Challenge Problems for Inductive Theorem Provers v1.0[*]

Louise A. Dennis[†] Jeremy Gow[‡] and Carsten Shürmann[§]

May 4, 2007

Abstract

Within the field of inductive theorem proving it is hard to assess claims for the superiority of any given system since there is naturally a tendency to report "successes" – difficult or challenging problems automatically proved. There is also a desire within the community to develop a store of shared knowledge about the challenges that face the automation of proof by mathematical induction.

A group of researchers within the community agreed that they would each put forward a number of "Challenge Problems". These should present interesting challenges to the automation of inductive proof or illustrate important features which an inductive prover should be able to handle.

This technical report represent the current state of this challenge problem set.

Inductive Theorem proving is a small field. The main theorem provers within this field are Nqthm [2] (now re-engineered as ACL2 [6]), INKA [1], the *Clam* series [4, 8] and RRL [5]. Twelf [7] also looks at the automation of inductive proof in the context of logical frameworks. Within the field it is hard to assess claims for the superiority of any given system since there is naturally a tendency to report "successes" – difficult or challenging problems automatically proved. There is also a desire within the community to develop a store of shared knowledge about the challenges that face the automation of proof by mathematical induction.

TPTP (Thousands of Problems for Theorem Provers) [10] is a library of test problems for first-order ATP systems. They provide the ATP community with a comprehensive library complete with unambiguous names and references. All the problems are stated in a standardised formulation of first-order logic and are widely used to benchmark first-order systems. They are also used as the test set for the CASC competition [9] which compares such systems. One of the benefits of the TPTP library to the ATP community is the existence of a common set of problems by which comparisons can be made.

It is not practical for inductive theorem provers to follow the pattern of the TPTP library. Various attempts have been made to build a similar corpus of problems requiring inductive reasoning. The most mature of these was based on the Boyer-Moore [2] corpus[1]. This corpus was unpopular partly because there was repetition within the problem set and partly because many problems depended on a few particular function definitions. But the major objection was that inductive theorem provers use a number of different logics, some of which are typed and some of which are not, which made it difficult to agree on a standard format. The use of other logics also raised translation issues and a fully automated process for converting the theorems, even into an agreed typed language was never produced.

A group of researchers within the community[2] agreed that instead of a large set of benchmarks in a standard logic they would each put forward a number of "Challenge Problems". These should present interesting challenges to the automation of inductive proof or illustrate important features which an inductive prover should be able to handle. A set of these problems would be collected which would remain sufficiently small that an individual could represent them within their own theorem proving system as they saw fit.

This technical report represent the current state of this challenge problem set.

[*] University of Liverpool, Department of Computer Science, Technical Report, ULCS-007-004

[†] Department of Computer Science, The University of Liverpool

[‡] UCL Interaction Centre (UCLIC), University College London

[§] Department of Theoretical Computer Science, IT University of Copenhagen

[1] This has become known as the Dmac corpus after David McAllester who translated a fragment of the NQTHM corpus into a simpler language.

[2] At the 2000 CADE Workshop on the Automation of Proof by Mathematical Induction.

Standard Notation

Logic	
$\top$	True
$\bot$	False
$\neg$	negation
$\vee$	or
$\wedge$	and
$\rightarrow$	implication
Natural Numbers	
s	successor
$+$	addition
$*$	multiplication
x^y	exponentiation
$even$	true if number is even (defined without using mutual recursion with odd)
$\sqrt[n]{x}$	nth root (inverse of exponentiation)
$\frac{x}{y}$	quotient (result of dividing x by y rounded down to nearest natural)
$x \, mod \, y$	the remainder of x divided by y
$<$	less than
$\leq$	less than or equal to
Lists	
$[]$	empty list
$::$	cons
$length(l)$	length of list l
$l_1 <> l_2$	append of lists, l_1 and l_2
$x \in l$	list membership
$hd(l)$	head of list
$tl(l)$	tail of list
$map(f, l)$	a function , f, applied to all elements l
$take(n, l)$	the first n elements of l
$drop(n, l)$	a list with the first n elements removed

The Challenges

1 IWC001a: First Order Version of the Arithmetic/Geometric Mean

1.1 Summary

- Unusual Induction Scheme.
- Needs Lemmas and extra functions.
- Presents challenges to Rippling [3].
- Goal is not equational.

1.2 Definitions

sum (first order)

$$\Sigma^0 L = 0 \quad (1)$$

$$\Sigma^N [] = 0 \quad (2)$$

$$\Sigma^{s(Y)} H :: T = H + \Sigma^Y T \quad (3)$$

prod (first order)

$$\Pi^0 L = 1 \tag{4}$$
$$\Pi^N [] = 1 \tag{5}$$
$$\Pi^{s(Y)} H :: T = H * \Pi^Y T \tag{6}$$

1.3 Theorem

$$\forall n, a.\, n = length(a) \rightarrow n^n * \Pi^n a \leq (\Sigma^n a)^n \tag{7}$$

1.4 Comments

The main challenge here is finding the appropriate induction scheme. One possible schemes involves two base cases (0 and 1) and two step cases $(P(n) \rightarrow P(2.n))$ and $(P(s(n)) \rightarrow P(n))$. However the step cases require a number of lemmas (including the introduction of a new functions). The new functions are called here oddlist, evenlist, sumlist, timeslist, ctimeslist, explist and are for respectively getting a list of the odd numbered elements of a list, the even numbered elements of a list, pairwise summation of all the elements of two lists, pairwise multiplication of two list, multiply every element of a list by a constant and raising every element of a list by a given exponent.

