Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers*

Gerhard Schellhorn

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 on the implementaion of depth-first search on graphs. The goal is to prove 54 consequences of the axioms specifying directed graphs. We present

- a structured algebraic specification of directed graphs with 165 axioms.
- 54 theorems, at least 46 of which can be proved without induction (some of the theorems rely on the 8 consequences, which have been proved in KIV with the help of induction)

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

Test files are available for the common syntax of the DFG-Schwerpunkt "Deduktion" ([RH96]), the Syntax of Otter ([WOLB92]) and as clauses for Setheo ([GLMS94]), the latter two using a functional encoding of sorts. 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. Finally Section 5 describes the test scenario

2 The Datatype of Directed Graphs

The set of theorems deals with a variant of the abstract datatype of directed graphs (no multiple edges), where the set of nodes is an initial segment $\{0, \ldots \mathrm{n} - 1\}$ of the natural numbers. This represenation allows efficient iteration over all nodes in the KIV-implementation of depth-first search, as well as an efficient implementation using adjacency lists.

A graph with node set $\{0 \ldots \mathrm{n} - 1\}$ and no edges can be constructed with *mkpg(n)*. For a graph *pg* with node set $\{0 \ldots \mathrm{n} - 1\}$, the new graph *pg ++* (where *++* is written postfix) contains one new node (so it has node set $\{0 \ldots \mathrm{n}\}$) and the same set of edges as *pg*. $\#_p$ *pg* gives the number of nodes in *pg*, so the test, whether node m is contained in pg, is $m < \#_p$ *pg*.

Edges are constructed as pairs of two natural numbers (source and target) by n => m (so => is an infix constructor for pairs). Adding an edge to a graph is done with *pg* $+_{pe}$ n => m ($+_{pe}$ is also written infix). This operation adds the edge n => m to the set of graph edges only if both nodes n and m are already contained in the graph, i.e. are below $\#_p$ *pg*. Otherwise it does not change the graph.

An edge can be deleted with *pg* $-_{pe}$ n => m (again $-_{pe}$ is infix). Membership in the set of graph edges can be checked with n => $m \in_{pe}$ *pg*. $\#_{pe}$ *pg* gives the number of edges of a graph, and finally *psuccs(pg,n)* gives the ordered list of all nodes m for which the graph contains an edge n => m (i.e. the successors of n).

To describe a datatype like directed graphs, KIV ([RSS95],[Rei95],[RSS97]) uses *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. 2 shows the specification graph for the datatype of graphs: 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, *Nat* is their union. Specification *List* specifies the datatype of lists with arbitrary elements. *Memlist* is an enrichment of lists with a membership function *in*, a function *last* to select the last element of a list, and an infix function *until*. *l until e* selects the prefix of the list *l* until the first occurence of *e*, or the whole list, if *e* is not in *l*.

Specification *Pair* defines generic pairs with arbitrary elements. All these specifications have been taken from the KIV-library of predefined specifications. Therefore they contain functions, which would not be neccessary for the task of defining directed Graphs in the toplevel specification *Graph*.

The toplevel specifications given in Fig. 2 uses pairs of natural numbers (specification *Edge*) as edges, and lists of natural number (*Natlist*) enriched with an *ordered*-predicate (*OrderedList*) as successors (as result of the function *psuccs*). The auxiliary specifications are all given in Fig. 2.

