Elementary engineering mechanics applications of the OTTER automated reasoning system

Nikolaos I. Ioakimidis

Division of Applied Mathematics and Mechanics, Department of Engineering Sciences, School of Engineering, University of Patras, GR-265 04 Patras, Greece
e-mail: n.ioakimidis@upatras.gr

Abstract The famous resolution-based McCune’s OTTER automated reasoning (or automated deduction) program has been used for the logical proof of elementary statements in mechanics on the basis of hypotheses either of general or of particular validity. Three such simple applications concerning fracture and structural mechanics in applied mechanics are described and the related OTTER’s complete input files and formal proofs are provided in all three cases. In the second of these applications, hypotheses including the disjunction of two (positive) possibilities are included in the related set, whereas in the third of these applications algebraic equality hypotheses are also present. The aim of these results is to attempt to use a very popular automated reasoning program (mainly being employed for formal proofs in pure mathematics and logic) in applied mechanics as a somewhat different possibility to the use of Prolog and expert systems and, in principle, with more general acceptable input clauses as well as a much more powerful reasoning–inference engine than Prolog. It is hoped that in the future there will become possible to combine numerical, symbolic and graphical computational facilities, already offered (simultaneously) by classical computer algebra systems (such as Maple and Mathematica), with automated reasoning systems (such as OTTER) in the same computational environment. Presently, the present results simply encourage the use of formal proofs (so popular in mathematics and logic) also in engineering mechanics, a fact already achieved here with moderate success with the help of OTTER. The encouragement to the axiomatization of simple engineering mechanics particular areas, although not attempted here, nevertheless seems also to constitute a possible by-product of the present approach, which seems to be new.

Keywords Automated proofs · Automated reasoning · Automated deduction · Formal proofs · Predicate logic · OTTER · Engineering mechanics · Fracture mechanics · Stress intensity factors · Cracks · Structural mechanics · Beams

Copyright © 1998 by the author

1Both the internal and the external links (all appearing in blue) were added by the author on 8 January 2018 for the online publication of this technical report.
Final publication details. This technical report was presented to the 5th National Congress on Mechanics, Ioannina, Greece, 27–30 August 1998. This Congress was organized by the Hellenic Society for Theoretical & Applied Mechanics (HSTAM) and the University of Ioannina. Therefore, the official publication of the present technical report is that included in the Proceedings of this Congress with the following details:


Although the style of this final, official publication is somewhat different from the style of the present technical report, nevertheless, the contents of the final publication essentially coincide with the contents of the present technical report.

1. Introduction

Several computational methods and the related software have proved powerful in engineering mechanics including purely numerical algorithms/computer languages, symbolic algorithms/computer algebra systems and mixed numerical–symbolic computations (also with the help of computer algebra systems). All of these approaches and the use of the related computer software/implementations have led to a variety of interesting results. Since the summer of 1989 we have been interested in symbolic computations (with the aid of computer algebra systems) including Gröbner bases, characteristic sets and computational quantifier elimination under simple inequality constraints (see, e.g., [1]).

Here our aim is somewhat different. More explicitly, we will use automated reasoning (automated deduction) software (based on classical mathematical/computational logic; see, e.g., [2–4]) as this software is used in computational logic, automated theorem proving (see, e.g., [5]) and automated deduction in general (see, e.g., [6]) and some areas of artificial intelligence long ago in order to become able to investigate its possible usefulness to elementary engineering mechanics applications. More explicitly, we have selected the famous OTTER (Organized Techniques for Theorem-proving and Effective Research, version 3.0.4 for MS-DOS) automated reasoning (automated deduction) system of W. McCune (prepared at Argonne National Laboratory, Argonne, IL, U.S.A. and freely available through the Internet (web page: http://www.cs.unm.edu/~mccune/otter) [7–11] and we have applied it to few elementary engineering mechanics problems having to do with fracture and structural mechanics and including logical statements and equalities. The basic inference techniques incorporated into OTTER are described in [12]. On the other hand, non-mechanical formal proof methods in mathematics are presented in [3, 13, 14].

The three engineering mechanics applications to be presented below are extremely simple and they have been derived with the help of a classical (in automated deduction) automated reasoning system, McCune’s OTTER. Nevertheless, these applications illustrate a probably interesting possibility at least from the theoretical–logical point of view and surely they may prove helpful in some instances, e.g. in the verification of the conclusions of alternative approaches such as the classical.