oddlist

$$oddlist([]) = [] \tag{8}$$
$$oddlist(H :: []) = H :: [] \tag{9}$$
$$oddlist(H_1 :: H_2 :: T) = H_1 :: oddlist(T) \tag{10}$$

evenlist:

$$evenlist([]) = [] \tag{11}$$
$$evenlist(H :: []) = [] \tag{12}$$
$$evenlist(H_1 :: H_2 :: T)) = H_2 :: evenlist(T) \tag{13}$$

sumlist:

$$sumlist([], L) = L \tag{14}$$
$$sumlist(L, []) = L \tag{15}$$
$$sumlist(H_1 :: T_1, H_2 :: T_2) = (H_1 + H_2) :: sumlist(T_1, T_2) \tag{16}$$

timeslist:

$$timeslist([], L) = L \tag{17}$$
$$timeslist(L, []) = L \tag{18}$$
$$timeslist(H_1 :: T_1, H_2 :: T_2) = (H_1 * H_2) :: timeslist(T_1, T_2) \tag{19}$$

ctimeslist:

$$ctimeslist(M, []) = [] \tag{20}$$
$$ctimeslist(M, H :: T) = (M * H) :: ctimeslist(M, T) \tag{21}$$

explist:

$$explist([], M) = [] \tag{22}$$
$$explist(H :: T, M) = (H^M) :: explist(T, M) \tag{23}$$

Lemmas used in a sample proof are:

$$\begin{array}{rcl}
(xy)^z & = & x^z y^z \\
(xy)z & = & y(xz) \\
\Pi^n ctimeslist(x,l) & = & x^n \Pi^n l \\
\Sigma^{2n} l & = & \Sigma^n sumlist(oddlist(l), evenlist(l)) \\
\Pi^{2n} l & = & \Pi^n prodlist(oddlist(l), evenlist(l)) \\
z^{yz} & = & (x^y)^z \\
\Pi^n explist(l,m) & = & (\Pi^n l)^m \\
\Sigma^n sumlist(l,m) & = & \Sigma^n l + \Sigma^n m \\
sumlist(l, ctimeslist(n,l)) & = & ctimeslist(s(n), l) \\
\Sigma^n ctimeslist(m,l) & = & m\Sigma^n l \\
xz \le xy & \rightarrow & z \le y \\
x^{s(n)} & = & xx^n
\end{array}$$

Even with these lemmas the rewriting involved presents several challenges to rippling (e.g. sinks need to be proved equal) and rippling has to take place in the induction hypothesis for the second step case.

1.5 Source

T. Walsh, *The Arithmetic/Geometric Mean Theorem*, Edinburgh MRG Group Blue Book Note 828.

2 IWC001b: Arithmetic/Geometric Mean

2.1 Summary

- Higher Order.
- Unusual Induction Scheme.
- Extra lemmas required.
- Challenges Rippling.
- Goal is not equational.

2.2 Definitions

sum

$$\begin{array}{rcl}
\Sigma^0 F & = & F(0) \qquad (24) \\
\Sigma^{s(Y)} F & = & F(s(Y)) + \Sigma^Y F \qquad (25)
\end{array}$$

prod

$$\begin{array}{rcl}
\Pi^0 F & = & F(0) \qquad (26) \\
\Pi^{s(Y)} F & = & F(s(Y)) * \Pi^Y F \qquad (27)
\end{array}$$

2.3 Theorem

$$\forall n, a.\, \Pi^n a \leq (\frac{\Sigma^n a}{n})^n \tag{28}$$

2.4 Comments

This is essentially the same theorem as IWC001c but reformulated and using a different step case in a way that makes the proof easier. Reformulation is attributed to Shankar.

There are at least two possible induction schemes that can be used for this proof. One with the step cases $(P(n) \rightarrow P(2n))$ and $(P(n) \rightarrow P(2n-1))$ and $(\forall m.(m < n) \rightarrow P(m)) \rightarrow P(n))$ with a casesplit on odd and even numbers in the step case. The discussion here is based upon a proof using the first of these.

It needs several lemmas (some given below)

$$\begin{aligned}
\Sigma^{2n} f &= \Sigma^n f + \Sigma^n \lambda i.f(n+i) \\
\Pi^{2n} f &= \Pi^n f.\Pi^n \lambda i.f(n+i) \\
\frac{x}{yz} &= \frac{\frac{x}{z}}{y} \\
\frac{x+y}{z} &= \frac{x}{z} + \frac{y}{z} \\
\sqrt[xy]{z} &= \sqrt[x]{\sqrt[y]{z}} \\
\sqrt[x]{yz} &= \sqrt[x]{y}.\sqrt[x]{z} \\
(x^n = y) &= (x = (\sqrt[n]{y}))
\end{aligned}$$

Even with these lemmas the rewriting involved presents several challenges to rippling (e.g. the "odd" step case does not appear to make use of the annotations).

2.5 Source

A. Bundy, *Shankar's Arithmetic/Geometric Mean Proof*, Edinburgh MRG Group Blue Book Note 951.

3 IWC001c: Arithmetic/Geometric Mean

3.1 Summary

- Higher Order.
- Unusual Induction Scheme.
- Extra lemmas required.
- Challenges Rippling.
- Goal is not equational.

sum

$$\Sigma^0 F = F(0) \tag{29}$$
$$\Sigma^{s(Y)} F = F(s(Y)) + \Sigma^Y F \tag{30}$$

prod

$$\Pi^0 F = F(0) \tag{31}$$
$$\Pi^{s(Y)} F = F(s(Y)) * \Pi^Y F \tag{32}$$

3.2 Theorem

$$\forall n, a. n^n * \Pi^n a \leq (\Sigma^n a)^n \tag{33}$$

3.3 Comments

There are at least two possible induction schemes that can be used for this proof. One with the step cases $(P(n) \rightarrow P(2n))$ and $(P(s(n)) \rightarrow P(n))$ and $\forall m.(m < n) \rightarrow P(m) \rightarrow P(n)$ called variously course of values induction, strong induction and other names. The discussion here is based upon a proof using the first of these.

