

Reductio ad Absurdum: Planning Proofs by Contradiction

Erica Melis, Martin Pollet, Jörg Siekmann

Universität des Saarlandes and
German Research Center for Artificial Intelligence (DFKI)
66123 Saarbrücken, Germany

Abstract. Sometimes it is pragmatically useful to prove a theorem by contradiction rather than finding a direct proof. Some reductio ad absurdum arguments have made mathematical history and the general issue of if and how a proof by contradiction can be replaced by a direct proof touches upon deep foundational issues such as the legitimacy of tertium non datur arguments in classical vs. intuitionistic foundations. In this paper we are interested in the pragmatic issue when and how to use this proof strategy in everyday mathematics in general and in particular in automated proof planning. Proof planning is a general technique in automated theorem proving that captures and makes explicit proof patterns and mathematical search control. So, how can we proof plan an argument by reductio ad absurdum and when is it useful to do so? What are the methods and decision involved?

1 Introduction

Heuristic guidance plays a major role in mathematical problem solving. This has been an issue in human search behavior for a mathematical proof, see, e.g., [Polya, 1945; Schoenfeld, 1985], as well as in artificial intelligence (see [Newell, 1981]) and in automated theorem proving. Newell argued that Polya's heuristics are beyond the current state of the art in artificial intelligence, which today is no longer the case in general (see [Melis and Meier, 2000]). However, it turned out that each of Polya's heuristics actually represents a whole class of related more specific heuristics.

Heuristics that are mainly *domain-independent* have been incorporated into some early AI-systems for mathematical problem solving, most notably into Gelernter's geometry prover [Gelernter, 1959], Lenat's AM system [Lenat, 1981], and in Woody Bledsoe's work. Gelernter's system used a given diagram to check the satisfiability of subgoals and assigned priorities to goals dependent on the expected length of their solution. AM was based on general heuristics to search for new concepts and conjectures.

The still dominating (purely logic-based) automated theorem proving paradigm – mostly based on the resolution principle [Robinson, 1965] – hardly uses any mathematical knowledge or mathematically inspired heuristics and makes up for this deficiency by its general refinements and ultra-fast search based on

well-engineered representational techniques (see [Robinson and Voronkov., 2001], vol.II, chapter 26). Of course, a resolution-based system always searches for a refutation, i.e, using the insight from Herbrandt's Theorem, the theorem to be shown is negated and there is a proof once the system derives the empty clause as the final contradiction.

The situation is different for proof planning systems [Bundy, 1988], where the proof steps are more general and more human-like holding the promise that traversing the search spaces can be based on (human) mathematical principles. Therefore, knowledge-based proof planning [Melis, 1998a; Melis and Siekmann, 1999; Melis *et al.*, 2006] uses extensive means for heuristic guidance and mathematical control knowledge, which has to be acquired.

In this paper, we propose some manually acquired control knowledge typical for theorems in an undergraduate textbook such as R.G. Bartle and D.R. Sherbert's 'Introduction to Real Analysis' [Bartle and Sherbert, 1982] from which our examples are taken. Our focus is on proofs by contradiction and this exercise serves the purpose to see what knowledge is available, how it can be expressed, and, generally, to shed more light on the use of domain-dependent mathematical control knowledge in automated theorem proving based on proof planning.

Why is the control knowledge interesting that helps to select the proof by contradiction strategy? A first answer is that although proofs by contradiction are relatively frequent in mathematics, mathematicians make this choice usually 'instinctively' and do not reason about it explicitly. That is, we have to bring this implicit knowledge to the surface and make it visible – for machines and students alike. Secondly, since this is obviously a proof strategy that humans use to their advantage, we like to give it to a machine as well.

