structure of a specification is visualized as a specification graph. Roughly, each arrow in such a specification graph indicates that one specification is based upon the other (for formal details see [Rei95]).

Fig. 1 shows the specification and its graph for finite Enumerations: Specification *NatBasic* describes natural numbers with zero (0), successor and predecessor (postfix +1 and −1). It is written like an ML ([MTH89]) datatype declaration. The axioms listed in Sect. 3.1 are generated automatically (including the induction principle "nat **generated by** 0,+1"). Specifications *Add* and *Sub* enrich *NatBasic* by addition an subtraction (written infix), *Nat* is their union. These specifications are all taken from the KIV library. Therefore, in addition to the axioms, they contain lemmas which can be used like axioms in the proofs of theorems over finite enumerations. *Elem* declares the sort *elem* of parameter elements of Enumerations. *Enum* and *DelEnum* specify finite enumerations.



NatBasic =
data specification
  nat = 0 | . +1 (. −1 : nat);
  variables m, n: nat;
  order predicates
    . < . : nat × nat;
end data specification

Add =
enrich Nat **with**
  functions
    . + . : nat × nat → nat ;
  axioms
    n + 0 = n,
    m + n +1 = (m + n) +1
end enrich

Sub =
enrich NatBasic **with**
  functions
    . − . : nat × nat → nat;
  axioms
    m − 0 = m,
    m − n +1 = (m − n) −1
end enrich

Nat =
Add + Sub

Elem =
specification
  sorts elem;
  variables d: elem;
end specification

Enum =
generic specification
  parameter Elem
  using Nat **target**
  sorts enum;
  constants ∅ : enum;
  functions
    . ⊕ . : enum × elem → enum ;
    num_of : elem × enum → nat ;
    el_of : nat × enum → elem ;
    # . : enum → nat ;
  predicates . ∈ . : elem × enum;
  variables en: enum;
  axioms
    enum **generated by** ∅, ⊕;
    d ∈ en → en ⊕ d = en,
    ¬ d ∈ ∅,
    d_1 ∈ en ⊕ d ↔ d = d_1 ∨ d_1 ∈ en,
    # ∅ = 0,
    ¬ d ∈ en → #(en ⊕ d) = (# en) +1,
    ¬ d ∈ en → num_of(d, en ⊕ d) = # en,
    d ≠ d_1 → num_of(d, en ⊕ d_1) = num_of(d, en),
    d ∈ en → el_of(num_of(d, en), en) = d
end generic specification

DelEnum =
enrich Enum **with**
  functions
    . ⊖ . : enum × elem → enum;
  axioms
    ¬ d ∈ en → en ⊖ d = en,
    d ∈ en → #(en ⊖ d) = (# en) −1,
    d_1 ∈ en ⊖ d ↔ d ≠ d_1 ∧ d_1 ∈ en,
      d ∈ en ∧ d_1 ∈ en
      ∧ num_of(d, en) < num_of(d_1, en)
    → num_of(d_1, en ⊖ d) = num_of(d_1, en) −1,
      d ∈ en ∧ d_1 ∈ en
      ∧ num_of(d_1, en) < num_of(d, en)
    → num_of(d_1, en ⊖ d) = num_of(d_1, en),
end enrich

Fig. 1. The Example Specification

3 The Axioms

3.1 Axioms and Lemmas from NatBasic

Axioms:

ax-1: $n + 1 - 1 = n$

ax-2: $n + 1 = n_0 + 1 \leftrightarrow n = n_0$

ax-3: $0 \neq n + 1$

ax-4: $n = 0 \vee n = n - 1 + 1$

ax-5: $\neg\ n < n$

ax-6: $n < n_0 \wedge n_0 < n_1 \rightarrow n < n_1$

ax-7: $\neg\ n < 0$

ax-8: $n_0 < n + 1 \leftrightarrow n_0 = n \vee n_0 < n$

Lemmas:

elim-pred: $m \neq 0 \rightarrow (n = m - 1 \leftrightarrow m = n + 1)$

lem-01: $0 < n \leftrightarrow n \neq 0$

lem-02: $m_1 + 1 < m_2 + 1 \leftrightarrow m_1 < m_2$

lem-03: $n \neq n + 1$

lem-04: $n \neq n + 1 + 1$

lem-05: $n - 1 + 1 = n \leftrightarrow n \neq 0$

lem-06: $m < n + 1 \leftrightarrow \neg\ n < m$

lem-07: $m + 1 < n \leftrightarrow m < n \wedge n \neq m + 1$

lem-08: $n - 1 = n \rightarrow n = 0$

lem-09: $n < n - 1 \rightarrow n = 0$

lem-10: $\neg\ 0 + 1 < n \leftrightarrow n = 0 \vee n = 0 + 1$

lem-11: $\neg\ m < n - 1 \rightarrow \neg\ m + 1 < n$

lem-12: $m \neq 0 + 1 \rightarrow (m - 1 = 0 \rightarrow m = 0)$

lem-13: $n - 1 < n \leftrightarrow n \neq 0$

lem-14: $m - 1 < n \rightarrow \neg\ n < m \wedge n \neq 0$

lem-15: $\neg\ n < m \rightarrow (\neg\ m - 1 < n \rightarrow m = 0)$

lem-16: $m < n \rightarrow (\neg\ m - 1 < n \rightarrow m = 0)$

lem-17: $m \neq 0 \rightarrow (m - 1 < n \leftrightarrow m < n + 1)$

lem-18: $m \neq 0 \rightarrow m - 1 + 1 = m$

3.2 Axioms and Lemmas from Add

Axioms:

ax-1: $n + 0 = n$

ax-2: $m + n + 1 = (m + n) + 1$

ax-3: $n < n_0 \vee n = n_0 \vee n_0 < n$

Lemmas:

ass: $(m + n) + k = m + n + k$

com: $m + n = n + m$

lem-01: $0 + n = n$

lem-02: $m + 1 + n = (m + n) + 1$

lem-03: $m + n = (m + k) + 1 \leftrightarrow n = k + 1$

lem-04: $m + k < n + k \leftrightarrow m < n$
lem-05: $m + n = m + k \leftrightarrow n = k$
lem-06: $m \neq (m + k) +1$
lem-07: $n \neq 0 \rightarrow m + n -1 = (m + n)\text{-}1$
lem-08: $m + n = (m + k) +1 +1 \leftrightarrow n = k +1 +1$
lem-09: $\neg\ m + n < m$
lem-10: $m + n = n +1 \leftrightarrow m = 0 +1$
lem-11: $m + n = m \leftrightarrow n = 0$
lem-12: $m < n + m \leftrightarrow n \neq 0$
lem-13: $k < m \wedge \neg\ n < n_0 \rightarrow k + n_0 < m + n$
lem-15: $\neg\ m + n \neq 0 \leftrightarrow m = 0 \wedge n = 0$
lem-16: $k \neq 0 \rightarrow (\neg\ (k + m)\text{-}1 < n \leftrightarrow n < k + m)$
lem-17: $m \neq 0 \rightarrow (\neg\ (k + m)\text{-}1 < n \leftrightarrow n < k + m)$
lem-18: $k + n = (k + m) +1 \leftrightarrow n = m +1$

3.3 Axioms and Lemmas from Sub

Axioms:

ax-01: $m - 0 = m$
ax-02: $m - n +1 = (m - n)\text{-}1$

Lemmas:

lem-01: $n - n = 0$
lem-02: $n +1 - n = 0 +1$
lem-03: $m -1 - n = (m - n) -1$
lem-07: $m < n \rightarrow n - n - m = m$
lem-08: $\neg\ n < m \rightarrow n - n - m = m$
lem-10: $n < m \wedge n \neq 0 \rightarrow m - n -1 = (m - n) +1$
lem-11: $\neg\ m < n \wedge n \neq 0 \rightarrow m - n -1 = (m - n) +1$
lem-13: $m < n \rightarrow n +1 - m = (n - m) +1$
lem-14: $\neg\ n < m \rightarrow n +1 - m = (n - m) +1$
lem-15: $\neg\ n < m \rightarrow n +1 - n - m = m +1$
lem-16: $m < n \rightarrow n +1 - (n - m)\text{-}1 = m +1 +1$
lem-17: $m < n \rightarrow n +1 - n -1 - m = m +1 +1$
lem-21: $n < m \wedge k < m \rightarrow (m - n < m - k \leftrightarrow k < n)$
lem-22: $n < m \wedge \neg\ m < k \rightarrow (m - n < m - k \leftrightarrow k < n)$
lem-23: $\neg\ m < n \wedge k < m \rightarrow (m - n < m - k \leftrightarrow k < n)$
lem-24: $\neg\ m < n \wedge \neg\ m < k \rightarrow (m - n < m - k \leftrightarrow k < n)$
lem-25: $n < m \wedge k < m \rightarrow (\neg\ m - n < m - k \leftrightarrow \neg\ k < n)$
lem-26: $n < m \wedge \neg\ m < k \rightarrow (\neg\ m - n < m - k \leftrightarrow \neg\ k < n)$
lem-27: $\neg\ m < n \wedge k < m \rightarrow (\neg\ m - n < m - k \leftrightarrow \neg\ k < n)$
lem-30: $\neg\ m < n \wedge \neg\ m < k \rightarrow (\neg\ m - n < m - k \leftrightarrow \neg\ k < n)$
lem-37: $n < n - m \rightarrow n < m$
lem-38: $n - m = 0 \rightarrow \neg\ m < n$