It needs several lemmas (given below)

$$\begin{array}{rcl}
(xy)^z & = & x^z y^z \\
(xy)z & = & y(xz) \\
\Pi^n \lambda i.xf(i) & = & x^n \Pi^n f \\
\Sigma^{2n} f & = & \Sigma^n \lambda i.f(2i-1) + f(2i) \\
\Pi^{2n} f & = & \Pi^n \lambda i.f(2i-1) + f(2i) \\
x^{yz} & = & (x^y)^z \\
\Sigma^n f + \Sigma^n g & = & \Sigma^n \lambda i.f(i) + g(i) \\
\Sigma^n \lambda i.xf(i) & = & x\Sigma^n f \\
xz \leq xy & \rightarrow & z \leq y
\end{array} \tag{34}$$

Even with these lemmas the rewriting involved presents several challenges to rippling (e.g. sinks need to be proved equal) and rippling has to take place in the induction hypothesis for the second step case.

3.4 Source

A. Bundy, *An Analysis of the Arithmetic/Geometric Mean Theorem* Edinburgh MRG Group Blue Book Note 524.

4 IWC002: Even Length Append

4.1 Summary

- Needs induction scheme on two variable.
- Needs a lemma.

4.2 Theorem

$$\forall x, y. even(length(X <> Y)) = even(length(Y <> X)) \tag{35}$$

4.3 Comments

Its the (non-mutually recursive) definition of even that requires a non-straigtforward induction scheme.

A lemma is also required (along the lines of)

$$length(X <> Y_1 :: (Y_2 :: Y)) = s(s(length(Y <> X)))$$

4.4 Source

Andrew Ireland

5 IWC003: Case Analysis

5.1 Summary

- Requires Case Analysis
- Not an equality theorem

5.2 Theorem:

$$\forall x, z.\, X \in Y \rightarrow X \in (Y <> Z) \tag{36}$$

5.3 Comments

This is here because λ*Clam* is no good at case splits. I'm not sure if this is a problem particular to λ*Clam* or a more general problem.

5.4 Source

Andrew Ireland

6 IWC004a: Rotate Length

6.1 Summary

- Requires a Generalisation.
- Needs lemmas.

6.2 Definitions

rotate

$$\begin{aligned} rotate(0, Z) &= Z & (37)\\ rotate(s(N), []) &= [] & (38)\\ rotate(s(N), Y :: Z) &= rotate(X, Z <> (Y :: [])) & (39) \end{aligned}$$

Allowed Lemmas

$$\begin{aligned} (X <> Y) <> Z &= X <> (Y <> Z) & (40)\\ (X <> (Y :: [])) <> Z &= X <> (Y :: Z)) & (41) \end{aligned}$$

6.3 Theorem

$$\forall x.\, rotate(length(X), X) = X \tag{42}$$

6.4 Comments

A generalisation step is required. One of these would be to transform the theorem into IWC004b.

6.5 Source

Andrew Ireland

7 IWC004b: Rotate Length

7.1 Summary

- Lemmas Required

7.2 Definitions

rotate

$$rotate(0, Z) = Z \tag{43}$$
$$rotate(s(N), []) = [] \tag{44}$$
$$rotate(s(N), Y :: Z) = rotate(X, Z <> (Y :: [])) \tag{45}$$

7.3 Theorem

$$\forall x, y.\, rotate(length(X), X <> Y) = Y <> X \tag{46}$$

7.4 Comments

Needs lemmas:

$$(X <> Y) <> Z = X <> (Y <> Z) \tag{47}$$
$$(X <> (Y :: [])) <> Z = X <> (Y :: Z)) \tag{48}$$

7.5 Source

Andrew Ireland

8 IWC005a: Binomial Theorems

8.1 Summary

- Non Trivial Lemmas needed

8.2 Definitions

choose

$$choose(X, 0) = s(0) \tag{49}$$
$$choose(0, s(Y)) = 0 \tag{50}$$
$$choose(s(X), s(Y)) = choose(X, s(Y)) + choose(X, Y) \tag{51}$$

sum

$$\begin{aligned}\Sigma_X^0 F &= F(0) && (52)\\ s(Y) < X \rightarrow \Sigma_X^{s(Y)} F &= 0 && (53)\\ \neg(s(Y) < X) \rightarrow \Sigma_X^{s(Y)} F &= F(s(Y)) + \Sigma_X^Y F && (54)\end{aligned}$$

8.3 Theorem

$$\forall x, n.\ s(x)^n = \Sigma_0^n(\lambda i.choose(n,i) * x^i) \quad (55)$$

8.4 Comments

Requires several non-trivial lemmas. A simple first order version has been proved by SPIKE.

Possible lemmas include:

$$\begin{aligned}choose(n,k) &= choose(n, n-k)\\ \Sigma_m^n \lambda i.f(i) + G(i) &= \Sigma_m^n f + \Sigma_m^n g\\ \Sigma_m^n \lambda i.t * g(i) &= t * \Sigma_m^n g\\ \neg s(m) < n \Rightarrow \Sigma_{s(m)}^n f &= f(n) + \Sigma_m^n \lambda i.f(s(i))\end{aligned}$$

8.5 Variants

$choose(n,i)$ can be replaced with $\frac{n!}{(i! * (n-i)!)}$

8.6 Source

T. Walsh, *The Binomial Theorem*, MRG Group BBNote 903.

9 IWC005b: Binomial Theorem (Variation)

9.1 Summary

- Non Trivial Lemmas needed.

