Structural vs. Cyclic Induction

Sorin Stratulat

Structural vs. cyclic induction
- a report on some experiments with Coq -

Sorin Stratulat
LITA, Department of Computer Science
Université de Lorraine
Metz, France
Email: sorin.stratulat@univ-lorraine. fr

***Abstract*—Structural and (Noetherian) cyclic induction are two instances of the Noetherian induction principle adapted to reason on first-order logic. From a theoretical point of view, every structural proof can be converted to a cyclic proof but the other way is only conjectured. From a practical point of view, i) structural induction principles are built-in or automatically issued from the analysis of recursive data structures by many theorem provers, and ii) the implementation of cyclic induction reasoning may require additional resources such as functional schemas, libraries and human interaction.**

In this paper, we firstly define a set of conjectures that can be proved by using cyclic induction and following a similar scenario. Next, we implement the cyclic induction reasoning in the Coq proof assistant. Finally, we show that the scenarios for proving these conjectures with structural induction differ in terms of the number of induction steps and lemmas, as well as proof scenario. We identified three conjectures from this set that are hard or impossible to be proved by structural induction.

***Index Terms*—mechanical reasoning, Noetherian induction, structural induction, cyclic induction, Coq**

I. Introduction

In [8], it has been shown that structural and (Noetherian) cyclic induction are two instances of the Noetherian induction principle adapted to reason on first-order logic. The cyclic induction principles are *formula-based* since they allow to prove the validity of sets of formulas. On the other hand, the structural induction principles are *term-based* because they allow to prove that a given formula holds for any value of some (induction) variable, provided that its datatype is recursively defined. From a theoretical point of view, every structural induction proof can be converted to a cyclic proof but the other way is only conjectured.

In this paper, we compare the two approaches while proving a given set of conjectures using the Coq proof assistant [11]. In Coq, the structural induction principles are automatically issued from the analysis of recursive data structures and their application can be easily performed by built-in tactics. This is not the case for the cyclic induction principles, even if they fit better to deal with i) mutual induction, i.e., when the proof of a formula ϕ_1 requires as induction hypothesis an instance of another formula ϕ_2 and, viceversa, the proof of ϕ_2 requires as induction hypothesis an instance of ϕ_1, and ii) lazy induction, i.e., the induction hypotheses need to be generated only when they are effectively applied in the proof.

The rest of the paper consists of four sections. In Section II, we define the set of conjectures serving to our comparison. Section III shows i) that the cyclic proof for any conjecture can be done following a similar scenario, and ii) how the cyclic reasoning can be implemented in Coq, using additional resources as functional schemas, libraries and non-trivial human interaction. Different scenarios for proving the conjectures using structural induction are presented in Section IV, and show that they differ in terms of number of induction steps and lemmas, as well as proof scenario.[1] Section V gives the conclusions and outlines future work.

II. Defining the set of conjectures

Let $\mathcal{R}$ be a set of ternary inductive predicate symbols taking natural numbers as arguments such that each symbol $R \in \mathcal{R}$ is defined by a set of axioms of the form:

$$R(0, u, 0) \tag{1}$$
$$R(x_1, y_1, y_2) \Rightarrow R(S(x), u, 0) \tag{2}$$
$$R(x'_1, y'_1, y'_2) \Rightarrow R(0, u, S(v)) \tag{3}$$
$$R(x_1, y_1, y_2) \wedge R(x'_1, y'_1, y'_2) \Rightarrow R(S(x), u, S(v)) \tag{4}$$

where S is the 'successor' function and the variables x, u, v are universally quantified. The values of the parameters x_1, y_1, y_2 and x'_1, y'_1, y'_2 of R occurring in the condition part of the axioms are defined in order to satisfy the following ordering constraints: i) $R(x_1, y_1, y_2) < R(S(x), u, 0)$, ii) $R(x'_1, y'_1, y'_2) < R(0, u, S(v))$, iii) $R(x_1, y_1, y_2) < R(S(x), u, S(v))$, and iv) $R(x'_1, y'_1, y'_2) < R(S(x), u, S(v))$, by using a well-founded ordering $<$. This ordering is defined such that $R(z_1, z_2, z_3) < R(z'_1, z'_2, z'_3)$ if $\{\{z_1, z_1\}, \{z_2, z_2, z_3\}\} \ll \{\{z'_1, z'_1\}, \{z'_2, z'_2, z'_3\}\}$, for any naturals $z_1, z_2, z_3, z'_1, z'_2, z'_3$. Here, $\ll$ is the multiset extension of an ordering over multisets of terms which, in turn, is the multiset extension of the rpo ordering [1], defined over naturals and denoted by $<_t$, based on the precedence over the function symbols stating that 0 is smaller than S. It can be shown that this rpo ordering is well-founded and satisfies, for example, that $0 <_t S(x)$ and $x <_t S(x)$, for any natural x. Since every multiset extension of a well-founded ordering is also well-founded, we conclude that $<$ is well-founded.

