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 .................................................. 1
2 The Datatype of Directed Graphs .............................. 2
3 The Axioms ..................................................... 6
   3.1 Axioms and Lemmas from NatBasic ...................... 6
   3.2 Axioms and Lemmas from Add ......................... 6
   3.3 Axioms and Lemmas from Sub ......................... 7
   3.4 Lemmas from Nat ...................................... 8
   3.5 Axioms and Lemmas from Pair (Edge Instances) ...... 8
   3.6 Axioms and Lemmas from List (NatList Instances) ... 8
   3.7 Axioms and Lemmas from MemList (NatList Instances) 9
   3.8 Axioms and Lemmas from ONatList .................... 10
   3.9 The Axioms from Graph ................................ 10
4 The Theorems ................................................ 11
5 The Test Scenario .......................................... 12
   5.1 Sequential Test Discipline ........................... 12
   5.2 Input Syntax ......................................... 13
   5.3 The Input Files ....................................... 13
   5.4 Inductive Theorems .................................. 14

1 Introduction

This paper describes a problem set for automated theorem provers taken from a KIV case study on the implementation 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 "Deduction" ([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 \( \{0, \ldots, n - 1\} \) of the natural numbers. This representation 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 \( \{0, \ldots, n - 1\} \) and no edges can be constructed with \( \text{mkpg}(n) \). For a graph \( pg \) with node set \( \{0, \ldots, n - 1\} \), the new graph \( pg + + \) (where ++ is written postfix) contains one new node (so it has node set \( \{0, \ldots, n\} \)) 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 \( pg + ne n => m \) (\(+ne \) 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 \( pg - ne n => m \) (again \(-ne \) is infix). Membership in the set of graph edges can be checked with \( n => m e ne pg \). \#ne \( pg \) gives the number of edges of a graph, and finally \( \text{psucc}(pg,n) \) 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 terms-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 occurrence 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 → graph;

. +_pe : graph × edge → graph;

. −_pe : graph × edge → graph;

#_p : graph → nat;

#_pe : graph → nat;

psucc : graph × nat → natlist;

++ : graph → graph;

predicates . ∈_pg : edge × graph;

variables pg₁, pg₂ : graph; n₄, n₃, n₂, n₁ : nat;

axioms

generated by mkpg, +_pe;

pg₁ = pg₂

ensitive

→ #_p pg₁ = #_p pg₂

∧ (∀ m, n. m < #_p pg₁ ∧ n < #_p pg₁

→ (m => n ∈ pg₁ ↔ m => n ∈ pg₂))

#_p mkpg(n) = n. #_p (pg +_pe pe) = #_p pg. #_p (pg −_pe pe) = #_p pg

#_p pg ++ = (#_p pg) + 1,

→ pe ∈ pg mkpg(n),

→ n₁ < #_p pg ∨ ¬ n₂ < #_p pg → pg +_pe n₁ => n₂ = pg

→ n₁ < #_p pg ∨ ¬ n₂ < #_p pg → pg −_pe n₁ => n₂ = pg

n₁ => n₂ ∈ pg pg ++ ↔ n₁ => n₂ ∈ pg

n₁ < #_p pg ∧ n₂ < #_p pg

→ ( n₃ => n₄ ∈ pg pg +_pe n₁ => n₂

↔ n₃ => n₄ = n₁ => n₂ ∨ n₃ => n₄ ∈ pg pg).

n₁ < #_p pg ∧ n₂ < #_p pg

→ ( n₃ => n₄ ∈ pg pg −_pe n₁ => n₂

↔ n₃ => n₄ ≠ n₁ => n₂ ∧ n₃ => n₄ ∈ pg pg).