---

2 These details were also added by the author on 8 January 2018 for the online publication of this technical report.
approaches in artificial intelligence. These approaches are based on logic programming and they generally use Prolog or one of its dialects (already proved so useful in engineering mechanics) either directly or on the basis of expert systems (see, e.g., [15, 16], respectively).

On the other hand, it seems that the present approach could establish some kind of link between computer algebra techniques (such as Gröbner bases) on the one hand and formal automated reasoning techniques (such as those traditionally used in automated theorem proving and based on computational logic).

Such already successful attempts, i.e. to introduce automated theorem proving methods to computer algebra systems, have been already made with the Clarke and Zhao Analytica [17] and, very recently, with the Buchberger much more powerful/ambitious Theorema [18] theorem provers (both implemented in Mathematica). Perhaps, in the future, one of the generalizations of the present results could be the simultaneous use of a computer algebra system and an automated reasoning system (with the help of appropriate linking auxiliary software) for reasoning in engineering mechanics as has been already the case in mathematics.

More interesting seems to be the enrichment of the classical tools for automated reasoning with the help of algebraic tools, which are much more algorithmic in nature and, therefore, predictable as far as the execution times and the final outputs are concerned. Recently, Lakmazaheri [19] has been able to successfully employ Gröbner bases for non-linear algebraic equality constraint-based reasoning in beam and truss problems of structural mechanics.

Under these circumstances, the present elementary and rather educational applications may prove helpful for the introduction of formal automated reasoning software (as opposed both to computer algebra systems and to Prolog and related logic-oriented languages) as one more approach for simple “theorem” verifications in engineering mechanics. This happens in some analogy with the mathematical/logical pure (although computerized) theorem proofs, many of which have been achieved with the help of OTTER automated reasoning system long ago especially in some branches of algebra.

Evidently, the integration of all of the aforementioned computational approaches falls completely outside our present aims, which are restricted just to the illustration of only one approach: that offered by automated reasoning/deduction software.

2. Automated reasoning applications in mechanics

In this section, we will present some applications of the automated reasoning system OTTER to elementary problems in engineering mechanics. These applications will give us a clear idea about what type of engineering mechanics applications are tractable with the help of OTTER. They will also permit us to get a concrete idea about the power of OTTER in such applications and, to some extent, to understand how OTTER works both in the computer input and in the computer output files. Finally, it is hoped that they will also permit (at least in the present simple cases) the axiomatic formalization of some engineering mechanics problems and the derivation of proofs analogous to those available in several classical fields of mathematics, where either OTTER or similar or even dissimilar automated deduction programs have already proved quite efficient.

2.1. A simple conclusion in fracture mechanics

We assume the validity of the following premises (i.e. statements that are assumed to be valid in advance, hypotheses, the equivalent of axioms in mathematics) in a very simple fracture mechanics problem concerning (possibly) a loaded edge crack in a plane elastic medium:

- An excessive loading causes the appearance of a very high stress intensity factor (SIF) at the crack tip.
Such a very high stress intensity factor causes the fracture of the specimen.

The fracture of the specimen has as a consequence its replacement.

The replacement of the specimen has as a consequence a related cost as well as the delay in the actual use of the specimen (possibly) in production.

A little more formally, these four premises can be written in an IF-THEN notation as follows:

- IF excessive loading, THEN very high stress intensity factor (SIF).
- IF very high stress intensity factor, THEN fracture.
- IF fracture, THEN replacement.
- IF replacement, THEN cost and delay.

OTTER uses a very simple syntax [7, 8] and, mainly, (logical) clauses (i.e. finite disjunctions of literals [12]). In the OTTER’s syntax the above four premises can be written as (as is easily, but logically, verified [8])

\[
\begin{align*}
\text{-EXCESSIVE\_LOADING} & | \text{VERY\_HIGH\_SIF}. \\
\text{-VERY\_HIGH\_SIF} & | \text{FRACTURE}. \\
\text{-FRACTURE} & | \text{REPLACEMENT}. \\
\text{-REPLACEMENT} & | \text{COST\_AND\_DELAY}. \\
\end{align*}
\]

with the - (minus) sign being used as the negation symbol and the | (vertical) sign being the disjunction (logical or) symbol, whereas the . (period) sign signals the end of a clause.