We use the fact that a multiset A is smaller than another multiset B w.r.t. the multiset extension of some ordering $\prec$ if, after pairwisely deleting the common elements from A and B we get the multisets A' and B', respectively, and, for each element x in A', there is an element y in B' such that $x \prec y$. In our case, the ordering constraints

[1]The full Coq scripts of the proofs can be found at `lita.univ-lorraine. fr/~stratula/synasc2016.zip`

1) i) and iii) are, respectively, $\{\{x_1, x_1\}, \{y_1, y_1, y_2\}\} \ll \{\{S(x), S(x)\}, \{u, u, 0\}\}$ and $\{\{x_1, x_1\}, \{y_1, y_1, y_2\}\} \ll \{\{S(x), S(x)\}, \{u, u, S(v)\}\}$.
   They are satisfied if x_1 is an element from $\{0, u, x\}$ and the pair (y_1, y_2) an element from $\{(0,0), (0,x), (x,0), (0,S(x)), (x,S(x)), (x,x)\}$.
   Notice that the pairs of the form $(S(x), \_)$ and the pairs including u cannot be assigned to (y_1, y_2);
2) ii) and iv) are, respectively, $\{\{x'_1, x'_1\}, \{y'_1, y'_1, y'_2\}\} \ll \{\{0,0\}, \{u, u, S(v)\}\}$ and $\{\{x'_1, x'_1\}, \{y'_1, y'_1, y'_2\}\} \ll \{\{S(x), S(x)\}, \{u, u, S(v)\}\}$. They are also satisfied if x'_1 is an element from $\{0, u, v\}$ and (y'_1, y'_2) is from $\{(0,0), (0,u), (0,v), (u,v), (v,u), (u,0), (v,0), (v,v)\}$. The pair (u,u) cannot be assigned to (y'_1, y'_2).

Therefore, the set $\mathcal{R}$ will have $3 \times 6 \times 3 \times 8 = 432$ inductive predicate symbols. Finally, the set of conjectures for our purpose is $\{\forall x\ u\ v, R(x,u,v) \mid R \in \mathcal{R}\}$.

III. Proving by cyclic induction

For any $R \in \mathcal{R}$, the conjecture $\forall x\ u\ v, R(x,u,v)$ can be proved by cyclic induction reasoning using only variable instantiations and unfoldings with the axioms defining R, as shown in the proof graph from Fig. 1.

The root node is labeled by $R(x,u,v)$, the other nodes being labeled by inductive atoms that are instances of it. Each non root-node n is pointed by a solid arrow starting from some other node n'. If p' is the inductive atom labelling n', then the inductive atom labelling n results either i) by instantiating some variable from p' by 0 and $S(x)$, where x is a fresh variable, or ii) by unfolding p' using one of the conditional axioms (3)-(4). In the first case, the instantiating substitution annotates the corresponding solid arrow. The inductive atom labeling each leaf node either instantiates (1) or the inductive atom labeling the root node. In the last case, a dashed arrow is firstly created by leading the leaf node to the root node, then annotated with the instantiating substitution, written in boldface.

The proof graph from Fig. 1 contains cycles by following the arrows in the graph. In general, not all proof derivations, for which the root formula is instantiated by leaf formulas, are sound. In our case, the soundness is guaranteed by the ordering constraints i) - iv), as shown by the cyclic induction method from [8].

The Coq implementation. Let us assume that R is one of the inductive predicates symbols from $\mathcal{R}$, defined by the axioms:

$$R(0,u,0) \tag{5}$$
$$R(u,x,S(x)) \Rightarrow R(S(x),u,0) \tag{6}$$
$$R(v,u,v) \Rightarrow R(0,u,S(v)) \tag{7}$$
$$R(u,x,S(x)) \wedge R(v,u,v) \Rightarrow R(S(x),u,S(v)) \tag{8}$$

We will show how the cyclic induction reasoning for proving $\forall x\ u\ v, R(x,u,v)$ can be certified in Coq.

R can be specified in Coq as an inductive predicate, denoted here by **R**:

```
Inductive R: nat → nat → nat → Prop :=
r_1: ∀ u, R 0 u 0 |
r_2: ∀ x u, R u x (S x) → R (S x) u 0 |
r_3: ∀ u v, R v u v → R 0 u (S v) |
r_4: ∀ x u v, R u x (S x) → R v u v → R (S x) u (S v).
```

The scenario from the cyclic proof from Fig. 1 can be reproduced if the conjecture to be proved is (temporarily) considered as an hypothesis before its usage.

```
Hypothesis R_admitted: ∀ x u v, R x u v.

Theorem R_assumption: ∀ x u v, R x u v.
Proof.
  destruct x ; intros.
  - Case "x=0". destruct v.
     + SCase "v=0". apply r_1.
     + SCase "v=S v". apply r_3. apply R_admitted.
  - Case "x=S x". destruct v.
     + SCase "v=0". apply r_2. apply R_admitted.
     + SCase "v=S v". apply r_4; apply R_admitted.
Qed.
```

The tactic `destruct`, when applied on a natural variable v, instantiates it by 0 and (S v).