#_p mkpg(n) = 0,

→ n₁ < #_p pg ∧ n₂ < #_p pg ∧ ¬ n₁ => n₂ ∈ pg pg

→ #_p (pg +_pe n₁ => n₂) = (#_p pg) + 1,

→ n₁ < #_p pg ∧ n₂ < #_p pg ∧ n₁ => n₂ ∈ pg pg

→ #_p (pg −_pe n₁ => n₂) = (#_p pg) − 1.

n in ppsucc(pg, m) ↔ m => n ∈ pg ordered(ppsucc(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  
by morphism  
  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 | . + . (car : elem, cdr : list);  
variables l: list;  
size functions # . : list → nat;  
order predicates . ≤ : list × list;  
end generic data specification  

MemList =  
enrich List with  
functions  
  last : list → elem ;  
  . until : list × elem → list ;  
variables ele1 : elem;  
axioms  
  nil until ele = nil,  
  (ele + t , l ) until ele = ele + t , nil,  
  ele ≠ ele1  
  → (ele1 + t , l ) until ele = ele1 + t until ele,  
  last(ele + t , nil) = ele,  
  last(ele + t , ele1 + t , l ) = last(ele1 + t , l ),  
  → ele in nil,  
  ele in ele1 + t l ↔ ele = ele1 ∨ ele in l  
end enrich  

NatList =  
enrich MemList with Nat  
by morphism  
  elem → nat, list → natlist,  
  nil → nil, + t → + n ,  
  car → ncar, cdr → ncdr,  
  # , → # n , ≤ → ≤ n ,  
  last → nilast, until → nuntil,  
  in → inn, l → nl  
end actualize  

ONatList =  
enrich NatList with  
functions  
  ordered : natlist;  
axioms  
  ordered(nil),  
  ordered(u + n nil),  
  ordered(m + n u + n nil)  
  ↔ m < n ∧ ordered(u + n nil)  
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: \( n = 0 \lor n = n -1 +1 \)
ax-5: \( \neg n < n \)
ax-6: \( n < n_0 \land n_0 < n_1 \rightarrow n < n_1 \)
ax-7: \( \neg n < 0 \)
ax-8: \( n_0 < n +1 \leftrightarrow n_0 = n \lor n_0 < n \)
genax-4: \( m = 0 \lor \exists m_0. m = m_0 +1 \)

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: \( m +1 < n \leftrightarrow m < n \land n \neq m +1 \)
lem-08: \( n -1 = n \rightarrow n = 0 \)
lem-09: \( n < n -1 \rightarrow n = 0 \)
lem-10: \( \neg 0 +1 < n \leftrightarrow n = 0 \lor n = 0 +1 \)
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: \( m -1 < n \rightarrow \neg n < m \land n \neq 0 \)
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: \( n < n_0 \lor n = n_0 \lor n_0 < n \)

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: \( m + n = (m + k) + 1 \iff n = k + 1 \)
lem-04: \( m + k < n + k \iff m < n \)
lem-05: \( m + n = m + k \iff n = k \)
lem-06: \( m \neq (m + k) + 1 \)
lem-07: \( n \neq 0 \iff m + n - 1 = (m + n) - 1 \)
lem-08: \( m + n = (m + k) + 1 \iff n = k + 1 + 1 \)
lem-09: \( \neg m + n < m \)
lem-10: \( m + n = n + 1 \iff m = 0 + 1 \)
lem-11: \( m + n = m \iff n = 0 \)
lem-12: \( m < n + m \iff n \neq 0 \)
lem-13: \( k < m \vee \neg n < n_0 \iff k + n_0 < m + n \)
lem-14: \( \neg m + n \neq 0 \iff m = 0 \land n = 0 \)
lem-15: \( k \neq 0 \iff (k + m) - 1 < n \iff n < k + m \)
lem-16: \( m \neq 0 \iff (k + m) - 1 < n \iff n < k + m \)
lem-17: \( k + n = (k + m) + 1 \iff n = m + 1 \)