2 Reductio ad Absurdum

Proofs by contradiction have a long history in mathematics and they were used to advantage already in Greek mathematics. Well known is Euclid's Theorem, which states that there are infinitely many prime numbers. The proof assumes¹ that this is not the case, i.e., the number of primes is finite. Let p_1, p_2, \dots, p_n be all these finitely many primes and consider the number $p = p_1 \cdot p_2 \cdot \dots \cdot p_n$. Now take the number $p + 1$. If this is a prime number, then it is greater than any of the p_i , which contradicts our assumption since it would be a prime not yet among the assumed ones. Or else, $p + 1$ has a prime divisor, say q , then q would have to be one of the p_i and consequently q divides $p + 1$ and p , which leads to a contradiction since q would also have to divide the difference $p + 1 - p$, which is 1.

Another famous proof, due to Hippasus from Metapontum, a student of Pythagoras and member of the Pythagorean School shows that $\sqrt{2}$ is not rational. Because of the geometrical interpretation of $\sqrt{2}$ (i.e., the diagonal in a square

¹ Lets take this formulation for the purpose of explanation: as Michael Beeson has pointed out, the proof is actually not necessarily by contradiction [Beeson, 1998; Beeson, 2001].

of length 1) this problem had already puzzled Indian mathematicians two millennia b.c. and the Babylonian estimated $\sqrt{2} = 1 \cdot 60^0 + 24 \cdot 60^{-1} + 51 \cdot 60^{-2} + 10 \cdot 60^{-3}$ (which is about correct for the first five decimals) as recorded in a cuneiform script from about 1800 b.c. We recapitulate the theorem and a proof.

Theorem: $\sqrt{2}$ is irrational.

Proof (Hippasus, 500 b.c.): Assume $\sqrt{2}$ is rational, i.e., there exist natural numbers m and n with no common divisor such that $\sqrt{2} = \frac{m}{n}$. Then $n \cdot \sqrt{2} = m$ and, thus $2n^2 = m^2$. Hence, m^2 is even and since odd numbers square to odds, m is even; say $m = 2k$. Then $2n^2 = (2k)^2 = 4k^2$, that is, $n^2 = 2k^2$. Thus, n^2 is even too, and so is n . That means that both n and m are even, which contradicts the fact that they do not have a common divisor.

This is a particularly interesting theorem not only because the Pythagorean's – believing in a rational world order – drowned Hippasus as a punishment for such offensive thinking but it was also posed as a challenge to 'the seventeen provers of the world': the results of this contest have been published in the Springer lecture notes [Wiedijk, 2006] to mark the 50th anniversary of the first theorem ever, that was proven by a computer.² Our system participated in this contest as well (see [Wiedijk, 2006]) but a fully automated proof planning of the proof, remarkably similar to the above proof of the Pythagorean School, was established a little later by Omega and has now been published in [Siekman *et al.*, 2003].

Proofs by contradiction have become an issue in the foundational discussion between classical versus intuitionistic mathematics. The above reasoning can be formulated as

$$\text{if } Ax \cup \{A\} \vdash F \text{ and } Ax \cup \{A\} \vdash \neg F \text{ then } Ax \vdash \neg A \quad (1)$$

or alternatively as

$$\text{if } Ax \cup \{\neg A\} \vdash F \text{ and } Ax \cup \{\neg A\} \vdash \neg F \text{ then } Ax \vdash A \quad (2)$$

The difference is that in (1) we conclude from the axioms and A as well as the contradiction $\{F, \neg F\}$ that $\neg A$ holds. In contrast, the second formulation states that we are allowed to conclude A from the axioms and $\neg A$ as well as the contradiction. Using the classical law:

$$\text{if } Ax \vdash \neg\neg A \text{ then } Ax \vdash A \quad (3)$$

the above (1) and (2) collapse into the same kind of reasoning. This holds, however, only, if we accept the tertium non datur postulate $F \vee \neg F$ from which (3) follows. Intuitionism rejects this postulate and hence, this is not a valid form of reasoning in intuitionism (see, e.g., [Dummet, 2000]).

² Martin Davis' program based on the decidable fragment of first order logic called Presburger Arithmetic, showed the remarkable theorem that the sum of two even numbers is again even.