In [9], it has been shown how to normalize the graph of a cyclic proof and boil down the induction reasoning to the strongly connected components of the normalized graph. It can be easily noticed that the cycles from the proof graph from Fig. 1 form a unique strongly connected component. The induction reasoning performed along these cycles can be captured by an explicit induction schema issued from the definition of a terminating and recursive boolean function, denoted by f_P, taking as argument a triplet of naturals.

```
Function f_P (a: nat × nat × nat) {wf (fun u v: nat×
nat×nat ⇒ match u,v with
      (u1, x1, y1), (u2, x2, y2) ⇒ mless ({{[u1,
u1]}} + {{[x1, x1, y1]}}) ({{[u2, u2]}} + {{[x2,
x2, y2]}}) end) a}: bool :=
 match a with
     | (x', y) ⇒ match x' with
                 | (u, x) ⇒
                   match u, x, y with
                     | 0, _, 0 ⇒ true
                     | (S x), u, 0 ⇒ f_P (u, x, (S x))
                     | 0, u, (S v) ⇒ f_P (v, u, v)
                     | (S x), u, (S v) ⇒ andb (f_P (v,
u, v)) (f_P (u, x, (S x)))
                   end
                 end
 end.
```

The function f_P firstly decomposes the triplet given as argument, then performs a case analysis on the resulting naturals to finally get a different (functional) representation of the definition of **R**.

As any function whose definition is accepted by Coq, f_P should terminate. The Coq environment generates proof obligations requiring that its argument should decrease after each recursive call w.r.t. the well-founded ordering provided

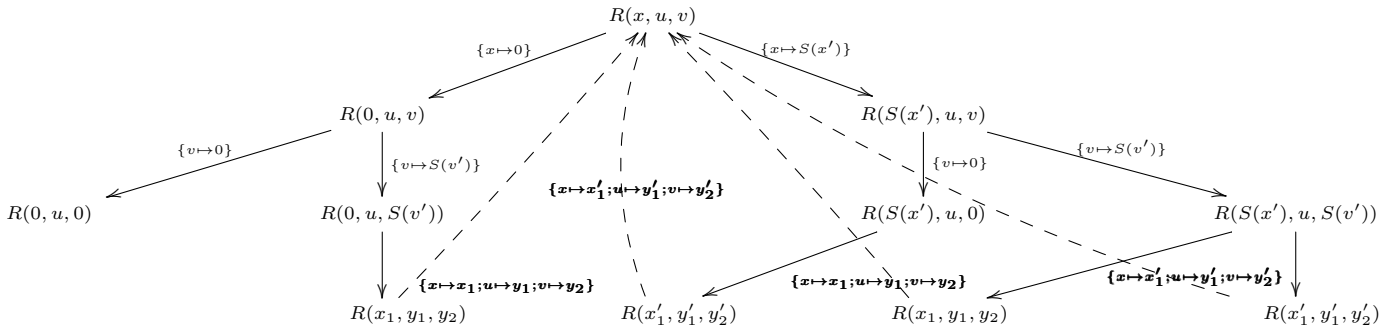

Fig. 1. The graph of the cyclic proof of $\forall x\ u\ v, R(x,u,v)$, for any $R \in \mathcal{R}$.

after the `wf` keyword, where mless is the multiset extension of the multiset extension of the 'less than' ordering over naturals, defined using the CoLoR library [2]. Once the proof obligations are discharged, Coq automatically generates a functional (induction) schema f_P_ind.

An explicit induction proof can be built for the theorem RP_true which is similar to R_assumption when using RP, the version of **R** with only one (triplet) argument.

```
Definition RP z := R (fst (fst z)) (snd (fst z)) (snd z).

Theorem RP_true: ∀ u x y, RP (u, x, y).
Proof.
  intros u x y. pattern (u, x, y). pattern (f_P (u, x, y)).
  apply f_P_ind; intros; unfold RP; simpl. (* apply the induction schema *)
  apply r_1.                    (* follow the cyclic proof *)
  apply r_2; unfold RP in H; simpl in H; trivial.
  apply r_3. unfold RP in H; simpl in H. trivial.
  apply r_4; unfold RP in H0; simpl in H0; trivial.
Qed.
```

Finally, the *R_admitted* hypothesis can be proved.

```
Theorem R_admitted: ∀ x u v, R x u v.
  (* the proof follows directly from RP_true *)
```

Similarly, $\forall\ x\ u\ v$, $R'(x,u,v)$ can be certified in Coq, for any $R' \in \mathcal{R}$, using a similar scenario for which only the termination proof is different.