9.2 Definitions

choose

$$\begin{aligned}choose(X,0) &= s(0) && (56)\\ choose(0, s(Y)) &= 0 && (57)\\ choose(s(X), s(Y)) &= choose(X, s(Y)) + choose(X,Y) && (58)\end{aligned}$$

sum

$$\begin{aligned}\Sigma_X^0 F &= F(0) && (59)\\ s(Y) < X \rightarrow \Sigma_X^{s(Y)} F &= 0 && (60)\\ \neg(s(Y) < X) \rightarrow \Sigma_X^{s(Y)} F &= F(s(Y)) + \Sigma_X^Y F && (61)\end{aligned}$$

9.3 Theorem:

$$\forall x, y, n.(x + y)^n = \Sigma_0^n(\lambda i.choose(n, i) * x^i * y^{n-i}) \tag{62}$$

9.4 Comments

9.5 Variants

$choose(n, i)$ can be replaced with $\frac{n!}{(i!*(n-i)!)}$

9.6 Source

10 IWC006: Two Definitions of Even are Equivalent

10.1 Summary

- Mutual Recursion

10.2 Definitions

evenm

$$evenm(0) = \top \tag{63}$$
$$evenm(s(N)) = oddm(N) \tag{64}$$

oddm

$$oddm(0) = \bot \tag{65}$$
$$oddm(s(N)) = evenm(N) \tag{66}$$

evenr

$$evenr(0) = \top \tag{67}$$
$$evenr(s(0)) = \bot \tag{68}$$
$$evenr(s(s(N))) = evenr(N) \tag{69}$$

10.3 Theorem

$$\forall n.evenm(n) = evenr(n) \tag{70}$$

10.4 Source

Alan Bundy

11 IWC007: All numbers are odd or even

11.1 Summary

- Mutual Recursion
- Not Equality

11.2 Definitions

evenm

$$\begin{aligned} evenm(0) &= \top & (71)\\ evenm(s(N)) &= oddm(N) & (72)\end{aligned}$$

oddm

$$\begin{aligned} oddm(0) &= \bot & (73)\\ oddm(s(N)) &= evenm(N) & (74)\end{aligned}$$

11.3 Theorem

$$\forall n. evenm(n) \vee oddm(n) \tag{75}$$

11.4 Source

Alan Bundy

12 IWC008a: Chinese Remainder Theorem

12.1 Summary

- Needs many lemmas (some of whose proofs are also challenging)
- An Existential Witness has to be Provided

12.2 Definitions

allcongruent

$$\begin{aligned} allcongruent(X, []) &= \top & (76)\\ allcongruent(X, Y :: Ys, Z :: Zs) &= allcongruent(X, Ys, Zs) \wedge (X \; mod \, Y) = (Z \; mod \, Y) & (77)\end{aligned}$$

allpositive true if all members of the list are greater than 0.

allprime2

$$\begin{aligned} allprime2([]) &= \top & (78)\\ allprime2(Y :: Z) &= prime2list(Y, Z) \wedge allprime2(Z) & (79)\end{aligned}$$

prime2 is true iff its arguments are relatively prime

prime2list

$$\begin{aligned} prime2list(X, []) &= \top & (80)\\ prime2list(X, Y :: Z) &= prime2(X, Y) \wedge prime2list(X, Z) & (81)\end{aligned}$$

products1 product of list.

12.3 Theorem

$$\begin{aligned}
\forall l_1, l_2.\exists x.allpositive(l_1) \wedge allprime2(l_1) &\rightarrow allcongruent(x, l_2, l_1) && (82)\\
\forall l_1, l_2, x, y.allpositive(l_1) \wedge allprime2(l_1) \wedge & \\
allcongruent(x, l_2, l_1) \wedge allcongruent(y, l_2, l_1) &\rightarrow (x - y)\, mod\, products(l) = 0 && (83)
\end{aligned}$$

12.4 Comment

Actually two proofs, existance and uniqueness. Proved in RRL by Zhang and Hua and there is a good deal of comment of the proof in their CADE-11 paper on the subject. They provide by hand the existential witness needed for the existance part of the proof.

12.5 Source

Proving the Chinese Remainder Theorem by the Cover Set Induction H. Zhang and X. Hua, CADE-11, D. Kapur (ed), 1992. Springer-Verlag.

13 IWC008b: Chinese Remainder Theorem (Higher Order)

13.1 Summary

- Needs many lemmas (some of whose proofs are also challenging).
- An Existential Witness has to be Provided.
- Higher Order.

13.2 Definitions

all

$$\begin{aligned}
all(X, 0, P) &= P(0) && (84)\\
s(Y) < X \rightarrow all(X, s(Y), P) &= \top && (85)\\
\neg(s(Y) < X) \rightarrow all(X, s(Y), P) &= P(s(Y)) \wedge all(X, Y, P) && (86)
\end{aligned}$$

prod

$$\begin{aligned}
\Pi(X, 0, F) &= F(0) && (87)\\
s(Y) < X \rightarrow \Pi(X, s(Y), F) &= s(0) && (88)\\
\neg(s(Y) < X) \rightarrow \Pi(X, s(Y), F) &= F(s(Y)) * \Pi(X, Y, F) && (89)
\end{aligned}$$

quot

$$\begin{aligned}
quot(X, 0) &= 0 && (90)\\
(X < Y) \wedge \neg(Y = 0) \rightarrow quot(X, Y) &= 0 && (91)\\
\neg(X < Y) \wedge \neg(Y = 0) \rightarrow quot(X, Y) &= s(quot(X - Y, Y)) && (92)
\end{aligned}$$

rprime X and Y are relatively prime.