3 Knowledge-Based Proof Planning

Proof planning is a technique for theorem proving in which proofs are planned at a higher level, where individual choices can be mathematically motivated by the semantics of the domain. In particular, proof planning tackles theorems not only with logical operators but also by using domain knowledge and explicitly encoded control [Melis, 1998a]. However, a monolithic proof planner, in which the order of problem solving operations is pre-defined does not take full advantage of the runtime knowledge that is available from the mathematical domain. For instance, failure analysis is a natural and important ingredient of mathematical proof construction.

Our experiments with proof planning in the past decade indicate, that the search process would benefit from more flexibility of choice [Melis *et al.*, 2006] and more and better control knowledge for specific domains and mathematical techniques – such as proofs by contradiction in real analysis – which is the subject of this paper.

The Ω MEGA project, which is essentially based on proof planning [Siekmann *et al.*, 2006] represents one of the major attempts to build an all encompassing assistant tool for the working mathematician, which combines interactive and automated proof construction for domains with rich and well-structured mathematical knowledge. The inference mechanism at the lowest level is an interactive theorem prover based on a higher order natural deduction (ND) variant of a soft-sorted version of Church’s simply typed λ -calculus [Church, 1940]. While this represents the “machine code” of the system, the user will seldom want to see, the search for a proof is conducted at a higher level by a proof planning process.

Proof planning differs from traditional search-based techniques in automated theorem proving not least with respect to its level of abstraction: the proof of a theorem is planned at an abstract level where an outline of the proof is found first. This outline, that is, the abstract proof plan, can be recursively expanded with operators and tactics eventually down to a proof within the logical calculus. The plan operators represent mathematical techniques familiar to a working mathematician.

Knowledge-based proof planning [Melis, 1998a; Melis and Siekmann, 1999] employs even more techniques from artificial intelligence such as hierarchical planning, constraint solving and control rules for meta-level reasoning. While the knowledge of a mathematical domain represented by operators (called *methods*) and control rules is specific to the mathematical field, the representational techniques and reasoning procedures are general-purpose.

The methods (partially) describe changes of proof states by pre- and postconditions which are called *premises* and *conclusions* in the following. The premises and conclusions of a method are formulae (more precisely, sequents) in a higher-order language and the conclusions are considered as logically inferable from the premises.

Hence, a mathematical theorem proving problem is expressed as a planning problem whose initial state consists of the proof assumptions and whose goal

description consists of the conjecture. Proof planning searches for a sequence (or a hierarchy) of instantiated methods, i.e. a *solution plan*, which transforms the initial state with assumptions into a state containing the conjecture.

Methods, Control Rules, and Strategies in a Context In order to make the ingredients of proof planning more explicit, let us repeat what methods and control rules contribute to proof planning and then extend the discussion to strategies and contexts.

Methods have been perceived by Alan Bundy as tactics augmented with pre-conditions and effects, called *premises* and *conclusions*, respectively. A method represents the inference of the conclusion from the premises. Backward methods reduce a goal (the conclusion) to new goals (the premises). Forward methods, in contrast, derive new conclusions from given premises.

For example, the following method **ComplexEstimate** is an essential ingredient in epsilon-delta proofs of limit theorems.

method: ComplexEstimate(a, b, e_1, ϵ)		
premises	$(0), \oplus(1), \oplus(2), \oplus(3)$	
conclusions	\ominus L12	
appl.cond	$\exists\sigma(\text{subst}(a, b) = \sigma) \&$ $\exists k, l(\text{caseextract}(a_\sigma, b) = (k, l)) \& b = k * a_\sigma + l$	
proof schema	(0). Δ	$\vdash a < e_1$ ()
	(1).	$\vdash a_\sigma < \epsilon / (2 * \mathbf{V})$ (OPEN)
	(2). Δ	$\vdash k \leq \mathbf{V}$ (OPEN)
	(3).	$\vdash 0 < \mathbf{V}$ (OPEN)
	(4). Δ	$\vdash l < \epsilon / 2$ (OPEN)
	L0.	$\vdash b = b$ (Ax)
	L1.	$\vdash b = k * a_\sigma + l$ (CAS;L0)
	.	$\vdash \dots$ (...)
	L12 Δ	$\vdash b < \epsilon$ (schema;L1,(3), (0),(1),(2),(4))