IV. Proving by structural induction

We will try to prove the conjectures $\forall\ x\ u\ v$, $R'(x,u,v)$, where $R' \in \mathcal{R}$, using this time *Peano induction* which is a structural induction principle issued from the analysis of the recursive definition of naturals: a natural can be either 0 or the successor of another natural. According to it, in order to prove a formula $P(x)$, where x is a natural variable to be instantiated, also called *induction variable*, it is enough to prove both $P(0)$ and $\forall x', P(x') \Rightarrow P(S(x'))$, where $P(x')$ is an *induction hypothesis* and x' a fresh variable. $P(x')$ can be soundly used in the proof of $P(S(x'))$ because the number of 'S' symbols in x' is smaller than in $S(x')$, for any natural x', and the 'less than' ordering over naturals is well-founded. In Coq, the Peano induction can be applied using the tactic `induction` which takes as argument the induction variable.

To distinguish each R' from $\mathcal{R}$, they are represented under the form of $Rijkl$, where

- i is the position of x_1 from the axioms (2) and (4) in the list $[0,u,x]$,
- j is the position of (y_1,y_2) from the axioms (2) and (4) in the list $[(0,0),(0,x),(x,0),(0,S(x)),(x,S(x)),(x,x)]$,
- k is the position of x'_1 from the axioms (3) and (4) in the list $[0,u,v]$, and
- l is the position of (y'_1,y'_2) from (3) and (4) in $[(0,0),(0,u),(0,v),(u,v),(v,u),(u,0),(v,0),(v,v)]$.

For example, the symbol R, defined by the axioms (5)-(8), will be referred to as $R2534$.

Similarly in Coq, the axioms defining $Rijkl$, denoted by **R**$ijkl$, are labelled as r$ijkl$_1, r$ijkl$_2, r$ijkl$_3, and r$ijkl$_4. In addition, the theorem to be proved is denoted as R$ijkl$_true.

All proofs of R$ijkl$_true have the following structure:

```
Theorem Rijkl_true: ∀ x u v, Rijkl x u v.
Proof.
  destruct x; destruct v; intros.
  (* case x=0, v=0 *) apply rijkl_1.
  (* case x=0, v=(S v) *) apply rijkl_3. (* to complete *)
  (* case x=(S x), v=0 *) apply rijkl_2. (* to complete *)
  (* case x=(S x), v=(S v) *) apply rijkl_4. (*to complete*)
Qed.
```

This proof scenario is similar to that from Fig. 1, excepting that the proof of the formulas labelling the lowest nodes in the proof graph should be provided.

Different proof scenarios can be distinguished to complete the proof of $Rijkl$_true. TABLE I presents some proof statistics for each case, where the second (resp., third) column gives the number of `induction` (resp., `destruct`) calls, and the fourth column the number of times '`apply` r$ijkl$_4' was called. The fifth column displays the number of intermediate lemmas and the last column the depth of the proof tree, issued by expanding the calls to the lemmas and by considering as one 'big step' each sequence of steps different from `induction`, `destruct`, `split` (for dealing with conjunctions) and '`apply` r$ijkl$_4'. We do not claim that our proofs have a graph with minimal depth, our intention

is only to give an idea about the degree of difficulty of each proof.

*a) Scenario 1: no induction steps. The case **R1111**:* this is the most trivial one.

```
Theorem R1111_true: ∀ x u v, R1111 x u v.
Proof.
  destruct x; destruct v; intros.
  apply r1111_1.
  apply r1111_3. apply r1111_1.
  apply r1111_2. apply r1111_1.
  apply r1111_4; apply r1111_1.
Qed.
```

Overall, 78 cases have no induction steps in their proofs. These proofs are performed mainly by instantiating variables and unfolding axioms, the maximal depth being of 7 (e.g., for R1132_true).

*b) Scenario 2: one induction step. The case **R1113**:* an induction step is performed in the proof of the additional lemma:

```
Theorem R1113_10v: ∀ v, R1113 0 0 v.
Proof.
  induction v.
   - Case "v=0" . apply r1113_1.
   - Case "v=S v". apply r1113_3. apply IHv.
Qed.

Theorem R1113_true: ∀ x u v, R1113 x u v.
Proof.
  destruct x; destruct v; intros. apply r1113_1.
  apply r1113_3. apply R1113_10v.
  apply r1113_2. apply r1113_1.
  apply r1113_4; apply R1113_10v.
Qed.
```

*c) Scenario 3: two induction steps. The case **R1115**:* two induction steps are performed in the proof of the conjunction lemma:

```
Theorem R1115_mix: ∀ u v, R1115 0 v u ∧ R1115 0 u v.
Proof.
  induction u; intros.
  - Case "u=0". split. apply r1115_1. destruct v.
apply r1115_1. apply r1115_3. apply r1115_1.
  - Case "u=S u". split. apply r1115_3. apply IHu.
induction v.
     + SCase "v=0". apply r1115_1.
     + SCase "v = S v". apply r1115_3. apply r1115_3.
apply IHu.
Qed.

Theorem R1115_true: ∀ x u v, R1115 x u v.
Proof.
  destruct x; destruct v; intros. apply r1115_1.
  apply r1115_3. apply R1115_mix.
  apply r1115_2. apply r1115_1.
  apply r1115_4; apply R1115_mix.
Qed.
```

*d) Scenario 4: more than two induction steps. The cases **R2535** and **R2534**:* the proof of R2535_true required 3 induction steps and its graph has the deepest depth (19). On the other hand, in spite of our efforts, the proof of R2534_true, as well as R2634_true and R2636_true, could not have been completed.