In fact, OTTER accepts only input files (in DOS, UNIX, etc.). It is understood that the above four clauses should be included in this file. In our present and very elementary application, we assume (in a special case) that we really have EXCESSIVE\_LOADING. OTTER generally works with refutational proofs. This means that having already assumed that we have EXCESSIVE\_LOADING, we further (and completely incorrectly) also assume that we do not have COST\_AND\_DELAY, i.e. we assume the validity of the negation, \text{-COST\_AND\_DELAY}, of our expected, true conclusion, i.e. to have COST\_AND\_DELAY. These two further (fifth and sixth) assumptions (the second being the false hypothesis) are easily written in the OTTER’s syntax as

\[
\begin{align*}
\text{EXCESSIVE\_LOADING}. \text{-COST\_AND\_DELAY}. \\
\end{align*}
\]

and they were really added to the input OTTER file accompanying our original four premises.

Under these conditions the complete input file for the present elementary application is simply

\[
\begin{align*}
\text{set(auto)}. \\
\text{list(usable)}. \\
\text{-EXCESSIVE\_LOADING} | \text{VERY\_HIGH\_SIF}. \\
\text{-VERY\_HIGH\_SIF} | \text{FRACTURE}. \\
\text{-FRACTURE} | \text{REPLACEMENT}. \\
\text{-REPLACEMENT} | \text{COST\_AND\_DELAY}. \\
\text{EXCESSIVE\_LOADING}. \text{-COST\_AND\_DELAY}. \\
\text{end\_of\_list}. \\
\end{align*}
\]
The above input file begins with a fundamental command, `set(auto)`, interpreted by OTTER that the user does not wish to intervene in the OTTER’s automated deduction process and, mainly, in the inference rules used in it by defining a more or less efficient strategy during the (possible) proof. Generally, this is not considered to be a recommended practice but for inexperienced users (such as the present author) this is generally the best possibility. Next, we have to provide to OTTER a list of clauses beginning with the command `list(usable)` and ending with the command `end_of_list`. Now what is missing is just the six aforementioned clauses (our assumptions), the first four of which are essentially our axioms, our hypotheses of quite a general validity, and the last two of which are the special hypotheses, i.e. that we have `EXCESSIVE_LOADING`, but, nevertheless, we do not have `COST_AND_DELAY`: `~COST_AND_DELAY`. Evidently, the way that OTTER will work will be the refutational one, leading to a contradiction, i.e. proving that the list of six clauses in the usable list of the above input file is simply incompatible according to the classical logic adopted both by OTTER and by us.

We display now the proof having been provided by OTTER inside the output file corresponding to the above input file

```markdown
---------------- PROOF ----------------
1 [][] -EXCESSIVE_LOADING|VERY_HIGH_SIF.
2 [][] -VERY_HIGH_SIF|FRACTURE.
3 [][] -FRACTURE|REPLACEMENT.
4 [][] -REPLACEMENT|COST_AND_DELAY.
5 [][] -COST_AND_DELAY.
6 [][] EXCESSIVE_LOADING.
7 [hyper,6,1] VERY_HIGH_SIF.
8 [hyper,7,2] FRACTURE.
9 [hyper,8,3] REPLACEMENT.
10 [hyper,9,4] COST_AND_DELAY.
11 [binary,10.1,5.1] $F$.
------------ end of proof -------------
```

From the above proof we observe that OTTER used (in the present case) all six clauses in the input file for its proof, which (just in the present simple case) is essentially identical to the proof that a man would follow. The main inference rule having been used was hyperresolution. By using this rule and combining the given clauses (1) and (6) in the above proof, OTTER proved that we have `VERY_HIGH_SIF` (in clause (7)). Next, it used again hyperresolution, but now combining the given clause (2) and the just derived clause (7), to deduce that we have also `FRACTURE` (clause (8) in the above proof). In an analogous way, next OTTER proved that we must have `REPLACEMENT` (clause (9)) and, finally, `COST_AND_DELAY` (clause (10)).

At this final point OTTER took into account that we had assumed not to have `COST_AND_DELAY` (in clause (5) of the proof, assumed valid in advance, inside the input file, and with the minus sign denoting negation as was already mentioned). In this way, OTTER was led (just in this case by using the inference rule of binary resolution) to `UNIT CONFLICT` (denoted by the symbol `$F$` in the proof, including the classical letter, `F`, for falsity contrary to `T` used for truth), i.e. to a contradiction of a logically deduced conclusion (from our hypotheses after few logical deductions), i.e. that we have `COST_AND_DELAY`, with an original assumption: `~COST_AND_DELAY`. This contradiction completes the present refutational proof, having been based on the false assumption that our (reasonable, logically correct) conclusion does not hold true. This is an extremely classical method of proof, which is now complete and, we hope, very easy to understand. Let us now proceed to a somewhat more difficult fracture mechanics application.
2.2. The appropriate numerical method in a crack problem

In this application, we will prove a simple conclusion in a fracture mechanics problem concerning a simple straight crack in a plane elastic medium. This problem is intended to be solved by the method of Cauchy-type singular integral equations in combination with an appropriate quadrature rule.