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: \( m < n \iff n - n - m = m \)
lem-08: \( \neg m < n \iff n - n - m = m \)
lem-10: \( n < m \land n \neq 0 \iff m - n - 1 = (m - n) + 1 \)
lem-11: \( \neg m < n \land n \neq 0 \iff m - n - 1 = (m - n) + 1 \)
lem-13: \( m < n \iff n + 1 - m = (n - m) + 1 \)
lem-14: \( \neg m < n \iff n + 1 - m = (n - m) + 1 \)
lem-15: \( \neg m < n \iff n + 1 - n - m = m + 1 \)
lem-16: \( m < n \iff n + 1 - n - m = m + 1 \)
lem-17: \( m < n \iff n + 1 - n - m - m = m + 1 + 1 \)
lem-21: \( n < m \land k < m \iff (m - n < m - k \iff k < n) \)
lem-22: \( n < m \land \neg m < k \iff (m - n < m - k \iff k < n) \)
lem-23: \( \neg m < n \land k < m \iff (m - n < m - k \iff k < n) \)
lem-24: \( \neg m < n \land \neg m < k \iff (m - n < m - k \iff k < n) \)
lem-25: \( n < m \land k < m \iff (\neg m - n < m - k \iff k < n) \)
lem-26: \( n < m \land \neg m < k \iff (\neg m - n < m - k \iff k < n) \)
lem-27: \( \neg m < n \land k < m \iff (\neg m - n < m - k \iff \neg k < n) \)
lem-28: \( n < m \land \neg m < k \iff (\neg m - n < m - k \iff \neg k < n) \)
lem-30: \( n < m \iff n - m \iff n < m \)
lem-37: \( n < m \iff n - m \iff n < m \)
lem-38: \( n - m = 0 \iff \neg m < n \)
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: \(m < n \rightarrow (n - m) - 1 + m = n - 1\)
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: \(n_1 < n \rightarrow (m < n - n_1 \leftrightarrow m + n_1 < n)\)
lem-36: \(n_1 < n \rightarrow (m < n - n_1 \leftrightarrow m + n_1 < n)\)

3.5 Axioms and Lemmas from Pair (Edge Instances)

Axioms:

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

Lemmas:

elim-pair \(n = pe . pe1 \land n_0 = pe . pe2 \leftrightarrow pe = n => n_0\)
lem-1: \(pe = pe . pe1 => n \leftrightarrow pe . pe2 = n\)
lem-2: \(pe = n => pe . pe2 \leftrightarrow pe . pe1 = n\)
lem-3: \(n_0 => n = n_1 => n \leftrightarrow n_0 = n_1\)
lem-4: \(n => n_0 = n => n_1 \leftrightarrow n_0 = n_1\)

3.6 Axioms and Lemmas from List (NatList Instances)

Axioms:

ax-01: \(#_n nil = 0\)
ax-02: \(#_n (n +_n nil) = (#_n nil) + 1\)
ax-1: \(\text{near}(n +_n nil) = n\)
ax-2: \(\text{ncdr}(n +_n nil) = nil\)
ax-3: \(n +_n nil = n_0 +_n nil_0 \leftrightarrow n = n_0 \land nil = nil_0\)
ax-4: \( \text{nnil} \neq n +_n \text{nl} \)
ax-5: \( n\text{l} = \text{nnil} \lor n\text{l} = \text{n}\text{car}(n\text{ll}) +_n \text{ncdr}(n\text{ll}) \)
ax-6: \( \lnot n \ll_n n\text{ll} \)
ax-7: \( n\text{l}_0 \ll_n n\text{l} \land n\text{l} \ll_n n\text{l}_1 \rightarrow n\text{l}_0 \ll_n n\text{l}_1 \)
ax-8: \( \lnot n \ll_n \text{nnil} \)
ax-9: \( n\text{l} \ll_n n +_n n\text{l}_0 \leftrightarrow n\text{l} = n\text{l}_0 \lor n\text{l} \ll_n n\text{l}_0 \)
genax-2: \( n\text{l}_1 = \text{nnil} \lor \exists m, n\text{l} = m +_n n\text{l} \)