13.3 Theorem

$$\begin{aligned}&\forall f, g, n, u_1, u_2, v_1, v_2.\\ &\exists x.all(0, n, \lambda i.(0 < f(i)))\\ &\land 0 \leq u_1 \leq n \ \land \ 0 \leq u_2 \leq n\\ &\land u_1 \neq u_2 \ \land \ rprime(f(u_1), f(u_2))\\ &\to all(0, n, \lambda i.x = (g(i) \; mod \; f(i))\end{aligned} \tag{93}$$

$$\begin{aligned}&\forall f, g, n, u_1, u_2, v_1, v_2, x_1, x_2.\\ &all(0, n, \lambda i.(0 < f(i)))\\ &\land 0 \leq u_1 \leq n \ \land \ 0 \leq u_2 \leq n\\ &\land u_1 \neq u_2 \ \land \ rprime(f(u_1), f(u_2))\\ &\land all(0, n, \lambda i.x_1 = (g(i) \; mod \; f(i))) \ \land \ all(0, n, \lambda i.x_2 = (g(i) \; mod \; f(i)))\\ &\to (x_1 \; mod \; \Pi(0, n, f)) = (x_2 \; mod \; \Pi(0, n, g))\end{aligned} \tag{94}$$

13.4 Comment

Actually two proofs, existance and uniqueness First Order version proved in RRL by Zhang and Hua (See IWC008a).

14 IWC009: "Pete's Nasty Theorem"

14.1 Summary

- Problems for Rippling (Hole-less Wave Front)

14.2 Definitions

split_list

$$split\_list([], W) = W \tag{95}$$

$$length(W) = 6 \to split\_list(A :: X, W) = W :: split\_list(A :: X, []) \tag{96}$$

$$\neg(length(W) = 6) \to split\_list(A :: X, W) = split\_list(X, W <> [A]) \tag{97}$$

new_split

$$new\_split([], W, D) = W \tag{98}$$

$$(D = 6) \to new\_split(A :: X, W, D) = W :: new\_split(A :: X, [], length([])) \tag{99}$$

$$\neg(D = 6) \to new\_split(A :: X, W, D) = new\_split(X, W <> [A], s(D)) \tag{100}$$

14.3 Theorem

$$\forall x, w.new\_split(x, w, (length(w))) = split\_list(x, w) \tag{101}$$

14.4 Comment

This poses a problem for rippling since it is hard to annotate the definitions of wave rules (although a number of solutions have been proferred I'm not aware that any have been implemented).

14.5 Source

Problem attributed to P. Madden.
A. Bundy, *How to Prove Pete's Nasty Theorem*, Edinburgh MRG Group BBNote 725
A. Bundy, *The Advantages of Binary Sinks*, Edinburgh MRG Group BBNote 1311

15 IWC010: Quicksort

15.1 Summary

- Destructor Style Induction Scheme
- Additional Lemmas

15.2 Definitions

grtlist

$$\begin{aligned} grtlist(X, []) &= [] &(102)\\ H \leq X \rightarrow grtlist(X, H :: T) &= grtlist(X, T) &(103)\\ \neg(H \leq X) \rightarrow grtlist(X, H :: T) &= H :: grtlist(X, T) &(104)\end{aligned}$$

leqlist

$$\begin{aligned} leqlist(X, []) &= [] &(105)\\ H \leq X \rightarrow leqlist(X, H :: T) &= H :: leqlist(X, T) &(106)\\ \neg(H \leq X) \rightarrow leqlist(X, H :: T) &= leqlist(X, T) &(107)\end{aligned}$$

occ

$$\begin{aligned} occ(X, []) &= 0 &(108)\\ X = H \rightarrow occ(X, H :: T) &= s(occ(X, T)) &(109)\\ X \neq H- > occ(X, H :: T) &= occ(X, T) &(110)\end{aligned}$$

qsort

$$\begin{aligned} qsort([]) &= [] &(111)\\ qsort(H :: T) &= qsort(leqlist(H, T)) <> H :: qsort(grtlist(H, T)) &(112)\end{aligned}$$

sorted

$$\begin{aligned} sorted([]) &= \top &(113)\\ sorted(X :: []) &= \top &(114)\\ (T \neq []) \wedge (\neg(H \leq hd(T))) \rightarrow sorted(H :: T) &= \bot &(115)\\ (T \neq []) \wedge (H \leq hd(T)) \rightarrow sorted(H :: T) &= sorted(T) &(116)\end{aligned}$$

15.3 Theorem

$$\forall l.sorted(qsort(l)) \qquad (117)$$

$$\forall l.occ(x, qsort(l)) = occ(x, l) \qquad (118)$$

15.4 Comments

Proved in RRL (using split, instead of $grtlist$ and $leqlist$, which takes $<$ and $>$ as arguments), Walther describes an approach in Mathematical Induction in the Handbook of Automated Reasoning and Bronsard, Reddy and Hasker look at the problem in Induction Using Term Orders in JAR 16.

A destructor style induction is needed.

Possible Lemmas:

$$
\begin{aligned}
sorted(L) \wedge sorted(M) &\rightarrow sorted(L <> M) && (119)\\
occ(X, L <> M) &= occ(X, L) + occ(X, M) && (120)\\
X \leq Y \rightarrow occ(X, leqlist(Y, L)) &= occ(X, L) && (121)\\
\neg(X \leq Y) \rightarrow occ(leqlist(Y, L) &= 0 && (122)\\
X \leq Y \rightarrow occ(X, grtlist(Y, L)) &= 0 && (123)\\
\neg(X \leq Y) \rightarrow occ(grtlist(Y, L)) &= occ(X, L) && (124)
\end{aligned}
$$

15.5 Source

F. Bronsard, U. Reddy, R. Hasker, Induction using Term Orders. *J. of Automated Reasoning Vol 16,* Nos 1-2, 3-37, 1996.

C. Walther, *Mathematical Induction.* In D. Gabbay, C. Hogger and J. Robinson (eds), Handbook of Logic in Artificial Intelligence and Logic Programming, v2. 127-228, OUP, 1994.

H. Zhang, D. Kapur, M. Krishnamoorthy. *A Mechnaizable Induction Principle for Equational Specifications.* In E. Lusk and R. Overbeek (eds). Proc 9th International Conference on Automated Induction, 152-181, Springer-Verlag, 1988.

16 IWC011: Verifying Abstractions in Model Checking (Safety Lemma for Removing the head of a list)

16.1 Summary

- Generalisation(?).
- Challenges for Rippling.
- Goal is not equational.