This frame-like data structure should be read as follows: the method's name is **ComplexEstimate** and it has the parameters a, b, e_1, ϵ . The premises are the lines (0), (1), (2), and (3) which are schematically detailed in the proof schema slot. The \oplus in the premises slot indicates these are added as subgoals by the application of the method. The conclusions is a goal schematically detailed in line L12 of the proof schema and the \ominus indicates that it is removed by the application of the method. The application condition is formulated in a meta-language and for **ComplexEstimate** it expresses that a and b have to be unifiable by a substitution σ and b can be decomposed into a linear combination of the substituted a (which is tested by a computer algebra system). The slot proof schema contains a sequence of (proof) lines that are introduced with the expansion of the method and used in the final proof. \mathbf{V} is a newly introduced (auxiliary) variable.

Control rules represent mathematical knowledge about how to proceed in a particular mathematical situation, and they guide the proof planning process. They can influence the planner's behavior at choice points (e.g., which goal to tackle next or which method to apply next) by preferring members of the list of possible goals or of the list of possible methods. This way promising search paths are preferred and the general search space can be pruned.

Methods and control rules are the main ingredients of current proof planning systems, however, they do not always provide enough structure and flexibility for the problem solving process as the past decade of experimentation revealed. First, there is a problem with the planning algorithm itself, which cannot be decomposed into its main components nor can new techniques easily be added.

Secondly, the proof planning process is too uniform, irrespective of the current context and independent of the kind of theorem to be shown.

For instance, if we want to prove a continuity theorem in the theory of analysis, it makes a difference whether we prove it via limit theorems, via an epsilon-delta technique, via converging sequences, or by contradiction which is the focus of this paper. Every mathematician has a variety of different strategies at her disposal to tackle such specific problems.

A *Strategy* as it is now used in our multi-strategy proof planning system employs a specific subset of the search algorithms, methods and control rules that are typical for the particular proof technique we want to simulate. For instance, one strategy may use an external system for some computation, another one may attack the problem with a completely different set of methods such as the epsilon-delta techniques, the methods and control rules typically used for a proof by induction or the methods and control rules for a proof by contradiction. It may cooperate with another strategy from a different theory, or it may use a different backtracking technique.

Meta-reasoning as to which strategy to employ on a problem introduces an additional explicit choice point and, thus, the system searches at the level of strategies as well.

All of this, however, is still in stark contrast to the situation of a mathematician who operates in a specific *context* which includes the current theory under development, preferences, knowledge representation techniques tailored to the specific context, definitions that are formulated in a way to serve the current purpose, typical techniques to prove this particular theorem as well as tricks of the trade typical for the field within which the theorem is stated. For example, a theorem about a continuous function requires certain methods, control rules and strategies to tackle its proof, which a student learns in the calculus courses, whereas a theorem, say in group theory, requires a very different set of methods and control, usually taught in an algebra class. The choice of either of them is *prior and above* a strategy and determined by the current context. The notion of a context provides a powerful structuring technique, in particular, for large-scale applications of mathematical assistant systems; this is ongoing work in the Omega group.

4 The Extended Limit Domain

The ‘Limit Domain’ is a well-known set of theorems to be shown by epsilon-delta-proofs. This domain typically comprises limit theorems and theorems about continuity. For example, the theorem that f is continuous at a , formulated as

$$\lim_{x \rightarrow 0}(f(a+x) - f(a)) = 0 \rightarrow \text{continuous}(f, a).$$

has been shown by our system (see [MeierMelis, 2004]).