At first, we assume that there are only three possibilities for the crack under consideration: (i) to be an interior crack (a possibility denoted by IC), (ii) to be an edge crack (EC) and (iii) to be an interface crack (FC). Moreover, as is well-known (and is assumed here to be the case), the following methods (based on the related quadrature rules) are applicable: (i) the Gauss–Chebyshev method (denoted by GC), (ii) the Lobatto–Chebyshev method (LC), (iii) the modified Gauss–Legendre method (MGL), (iv) the modified Lobatto–Legendre method (MLL), (v) the Gauss–Jacobi method (GJ) and (vi) the Lobatto–Jacobi method (LJ). These numerical methods are assumed here to be independent of each other and our sole possibilities. Moreover, for the final actual computation of the stress intensity factor(s), three numerical possibilities are present: (i) the polynomial extrapolation (denoted here by PE), (ii) the natural extrapolation (NE) and (iii) no extrapolation (NO).

Evidently, as is well understood, we cannot arbitrarily combine all of the above possibilities. More explicitly, e.g. the Gauss– and Lobatto–Chebyshev methods are associated with interior cracks and the avoidance of extrapolation with Lobatto-type methods. All of these facts were inserted into the related OTTER input file just below as logical rules (in 27 clauses). In our particular case, we also assume that we have an interface crack and that we wish to use either polynomial or natural extrapolation for the computation of the stress intensity factors. We have the impression that our sole possibility under the present assumptions is the use of the Gauss–Jacobi method. We will leave OTTER to prove that this is really the case, i.e. that no other of the aforementioned six quadrature methods is applicable.

We display below the related OTTER input file based again on the refutational method, i.e. including the negation/denial of our conclusion to be proved (−GJ). This wrong negative clause in the file will naturally lead to a contradiction and to the actual proof of our correct conclusion.

```
set(hyper_res).
list(sos).
% RULES
-IC | GC | LC. -IC | -MGL. -IC | -GJ. -IC | -LJ.
-EC | MGL | MLL. -EC | -GC. -EC | -LC. -EC | -GJ. -EC | -LJ.
-FC | GJ | LJ. -FC | -GC. -IC | -LC. -FC | -MGL. -IC | -MLL.
-PE | GC | MGL | GJ. -PE | -LC. -PE | -MLL. -PE | -LJ.
-NE | GC | MGL | GJ. -NE | -LC. -NE | -MLL. -NE | -LJ.
-NO | LC | MLL | LJ. -NO | -GC. -NO | -MGL. -NO | -GJ.
% PARTICULAR ASSUMPTIONS
FC. PE | NE.
end_of_list.
list(passive).
-GJ.
end_of_list.
```

In the above file we did not employ OTTER in its automatic mode but rather we preferred to use the command `set(hyper_res)` enabling the use of hyperresolution as the principal inference method. Moreover, we used two lists: (i) the set of support, where we put all of our original hypotheses (both the rules and the particular assumptions) and (ii) the passive list, where we put
just the negation of our conclusion \(-GJ\), which is clearly a false statement. This has permitted a
direct proof (evidently again by contradiction) of the conclusion that the Gauss–Jacobi method is
the sole appropriate one in the sense that the false assumption \(-GJ\) has not been used at all during
the proof with the obvious exception of the last inference: the contradictory one leading to “false”
\((F)\). The related proof is sufficiently short and it is displayed just below

\begin{verbatim}
---------------- PROOF ----------------
11 [] -FC|G|LJ.
19 [] -PE| -LJ.
23 [] -NE| -LJ.
28 [] FC.
29 [] PE|NE.
30 [] -GJ.
31 [hyper,11,28] G|LJ.
32 [hyper,31,19,29] G|NE.
33 [hyper,32,23,31] G.
34 [binary,33.1,30.1] $F.
-------------- end of proof -------------
\end{verbatim}

It seems that it is rather simple to verify the correctness of this proof. We also observe that
OTTER used just few of the rules in the input file (3 among 27). Yet, the inclusion of all of the
available rules there is recommended since under different conditions (e.g. for an edge crack and/or
for no extrapolation) additional/different rules would/might be used.