3.4 Lemmas from Nat

Lemmas:

elim: $\neg\ m < n \rightarrow k = m - n \leftrightarrow m = k + n$

lem-04: $(m + n) - n = m$
lem-05: $m - n + n_1 = (m - n) - n_1$
lem-06: $(m + n) +1 - n = m +1$
lem-09: $\neg\ n < n_1 \rightarrow (n - n_1) + m = (n + m) - n_1$
lem-12: $m < n \rightarrow (n - m)\text{-}1 + m = n - 1$
lem-18: $\neg\ n < m \rightarrow (n - m) + m = n$
lem-19: $\neg\ n < m \rightarrow m + n - m = n$
lem-20: $n_1 < n \rightarrow (n - n_1) + m = (n + m) - n_1$
lem-28: $\neg\ k < m \rightarrow (\neg\ k - m < n \leftrightarrow \neg\ k < m + n)$
lem-29: $\neg\ k < m \rightarrow (k - m < n \leftrightarrow k < m + n)$
lem-31: $\neg\ m < n_1 \rightarrow (\neg\ m - n_1 < n \leftrightarrow \neg\ m < n + n_1)$
lem-32: $\neg\ m < n_1 \rightarrow (m - n_1 < n \leftrightarrow m < n + n_1)$
lem-33: $\neg\ n < n_1 \rightarrow (\neg\ m < n - n_1 \leftrightarrow \neg\ m + n_1 < n)$
lem-34: $n_1 < n \rightarrow (\neg\ m < n - n_1 \leftrightarrow \neg\ m + n_1 < n)$
lem-35: $\neg\ n < n_1 \rightarrow (m < n - n_1 \leftrightarrow m + n_1 < n)$
lem-36: $n_1 < n \rightarrow (m < n - n_1 \leftrightarrow m + n_1 < n)$

3.5 Axioms from Enum

Axioms:

ax-01: $d \in en \rightarrow en \oplus d = en$
ax-04: $d_1 \in en \oplus d \leftrightarrow d = d_1 \vee d_1 \in en$
ax-06: $\#\ \emptyset = 0$
ax-07: $\neg\ d \in en \rightarrow \#(en \oplus d) = (\#\ en) +1$
ax-09: $\neg\ d \in en \rightarrow num\_of(d, en \oplus d) = \#\ en$
ax-10: $d \neq d_1 \rightarrow num\_of(d, en \oplus d_1) = num\_of(d, en)$
ax-13: $d \in en \rightarrow el\_of(num\_of(d, en), en) = d$

3.6 Axioms from DelEnum

Axioms:

ax-02: $\neg\ d \in en \rightarrow en \ominus d = en$
ax-03: $\neg\ d \in \emptyset$
ax-05: $d_1 \in en \ominus d \leftrightarrow d \neq d_1 \wedge d_1 \in en$
ax-08: $d \in en \rightarrow \#(en \ominus d) = (\#\ ix) - 1$
ax-11: $d \in en \wedge d_1 \in en \wedge num\_of(d, en) < num\_of(d_1, en) \rightarrow num\_of(d_1, en \ominus d) = num\_of(d_1, en) - 1$
ax-12: $d \in en \wedge d_1 \in en \wedge num\_of(d_1, en) < num\_of(d, en) \rightarrow num\_of(d_1, en \ominus d) = num\_of(d_1, en)$