16.2 Definitions

aelem enumerated type containing elements $\{e_1, e_2, n_e\}$. This is intended to be an abstraction of the natural numbers: an $\alpha \in$ is equal to 2, 5 or some other natural.

$\alpha_\in$ Converts naturals to aelems.

$$
\begin{aligned}
\alpha_\in(2) &= e_1 && (125)\\
\alpha_\in(5) &= e_2 && (126)\\
\neg((N = 2) \vee (N = 5)) \rightarrow \alpha_\in(N) &= n_e && (127)
\end{aligned}
$$

order enumerated type containing elements $\{e[], e_1l, e_2l, e_1e_2l, e_2e_1l, error\}$. This abstracts lists of aelems to the information about whether there are an occurences of e_1 and e_2 in the list and in which order they come. $error$ indicates that there is more than 1 occurence of either e_1 or e_2 in the list.

combine Represents the effect of inserting aelems in orders.

$$combine(n_e, X) = X \tag{128}$$
$$combine(e_1, e[]) = e_1l \tag{129}$$
$$combine(e_1, e_2l) = e_1e_2l \tag{130}$$
$$combine(e_2, e[]) = e_2l \tag{131}$$
$$combine(e_2, e_1l) = e_2e_1l \tag{132}$$
$$\neg((X = e[]) \vee (X = e_2l)) \rightarrow combine(e_1, X) = error \tag{133}$$
$$\neg((X = e[])or(X = e_1l)) \rightarrow combine(e_2, X) = error \tag{134}$$

α**order** Converts a regular list of naturals to an order

$$\alpha order([]) = e[] \tag{135}$$
$$\alpha order(H :: T) = combine(\alpha_\in(H), \alpha order(T)) \tag{136}$$

alist New Type. A quadruple of two booleans, an aelem and an order. A refinement of αorder. The first boolean indicates whether the list is nonempty, the second whether it has only one element, the third element gives the head of the list and the third the ordering information held in the order type.

α This converts a regular list of naturals to an alist.

$$\alpha([]) = \langle \bot, \bot, n_e, e[] \rangle \tag{137}$$
$$\alpha(H_1 :: T) = \langle \top, (T = []), \alpha_\in(H_1), \alpha order(T) \rangle \tag{138}$$

set_l New type. Constructors:

$set_\emptyset$

$set_i(alist, set_l)$

rmhd

$$rmhd([]) = [] \tag{139}$$
$$rmhd(H :: T) = T \tag{140}$$

set_elem

$$set\_elem(E, set_\emptyset) = \bot \tag{141}$$
$$set\_elem(E, set_i(E, X)) = \top \tag{142}$$
$$\neg(E = X) \rightarrow set\_elem(E, set_i(X, T)) = set\_elem(E, T) \tag{143}$$

armhd

$$armhd(\langle \bot, A, B, C \rangle) = set_i(\langle \bot, \bot, n_e, e[] \rangle, set_\emptyset) \tag{144}$$
$$armhd(\langle A, \top, B, C \rangle) = set_i(\langle \bot, \bot, n_e, e[] \rangle, set_\emptyset) \tag{145}$$
$$armhd(\langle \top, \bot, X, e[] \rangle) = set_i(\langle \top, \bot, n_e, e[] \rangle, set_i(\langle \top, \top, n_e, e[] \rangle, set_\emptyset)) \tag{146}$$
$$armhd(\langle \top, \bot, X, e_1l \rangle) = set_i(\langle \top, \bot, e_1, e[] \rangle, set_i(\langle \top, \top, e_1, e[] \rangle, set_i(\langle \top, \bot, n_e, e_1l \rangle, set_\emptyset))) \tag{147}$$

$$armhd(\langle\top,\bot,X,e_2l\rangle) = set_i(\langle\top,\bot,e_2,e[]\rangle, set_i(\langle\top,\top,e_2,e[]\rangle, set_i(\langle\top,\bot,n_e,e_2l\rangle, set_\emptyset))) \quad (148)$$

$$armhd(\langle\top,\bot,X,e_1e_2l\rangle) = set_i(\langle\top,\bot,n_e,e_1e_2l\rangle, set_i(\langle\top,\bot,e_1,e_2l\rangle, set_\emptyset)) \quad (149)$$

$$armhd(\langle\top,\bot,X,e_1e_1l\rangle) = set_i(\langle\top,\bot,n_e,e_1e_1l\rangle, set_i(\langle\top,\bot,e_2,e_1l\rangle, set_\emptyset)) \quad (150)$$

$$\begin{aligned} armhd(\langle\top,\bot,X,error\rangle) = {} & set_i(\langle\top,\bot,n_e,error\rangle, set_i(\langle\top,\bot,e_1,e_1l\rangle, \\ & set_i(\langle\top,\bot,e_1,e_1e_2l\rangle, set_i(\langle\top,\bot,e_1,e_2e_1l\rangle, \\ & set_i(\langle\top,\bot,e_1,error\rangle, set_i(\langle\top,\bot,e_2,e_2l\rangle, \\ & set_i(\langle\top,\bot,e_2,e_1e_2l\rangle, set_i(\langle\top,\bot,e_2,e_2e_1l\rangle, \\ & set_i(\langle\top,\bot,e_2,error\rangle, set_\emptyset))))))))) \end{aligned} \quad (151)$$

16.3 Theorem

$$\forall l. set\_elem(\alpha(rmhd(l)), armhd(\alpha(l))) \quad (152)$$

16.4 Comments

Provided by Dieter Hutter for 2000 Challenges. The problem is originally attributed to Dennis Dams of Eindhoven University. Believe the problems involve the need for generalisations and some challenges for Rippling.

17 IWC012: Verifying Abstractions in Model Checking (Safety Lemma for the prefix Operation)

17.1 Summary

- Generalisation(?).
- Lemmas.
- Challenges for Rippling.
- Goal is not equational.