Beyond the sufficiently larger data base of rules in the present application (compared to the
previous one), it is also important to remark that in the present case we did not have just Horn
clauses (i.e. clauses with only one positive/non-negated literal) leading to just one conclusion if a set
of hypotheses is simultaneously valid. For example, in the above input file (and in the related proof)
we observe that in the case of an interface crack (FC) we can use either the Gauss–Jacobi (GJ) or
the Lobatto–Jacobi (LJ) method in the related essentially IF-THEN statement. Therefore, we have
a disjunction in the “head” of this statement contrary to what happens in Horn clauses essentially
exclusively used in Prolog. Of course, by using two more rules as well, OTTER understood that
only the Gauss–Jacobi method (GJ) is applicable since we assumed either polynomial extrapolation
(PE) or natural extrapolation (NE) in the numerical method to be used in the final stage of the
approach for the solution of the singular integral equation.

Now we will proceed to a third and final application concerning the case of the existence of
equalities (in the ordinary, real-variable sense) inside the input file as well.

2.3. The linearity property for the deflections of beams

As a final application of OTTER we will consider a simple linearly elastic beam \(B\) under a
normal loading \(Q = Q(x)\). In this problem, we will assume that

\begin{itemize}
  \item IF \(B\) is a beam, \(C\) a real constant and \(Q\) the loading of \(B\), THEN the deflection corresponding
to the loading \(CQ\) is equal to \(C\) times the deflection corresponding to the loading \(Q\).
  \item Similarly, IF \(B\) is a beam and both \(Q_1\) and \(Q_2\) are possible, acceptable loadings of \(B\), THEN
the deflection corresponding to the sum of the loadings \(Q_1 + Q_2\) is equal to the sum of the
deflections corresponding to each particular loading \(Q_1\) and \(Q_2\).
\end{itemize}
These are our two fundamental premises (or rules or axioms) in the present beam problem. Now our task will be simply to prove the linearity property for the present beam $B$, i.e.

- If $B$ is a beam and both $C_1$ and $C_2$ are real constants and, further, both $Q_1$ and $Q_2$ are possible, acceptable loadings of $B$, then the deflection of $B$ corresponding to the linear combination of the loadings $C_1Q_1 + C_2Q_2$ is equal to the same linear combination of the deflections corresponding to each particular loading $Q_1$ and $Q_2$.

If we use the symbols $V_1$ and $V_2$ for these deflections corresponding to the loadings $Q_1$ and $Q_2$, respectively, then we can ask OTTER to prove the aforementioned linear property, i.e.

$$\text{DEFLECTION}(B;C_1Q_1 + C_2Q_2) = C_1V_1 + C_2V_2$$

evidently by adopting again the refutational method (consisting in the denial of the above equation–conclusion), i.e. assuming that the above equation does not hold true. Using the automatic mode of operation of OTTER, we have easily prepared the input file below:

```
set(auto).
op(400, xfx, [*,+]). op(300, yf, @).
set(prolog_style_variables).
list(usable).
% RULES
- BEAM(B) | -REAL(C) | -LOADING(B,Q)
  | DEFLECTION(B,C * Q) = C * DEFLECTION(B,Q).
- BEAM(B) | -LOADING(B,Q1) | -LOADING(B,Q2)
  | DEFLECTION(B,Q1 + Q2) = DEFLECTION(B,Q1) + DEFLECTION(B,Q2).
- BEAM(B) | -REAL(C) | -LOADING(B,Q) | LOADING(B, C * Q).
- BEAM(B) | -LOADING(B,Q1) | -LOADING(B,Q2) | LOADING(B, Q1+Q2).
% DATA
BEAM(b). REAL(c1). REAL(c2). LOADING(b,q1). LOADING(b,q2).
DEFLECTION(b,q1) = v1. DEFLECTION(b,q2) = v2.
DEFLECTION(b, (c1 * q1) + (c2 * q2)) != (c1 * v1) + (c2 * v2).
end_of_list.
```

(with $!=$ denoting the not-equal sign). In this file, we introduced, beyond the two aforementioned fundamental rules, two auxiliary but also, possibly, useful to OTTER rules stating that

- If $B$ is a beam, $C$ a real constant and $Q$ an acceptable loading of $B$, then $CQ$ is also an acceptable loading of $B$.

- Moreover, if $B$ is a beam and both $Q_1$ and $Q_2$ are acceptable loadings of $B$, then $Q_1 + Q_2$ is also an acceptable loading of $B$.

Of course, it is understood that these are just assumptions for the present proof of the linearity property in analogy to elementary mathematics. Frequently, because of fracture or even plasticity/non-linear behaviour phenomena, our original hypotheses may not hold true.