4 The Theorems

Lemmas:

th-01: $\neg\ d \in en \ominus d$
th-02: $d \in en \oplus d$

th-03: $d \in en \rightarrow num\_of(d, en) < \# en$

th-04: $\neg\ d \in en \rightarrow el\_of(\# en, en \oplus d) = d$

th-05: $d \neq d_1 \wedge d \in en \wedge d_1 \in en \rightarrow num\_of(d, en) \neq num\_of(d_1, en)$

th-06: $\emptyset \neq en \oplus d$

th-07: $\# en = 0 \leftrightarrow en = \emptyset$

th-08: $\neg\ d \in en \rightarrow num\_of(d, en) = num\_of(d, \emptyset)$

th-09: $\neg\ d \in en \wedge \neg\ d_0 \in en_0 \wedge d \neq d_0 \rightarrow en \oplus d \neq en_0 \oplus d_0$

th-10: $\#(\emptyset \oplus d) = 0 + 1$

th-11: $d_0 \neq d \wedge \neg\ d_0 \in en \rightarrow \#((en \oplus d) \oplus d_0) = (\#(en \oplus d)) + 1$

th-12: $\neg\ d \in en \wedge \neg\ d_0 \in en \rightarrow (en \oplus d = en \oplus d_0 \leftrightarrow d = d_0)$

th-13: $en \neq \emptyset \rightarrow (\exists\ d_1, en_1.\ en = en_1 \oplus d_1 \wedge \neg\ d_1 \in en_1)$

th-14: $d \in en \rightarrow num\_of(d, en) \neq \# en$

th-15: $n < \# en \rightarrow (\exists\ d.\ d \in en \wedge num\_of(d, en) = n)$

th-16: $\neg\ d \in en \wedge n < \# en \rightarrow el\_of(n, en \oplus d) = el\_of(n, en)$

th-17: $n < \# en \rightarrow el\_of(n, en) \in en$

th-18: $\neg\ d \in en \wedge n < \# en \rightarrow el\_of(n, en) \neq d$

th-19: $n < \# en \rightarrow num\_of(el\_of(n, en), en) = n$

th-20: $n_1 < \# en \wedge n_2 < \# en \rightarrow (el\_of(n_1, en) = el\_of(n_2, en) \leftrightarrow n_1 = n_2)$

th-21: $n \neq n_1 \wedge n < \# en \wedge n_1 < \# en \rightarrow el\_of(n, en) \neq el\_of(n_1, en)$

th-22: $en = en_0 \leftrightarrow \# en = \# en_0 \wedge$
$(\forall\ n.\ n < \# en \rightarrow el\_of(n, en) = el\_of(n, en_0))$

th-23: $en = en_0 \leftrightarrow \# en = \# en_0 \wedge$
$(\forall\ d.\ d \in en \leftrightarrow d \in en_0) \wedge (\forall\ d.\ num\_of(d, en) = num\_of(d, en_0))$

th-24: $\neg\ d_0 \in en \wedge d_0 \neq d \rightarrow (en \oplus d) \ominus d_0 = en \oplus d$

th-25: $d_0 \neq d \wedge \neg\ d_0 \in en \rightarrow \#((en \ominus d) \oplus d_0) = (\#(en \ominus d)) + 1$

th-26: $d_0 \in en \wedge d_0 \neq d \rightarrow (en \ominus d) \oplus d_0 = en \ominus d$

th-27: $d_0 \in en \wedge d_0 \neq d \rightarrow (en \oplus d) \oplus d_0 = en \oplus d$

th-28: $\neg\ d_0 \in en \wedge d_0 \neq d \rightarrow (en \ominus d) \ominus d_0 = en \ominus d$

th-29: $d_0 \in en \wedge d_0 \neq d \rightarrow \#((en \oplus d) \ominus d_0) = (\#(en \oplus d)) - 1$

th-30: $\neg\ d \in en \wedge d_0 \neq d \rightarrow \#((en \oplus d) \ominus d_0) = (\#(en \ominus d_0)) + 1$

th-31: $num\_of(d, en \ominus d) = num\_of(d, \emptyset)$

th-32: $d_0 \neq d \rightarrow num\_of(d, (en \ominus d) \oplus d_0) = num\_of(d, \emptyset)$

th-33: $\neg\ d_1 \in en \wedge d \neq d_1 \rightarrow num\_of(d_1, (en \ominus d) \oplus d_1) = \#(en \ominus d)$

th-34: $d \in en \rightarrow \neg\ \# en < num\_of(d, en)$

th-35: $\neg\ d \in en \rightarrow num\_of(d, en \ominus d_0) = num\_of(d, \emptyset)$

th-36: $\neg\ d \in en \wedge d_1 \neq d \rightarrow num\_of(d, (en \ominus d_0) \oplus d_1) = num\_of(d, \emptyset)$

th-37: $\neg\ d \in en \wedge d_1 \neq d \rightarrow num\_of(d, (en \oplus d_1) \ominus d_0) = num\_of(d, \emptyset)$