V. Conclusions and future work

We have defined a set of conjectures that can be proved using cyclic induction by following a similar scenario. We have shown how to implement the cyclic induction reasoning in Coq, by means of external libraries and functional schemas issued from the definition of new function symbols. On the other hand, all but three conjectures have also been proved by structural induction. We have shown that these proofs are very different in terms of scenario, length and difficulty.

We think that the cyclic proofs can be easily automatised in Coq, the main challenge being the automatic generation of the termination proof when defining the new function symbols. This is not the case for the proofs by structural induction which may require major human interaction.

In the future, we plan to generalize the implementation in Coq of the cyclic reasoning for arbitrary proofs. For this, we may use different libraries of orderings, like Coccinelle [4], [5], and follow ideas that have been successfully tested for implementing the implicit induction reasoning [6], [7], [9], [10]. From a theoretical point of view, we plan to use the three 'not yet' proved conjectures to check the Brotherston-Simpson conjecture [3], asserting the equivalence between cyclic and explicit induction reasoning for first-order logic with inductive definitions.

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[2] F. Blanqui and A. Koprowski. CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *MSCS*, 21(4):827–859, 2011.

[3] J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

[4] E. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. *Frontiers of Combining Systems*, pages 148–162, 2007.

[5] E. Contejean, A. Paskevich, X. Urbain, P. Courtieu, O. Pons, and J. Forest. A3PAT, an approach for certified automated termination proofs. In J. P. Gallagher and J. Voigtländer, editors, *PEPM - Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2010, Madrid, Spain, January 18-19, 2010*, pages 63–72. ACM, 2010.

[6] A. Henaien and S. Stratulat. Performing implicit induction reasoning with certifying proof environments. In A. Bouhoula, T. Ida, and F. Kamareddine, editors, *Proceedings Fourth International Symposium on Symbolic Computation in Software Science, Gammarth, Tunisia, 15-17 December 2012*, volume 122 of *Electronic Proceedings in Theoretical Computer Science*, pages 97–108. Open Publishing Association, 2013.

[7] S. Stratulat. Integrating implicit induction proofs into certified proof environments. In *IFM'2010 (8th International Conference on Integrated Formal Methods)*, volume 6396 of *Lecture Notes in Computer Science*, pages 320–335, 2010.

[8] S. Stratulat. A unified view of induction reasoning for first-order logic. In A. Voronkov, editor, *Turing-100 (The Alan Turing Centenary Conference)*, volume 10 of *EPiC Series*, pages 326–352. EasyChair, 2012.

[9] S. Stratulat. Mechanically certifying formula-based Noetherian induction reasoning. *Journal of Symbolic Computation*, 80, Part 1:209–249, 2017.

[10] S. Stratulat and V. Demange. Automated certification of implicit induction proofs. In *CPP'2011 (First International Conference on Certified Programs and Proofs)*, volume 7086 of *Lecture Notes Computer Science*, pages 37–53. Springer Verlag, 2011.

[11] The Coq development team. *The Coq Reference Manual - version 8.5*. INRIA, 2016.

TABLE I: Statistics about the proofs by Peano induction of $Rijkl$_true.