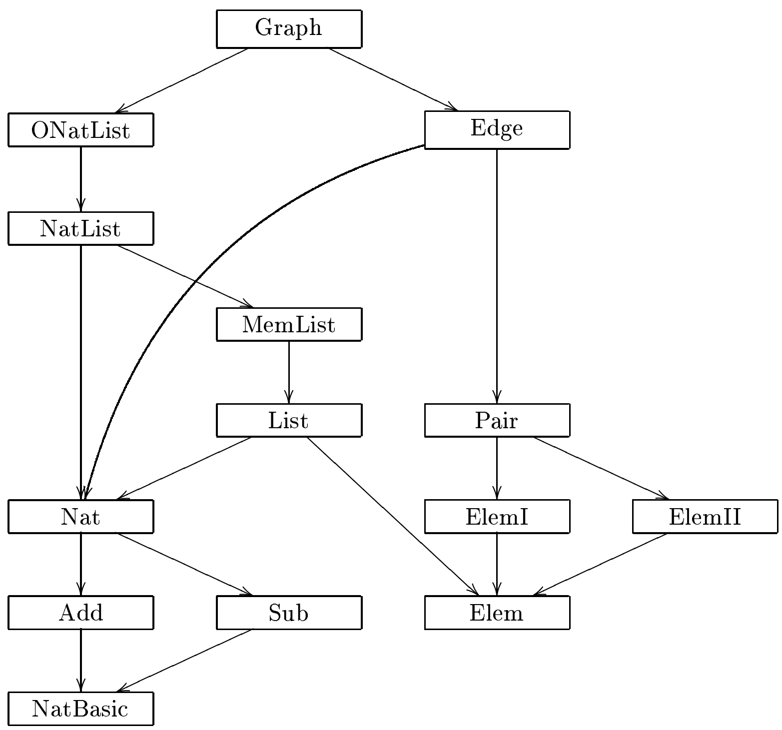

Fig. 1. Specification graph

Graph =
enrich ONatList, Edge **with**
sorts graph;
functions

mkpg	: nat	$\to$ graph	;
$. +_{pe} .$	: graph $\times$ edge	$\to$ graph	;
$. -_{pe} .$	: graph $\times$ edge	$\to$ graph	;
$\#_{p}\ .$	: graph	$\to$ nat	;
$\#_{pe}\ .$	: graph	$\to$ nat	;
psuccs	: graph $\times$ nat	$\to$ natlist	;
$. ++$	: graph	$\to$ graph	;

predicates $. \in_{pg} .$: edge $\times$ graph;
variables pg_2, pg_1, pg: graph; n_4, n_3, n_2, n_1: nat;
axioms
graph **generated by** mkpg, $+_{pe}$;

$$\begin{array}{l} \quad \mathrm{pg}_1 = \mathrm{pg}_2 \\ \leftrightarrow \quad \#_p\ \mathrm{pg}_1 = \#_p\ \mathrm{pg}_2 \\ \quad \wedge\ (\forall\ \mathrm{m, n.} \quad \mathrm{m} < \#_p\ \mathrm{pg}_1 \wedge \mathrm{n} < \#_p\ \mathrm{pg}_1 \\ \qquad\qquad \to (\mathrm{m} => \mathrm{n} \in_{pg} \mathrm{pg}_1 \leftrightarrow \mathrm{m} => \mathrm{n} \in_{pg} \mathrm{pg}_2)), \end{array}$$

$\#_p$ mkpg(n) = n, $\#_p(\mathrm{pg} +_{pe} \mathrm{pe}) = \#_p\ \mathrm{pg}$, $\#_p(\mathrm{pg} -_{pe} \mathrm{pe}) = \#_p\ \mathrm{pg}$,
$\#_p\ \mathrm{pg} ++ = (\#_p\ \mathrm{pg}) +1$,
$\neg\ \mathrm{pe} \in_{pg} \mathrm{mkpg(n)}$,
$\neg\ \mathrm{n}_1 < \#_p\ \mathrm{pg} \vee \neg\ \mathrm{n}_2 < \#_p\ \mathrm{pg} \to \mathrm{pg} +_{pe} \mathrm{n}_1 => \mathrm{n}_2 = \mathrm{pg}$,
$\neg\ \mathrm{n}_1 < \#_p\ \mathrm{pg} \vee \neg\ \mathrm{n}_2 < \#_p\ \mathrm{pg} \to \mathrm{pg} -_{pe} \mathrm{n}_1 => \mathrm{n}_2 = \mathrm{pg}$,
$\mathrm{n}_1 => \mathrm{n}_2 \in_{pg} \mathrm{pg} ++ \leftrightarrow \mathrm{n}_1 => \mathrm{n}_2 \in_{pg} \mathrm{pg}$,

$$\begin{array}{l} \quad \mathrm{n}_1 < \#_p\ \mathrm{pg} \wedge \mathrm{n}_2 < \#_p\ \mathrm{pg} \\ \to (\quad \mathrm{n}_3 => \mathrm{n}_4 \in_{pg} \mathrm{pg} +_{pe} \mathrm{n}_1 => \mathrm{n}_2 \\ \quad \leftrightarrow \mathrm{n}_3 => \mathrm{n}_4 = \mathrm{n}_1 => \mathrm{n}_2 \vee \mathrm{n}_3 => \mathrm{n}_4 \in_{pg} \mathrm{pg}), \end{array}$$

$$\begin{array}{l} \quad \mathrm{n}_1 < \#_p\ \mathrm{pg} \wedge \mathrm{n}_2 < \#_p\ \mathrm{pg} \\ \to (\quad \mathrm{n}_3 => \mathrm{n}_4 \in_{pg} \mathrm{pg} -_{pe} \mathrm{n}_1 => \mathrm{n}_2 \\ \quad \leftrightarrow \mathrm{n}_3 => \mathrm{n}_4 \neq \mathrm{n}_1 => \mathrm{n}_2 \wedge \mathrm{n}_3 => \mathrm{n}_4 \in_{pg} \mathrm{pg}), \end{array}$$

$\#_{pe}$ mkpg(n) = 0,

$$\begin{array}{l} \quad \mathrm{n}_1 < \#_p\ \mathrm{pg} \wedge \mathrm{n}_2 < \#_p\ \mathrm{pg} \wedge \neg\ \mathrm{n}_1 => \mathrm{n}_2 \in_{pg} \mathrm{pg} \\ \to \#_{pe}(\mathrm{pg} +_{pe} \mathrm{n}_1 => \mathrm{n}_2) = (\#_{pe}\ \mathrm{pg}) +1, \end{array}$$

$$\begin{array}{l} \quad \mathrm{n}_1 < \#_p\ \mathrm{pg} \wedge \mathrm{n}_2 < \#_p\ \mathrm{pg} \wedge \mathrm{n}_1 => \mathrm{n}_2 \in_{pg} \mathrm{pg} \\ \to \#_{pe}(\mathrm{pg} -_{pe} \mathrm{n}_1 => \mathrm{n}_2) = (\#_{pe}\ \mathrm{pg}) -1, \end{array}$$

n inn psuccs(pg, m) $\leftrightarrow$ m => n $\in_{pg}$ pg, ordered(psuccs(pg, m))
end enrich