th-38: $d \neq d_1 \rightarrow (en \oplus d_1) \ominus d = (en \ominus d) \oplus d_1$

th-39: $el\_of(n, en) \in en \wedge n < \# en \rightarrow num\_of(el\_of(n, en), en) = n$

th-40: $d \in \emptyset \oplus d_1 \leftrightarrow d = d_1$

th-41: $(en \ominus d) \ominus d = en \ominus d$

th-42: $(\emptyset \oplus d) \ominus d = \emptyset$

th-43: $\emptyset \ominus d = \emptyset$

th-44: $el\_of(0, \emptyset \oplus d) = d$

th-45: $\# en = 0 + 1 \rightarrow \emptyset \oplus el\_of(0, en) = en$

th-46: $\# en = 0 + 1 \rightarrow (d \in en \leftrightarrow d = el\_of(0, en))$

th-47: $\# en = 0 + 1 \rightarrow (en = \emptyset \oplus d \leftrightarrow el\_of(0, en) = d)$

th-48: $\#\ en = 0 + 1 \rightarrow el\_of(0, en) \in en$
th-49: $\#\ en = 0 + 1 \rightarrow en \ominus el\_of(0, en) = \emptyset$
th-50: $\neg\ d \in en \rightarrow (en \oplus d) \ominus d = en$
th-51: $\#\ en = 0 + 1 \rightarrow (d \in en \leftrightarrow en = \emptyset \oplus d)$
th-52: $\neg\ d \in en \wedge \neg\ d_0 \in en_0 \rightarrow (en \oplus d = en_0 \oplus d_0 \leftrightarrow en = en_0 \wedge d = d_0)$

Remark: Of course, we could have split these theorems in two blocks, and placed the theorems not using $\ominus$ in *Enum*, the others in *DelEnum*

5 The Test Scenario

5.1 Sequential Test Discipline

The proof of each of the theorems shown in Sect. 4 could be tried using the 102 axioms from Sect. 3. A far better strategy is the following: to prove theorem th-n all the n-1 previously proved theorems as lemmas to the theory. Although this enlarges the theory, the effect is positive: With the redundant 77 lemmas of *Nat* and the discipline to add all previously proved test examples to the theory, the success rate of the five provers was doubled (since proof lengths become much shorter, and the number of proofs which require induction decreases).

The order of the theorems is generated such that it is compatible with the partial order induced by the hierarchy of proofs in KIV (i.e. if the KIV proof of a theorem T uses another theorem L as a lemma, then the number of L is lower than that of T).

The sequential test discipline results in one input file for each of the 45 theorems to prove and for each of the 5 provers.

5.2 Axiom Reduction

The full number of axioms and lemmas in a specification

is usually quite large (here up to 153, several hundreds in many case studies). Most of them are redundant for the proof of a particular theorem. Therefore we have developed a reduction algorithm to find out irrelevant axioms in large specifications. It exploits the specification hierarchy and other properties of axioms. It is described in [RS97]. Often the number of relevant axioms and lemmas can be reduced drastically.

To study the effects of axiom reduction, for each of the 45 noninductive theorems (and for each prover) another input file was generated for each prover, which contains only the reduced set of axioms as computed by the algorithm. This set depends on the signature, which occurs in the theorem:

- Case 1: The theorem does not contain any of $\ominus$, $<$ or -1.
- Case 2: The theorem does not contain $\ominus$, but it contains the less predicate or the predecessor ($<$ or -1).
- Case 3: The theorem contains $\ominus$

Case 1 applies to theorems th-02, th-04 – th-14, th-23, th-27, th-40, th-44 – th-48, th-51 and th-52, case 2 applies to th-03, th-15 – th-22, th-34 and th-39. Case 3 applies to the remaining theorems. Depending on the case, axiom reduction computes for each theorem a set of *relevant* axioms:

- For case 1 the axioms from Enum and the theorems of case 1 with lower number are relevant. Also the axioms ax-2 and ax-3 and the theorems lem-03 and lem-04 from NatBasic, which mention neither $<$ nor -1, are relevant.
- For case 2 the axioms from Enum and the theorems with lower number in cases 1 and 2 are relevant. Also the axioms and theorems of NatBasic are relevant.
- For case 3 only the axioms and theorems from Nat, Add and Sub can be dropped. All axioms and theorems from NatBasic are relevant. Also all axioms from Enum and DelEnum, and all theorems with lower number are relevant.