Case	IS	DS	CR	Lemmas	Depth	Case	IS	DS	CR	Lemmas	Depth
R1111	0	2	1	0	3	**R2411**	0	3	2	1	5
R1112	0	3	1	1	4	**R2412**	0	4	2	2	5
R1113	1	2	1	1	4	**R2413**	1	3	2	2	6
R1114	1	2	1	1	4	**R2414**	1	3	2	2	6
R1115	2	2	1	1	6	**R2415**	1	5	2	2	5
R1116	0	2	1	0	3	**R2416**	0	3	2	1	5
R1117	0	2	1	0	3	**R2417**	0	3	2	1	5
R1118	1	2	1	1	4	**R2418**	1	3	2	2	6
R1121	0	3	1	1	4	**R2421**	0	4	2	2	6
R1122	1	2	2	1	5	**R2422**	0	4	2	2	5
R1123	1	4	2	1	6	**R2423**	3	6	3	2	9
R1124	1	4	2	1	6	**R2424**	4	5	3	2	9
R1125	2	2	3	1	7	**R2425**	1	4	5	2	9
R1126	0	3	1	1	4	**R2426**	0	4	2	2	6
R1127	0	3	1	1	4	**R2427**	0	4	2	2	6
R1128	1	4	2	1	6	**R2428**	1	5	5	3	9
R1131	0	3	1	1	4	**R2431**	1	3	2	2	6
R1132	0	6	2	1	7	**R2432**	4	8	3	2	9
R1133	1	2	2	1	5	**R2433**	1	3	3	2	7
R1134	1	2	2	1	5	**R2434**	7	5	11	2	12
R1135	1	6	3	1	8	**R2435**	3	6	7	2	12
R1136	0	4	1	1	5	**R2436**	3	5	2	2	7
R1137	0	3	1	1	4	**R2437**	1	6	4	2	8
R1138	1	2	2	1	5	**R2438**	1	3	4	2	8
R1211	0	3	1	1	4	**R2511**	1	3	4	1	7
R1212	0	3	1	1	4	**R2512**	1	5	4	2	8
R1213	1	2	1	1	3	**R2513**	2	4	4	2	8
R1214	1	2	1	1	4	**R2514**	2	5	3	2	9
R1215	1	4	1	1	5	**R2515**	2	6	4	2	9
R1216	0	3	1	1	4	**R2516**	1	4	4	2	8
R1217	0	3	1	1	4	**R2517**	1	4	4	1	7
R1218	1	3	1	2	4	**R2518**	2	4	4	2	8
R1221	0	4	1	2	4	**R2521**	2	4	4	2	8
R1222	0	4	2	2	6	**R2522**	2	4	5	2	9
R1223	1	4	2	2	7	**R2523**	4	6	4	2	11
R1224	2	4	2	2	6	**R2524**	4	4	7	2	11
R1225	1	4	3	2	8	**R2525**	3	6	16	3	16
R1226	0	4	1	2	5	**R2526**	2	4	8	2	11
R1227	0	4	1	2	5	**R2527**	2	5	8	3	11
R1228	1	5	3	3	10	**R2528**	2	6	17	3	18
R1231	1	2	1	3	5	**R2531**	2	4	4	2	8
R1232	1	4	2	3	8	**R2532**	2	6	7	3	13
R1233	1	3	2	3	7	**R2533**	2	4	7	2	9
R1234	2	2	3	3	8	**R2534**	-	-	-	-	-
R1235	2	5	3	3	9	**R2535**	3	7	18	3	19
R1236	1	3	1	3	6	**R2536**	2	5	9	3	13
R1237	1	2	1	3	5	**R2537**	2	4	11	2	12
R1238	1	2	1	3	6	**R2538**	2	4	14	2	13
R1311	0	2	1	0	3	**R2611**	1	5	3	1	8
R1312	0	3	1	1	4	**R2612**	1	8	3	2	8
R1313	1	2	1	1	4	**R2613**	2	7	3	2	9
R1314	1	2	1	1	4	**R2614**	2	8	4	2	11
R1315	1	4	1	1	5	**R2615**	2	5	4	2	10
R1316	0	3	1	1	4	**R2616**	1	6	3	1	8

TABLE I – Continued

Case	IS	DS	CR	Lemmas	Depth	Case	IS	DS	CR	Lemmas	Depth
R1317	0	2	1	0	3	**R2617**	1	6	3	1	8
R1318	1	2	1	1	4	**R2618**	2	7	3	2	9
R1321	0	3	1	1	4	**R2621**	2	8	3	2	10
R1322	0	3	1	1	4	**R2622**	2	8	5	2	12
R1323	1	4	2	1	6	**R2623**	6	9	5	3	14
R1324	1	4	2	1	6	**R2624**	10	2	7	3	13
R1325	1	3	3	1	8	**R2625**	3	7	17	3	18
R1326	0	3	1	1	4	**R2626**	2	6	5	2	14
R1327	0	3	1	1	4	**R2627**	2	7	6	3	15
R1328	1	4	2	1	6	**R2628**	2	7	8	2	16
R1331	0	3	1	1	4	**R2631**	2	7	3	2	12
R1332	0	7	2	1	7	**R2632**	4	10	5	3	13
R1333	1	2	2	1	5	**R2633**	2	6	4	2	13
R1334	1	2	2	1	5	**R2634**	-	-	-	-	-
R1335	1	6	3	1	8	**R2635**	3	9	11	4	16
R1336	0	3	1	1	4	**R2636**	-	-	-	-	-
R1337	0	3	1	1	4	**R2637**	2	7	6	2	15
R1338	1	2	2	1	5	**R2638**	2	7	6	2	14
R1411	0	2	1	1	3	**R3111**	1	2	1	1	4
R1412	0	3	1	2	4	**R3112**	1	3	1	1	4
R1413	1	2	1	2	4	**R3113**	2	2	1	2	4
R1414	1	2	1	2	4	**R3114**	2	2	1	2	4
R1415	1	4	1	1	5	**R3115**	2	4	1	2	5
R1416	0	3	1	2	4	**R3116**	1	2	1	1	4
R1417	0	2	1	1	3	**R3117**	1	2	1	1	4
R1418	1	2	1	2	4	**R3118**	2	2	1	2	4
R1421	0	3	1	2	4	**R3111**	1	2	1	1	4
R1422	0	3	2	2	5	**R3122**	1	3	2	2	5
R1423	3	4	2	1	7	**R3123**	4	4	2	2	7
R1424	2	4	2	2	6	**R3124**	2	4	2	2	7
R1425	1	3	3	2	7	**R3125**	2	4	4	2	8
R1426	0	3	1	2	4	**R3126**	1	3	1	2	4
R1427	0	3	1	2	4	**R3127**	1	3	1	2	5
R1428	2	4	3	3	8	**R3128**	3	5	4	2	9
R1431	1	2	1	2	4	**R3131**	1	3	1	2	5
R1432	4	5	2	1	8	**R3132**	1	4	2	2	7
R1433	1	2	2	2	5	**R3133**	2	2	2	2	6
R1434	2	2	3	2	5	**R3134**	2	2	2	2	6
R1435	2	5	3	3	9	**R3135**	4	8	4	2	11
R1436	1	3	1	3	5	**R3136**	1	4	1	2	6
R1437	1	2	1	2	4	**R3137**	1	3	1	2	5
R1438	1	2	2	2	5	**R3138**	2	2	2	2	6
R1511	0	2	1	1	3	**R3211**	1	2	2	1	5
R1512	0	3	1	2	4	**R3212**	1	3	2	2	5
R1513	1	2	1	1	4	**R3213**	1	3	2	2	5
R1514	1	3	1	2	5	**R3214**	3	2	2	2	6
R1515	1	4	1	2	5	**R3215**	2	4	2	2	5
R1516	0	3	1	2	4	**R3216**	1	2	2	1	5
R1517	0	2	1	1	3	**R3217**	1	2	2	1	5
R1518	1	2	1	2	4	**R3218**	2	2	2	2	6
R1521	1	2	1	2	4	**R3211**	1	3	2	2	6
R1522	1	2	2	2	5	**R3222**	2	2	3	2	5
R1523	4	4	3	3	7	**R3223**	5	4	3	2	7
R1524	3	2	2	2	6	**R3224**	3	4	3	2	9