Fig. 2. Toplevel Specification of Directed Graphs

ElemI =
rename Elem **by morphism**
elem → elem'
end rename

ElemII =
rename Elem **by morphism**
elem → elem"
end rename

Pair =
generic data specification
parameter ElemI + ElemII
pair = mkp(. .p1 : elem', . .p2 : elem");
variables p: pair;
end generic data specification

Edge =
actualize Pair **with** Natbasic
bymorphism
elem' → nat, elem" → nat,
pair → edge, mkp → =>,
.p1 → .pe1, .p2 → .pe2,
p → pe
end actualize

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;
end specification

List =
generic data specification
parameter Elem **using** Nat
list = nil | . $+_l$. (car : elem, cdr : list);
variables l: list;
size functions $\#_l$. : list → nat ;
order predicates . ≪ . : list × list;
end generic data specification

MemList =
enrich List **with**
functions
last : list → elem ;
. until . : list × elem → list ;
predicates . in . : elem × list;
variables ele_1: elem;
axioms
nil until ele = nil,
(ele $+_l$ l) until ele = ele $+_l$ nil,
ele ≠ ele_1
→ (ele_1 $+_l$ l) until ele = ele_1 $+_l$ l until ele,
last(ele $+_l$ nil) = ele,
last(ele $+_l$ ele_1 $+_l$ l) = last(ele_1 $+_l$ l),
¬ ele in nil,
ele in ele_1 $+_l$ l ↔ ele = ele_1 ∨ ele in l
end enrich

NatList =
actualize MemList **with** Nat
bymorphism
elem → nat, list → natlist,
nil → nnil, $+_l$ → $+_n$,
car → ncar, cdr → ncdr,
$\#_l$ → $\#_n$, ≪ → $\ll_n$,
last → nlast, until → nuntil,
in → inn, l → nl
end actualize

ONatList =
enrich NatList **with**
predicates ordered : natlist;
axioms
ordered(nnil),
ordered(n $+_n$ nnil),
ordered(m $+_n$ n $+_n$ nl)
↔ m < n ∧ ordered(n $+_n$ nl)
end enrich

Fig. 3. Subspecifications of the Specification of Directed Graphs

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:	$n_0 < n +1 \leftrightarrow n_0 = n \vee n_0 < n$
genax-4:	$m = 0 \vee \exists\, m_0.\ m = m_0 +1$

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: $n \neq 0 \to m + n -1 = (m + n)\text{-}1$
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: $k < m \wedge \neg\ n < n_0 \to k + n_0 < m + n$
lem-15: $\neg\ m + n \neq 0 \leftrightarrow m = 0 \wedge n = 0$
lem-16: $k \neq 0 \to (\neg\ (k + m)\text{-}1 < n \leftrightarrow n < k + m)$
lem-17: $m \neq 0 \to (\neg\ (k + m)\text{-}1 < n \leftrightarrow n < k + m)$
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: $m - n +1 = (m - n) -1$

Lemmas:

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

3.4 Lemmas from Nat

Lemmas:

elim:	$\neg\, m < n \to k = m - n \leftrightarrow m = k + n$
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:	$\neg\, n < n_1 \to (n - n_1) + m = (n + m) - n_1$
lem-12:	$m < n \to (n - m) -1 + m = n -1$
lem-18:	$\neg\, n < m \to (n - m) + m = n$
lem-19:	$\neg\, n < m \to m + n - m = n$
lem-20:	$n_1 < n \to (n - n_1) + m = (n + m) - n_1$
lem-28:	$\neg\, k < m \to (\neg\, k - m < n \leftrightarrow \neg\, k < m + n)$
lem-29:	$\neg\, k < m \to (k - m < n \leftrightarrow k < m + n)$
lem-31:	$\neg\, m < n_1 \to (\neg\, m - n_1 < n \leftrightarrow \neg\, m < n + n_1)$
lem-32:	$\neg\, m < n_1 \to (m - n_1 < n \leftrightarrow m < n + n_1)$
lem-33:	$\neg\, n < n_1 \to (\neg\, m < n - n_1 \leftrightarrow \neg\, m + n_1 < n)$
lem-34:	$n_1 < n \to (\neg\, m < n - n_1 \leftrightarrow \neg\, m + n_1 < n)$
lem-35:	$\neg\, n < n_1 \to (m < n - n_1 \leftrightarrow m + n_1 < n)$
lem-36:	$n_1 < n \to (m < n - n_1 \leftrightarrow m + n_1 < n)$