17.2 Definitions

aelem Defined as for IWC011.

order Defined as for IWC011.

alist Defined as for IWC011.

α**elem** Defined as for IWC011.

combine Defined as for IWC011.

α**order** Defined as for IWC011.

α Defined as for IWC011.

prefix

$$prefix([], L) = \top \quad (153)$$

$$prefix(h :: t1, h :: t2) = prefix(t1, t2) \quad (154)$$

$$(h1 \neq h2) \rightarrow prefix(h1 :: t1, h2 :: t2) = \bot \quad (155)$$

aprefix

$$
\begin{aligned}
aprefix(\langle \bot, A, B, C\rangle, \langle E, F, G, H\rangle) &= \top && (156)\\
aprefix(\langle A, B, C, D\rangle, \langle E, F, G, error\rangle) &= \top && (157)\\
aprefix(\langle A, \top, D, e[]\rangle, \langle \top, F, G, H\rangle) &= \top && (158)\\
aprefix(\langle A, B, D, e[]\rangle, \langle \top, \bot, G, H\rangle) &= \top && (159)\\
aprefix(\langle A, \top, D, e_1 l\rangle, \langle E, F, G, e_1 l\rangle) &= \top && (160)\\
aprefix(\langle A, B, D, e_1 l\rangle, \langle E, \bot, G, e_1 l\rangle) &= \top && (161)\\
aprefix(\langle A, \top, D, e_1 l\rangle, \langle E, F, G, e_1 e_2 l\rangle) &= \top && (162)\\
aprefix(\langle A, \top, D, e_2 l\rangle, \langle \top, \bot, G, e_2 l\rangle) &= \top && (163)\\
aprefix(\langle A, B, D, e_2 l\rangle, \langle E, \bot, G, e_2 l\rangle) &= \top && (164)\\
aprefix(\langle A, B, D, e_2 l\rangle, \langle E, F, G, e_2 e_1 l\rangle) &= \top && (165)\\
otherwise false & && (166)
\end{aligned}
$$

17.3 Theorem

$$\forall l_1, l_2. prefix(l_1, l_2) \rightarrow aprefix(\alpha(l_1), \alpha(l_2)) \qquad (167)$$

17.4 Comments

Provided by Dieter Hutter for 2000 Challenges. Believe the problems involve the need for generalisations and some challenges for Rippling.

Lemmas required by INKA include:

$$aprefix(\langle \top, \bot, U, Y\rangle, \langle \top, \bot, V, Z\rangle) \rightarrow aprefix(\langle \top, \bot, W, combine(X, Y)\rangle, \langle \top, \bot, W, combine(X, Z)\rangle)$$

$$aprefix(\langle \top, \bot, U, combine(U, e[])\rangle, \langle \top, \bot, U, combine(U, V)\rangle)$$

$$prefix(Y, Z) \rightarrow aprefix(\langle \top, \bot, Z, \alpha order(Y)\rangle, \langle \top, \bot, X, \alpha order(Z)\rangle)$$

18 IWC013: Divide and Conquer

18.1 Summary

- Destructor style induction (at least)

18.2 Definitions

dc

$$
\begin{aligned}
dc(F, B, []) &= B && (168)\\
dc(F, B, X :: []) &= X && (169)\\
dc(F, B, H_1 :: (H_2 :: L)) &= F(dc(F, B, take(\frac{length(L)}{2}, L)), dc(F, B, drop(\frac{length(L)}{2}, L))) && (170)
\end{aligned}
$$

foldr

$$
\begin{aligned}
foldr(F, A, []) &= A && (171)\\
foldr(F, A, H :: T) &= F(H, foldr(F, A, T)) && (172)
\end{aligned}
$$

split

$$split(0, L) = L :: [] \quad (173)$$
$$split(s(0), L) = L :: [] \quad (174)$$
$$split(s(s(X)), L) = take(\frac{length(L)}{2}, L) :: split(1, drop(\frac{length(L)}{2}, L)) \quad (175)$$

18.3 Theorem

$$\forall f, b, x, l, n.\ f(b, x) = x \rightarrow dc(f, b, l) = foldr(f, b, foldr(<>, [], map(map(\lambda x.f(b, x), split(n, l)))))$$

18.4 Comments

Prove the equivalence of two divide and conquer algorithms.

18.5 Source:

Greg Michaelson.

19 IWC014: Harald's Problem

19.1 Summary

- Non-standard induction scheme.
- Needs a Lemma
- Higher Order

19.2 Definitions

foldl

$$foldl(F, A, []) = A \quad (176)$$
$$foldl(F, A, H :: T) = foldl(F, F(A, H), T) \quad (177)$$

foldr

$$foldr(F, A, []) = A \quad (178)$$
$$foldr(F, A, H :: T) = F(H, foldr(F, A, T)) \quad (179)$$

19.3 Theorem

$$\forall o_1, o_2, l, x, y, z, a.\ (o_1(a, x) = o_2(x, a) \wedge o_1(o_2(x, y), z) = o_2(x, o_1(y, z))) \rightarrow foldl(o_1, a, l) = foldr(o_2, a, l)$$

19.4 Comments

Allegedly easy to understand and prove when expressed as Ellipsis. A key lemma suggested is $foldl(F, A, (L <> [e])) = F(foldl(F, A, L), E)$ (NB. It is probably not unreasonable to assume the presence of this lemma in a well developed theory.) and a suggested induction rule is

$$\frac{P([]), P([E]), P(T) \wedge P(H :: T) \wedge P(T <> [E]) \Rightarrow P(H :: T <> [E])}{\forall L.P(L)}$$

which gives 3 induction hypotheses in the step case.