5.3 Input Syntax

Although each of the provers we tested has a different input syntax, a common translation for symbols was used in the generation of the 90 input files. Since most automated theorem provers cannot handle infix symbols or graphic symbols, as they are used in KIV, the symbols of the previous sections had to be translated to ASCII symbols (also some of the symbols are named differently in the KIV case study than in this paper). The following table gives the translation from the notation used here to the ASCII notation.

here	ASCII	here	ASCII	here	ASCII	here	ASCII
elem	data	$-$ (infix)	jsub	# (prefix)	jsizix	m	m
nat	nat	$\oplus$ (infix)	jaddix	$<$ (infix)	jls	n_0	n0
enum	index	$\ominus$ (infix)	jsubix	$\in$ (infix)	inx	en	ix
$+1$ (postfix)	jsuc	num_of	ix_of	d	d	en_0	ix0
-1 (posfix)	jpre	el_of	el_of	n_1	n1	d_1	d1
$+$ (infix)	jadd			n	n		

6 Experimental Results with 5 Provers

6.1 The provers and their settings

All experiments were done on a SPARC 20 with Solaris 2.4. The provers were given a time limit of 2 minutes for the proof time (excluding preprocessing).

Otter Otter (version 3.0.4,[WOLB92]) has a built-in equality predicate, but no sorts. These were encoded by mapping terms t of sort s to pairs *mk(t,s)*, where s is a constant. Otter was used with the settings of auto-mode, but with the (negated) theorem to prove as the set of support (sos, see p. 552 of [WOLB92]) and with binary resolution instead of hyper resolution (since the

latter is not complete when combined with sos). The settings are contained in a file named 'settings'. This file must be prepended to the input file, to call Otter successfully. We also tried some other settings, but all resulted in fewer provable theorems. E.g. in auto-mode, 19 resp. 26 theorems could be proved without resp. with reduction. For th-48 auto-mode finds the rather trivial two-step proof (application of th-46 and reflexivity) in 0.10 seconds, while our setting only finds a complex proof in 138 seconds.

Setheo Setheo (V 3.3,[GLMS94]) can handle only clauses from an unsorted logic, and has no built-in equality predicate. Therefore we had to use a standard algorithm for encoding formulas as clauses. The resulting clauses are the same as the ones that Otter generates, except that clauses $\{x \neq t, L_1, \ldots L_1\}$ with $x \notin Vars(t)$ are optimized to $\{L_1[x \leftarrow t], \ldots L_1[x \leftarrow t]\}$. Sorts were encoded as additional arguments to functions, similar to Otter, but dropping the *mk*-function. The equality predicate therefore has arity 4 and was explicitly axiomatized. Setheo was used with the '-wdr' option, which performed significantly better than the '-dr' option.

Protein Protein (v 2.12, [BF94]) was given the same input as Setheo. It was used with the model elimination calculus, which gave the best results.

Spass Spass (v 0.55, [WGR96]) is a theorem prover for unsorted first-order logic with equality. Since unary predicates are treated by special "sort" inference rules we encoded sorts as unary predicates.

${}_3T^AP$ ${}_3T^AP$ (version 4.0, [BHOS96]) is a theorem prover for full first-order logic with equality. In a project together with the developers of ${}_3T^AP$ on the integration of interactive and automated theorem proving an interface between KIV and ${}_3T^AP$ has already been built. Therefore, in contrast to all other provers, it was not necessary to generate separate input files for each theorem. Instead ${}_3T^AP$ could be called directly from KIV. The experiments with ${}_3T^AP$ benefited from the fact, that KIV gives only a subset of the previously proved lemmas to ${}_3T^AP$.

6.2 The input and output files

Some of the theorems, namely th-03, th-07, th-08, th-13, th-15, th-22 and th-23 can only be proved by structural induction. Currently no input files are generated (a later release may include the two goals for induction base and step), but the theorems are used as lemmas. For any other theorem th-*n*, a file 'v0-th-*n*' with the full set of axioms and a file 'v*k*-th-*n*' with the reduced set of axioms is generated, where k is 4,3,1, if the theorem falls under case 1,2,3 (see Sect. 5.2) respectively (for historical reasons). The files have suitable extensions .tme, .lop, .cl, .in etc. for the provers. The output generated from the provers for input file 'v*k*-th-*n*' can be found in 'res-v*k*-th-*n*' for Spass and Protein. The output file for Otter is 'v*k*-th-*n*.out', and for Setheo it is 'v*k*-th-*n*.log'.