3.5 Axioms and Lemmas from Pair (Edge Instances)

Axioms:

ax-1:	$(n_0 \Rightarrow n).pe1 = n_0$
ax-2:	$(n \Rightarrow n_0).pe2 = n_0$
ax-3:	$n \Rightarrow n_1 = n_0 \Rightarrow n_2 \leftrightarrow n = n_0 \wedge n_1 = n_2$
ax-4:	$pe.pe1 \Rightarrow pe.pe2 = pe$
genax-3:	$\exists\, m, m_0.\ pe = m \Rightarrow m_0$

Lemmas:

elim-pair	$n = pe.pe1 \wedge n_0 = pe.pe2 \leftrightarrow pe = n \Rightarrow n_0$
lem-1:	$pe = pe.pe1 \Rightarrow n \leftrightarrow pe.pe2 = n$
lem-2:	$pe = n \Rightarrow pe.pe2 \leftrightarrow pe.pe1 = n$
lem-3:	$n_0 \Rightarrow n = n_1 \Rightarrow n \leftrightarrow n_0 = n_1$
lem-4:	$n \Rightarrow n_0 = n \Rightarrow n_1 \leftrightarrow n_0 = n_1$

3.6 Axioms and Lemmas from List (NatList Instances)

Axioms:

ax-01:	$\#_n\ nnil = 0$
ax-02:	$\#_n(n +_n nl) = (\#_n\ nl)+1$
ax-1:	$ncar(n +_n nl) = n$
ax-2:	$ncdr(n +_n nl) = nl$
ax-3:	$n +_n nl = n_0 +_n nl_0 \leftrightarrow n = n_0 \wedge nl = nl_0$

ax-4: $\text{nnil} \neq \text{n} +_n \text{nl}$
ax-5: $\text{nl} = \text{nnil} \vee \text{nl} = \text{ncar(nl)} +_n \text{ncdr(nl)}$
ax-6: $\neg\ \text{nl} \ll_n \text{nl}$
ax-7: $\text{nl}_0 \ll_n \text{nl} \wedge \text{nl} \ll_n \text{nl}_1 \rightarrow \text{nl}_0 \ll_n \text{nl}_1$
ax-8: $\neg\ \text{nl} \ll_n \text{nnil}$
ax-9: $\text{nl} \ll_n \text{n} +_n \text{nl}_0 \leftrightarrow \text{nl} = \text{nl}_0 \vee \text{nl} \ll_n \text{nl}_0$
genax-2: $\text{nl}_1 = \text{nnil} \vee \exists\ \text{m, nl.}\ \text{nl}_1 = \text{m} +_n \text{nl}$

Lemmas:
elim-carcdr: $\text{nl} \neq \text{nnil} \rightarrow \text{n} = \text{ncar(nl)} \wedge \text{nl}_0 = \text{ncdr(nl)} \leftrightarrow \text{nl} = \text{n} +_n \text{nl}_0$
lem-01: $\text{ncdr(nl)} \ll_n \text{nl} \leftrightarrow \text{nl} \neq \text{nnil}$
lem-02: $\text{nnil} \ll_n \text{n} +_n \text{nl}$
lem-03: $\text{nl} \neq \text{nnil} \rightarrow \text{ncar(nl)} +_n \text{ncdr(nl)} = \text{nl}$
lem-04: $\text{nl} \neq \text{nnil} \rightarrow (\text{nl} = \text{n} +_n \text{ncdr(nl)} \leftrightarrow \text{ncar(nl)} = \text{n})$
lem-05: $\text{nl} \neq \text{nnil} \rightarrow (\text{nl} = \text{ncar(nl)} +_n \text{nl}_0 \leftrightarrow \text{ncdr(nl)} = \text{nl}_0)$
lem-06: $\text{nl} \neq \text{nnil} \rightarrow (\text{nl} \neq \text{n} +_n \text{ncdr(nl)} \leftrightarrow \text{ncar(nl)} \neq \text{n})$
lem-07: $\text{nl} \neq \text{nnil} \rightarrow (\text{nl} \neq \text{ncar(nl)} +_n \text{nl}_0 \leftrightarrow \text{ncdr(nl)} \neq \text{nl}_0)$
lem-08: $\text{ncdr(nl)} \neq \text{nnil} \rightarrow (\text{nl} = \text{n} +_n \text{nnil} \leftrightarrow \text{false})$
lem-09: $\#_n\ \text{nl} = 0 \leftrightarrow \text{nl} = \text{nnil}$
lem-10: $\text{nl} \neq \text{nnil} \wedge \text{ncdr(nl)} = \text{nnil} \rightarrow \text{ncar(nl)} +_n \text{nnil} = \text{nl}$