Another well-known example is the LIM^+ theorem which was posed by Woody Bledsoe as a challenge to (classical) automated theorem proving systems. It states that the limit of the sum of two functions f and g equals the sum of their limits. Hence there are two assumptions which define the limit of f and g .

$$\forall \epsilon_1(0 < \epsilon_1 \Rightarrow \exists \delta_1(0 < \delta_1 \wedge \forall x_1(|x_1 - a| > 0 \wedge |x_1 - a| < \delta_1 \Rightarrow |f(x_1) - l_1| < \epsilon_1))) \quad (4)$$

and

$$\forall \epsilon_2(0 < \epsilon_2 \Rightarrow \exists \delta_2(0 < \delta_2 \wedge \forall x_2(|x_2 - a| > 0 \wedge |x_2 - a| < \delta_2 \Rightarrow |g(x_2) - l_2| < \epsilon_2))). \quad (5)$$

and the theorem is:

$$\forall \epsilon(0 < \epsilon \Rightarrow \exists \delta(0 < \delta \wedge \forall x(|x - a| > 0 \wedge |x - a| < \delta \Rightarrow |(f(x) + g(x)) - (l_1 + l_2)| < \epsilon))) \quad (6)$$

This and many more open challenge problems in this domain have been solved now with proof planning. Here is the proof of LIM^+ , first provided in [Melis, 1998b]:

The system first decomposes the conjecture and the assumptions. Among others, this yields the new assumptions³ $|f(v_{x_1}) - l_1| < v_{\epsilon_1}$ and $|g(v_{x_2}) - l_2| < v_{\epsilon_2}$ and the two new goals $0 < v_\delta$ and $|(f(c_x) + g(c_x)) - (l_1 + l_2)| < c_\epsilon$.⁴ The first goal, $0 < v_\delta$, is closed by the method `TellCS` which closes the goal and adds it to the constraint store of the constraint solver `COSIE` [Zimmer and Melis, 2004]. The second goal $|(f(c_x) + g(c_x)) - (l_1 + l_2)| < c_\epsilon$ requires further decomposition, which is done by `ComplexEstimate` [Melis, 1998b]

In the concrete example LIM^+ , `ComplexEstimate` employs the new assumption $|f(v_{x_1}) - l_1| < v_{\epsilon_1}$ and yields four new goals:

$$\epsilon_1 < \frac{c_\epsilon}{2 * v} \quad (7)$$

$$|1| \leq v \quad (8)$$

$$0 < v \quad (9)$$

$$|g(c_x) - l_2| < \frac{c_\epsilon}{2} \quad (10)$$

³ Notation: Proof planning replaces quantified variables either by constants or placeholder variables. The placeholder variable substituted for a quantified variable x is denoted by v_x . The constant substituted for a quantified variable x is denoted by c_x .

⁴ During the decomposition of the assumptions further goals are created and the decomposition of the conjecture yields further assumptions are derived. However, in order to illustrate the basic proof planning approach we ignore these details.

(7), (8), (9) can be closed by **Te11CS**. Goal (10) is reduced by a method called **Solve** using the derived assumption $|g(v_{x_2}) - l_2| < v_{\epsilon_2}$ and yields the subgoals $v_{\epsilon_2} \leq \frac{c_x}{2}$ and $v_{x_2} = c_x$ which can be closed by the method **Te11CS**.

When all goals are closed, the constraint solver **COSIE** computes appropriate instances for variables that are consistent with the collected constraints. In this case, it generates the following instantiation:

$v_\delta \mapsto \min(c_{\delta_1}, c_{\delta_2})$, $v_{\epsilon_1} \mapsto \frac{c_x}{2}$, $v_{\epsilon_2} \mapsto \frac{c_x}{2}$. Note, that these happen to be the same values that are used in a typical human proof of LIM^+ , say, in a standard textbook such as [Bartle and Sherbert, 1982]. (end of proof)

In the meantime, we investigated and solved many more problems from the Extended Limit Domain whose problems are all the theorems, examples, and exercises of two chapters of the introductory textbook for Real Analysis [Bartle and Sherbert, 1982] that deal with limits of sequences and limits of functions. For many of these theorems there are several ways to prove them, e.g., epsilon-delta-proofs, proofs using other limit theorems, and proofs involving the estimation of an upper bound. The latter involves the use of the **Dominance** method. **Dominance**(a_n), where the parameter (a_n) denotes a sequence converging to zero, reduces a goal $\lim(x_n) = l$ to the goal