19.5 Source

Attributed to Harald Ganzinger. A. Bundy, *Rippling in Harald's Problem*, Edinburgh MRG Group BBNote 978

20 IWC015: Paulson's Problem

20.1 Summary

- Non-standard induction scheme
- Lemma needed
- Problems for Rippling
- Higher Order

20.2 Definitions

foldl

$$\begin{aligned} foldl(F, A, []) &= A &(180)\\ foldl(F, A, H :: T) &= foldl(F, F(A, H), T) &(181)\end{aligned}$$

20.3 Theorem

$$\forall o, l, x, y, e.\ o(x, e) = x \wedge o(o(x, y), z) = o(x, o(y, z)) \rightarrow o(y, foldl(o, e, l)) = foldl(o, y, l)$$

20.4 Comments

Paulson presents two proofs of this. In both cases he uses non-structural induction to justify the use of a non-standard but structural induction scheme.

The two schemes are:

$$\frac{P([]), P(l) \rightarrow P(L <> [X])}{\forall L.P(L)}$$

and

$$\frac{P([]), P([x]), P(x :: l) \rightarrow P(x_1 :: (x_2 :: l))}{\forall L.P(L)}.$$

With the first of these schemes a rippling proof goes through fairly easily, given the lemma

$$foldl(F, X, L <> M) = foldl(F, foldl(F, X, L), M)$$

The second presents some fairly serious challenges to rippling.

Another suggested lemma is:

$$foldl(F, X, L <> H :: T) = F(foldl(F, X, L), H)$$

20.5 Source

L. C. Paulson, ML for the Working Programmer.
A. Bundy, *Non-Structural Inductions and Rippling*, Edinburgh MRG Group BBNote 1188

21 IWC016: The Whisky Problem

21.1 Summary

- Needs some sort of Generalisation.
- Goal is not an equation.

21.2 Definitions

$$\begin{aligned} & p(0,0) & & \quad (182) \\ p(X,0) & \rightarrow & p(h(X),0) & \quad (183) \\ p(h(X),Y) & \rightarrow & p(X,s(Y)) & \quad (184) \end{aligned}$$

21.3 Theorem

$$\forall y.p(0,y)$$

21.4 Comments

This is a proof from the domain of first order temporal logic. For it to go through the original conjecture needs to be generalised to: $\forall y,n.p(h^n(0),y)$ (where h^n means n applications of h - which is second order) or a new function h^* needs to be introduced

$$\begin{aligned} h^*(0,X) & = & X & \quad (185) \\ h^*(s(n),X) & = & h(h^*(n,X)) & \quad (186) \end{aligned}$$

and the conjecture generalised to

$$\forall y,n.p(h^*(n,0),y)$$

21.5 Source

Logics group at Liverpool. In particular Michael Fisher and Anatoli Degtyarev. Problem originally attributed to Regimantas Pliuskevicius.
L. A. Dennis and A. Bundy, A Comparison of two Proof Critics: Power vs. Robustness, in V. A Carreno, C. A. Munoz, S. Tahar (Eds.): *Proceedings of Theorem Proving in Higher Order Logics, 15th International Conference, TPHOLs 2002*, Hampton, VA, USA, August 20-23, 2002. pp 182-198. LNCS 2410. Springer.

22 IWC017: Dixon's Problem

22.1 Summary

22.2 Definitions

$$\begin{aligned} P0 & \iff & (P0 \iff P1) \\ P1 & \iff & (P1 \iff P2) \\ & \ldots & \\ Pi & \iff & (Pi \iff P(i+1)) \\ & \ldots & \\ Pn & \iff & (Pn \iff P0) \end{aligned}$$

Which can be captured in something like the following equation

$$q(N) = (\forall 0 \leq i < N.p(i) \iff (p(i) \iff p(s(i))) \wedge p(N) \iff (p(N) \iff p(0))) \tag{187}$$

22.3 Theorem

$$\forall n.\, q(n) \rightarrow p(0)$$

22.4 Comments

Interestingly this problem is easy for classical logic, but less easy if you are using intuitionistic logic (but still true and provable). Either way induction is a natural way to solve the problem.

22.5 Source

Lucas Dixon, Edinburgh MRG group.

References

[1] S. Autexier, D. Hutter, H. Mantel, and A. Schairer. System description: INKA 5.0 - a logical voyager. In H. Ganzinger, editor, *16th International Conference on Automated Deduction, CADE-16*, volume 1732 of *Lecture Notes in Artificial Intelligence*, Trento, 1999. Springer.

[2] R. S. Boyer and J S. Moore. *A Computational Logic.* ACM monograph series. Academic Press, New York, 1979.

[3] A. Bundy, D. Basin, D. Hutter, and A. Ireland. *Rippling: Meta-Level Guidance for Mathematical Reasoning*, volume 56 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2005.

[4] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 647–648. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449. Also available from Edinburgh as DAI Research Paper 507.

[5] D. Kapur and H. Zhang. An overview of rewrite rule laboratory (RRL). *J. Computer and Mathematics with Applications*, 29(2):91–114, 1995.

[6] M. Kaufmann and J S. Moore. ACL2: An industrial strength version of Nqthm. In *Compass'96: Eleventh Annual Conference on Computer Assurance*, page 23, Gaithersburg, Maryland, 1996. National Institute of Standards and Technology.

[7] F. Pfenning and C. Schürmann. System description: Twelf — A meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, 1999. Springer-Verlag LNAI 1632.

[8] J.D.C. Richardson, A. Smaill, and I. Green. System description: Proof planning in higher-order logic with lambda-clam. In C. Kirchner and H. Kirchner, editors, *Conference on Automated Deduction (CADE'98)*, volume 1421 of *Lecture Notes in Computer Science*, pages 129–133. Springer-Verlag, 1998.

[9] G. Sutcliffe. The CADE-17 ATP system competition. *Journal of Automated Reasoning*, 27(3):227–250, 2001.

[10] G. Sutcliffe and C. Suttner. The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.