3.7 Axioms and Lemmas from MemList (NatList Instances)

Axioms:
ax-01: $\neg\ \text{n inn nnil}$
ax-02: $\text{n}_0\ \text{inn n} +_n \text{nl} \leftrightarrow \text{n}_0 = \text{n} \vee \text{n}_0\ \text{inn nl}$
ax-03: $\text{nlast(n} +_n \text{nnil)} = \text{n}$
ax-04: $\text{nlast(n} +_n \text{n}_0 +_n \text{nl)} = \text{nlast(n}_0 +_n \text{nl)}$
ax-05: $\text{nnil nuntil n} = \text{nnil}$
ax-06: $(\text{n} +_n \text{nl})\ \text{nuntil n} = \text{n} +_n \text{nnil}$
ax-07: $\text{n}_0 \neq \text{n} \rightarrow (\text{n} +_n \text{nl})\ \text{nuntil n}_0 = \text{n} +_n \text{nl nuntil n}_0$

Lemmas:
lem-01: $\text{ncar((n}_0 +_n \text{nl) nuntil n)} = \text{n}_0$
lem-02: $\text{n inn nl} \rightarrow \text{nlast(nl nuntil n)} = \text{n}$
lem-03: $\text{n inn nl} \wedge \text{nl} \ll_n \text{nl}_0 \rightarrow \text{n inn nl}_0$
lem-04: $(\text{n} +_n \text{nl})\ \text{nuntil n}_0 \neq \text{nnil}$
lem-05: $\text{nl} \neq \text{nnil} \rightarrow \text{nlast(nl) inn nl}$
lem-06: $\text{nl} \neq \text{nnil} \wedge \text{n inn ncdr(nl)} \rightarrow \text{n inn nl}$
lem-07: $\text{nl} \neq \text{nnil} \rightarrow \text{ncar(nl) inn nl}$
lem-08: $\text{n inn n} +_n \text{nl}$
lem-09: $\text{nl} \neq \text{nnil} \rightarrow \text{nlast(n} +_n \text{nl)} = \text{nlast(nl)}$

3.8 Axioms and Lemmas from ONatList

Axioms:

ax-01: ordered(nnil)

ax-02: ordered(n $+_n$ nnil)

ax-03: ordered(m $+_n$ n $+_n$ nl) $\leftrightarrow$ m $<$ n $\land$ ordered(n $+_n$ nl)

Lemmas:

ext: ordered(nl_1) $\land$ ordered(nl_2)
$\rightarrow$ (nl_1 = nl_2 $\leftrightarrow$ ($\forall$ n.n inn nl_1 $\leftrightarrow$ n inn nl_2))

lem-01: ordered(n $+_n$ nl) $\rightarrow$ ordered(nl)

lem-02: ordered(n $+_n$ nl) $\rightarrow$ $\neg$ n inn nl

lem-03: ordered(n $+_n$ nl) $\land$ n_0 $<$ n $\rightarrow$ $\neg$ n_0 inn nl

lem-04: ordered(nl) $\land$ nlast(nl) $<$ k $\rightarrow$ $\neg$ k inn nl

lem-05: ordered(n $+_n$ nl) $\rightarrow$ $\neg$ nlast(n $+_n$ nl) $<$ n

lem-06: nl $\neq$ nnil $\land$ ordered(nl) $\rightarrow$ ordered(ncdr(nl))

lem-07: nl $\neq$ nnil $\land$ ordered(nl) $\rightarrow$ $\neg$ ncar(nl) inn ncdr(nl)

lem-08: ordered(nl) $\rightarrow$ (ordered(n $+_n$ nl) $\leftrightarrow$ nl = nnil $\lor$ n $<$ ncar(nl))

3.9 The Axioms from Graph

Axioms:

ax-01: pg_1 = pg_2
$\leftrightarrow$ $\#_p$ pg_1 = $\#_p$ pg_2
$\land$ ($\forall$ m, n. m $<$ $\#_p$ pg_1 $\land$ n $<$ $\#_p$ pg_1
$\rightarrow$ (m => n $\in_{pg}$ pg_1 $\leftrightarrow$ m => n $\in_{pg}$ pg_2))

ax-02: $\#_p$ mkpg(n) = n

ax-03: $\#_p$(pg $+_{pe}$ pe) = $\#_p$ pg

ax-04: $\#_p$(pg $-_{pe}$ pe) = $\#_p$ pg

ax-05: $\neg$ pe $\in_{pg}$ mkpg(n)

ax-06: $\neg$ n_1 $<$ $\#_p$ pg $\lor$ $\neg$ n_2 $<$ $\#_p$ pg $\rightarrow$ pg $+_{pe}$ n_1 => n_2 = pg

ax-07: $\neg$ n_1 $<$ $\#_p$ pg $\lor$ $\neg$ n_2 $<$ $\#_p$ pg $\rightarrow$ pg $-_{pe}$ n_1 => n_2 = pg

ax-08: n_1 $<$ $\#_p$ pg $\land$ n_2 $<$ $\#_p$ pg
$\rightarrow$ (n_3 => n_4 $\in_{pg}$ pg $+_{pe}$ n_1 => n_2
$\leftrightarrow$ n_3 => n_4 = n_1 => n_2 $\lor$ n_3 => n_4 $\in_{pg}$ pg)