TABLE I – Continued

Case	IS	DS	CR	Lemmas	Depth	Case	IS	DS	CR	Lemmas	Depth
R1525	2	3	4	3	7	**R3225**	2	4	5	2	9
R1526	1	2	1	2	4	**R3226**	1	3	2	2	6
R1527	1	2	1	2	4	**R3227**	1	3	2	2	6
R1528	1	5	5	2	10	**R3228**	1	4	5	2	11
R1531	1	2	1	2	4	**R3231**	1	2	2	1	6
R1532	1	2	3	3	9	**R3232**	1	4	3	2	9
R1533	1	2	2	2	5	**R3233**	1	4	2	1	6
R1534	1	2	2	2	5	**R3234**	2	2	3	2	7
R1535	2	6	6	3	13	**R3235**	2	6	5	2	13
R1536	1	3	1	3	5	**R3236**	1	3	2	2	7
R1537	1	2	1	2	4	**R3237**	1	2	2	1	6
R1538	1	2	2	2	5	**R3238**	1	2	3	1	6
R1611	0	3	1	1	4	**R3311**	1	2	2	1	4
R1612	0	4	1	2	5	**R3312**	1	3	1	2	4
R1613	1	3	1	2	5	**R3313**	2	2	1	2	4
R1614	1	3	1	2	5	**R3314**	2	2	1	2	4
R1615	1	4	1	2	5	**R3315**	2	4	1	2	5
R1616	0	3	1	2	4	**R3316**	1	2	1	1	4
R1617	0	3	1	2	4	**R3317**	1	2	1	1	4
R1618	1	3	1	2	5	**R3318**	2	2	1	2	4
R1621	2	2	1	2	5	**R3311**	1	3	1	2	5
R1622	1	2	3	1	6	**R3322**	1	3	1	2	5
R1623	4	5	3	2	10	**R3323**	3	5	2	2	7
R1624	8	3	3	3	11	**R3324**	2	4	2	2	7
R1625	2	4	7	2	11	**R3325**	2	4	4	2	8
R1626	1	3	1	2	5	**R3326**	1	3	2	2	5
R1627	1	3	1	2	3	**R3327**	1	4	2	2	5
R1628	2	4	4	2	8	**R3328**	2	6	4	4	10
R1631	1	2	1	1	5	**R3331**	1	3	1	2	5
R1632	3	4	3	2	10	**R3332**	1	7	2	2	8
R1633	1	2	2	1	6	**R3333**	2	2	2	2	7
R1634	1	2	2	1	6	**R3334**	2	2	2	2	6
R1635	2	6	5	2	12	**R3335**	2	4	3	2	9
R1636	1	2	1	1	5	**R3336**	1	3	1	2	5
R1637	1	2	1	1	5	**R3337**	1	3	1	2	5
R1638	1	2	2	1	6	**R3338**	2	2	2	2	6
R2111	0	3	1	1	4	**R3411**	1	2	2	1	5
R2112	0	4	1	2	4	**R3412**	1	3	2	2	5
R2113	1	3	1	2	4	**R3413**	2	2	2	2	6
R2114	1	3	1	2	4	**R3414**	3	2	2	2	6
R2115	1	5	1	2	5	**R3415**	2	4	2	2	5
R2116	0	4	1	2	4	**R3416**	1	3	2	2	5
R2117	0	4	2	2	4	**R3417**	1	2	2	1	5
R2118	1	3	1	2	4	**R3418**	2	2	2	2	6
R2121	0	3	1	1	4	**R3421**	1	3	2	2	6
R2122	0	4	2	2	4	**R3422**	1	3	2	2	6
R2123	2	5	2	2	8	**R3423**	4	5	3	2	9
R2124	1	5	2	2	6	**R3424**	3	4	3	2	9
R2125	1	6	4	2	9	**R3425**	2	4	4	2	9
R2126	0	4	1	2	4	**R3426**	1	3	2	2	6
R2127	0	4	1	2	4	**R3427**	1	3	2	2	6
R2128	1	5	2	2	5	**R3428**	1	4	5	2	10
R2131	0	3	1	1	4	**R3431**	1	3	2	2	6
R2132	0	6	2	2	6	**R3432**	1	7	3	2	9