Finally, the above input file to OTTER contains our data (with the symbols there concerning a special beam $b$ and special values, $c_{1,2}$ and $q_{1,2}$, for $C_{1,2}$ and $Q_{1,2}$, respectively) as well as the denial/refutation (i.e. an inequality) of our conclusion so that OTTER can be led to a contradiction. Incidentally, we have also adopted the Prolog style for the symbols of the constants/variables although OTTER took the liberty to change our symbols for the variables.
On the basis of the above input file, OTTER has been really able to find a proof of our assertion that the linearity property holds true in the present beam problem. OTTER’s proof has as follows:

\[ \text{---------------- PROOF ----------------} \]
\[ 1 \mapsto \text{BEAM(A)} \land \text{REAL(B)} \land \text{LOADING(A,C)} \land \text{DEFLECTION(A,B*C)=B*DEFLECTION(A,C)}. \]
\[ 2 \mapsto \text{BEAM(A)} \land \text{LOADING(A,B)} \land \text{LOADING(A,C)} \land \text{DEFLECTION(A,B+C)=DEFLECTION(A,B)+DEFLECTION(A,C)}. \]
\[ 3 \mapsto \text{BEAM(A)} \land \text{REAL(B)} \land \text{LOADING(A,C)} \land \text{LOADING(A,B*C)}. \]
\[ 5 \mapsto \text{DEFLECTION(b,(c1*q1)+(c2*q2))!=(c1*v1)+(c2*v2)}. \]
\[ 7 \mapsto \text{BEAM(b)}. \]
\[ 8 \mapsto \text{REAL(c1)}. \]
\[ 9 \mapsto \text{REAL(c2)}. \]
\[ 10 \mapsto \text{LOADING(b,q1)}. \]
\[ 11 \mapsto \text{LOADING(b,q2)}. \]
\[ 13, 12 \mapsto \text{DEFLECTION(b,q1)=v1}. \]
\[ 15, 14 \mapsto \text{DEFLECTION(b,q2)=v2}. \]
\[ 18 \mapsto \text{hyper,10,3,7,8 LOADING(b,c1*q1)}. \]
\[ 24, 23 \mapsto \text{hyper,10,1,7,8,demod,13 DEFLECTION(b,c1*q1)=c1*v1}. \]
\[ 28 \mapsto \text{hyper,11,3,7,9 LOADING(b,c2*q2)}. \]
\[ 37, 36 \mapsto \text{hyper,11,1,7,9,demod,15 DEFLECTION(b,c2*q2)=c2*v2}. \]
\[ 283 \mapsto \text{hyper,28,2,7,18,demod,24,37 DEFLECTION(b,(c1*q1)+(c2*q2))=(c1*v1)+(c2*v2)}. \]
\[ 285 \mapsto \text{binary,283.1,5.1 $F$.} \]
\[ \text{------------ end of proof ------------} \]

From the above proof we observe that in the automatic mode OTTER employed not only hyperresolution and binary resolution but also demodulation. As was also observed from the complete output file, it also introduced the Knuth–Bendix strategy beyond hyperresolution because of the existing equalities in the related input file.

3. Conclusions

From the above results it is concluded that at least in simple cases automated reasoning/deduction systems (such as here the McCune program OTTER) are able to prove elementary statements concerning engineering mechanics applications. This requires the introduction of the appropriate premises: axioms and data. In rather simple problems (such as the above ones), the selection of a strategy by the user is not necessary and just the OTTER automatic mode is efficiently applicable. On the other hand, in difficult situations, OTTER is not expected to find a proof (even of a correct conclusion) without help. The present results and analogous (even somewhat more difficult) proofs having been derived in fracture and structural mechanics by the author mainly aim at the use of automated reasoning/deduction systems in the broad area of engineering mechanics. Here OTTER was used alone but, generally speaking, the preparation of a convenient, integrated computational environment including numerical and symbolic computations, graphics and automated reasoning capabilities would be extremely welcome. This being a very ambitious task, just the exchange of data (even through computer files) between Maple or Mathematica on the one hand and OTTER on the other hand may be a more realistic extension of the present approach. Moreover, the axiomatization in very special areas of engineering mechanics could also receive the due attention although, undoubtedly, by no means is it an easy task even in elementary cases. In any case, it is hoped that the infancy of the use of automated reasoning/deduction systems in engineering mechanics should and can start in the near future exactly as this happened in mathematics and logic more than forty years ago.
References