ax-09: n_1 $<$ $\#_p$ pg $\land$ n_2 $<$ $\#_p$ pg
$\rightarrow$ (n_3 => n_4 $\in_{pg}$ pg $-_{pe}$ n_1 => n_2
$\leftrightarrow$ n_3 => n_4 $\neq$ n_1 => n_2 $\land$ n_3 => n_4 $\in_{pg}$ pg)

ax-10: n inn psuccs(pg, m) $\leftrightarrow$ m => n $\in_{pg}$ pg

ax-11: ordered(psuccs(pg, m))

ax-12: $\#_p$ pg ++ = ($\#_p$ pg)+1

ax-13: n_1 => n_2 $\in_{pg}$ pg ++ $\leftrightarrow$ n_1 => n_2 $\in_{pg}$ pg

ax-14: $\#_{pe}$ mkpg(n) = 0

ax-15: n_1 $<$ $\#_p$ pg $\land$ n_2 $<$ $\#_p$ pg $\land$ $\neg$ n_1 => n_2 $\in_{pg}$ pg
$\rightarrow$ $\#_{pe}$(pg $+_{pe}$ n_1 => n_2) = ($\#_{pe}$ pg)+1

ax-16: $n_1 < \#_p\ pg \wedge n_2 < \#_p\ pg \wedge n_1 \Rightarrow n_2 \in_{pg} pg$
$\rightarrow \#_{pe}(pg -_{pe} n_1 \Rightarrow n_2) = (\#_{pe}\ pg) - 1$

genax-1: $\exists\ m.\ pg = mkpg(m) \vee \exists\ pe, pg_0.\ pg = pg_0 +_{pe} pe$

4 The Theorems

th-1: $\neg\ n_1 < \#_p\ pg \rightarrow pg +_{pe} n_1 \Rightarrow n_2 = pg$

th-2: $\neg\ n_2 < \#_p\ pg \rightarrow pg +_{pe} n_1 \Rightarrow n_2 = pg$

th-3: $\neg\ n_1 < \#_p\ pg \rightarrow \neg\ n_1 \Rightarrow n_2 \in_{pg} pg$

th-4: $\neg\ n_2 < \#_p\ pg \rightarrow \neg\ n_1 \Rightarrow n_2 \in_{pg} pg$

th-5: $m \Rightarrow n \in_{pg} pg \rightarrow m < \#_p\ pg$

th-6: $m \Rightarrow n \in_{pg} pg \rightarrow n < \#_p\ pg$

th-7: $n_1 \Rightarrow n_2 \in_{pg} pg +_{pe} n_1 \Rightarrow n_2 \leftrightarrow n_1 < \#_p\ pg \wedge n_2 < \#_p\ pg$

th-8: $\neg\ n_1 < \#_p\ pg \rightarrow pg -_{pe} n_1 \Rightarrow n_2 = pg$

th-9: $\neg\ n_2 < \#_p\ pg \rightarrow pg -_{pe} n_1 \Rightarrow n_2 = pg$

th-10: $\neg\ n_3 \Rightarrow n_4 \in_{pg} pg$
$\rightarrow (\quad n_3 \Rightarrow n_4 \in_{pg} pg +_{pe} n_1 \Rightarrow n_2$
$\leftrightarrow n_1 = n_3 \wedge n_2 = n_4 \wedge n_1 < \#_p\ pg \wedge n_2 < \#_p\ pg)$

th-11: $n_3 \Rightarrow n_4 \in_{pg} pg \rightarrow n_3 \Rightarrow n_4 \in_{pg} pg +_{pe} n_1 \Rightarrow n_2$

th-12: $n_1 \neq n_3 \rightarrow (n_3 \Rightarrow n_4 \in_{pg} pg +_{pe} n_1 \Rightarrow n_2 \leftrightarrow n_3 \Rightarrow n_4 \in_{pg} pg)$

th-13: $n_2 \neq n_4 \rightarrow (n_3 \Rightarrow n_4 \in_{pg} pg +_{pe} n_1 \Rightarrow n_2 \leftrightarrow n_3 \Rightarrow n_4 \in_{pg} pg)$

th-14: $n_1 \Rightarrow n_2 \in_{pg} pg +_{pe} n_1 \Rightarrow n_2$
$\leftrightarrow \neg\ (\neg\ n_1 < \#_p\ pg \vee \neg\ n_2 < \#_p\ pg)$

th-15: $m \Rightarrow n \in_{pg} pg \rightarrow \neg\ \#_p\ pg < m$

th-16: $\neg\ n_1 < \#_p\ pg \rightarrow \neg\ n_1 \Rightarrow n_2 \in_{pg} pg -_{pe} pe$

th-17: $\neg\ n_1 \Rightarrow n_2 \in_{pg} pg -_{pe} n_1 \Rightarrow n_2$

th-18: $m \Rightarrow n \in_{pg} pg \rightarrow pg +_{pe} m \Rightarrow n = pg$

th-19: $n \neq n_1 \rightarrow psuccs(pg +_{pe} n_1 \Rightarrow n_2, n) = psuccs(pg, n)$