Lemmas:
\begin{enumerate}
\item elim-carcdr: \( n\text{l} \neq \text{nnil} \rightarrow n = \text{n}\text{car}(n\text{l}) \land n\text{l}_0 = \text{ncdr}(n\text{l}) \leftrightarrow n = n +_n n\text{l}_0 \)
\item lem-01: \( \text{ncdr}(n\text{l}) \ll_n n\text{l} \leftrightarrow n\text{l} \neq \text{nnil} \)
\item lem-02: \( \text{nnil} \ll_n n +_n n\text{l} \)
\item lem-03: \( n\text{l} \neq \text{nnil} \rightarrow \text{n}\text{car}(n\text{l}) +_n \text{ncdr}(n\text{l}) = \text{nnil} \)
\item lem-04: \( n\text{l} \neq \text{nnil} \rightarrow (n\text{l} = n +_n \text{ncdr}(n\text{l}) \leftrightarrow \text{n}\text{car}(n\text{l}) = n) \)
\item lem-05: \( n\text{l} \neq \text{nnil} \rightarrow (n\text{l} = \text{n}\text{car}(n\text{l}) +_n n\text{l}_0 \leftrightarrow \text{ncdr}(n\text{l}) = n\text{l}_0) \)
\item lem-06: \( n\text{l} \neq \text{nnil} \rightarrow (n\text{l} \neq n +_n \text{ncdr}(n\text{l}) \leftrightarrow \text{n}\text{car}(n\text{l}) \neq n) \)
\item lem-07: \( n\text{l} \neq \text{nnil} \rightarrow (n\text{l} \neq \text{n}\text{car}(n\text{l}) +_n n\text{l}_0 \leftrightarrow \text{ncdr}(n\text{l}) \neq n\text{l}_0) \)
\item lem-08: \( \text{ncdr}(n\text{l}) \neq \text{nnil} \rightarrow (n\text{l} = n +_n \text{nnil} \leftrightarrow \text{false}) \)
\item lem-09: \( \#_n n\text{l} = 0 \leftrightarrow n\text{l} = \text{nnil} \)
\item lem-10: \( n\text{l} \neq \text{nnil} \land \text{ncdr}(n\text{l}) = \text{nnil} \rightarrow \text{n}\text{car}(n\text{l}) +_n \text{nnil} = n\text{l} \)
\end{enumerate}

3.7 Axioms and Lemmas from MemList (NatList Instances)

Axioms:
\begin{enumerate}
\item ax-01: \( \lnot n \text{inn nnnil} \)
\item ax-02: \( n_0 \text{inn n +}_n \text{nl} \leftrightarrow n_0 = n \lor n_0 \text{inn nl} \)
\item ax-03: \( \text{nlast}(n +_n \text{nnil}) = n \)
\item ax-04: \( \text{nlast}(n +_n n_0 +_n n\text{l}) = \text{nlast}(n_0 +_n n\text{l}) \)
\item ax-05: \( \text{nnil until n = nnnil} \)
\item ax-06: \( (n +_n n\text{l}) \text{until n = n +}_n \text{nnil} \)
\item ax-07: \( n_0 \neq n \rightarrow (n +_n n\text{l}) \text{until n}_0 = n +_n n\text{l} \text{until n}_0 \)
\end{enumerate}

Lemmas:
\begin{enumerate}
\item lem-01: \( \text{n}\text{car}((n_0 +_n n\text{l}) \text{until n}) = n_0 \)
\item lem-02: \( n \text{inn nl} \rightarrow \text{nlast(nl until n) = n} \)
\item lem-03: \( n \text{inn nl} \land n \ll_n n\text{l}_0 \rightarrow n \text{inn n}_0 \)
\item lem-04: \( (n +_n n\text{l}) \text{until n}_0 \neq \text{nnil} \)
\item lem-05: \( n\text{l} \neq \text{nnil} \rightarrow \text{nlast(nl) inn nl} \)
\item lem-06: \( n\text{l} \neq \text{nnil} \land n \text{inn ncdr(nl)} \rightarrow n \text{inn nl} \)
\item lem-07: \( n\text{l} \neq \text{nnil} \rightarrow \text{ncar(nl) inn nl} \)
\item lem-08: \( n \text{inn n +}_n n\text{l} \)
\item lem-09: \( n\text{l} \neq \text{nnil} \rightarrow \text{nlast(n +}_n n\text{l}) = \text{nlast(nl)} \)

9
3.8 Axioms and Lemmas from ONatList

Axioms:
ax-01: \( \text{ordered}(\text{null}) \)
ax-02: \( \text{ordered}(n +_n \text{null}) \)
ax-03: \( \text{ordered}(m +_n n +_n n) \leftrightarrow m < n \land \text{ordered}(n +_n n) \)