TABLE I – Continued

Case	IS	DS	CR	Lemmas	Depth	Case	IS	DS	CR	Lemmas	Depth
R2133	1	3	2	2	5	**R3433**	1	2	4	1	7
R2134	1	3	2	2	5	**R3434**	2	2	5	2	9
R2135	1	7	3	2	8	**R3435**	2	5	4	2	11
R2136	0	5	1	2	5	**R3436**	1	3	2	2	7
R2137	0	4	1	2	4	**R3437**	1	2	2	1	7
R2138	1	3	2	2	5	**R3438**	1	2	4	1	7
R2211	0	7	2	1	7	**R3511**	1	2	2	1	5
R2212	0	5	2	2	7	**R3512**	1	3	2	2	5
R2213	1	4	2	2	7	**R3513**	2	2	2	2	6
R2214	1	6	2	2	7	**R3514**	2	4	2	2	7
R2215	1	9	2	2	7	**R3515**	2	4	2	2	7
R2216	0	7	2	1	6	**R3516**	1	3	2	2	5
R2217	0	7	2	1	6	**R3517**	1	2	2	1	5
R2218	1	7	2	2	8	**R3518**	2	2	2	2	6
R2221	0	9	2	2	7	**R3521**	2	2	2	2	6
R2222	0	9	3	2	7	**R3522**	1	3	4	2	8
R2223	3	4	2	1	7	**R3523**	4	5	4	2	10
R2224	4	6	3	2	10	**R3524**	3	2	2	1	6
R2225	1	9	4	2	11	**R3525**	2	5	9	2	12
R2226	0	8	2	2	8	**R3526**	1	3	2	2	8
R2227	0	9	2	2	8	**R3527**	1	3	2	2	6
R2228	0	6	3	2	9	**R3528**	1	4	9	2	12
R2231	0	4	2	2	8	**R3531**	1	3	2	2	6
R2232	1	4	2	2	8	**R3532**	1	4	4	2	10
R2233	1	4	2	2	9	**R3533**	1	2	4	1	7
R2234	2	4	4	3	9	**R3534**	1	2	4	1	7
R2235	2	11	6	3	17	**R3535**	2	6	5	2	13
R2236	1	5	2	3	8	**R3536**	1	3	2	2	7
R2237	1	4	3	2	9	**R3537**	1	2	2	1	6
R2238	1	5	4	2	10	**R3538**	1	2	4	1	7
R2311	1	4	1	1	5	**R3611**	1	2	2	1	5
R2312	1	5	1	2	5	**R3612**	1	3	2	2	5
R2313	2	4	1	2	5	**R3613**	2	2	2	2	6
R2314	2	4	1	2	5	**R3614**	2	4	2	2	7
R2315	2	6	1	2	5	**R3615**	2	4	2	2	7
R2316	0	6	1	2	5	**R3616**	1	2	2	1	5
R2317	1	4	1	1	5	**R3617**	1	2	2	1	5
R2318	2	4	1	2	5	**R3618**	2	2	2	2	6
R2321	1	5	1	2	5	**R3621**	1	3	2	2	6
R2322	1	5	1	2	5	**R3622**	1	3	4	2	8
R2323	3	7	2	2	7	**R3623**	4	5	5	2	10
R2324	2	6	2	2	8	**R3624**	3	2	2	1	6
R2325	2	6	4	2	10	**R3625**	2	4	9	2	12
R2326	1	5	1	2	6	**R3626**	1	3	2	2	6
R2327	1	4	1	1	5	**R3627**	1	3	2	2	6
R2328	3	7	4	2	10	**R3628**	1	4	7	2	12
R2331	1	3	1	2	5	**R3631**	1	3	2	2	7
R2332	1	6	2	2	8	**R3632**	1	7	4	2	10
R2333	2	4	2	2	5	**R3633**	1	2	3	1	6
R2334	2	4	2	2	5	**R3634**	1	2	4	1	7
R2335	2	7	3	2	10	**R3635**	2	9	6	2	14
R2336	1	5	1	2	6	**R3636**	1	3	2	2	7
R2337	1	4	1	1	5	**R3637**	1	2	2	1	6
R2338	2	4	2	2	7	**R3638**	1	2	2	1	5