th-20: $n_1 < \#_p\ pg \wedge n_2 < \#_p\ pg$
$\rightarrow (pg +_{pe} n_1 \Rightarrow n_2)++ = pg ++ +_{pe} n_1 \Rightarrow n_2$

th-21: $\neg\ \#_p\ pg \Rightarrow n \in_{pg} pg$

th-22: $\neg\ m \Rightarrow \#_p\ pg \in_{pg} pg$

th-23: $\neg\ n_1 < \#_p\ pg \rightarrow \neg\ n_1 \Rightarrow n_2 \in_{pg} pg +_{pe} pe$

th-24: $\neg\ n_2 < \#_p\ pg \rightarrow \neg\ n_1 \Rightarrow n_2 \in_{pg} pg +_{pe} pe$

th-25: $\neg\ n_2 < \#_p\ pg \rightarrow \neg\ n_1 \Rightarrow n_2 \in_{pg} pg -_{pe} pe$

th-26: $n_1 \neq n_3 \rightarrow (n_3 \Rightarrow n_4 \in_{pg} pg -_{pe} n_1 \Rightarrow n_2 \leftrightarrow n_3 \Rightarrow n_4 \in_{pg} pg)$

th-27: $n_1 \Rightarrow n_3 \in_{pg} pg -_{pe} n_1 \Rightarrow n_2 \leftrightarrow n_1 \Rightarrow n_3 \in_{pg} pg \wedge n_2 \neq n_3$

th-28: $\neg\ n < \#_p\ pg \rightarrow psuccs(pg, n) = nnil$

th-29: $m \Rightarrow n \in_{pg} pg \rightarrow \#_{pe}\ pg \neq 0$

th-30: $mkpg(n)++ = mkpg(n + 1)$

th-31: $m < \#_p\ pg \wedge n < \#_p\ pg \rightarrow mkpg(k) \neq pg +_{pe} m \Rightarrow n$

th-32: $n < n_1 \rightarrow psuccs(pg +_{pe} n_1 \Rightarrow n_2, n) = psuccs(pg, n)$

th-33: $n_1 < n \rightarrow psuccs(pg +_{pe} n_1 \Rightarrow n_2, n) = psuccs(pg, n)$

th-34: $psuccs(mkpg(m), n) = nnil$

th-35: $\#_{pe}$ pg ++ = $\#_{pe}$ pg

th-36: m => n $\in_{pg}$ pg $\rightarrow \neg$ $\#_{p}$ pg < n

th-37: psuccs(pg_2 ++, $\#_{p}$ pg_2) = nnil

th-38: $\neg\neg$ n_1 => n_3 $\in_{pg}$ pg $+_{pe}$ n_1 => n_2
$\leftrightarrow \neg\neg$ (n_1 => n_3 $\in_{pg}$ pg $\wedge$ $n_2 \neq n_3$
$\vee$ $n_2 = n_3$ $\wedge$ $n_1 <$ $\#_{p}$ pg $\wedge$ $n_3 <$ $\#_{p}$ pg)

th-39: $\neg\neg$ n_1 => n_3 $\in_{pg}$ pg $+_{pe}$ n_2 => n_3
$\leftrightarrow \neg\neg$ (n_1 => n_3 $\in_{pg}$ pg $\wedge$ $n_1 \neq n_2$
$\vee$ $n_1 = n_2$ $\wedge$ $n_1 <$ $\#_{p}$ pg $\wedge$ $n_3 <$ $\#_{p}$ pg)

th-40: $n_2 \neq n_4$ $\rightarrow$ (n_3 => n_4 $\in_{pg}$ pg $-_{pe}$ n_1 => n_2 $\leftrightarrow$ n_3 => n_4 $\in_{pg}$ pg)

th-41: n_1 => n_3 $\in_{pg}$ pg $-_{pe}$ n_2 => n_3 $\leftrightarrow$ n_1 => n_3 $\in_{pg}$ pg $\wedge$ $n_1 \neq n_2$

th-42: psuccs(pg, $\#_{p}$ pg) = nnil

th-43: $\neg$ n => ($\#_{p}$ pg)+1 $\in_{pg}$ pg

th-44: m = $\#_{p}$ pg $\rightarrow \neg$ n => m $\in_{pg}$ pg

th-45: m = ($\#_{p}$ pg)+1 $\rightarrow \neg$ n => m $\in_{pg}$ pg

th-46: $\neg$ ($\#_{p}$ pg)+1 => n $\in_{pg}$ pg

th-47: n = $\#_{p}$ pg $\rightarrow \neg$ n => m $\in_{pg}$ pg

th-48: n = ($\#_{p}$ pg)+1 $\rightarrow \neg$ n => m $\in_{pg}$ pg

th-49: pg $\neq$ mkpg($\#_{p}$ pg)
$\leftrightarrow$ ($\exists$ m, n.m < $\#_{p}$ pg $\wedge$ n < $\#_{p}$ pg $\wedge$ m => n $\in_{pg}$ pg)