6.3 Proof Times and Success Rates

The following table gives the proof times (in seconds) for the automated theorem provers. The results are given with (columns marked with *red*) and without

axiom reduction. Lines marked with an exclamation mark have not been tried, since the proof requires induction. The last column gives the number of interactions, which were required in KIV to prove a goal. An exclamation mark in this column indicates that the proof in KIV used induction (for th-19 and th-51 induction would not have been necessary). For th-14 the positions marked with an 'X' refer to the fact, that it is not provable with the reduced set of axioms, since it must be generalized to th-03, which requires the additional <-predicate. The success rates are indicated in the bottom row. Σ indicates how many of the 45 theorems could be proved with and without reduction for all the provers.

theorem	Otter	Otter red	Protein	Protein red	Setheo	Setheo red	Spass	Spass red	${}_3T^AP$	${}_3T^AP$ red	KIV
th-01	0.03	0.02	0.00	0.00	0.99	0.40	—	51.48	6.54	0.34	0
th-02	0.02	0.01	0.00	0.00	0.99	0.18	78.72	0.23	0.31	0.13	0
th-03	!	!	!	!	!	!	!	!	!	!	0!
th-04	1.86	0.04	13.15	0.82	5.49	0.35	—	0.25	—	20.9	1
th-05	0.04	0.01	—	21.73	—	0.84	53.10	0.15	—	—	2
th-06	0.01	0.02	0.00	0.02	1.01	0.18	1.07	0.13	—	5.49	2
th-07	!	!	!	!	!	!	!	!	!	!	0!
th-08	!	!	!	!	!	!	!	!	!	!	0!
th-09	82.97	10.96	—	—	—	—	97.98	1.40	—	—	4
th-10	10.08	—	0.03	0.00	1.92	0.23	58.83	0.28	—	7.77	0
th-11	7.20	0.74	—	0.02	2.35	0.26	—	0.47	31.42	1.81	0
th-12	0.42	0.81	0.07	0.03	1.08	0.25	60.07	0.37	—	—	0
th-13	!	!	!	!	!	!	!	!	!	!	0!
th-14	0.01	X	0.18	X	1.97	X	46.40	—	—	X	1
th-15	!	!	!	!	!	!	!	!	!	!	3!
th-16	—	—	—	—	—	—	—	—	—	—	3
th-17	0.27	0.05	—	—	—	—	97.92	16.78	—	—	1
th-18	0.15	0.08	0.02	0.03	1.20	0.40	48.63	7.92	3.64	0.45	0
th-19	0.28	0.05	—	—	—	40.51	—	20.83	—	—	2!
th-20	—	—	—	8.42	11.41	0.49	—	17.87	—	—	1
th-21	0.49	0.16	0.00	0.00	1.14	0.42	119.88	19.52	—	—	0
th-22	!	!	!	!	!	!	!	!	!	!	5!
th-23	!	!	!	!	!	!	!	!	!	!	9!
th-24	1.00	0.86	0.48	0.27	1.31	0.68	—	89.57	—	5.92	0
th-25	0.16	0.16	0.05	0.02	1.31	0.63	108.33	56.43	—	12.53	0
th-26	1.70	1.16	0.57	0.38	1.28	0.68	—	71.50	—	—	0
th-27	0.42	0.16	0.02	0.00	1.26	0.31	117.92	1.12	—	1.41	0
th-28	0.19	0.17	0.02	0.02	1.24	0.65	90.07	53.20	5.49	1.91	0
th-29	0.41	0.31	0.05	0.02	1.32	0.70	—	86.52	—	15.93	0
th-30	—	—	—	—	—	—	—	—	—	—	0
th-31	0.00	0.00	0.00	0.00	1.32	0.75	45.02	28.15	—	—	1
th-32	2.76	2.53	0.12	0.05	1.39	0.68	—	104.28	5.2	1.5	0
th-33	0.21	0.20	0.05	0.00	1.32	0.70	—	56.23	—	0.65	0
th-34	31.50	5.85	0.07	0.00	1.58	0.53	46.12	9.22	5.57	1.06	3
th-35	0.16	0.15	0.12	0.07	1.35	0.68	104.48	52.63	4.31	0.72	1
th-36	—	105.21	45.87	0.30	2.87	0.89	—	109.28	23.29	0.67	0
th-37	23.29	21.27	0.13	0.05	1.31	0.74	—	105.13	—	1.03	1