Lemmas:
\begin{align*}
\text{ext}: & \quad \text{ordered}(n_{l1}) \land \text{ordered}(n_{l2}) \\
& \quad \rightarrow (n_{l1} = n_{l2} \leftrightarrow (\forall n. n \text{ in } l_{1} \leftrightarrow n \text{ in } l_{2}))
\end{align*}

lem-01: \( \text{ordered}(n +_n n) \rightarrow \text{ordered}(n) \)
lem-02: \( \text{ordered}(n +_n n) \rightarrow \neg n \text{ in } n \)
lem-03: \( \text{ordered}(n +_n n) \land n_0 < n \rightarrow \neg n_0 \text{ in } n \)
lem-04: \( \text{ordered}(n) \land \text{nlast}(n) < k \rightarrow \neg k \text{ in } n \)
lem-05: \( \text{ordered}(n +_n n) \rightarrow \neg \text{nlast}(n +_n n) < n \)
lem-06: \( n \neq \text{null} \land \text{ordered}(n) \rightarrow \text{ordered}(\text{ncdr}(n)) \)
lem-07: \( n \neq \text{null} \land \text{ordered}(n) \rightarrow \neg \text{ncar}(n) \land \text{in} \text{ ncdr}(n) \)
lem-08: \( \text{ordered}(n) \rightarrow (\text{ordered}(n +_n n) \leftrightarrow n = \text{null} \lor n < \text{ncar}(n)) \)

3.9 The Axioms from Graph

Axioms:
ax-01: \( p_{g1} = p_{g2} \)
\begin{align*}
\leftrightarrow & \quad \#_p p_{g1} = \#_p p_{g2} \\
& \land (\forall m, n. m < \#_p p_{g1} \land n < \#_p p_{g1} \\
& \rightarrow (m => n \in p_{g1} \leftrightarrow m => n \in p_{g2}))
\end{align*}
ax-02: \( \#_p \text{mkpg}(n) = n \)
ax-03: \( \#_p (p_{g} +_{pe} pe) = \#_p p_{g} \)
ax-04: \( \#_p (p_{g} \rightarrow_{pe} pe) = \#_p p_{g} \)
ax-05: \( \neg pe \in p_{g} \text{mkpg}(n) \)
ax-06: \( \neg n_1 < \#_p p_{g} \lor \neg n_2 < \#_p p_{g} \rightarrow p_{g} +_{pe} n_1 => n_2 = p_{g} \)
ax-07: \( \neg n_1 < \#_p p_{g} \lor \neg n_2 < \#_p p_{g} \rightarrow p_{g} \rightarrow_{pe} n_1 => n_2 = p_{g} \)
ax-08: \( n_1 < \#_p p_{g} \land n_2 < \#_p p_{g} \)
\begin{align*}
& \rightarrow (n_3 => n_4 \in p_{g} +_{pe} n_1 => n_2 \\
& \lor n_3 => n_4 = n_1 => n_2 \lor n_3 => n_4 \in p_{g})
\end{align*}
ax-09: \( n_1 < \#_p p_{g} \land n_2 < \#_p p_{g} \)
\begin{align*}
& \rightarrow (n_3 => n_4 \in p_{g} \rightarrow_{pe} n_1 => n_2 \\
& \lor n_3 => n_4 \neq n_1 => n_2 \land n_3 => n_4 \in p_{g})
\end{align*}
ax-10: \( n \text{ in } \text{psuccs}(p_{g}, m) \leftrightarrow m => n \in p_{g} \text{ pg} \)
ax-11: \( \text{ordered}(\text{psuccs}(p_{g}, m)) \)
ax-12: \( \#_p p_{g} + = (\#_p p_{g}) + 1 \)
ax-13: \( n_1 => n_2 \in p_{g} \text{ pg} + => n_1 => n_2 \in p_{g} \text{ pg} \)
ax-14: \( \#_{pe} \text{mkpg}(n) = 0 \)
ax-15: \( n_1 < \#_p p_{g} \land n_2 < \#_p p_{g} \land \neg n_1 => n_2 \in p_{g} \text{ pg} \)
\begin{align*}
& \rightarrow \#_{pe} (p_{g} +_{pe} n_1 => n_2) = (\#_{pe} p_{g}) + 1
\end{align*}
ax-16: \[ n_1 < \#_p \pg \land n_2 < \#_p \pg \land n_1 \Rightarrow n_2 \in \pg \pg \]
\[ \rightarrow \#_p (\pg \rightarrow n_1 \Rightarrow n_2) = (\#_p \pg) - 1 \]
genax-1: \[ \exists \ m. \ pg = \mkpg(m) \lor \exists \ pe. \ pg \in \pg + \pe \]