th-50: $\#_{pe}$ pg = 0 $\leftrightarrow$ pg = mkpg($\#_{p}$ pg)

th-51: psuccs(pg, m) = nnil $\rightarrow \neg$ m => n $\in_{pg}$ pg

th-52: m => n $\in_{pg}$ pg $\rightarrow$ (pg $-_{pe}$ m => n) $+_{pe}$ m => n = pg

th-53: m => n $\in_{pg}$ pg $\rightarrow$ $\#_{pe}$(pg $-_{pe}$ m => n) = ($\#_{pe}$ pg) −1

th-54: (pg $+_{pe}$ n_1 => n_2) $+_{pe}$ n_1 => n_2 = pg $+_{pe}$ n_1 => n_2

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 54 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 111 lemmas of *NatBasic*,*Sub*,*Nat*, *List*, ... (together 165) and the discipline to add all previously proved test examples to the theory, the success rate of automated theorem provers is much better (since proof lengths become much shorter, and the number of proofs which require induction decreases drastically).

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 theorem th-n uses another theorem th-m as a lemma, then $m < n$).

The sequential test discipline results in three input files for each of the 54 theorems, one in DFG-Syntax, one in Setheo-Syntax and one in Otter-Syntax. The file for th-n contains $165+n-1$ axioms.

5.2 Input Syntax

Although DFG-, Otter- and Setheo-Syntax differ, a common translation for symbols was used. 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 a few 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
natlist	natlist	nlast	nlast	psuccs	psuccs	nl	nl
nat	nat	nuntil	nuntil	++	jaddjadd	k	k
edge	primedge	$+1$	jsuc	<=	jle	n	n
graph	primgraph	-1	jpre	$>$	jgr	n_0	n0
nnil	nnil	=>	jeqjeqjgr	$\ll_n$	jlsjlsn	pe	pe
0	jzer	.pe1	jdotpe1	inn	inn	pg	pg
$-$	jsub	.pe2	jdotpe2	ordered	ordered	pg_1	pg1
$+$	jadd	mkpg	mkpg	$<$	jls	pg_2	pg2
+n	jaddn	$+_{pe}$	jaddpe	$\in$pg	jinpg	n_1	n1
ncar	ncar	$-_{pe}$	jsubpe	m	m	n_2	n2
ncdr	ncdr	$\#_p$	jsizp	nl_1	nl1	n_3	n3
$\#_n$	jsizn	$\#_{pe}$	jsizpe	nl_0	nl0	n_4	n4

5.3 The Input Files

The input files in DFG-syntax are given as a file graph-DFG.tar.gz. Unzipping and untaring them (use either ‘tar -xzf graph-DFG.tar.gz’ if you have the GNU-version of tar,or first ‘gunzip graph-DFG.tar.gz’ then ‘tar -xf graph-DFG.tar’) creates a directory ‘DFG’, which contains files ‘th-1’ ... ‘th-54’ with the goals to prove.

Similarly the files in Otter-Syntax are given as a file graph-Otter.tar.gz. Unpacking this file creates a directory ‘Otter’, with the input files ‘th-1.in’ ... ‘th-54.in’ and a file named ‘settings’.

Unpacking the files in Setheo-Syntax (graph-Setheo.tar.gz) gives a directory ‘Setheo’, with input files th-1.lop ... th-54.lop.

To be suitable for Otter and Setheo, terms t of sort s from KIV have been “functionally encoded” as *s(t)*. For Otter, they have also been partioned into a “set of support” for the theorem to prove (see p. 552 of [WOLB92]) and the rest of the clauses. The file ‘settings’ contains some settings for Otter, which gave good results for some other examples we have already tried (see [SR97]; in particular, these settings performed far better than auto-mode on our examples). If you find better settings, please let us know.

To feed an example into otter, use the command:

cat settings th-1.in | otter > th-1.out

For Setheo, clauses have been generated using a standard algorithm. Equality has been explicitly axiomatised (with relexivity, symmetry, transitivity and congruence axioms). Clauses of the form $\{x \neq t, L_1, \ldots L_1\}$ with $x \notin Vars(t)$ have been optimized to $\{L_1[x \leftarrow t], \ldots L_1[x \leftarrow t]\}$ and tautological clauses have been removed.

5.4 Inductive Theorems

th-3, th-4, th-5, th-6, th-29, th-35, th-49 and th-50 were proved in KIV using induction. For these 8 theorems a noninductive proof *may or may not* exist (the use of induction in KIV might have been unnecessary). All other 46 theorems are guaranteed to be provable wtihout induction.

References

[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.

[MTH89] R. Milner, M. Tofte, and R. 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/.

[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.

[SR97] G. Schellhorn and W. Reif. Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers. Ulmer Informatik-Berichte 97-12, Universität Ulm, Fakultät für Informatik, 1997.

[WOLB92] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning, Introduction and Applications (2nd ed.).* McGraw Hill, 1992.