theorem	Otter	Otter red	Protein	Protein red	Setheo	Setheo red	Spass	Spass red	$_3T^AP$	$_3T^AP$ red	KIV
th-38	—	—	—	—	—	—	—	—	—	—	11
th-39	0.36	0.17	0.00	0.02	1.35	0.48	109.79	22.95	—	—	0
th-40	0.11	0.03	15.50	0.42	7.94	0.51	—	1.67	—	—	0
th-41	0.03	0.02	0.00	0.00	1.33	0.78	1.97	1.02	—	—	0
th-42	0.68	0.61	—	—	—	11.07	—	—	—	23.47	4
th-43	0.01	0.70	0.00	0.00	1.45	0.76	2.08	1.50	—	4.59	0
th-44	—	3.23	7.83	0.48	6.94	0.58	99.73	3.53	—	—	1
th-45	—	11.05	—	—	—	—	—	66.22	—	—	1
th-46	—	105.09	—	—	90.54	4.45	—	6.22	—	—	0
th-47	36.18	6.94	119.03	2.88	2.40	0.53	—	2.08	—	—	0
th-48	1.19	—	0.00	0.00	1.38	0.34	53.22	2.93	—	—	0
th-49	1.76	0.70	0.65	0.43	5.90	1.42	—	—	—	—	1
th-50	—	—	—	—	—	—	—	—	—	—	5
th-51	—	21.60	—	—	—	56.54	—	9.47	—	20.19	0!
th-52	12.29	—	—	—	117.15	—	118.03	—	—	—	5
Σ	35	36	30	32	34	36	22	37	9	21	52

References

[BF94] P. Baumgartner and U. Furbach. Protein: A prover with a theory extension interface. In *Proc. 12th CADE*, LNCS 804. Springer, 1994. for the newest version of Protein, see the URL: http://www.uni-koblenz.de/ag-ki/-Implementierungen/Protein/.

[BHOS96] Bernhard Beckert, Reiner Hähnle, Peter Oel, and Martin Sulzmann. The tableau-based theorem prover $_3T^AP$, version 4.0. In Michael McRobbie, editor, *Proc. 13th CADE, New Brunswick/NJ, USA*, LNCS 1104, pages 303–307. Springer, 1996. for the newest version of $_3T^AP$, see the URL: http://i12www.ira.uka.de/~threetap/.

[GLMS94] C. Goller, R. Letz, K. Mayr, and J. Schumann. Setheo v3.2: Recent developments – system abstract. In A. Bundy, editor, *12th Int. Conf. on Automated Deduction, CADE-12*, Springer LNCS 814. Nancy, France, 1994. for the newest version of Setheo, see the URL: http://wwwjessen.informatik.-tu-muencen.de/forschung/reasoning/setheo.html.

[MTH89] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. MIT Press, Cambridge, MA, 1989.

[Rei95] W. Reif. The KIV-approach to Software Verification. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software – Final Report*. Springer LNCS 1009, 1995.

[RH96] C. Weidenbach R. Hähnle, M. Kerber. Common Syntax of the DFG-Schwerpunktprogramm "Deduktion". Technical Report 10/96, Fakultät für Informatik, Universität Karlsruhe, Germany, 1996. current version available from the DFG-Schwerpunktprogramm homepage: http://www.uni-koblenz.de/ag-ki/Deduktion/.

[RS97] W. Reif and G. Schellhorn. Theorem Proving in Large Theories. CADE 97, Workshop on Automated Theorem Proving in Software Engineering, to appear, 1997.

[RSS95] W. Reif, G. Schellhorn, and K. Stenzel. Interactive Correctness Proofs for Software Modules Using KIV. In *Tenth Annual Conference on Computer Assurance*, IEEE press. NIST, Gaithersburg (MD), USA, 1995.

[RSS97] W. Reif, G. Schellhorn, and K. Stenzel. Proving System Correctness with KIV 3.0. In *14th International Conference on Automated Deduction. Proceedings*. Townsville, Australia, Springer LNCS, 1997. to appear.

[WGR96] C. Weidenbach, B. Gaede, and G. Rock. Spass & flotter, version 0.42. In *13th International Conference on Automated Deduction, CADE-13*, Springer LNCS, 1996. for the newest version of Spass, see the URL: http://www.mpi-sb.mpg.de/guide/software/spass.html.

[WOLB92] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning, Introduction and Applications (2nd ed.)*. McGraw Hill, 1992. for the newest version of OTTER, see the URL: http://www.mcs.anl.gov/home/mccune/ar/-otter/#doc.