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: \[ \neg n_1 < \#_p \pg \rightarrow \neg n_1 = \Rightarrow n_2 \in \pg \]
th-4: \[ \neg n_2 < \#_p \pg \rightarrow \neg n_1 = \Rightarrow n_2 \in \pg \]
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: \[ n_1 = \Rightarrow n_2 \in \pg \pg + \pe \ n_1 \Rightarrow n_2 \leftrightarrow n_1 < \#_p \pg \land n_2 < \#_p \pg \]
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: \[ \neg n_3 = \Rightarrow n_4 \in \pg \pg \]
\[ \leftrightarrow n_1 = n_3 \land n_2 = n_4 \land n_1 < \#_p \pg \land n_2 < \#_p \pg \]
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: \[ n_1 = \Rightarrow n_2 \in \pg \pg + \pe \ n_1 = \Rightarrow n_2 \]
\[ \leftrightarrow \neg (\neg n_1 < \#_p \pg \lor \neg n_2 < \#_p \pg) \]
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: \[ n_1 < \#_p \pg \land n_2 < \#_p \pg \]
\[ \rightarrow (\pg + \pe \ n_1 = \Rightarrow n_2) \neq n \neq \pg + \pe \ n_1 = \Rightarrow n_2 \]
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: \[ n_1 = \Rightarrow n_3 \in \pg \pg + \pe \ n_1 = \Rightarrow n_2 \leftrightarrow n_1 = \Rightarrow n_3 \in \pg \pg \land n_2 \neq n_3 \]
th-28: \[ \neg n < \#_p \pg \rightarrow \psuccs(\pg, n) \neq \nnil \]
th-29: \[ m = \Rightarrow n \in \pg \pg \rightarrow \#_p \pg \neq 0 \]
th-30: \[ \mkpg(n) \neq n = \mkpg(n + 1) \]
th-31: \[ m < \#_p \pg \land n < \#_p \pg \rightarrow \mkpg(k) \neq \pg + \pe \ m = \Rightarrow n \]
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: \( \#_p \text{ pg} + n = \#_p \text{ pg} \)

th-36: \( m \Rightarrow n \in_{pg} \text{ pg} \Rightarrow \neg \#_p \text{ pg} < n \)

th-37: \( \text{psucc}(pg_2 + n, \#_p \text{ pg} + n) = \text{null} \)

th-38: \( \neg n_1 \Rightarrow n_3 \in_{pg} \text{ pg} + _p n_1 \Rightarrow n_2 \\
\leftrightarrow \neg \neg ( n_1 \Rightarrow n_3 \in_{pg} \text{ pg} \land n_2 \neq n_3 \\
\land n_2 = n_3 \land n_1 < \#_p \text{ pg} \land n_3 < \#_p \text{ pg}) \)

th-39: \( \neg n_1 \Rightarrow n_2 \in_{pg} \text{ pg} + _p n_2 \Rightarrow n_2 \\
\leftrightarrow \neg \neg ( n_1 \Rightarrow n_3 \in_{pg} \text{ pg} \land n_1 \neq n_2 \\
\land n_1 = n_2 \land n_1 < \#_p \text{ pg} \land n_2 < \#_p \text{ pg}) \)

th-40: \( n_2 \neq n_4 \Rightarrow (n_3 \Rightarrow n_4 \in_{pg} \text{ pg} + _p n_1 \Rightarrow n_2 \leftrightarrow n_3 \Rightarrow n_4 \in_{pg} \text{ pg}) \)

th-41: \( n_1 \Rightarrow n_3 \in_{pg} \text{ pg} + _p n_2 \Rightarrow n_3 \leftrightarrow n_1 \Rightarrow n_3 \in_{pg} \text{ pg} \land n_1 \neq n_2 \)

th-42: \( \text{psucc}(pg, \#_p \text{ pg}) = \text{null} \)

th-43: \( \neg n \Rightarrow (\#_p \text{ pg}) + 1 \in_{pg} \text{ pg} \)

th-44: \( m = \#_p \text{ pg} \Rightarrow \neg n \Rightarrow m \in_{pg} \text{ pg} \)

th-45: \( m = (\#_p \text{ pg}) + 1 \Rightarrow \neg n \Rightarrow m \in_{pg} \text{ pg} \)

th-46: \( \neg (\#_p \text{ pg}) + 1 \Rightarrow n \in_{pg} \text{ pg} \)

th-47: \( n = \#_p \text{ pg} \Rightarrow \neg n \Rightarrow m \in_{pg} \text{ pg} \)

th-48: \( n = (\#_p \text{ pg}) + 1 \Rightarrow \neg n \Rightarrow m \in_{pg} \text{ pg} \)

th-49: \( \text{pg} \neq \text{mkpg}(\#_p \text{ pg}) \\
\leftrightarrow (\exists m, n, m < \#_p \text{ pg} \land n < \#_p \text{ pg} \land m \Rightarrow n \in_{pg} \text{ pg}) \)

th-50: \( \#_p \text{ pg} = 0 \leftrightarrow \text{pg} = \text{mkpg}(\#_p \text{ pg}) \)

th-51: \( \text{psucc}(pg, m) = \text{null} \Rightarrow \neg m \Rightarrow n \in_{pg} \text{ pg} \)

th-52: \( m \Rightarrow n \in_{pg} \text{ pg} \Rightarrow \text{(pg} + _p m \Rightarrow n) + _p m \Rightarrow n = \text{pg} \)

th-53: \( m \Rightarrow n \in_{pg} \text{ pg} \Rightarrow \#_p (\text{pg} + _p m \Rightarrow n) = (\#_p \text{ pg}) - 1 \)

th-54: \( (\text{pg} + _p n_1 \Rightarrow n_2) + _p n_1 \Rightarrow n_2 = \text{pg} + _p n_1 \Rightarrow n_2 \)

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.

<table>
<thead>
<tr>
<th>here</th>
<th>ASCII</th>
<th>here</th>
<th>ASCII</th>
<th>here</th>
<th>ASCII</th>
<th>here</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>natlist</td>
<td>natlist</td>
<td>nlast</td>
<td>nlast</td>
<td>psuccs</td>
<td>psuccs</td>
<td>nl</td>
<td>nl</td>
</tr>
<tr>
<td>nat</td>
<td>nat</td>
<td>nuntil</td>
<td>nuntil</td>
<td>++</td>
<td>jadd</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>edge</td>
<td>primedge</td>
<td>+1</td>
<td>jsuc</td>
<td>&lt;=</td>
<td>jle</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>graph</td>
<td>primgraph</td>
<td>-1</td>
<td>jpre</td>
<td>&gt;</td>
<td>jgr</td>
<td>n0</td>
<td>n0</td>
</tr>
<tr>
<td>nnil</td>
<td>nnil</td>
<td>=&gt;</td>
<td>jeq</td>
<td>&lt;=n</td>
<td>jlsjnsn</td>
<td>pe</td>
<td>pe</td>
</tr>
<tr>
<td>0 jzer</td>
<td>.pe1</td>
<td>jdotpe1</td>
<td>im</td>
<td>inn</td>
<td>pg</td>
<td>pg</td>
<td></td>
</tr>
<tr>
<td>- jsub</td>
<td>.pe2</td>
<td>jdotpe2</td>
<td>ordered</td>
<td>ordered</td>
<td>pg1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ jadd</td>
<td>mkpg</td>
<td>mkpg</td>
<td>&lt; jls</td>
<td>pg2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+n jadn</td>
<td>+pe</td>
<td>jaddpe</td>
<td>€ pg</td>
<td>jinp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ncar</td>
<td>ncar</td>
<td>-pe</td>
<td>jsubpe</td>
<td>m</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ncdr</td>
<td>ncdr</td>
<td>#p</td>
<td>jsizp</td>
<td>nl1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#n jsizn</td>
<td>#p</td>
<td>jsizpe</td>
<td>nl0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 -xf 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 partitioned 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 reflexivity, symmetry, transitivity and congruence axioms). Clauses of the form \( \{ x \neq t, L_1, \ldots L_l \} \) with \( x \not\in \text{Vars}(t) \) have been optimized to \( \{ L_1[x \leftarrow t], \ldots L_l[x \leftarrow t] \} \) 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 without induction.

### References