$\exists k \forall n (k \in \mathbf{N} \wedge (n \in \mathbf{N} \rightarrow (n > k \rightarrow \frac{|x_n - l|}{|a_n|} < c))$ for a constant c . This is justified by the theorem that says

if there is a sequence (a_n) converging to zero, a constant number c , and a natural number k such that for all $n \geq k$ holds $|x_n - l| < c \cdot a_n$, then $\lim(x_n) = l$.

Not without surprise there are also many theorems in Bartle and Sherbert that are shown by contradiction, although in principle they could be proven directly. But the authors found it simpler and esthetically more pleasing to do otherwise.

5 Planning Proofs by Contradiction

In order to show the conjecture T under the assumptions A_1, \dots, A_n , a *proof by contradiction* assumes $\neg T$ and the proof assumptions A_1, \dots, A_n and tries to prove a contradiction $F \wedge \neg F$ from these assumptions for some formula F .

In common undergraduate textbooks such as Bartle and Sherbert's introduction to real analysis, this principle is used often whenever it is convenient. In the past we looked at this textbook quite frequently for inspiration to proof planning and in the meantime we have solved (almost) all of these problems with our system.

But there remained several proofs by contradiction: so the first question to be answered is, under which conditions is a proof by contradiction appropriate. And secondly, the difficulty and the 'creative trick' in human proofs by contradiction (unlike say in a refutation by resolution) is to find an appropriate fact F which can be contradicted.

In order to solve the problems in Bartle and Sherbert we experimented with control knowledge on *when* to prove a conjecture by contradiction and *which* formula F to refute. In all the examples cited below we used the heuristic that

F is one of the assumptions (A_k). Now this is not much of a restriction, if we include anything among the assumptions A_i . So, what we have in mind is the common situation in a textbook, where the assumptions A_1, \dots, A_n are immediately given either in the theorem itself as its hypotheses or more or less immediately in the chapter prior to the theorem, i.e., in the *current context*.

Proofs by contradiction of this nature are now realized in our system by the method $\text{Contradict}(A_k)$, which has the parameter A_k .

Method : Contradict (A_k)	
Premises	$L_0, \oplus L_3$
Conclusions	$\ominus L_4$
Parameter	L_0
Proof Schema	$(L_0) \quad \vdash A_k \quad ()$
	$(L_1) \quad \Delta \quad \vdash \neg T \quad (\text{Hyp})$
	$(L_2) \quad \Delta, L_1 \vdash \neg A_k \quad (\text{OPEN})$
	$(L_3) \quad \Delta, L_1 \vdash A_k \wedge \neg A_k \quad (\wedge\text{I}; L_0 L_2)$
	$(L_4) \quad \Delta \quad \vdash T \quad (\neg\text{E}; L_1 L_3)$

The $\ominus L_4$ conclusion of the method indicates that the sequent in line L_4 , T , is removed as a goal from the state when $\text{Contradict}(A_k)$ has been applied. The L_0 premise which is not annotated indicates that the sequent of L_0 has to be an assumption in the state before $\text{Contradict}(A_k)$ is applied. The $\oplus L_3$ premise indicates that the sequent $\Delta, L_1 \vdash A_k \wedge \neg A_k$ of line L_3 in the Proof Schema is added as a new subgoal to the state when $\text{Contradict}(A_k)$ has been applied, where Δ, L_1 is the union of A_1, \dots, A_n and $\neg T$. The Proof Schema contains the lines/nodes that are inserted into the partial proof plan when $\text{Contradict}(A_k)$ is expanded.

Now, when should this method be used in the proof planning process? Some very general guidelines for choosing a proof by contradiction are given in [Sieg and Byrnes, 1998] for proofs in the natural deduction calculus of Wilfried Sieg. These guidelines say: if you cannot derive a goal forwardly from the proof assumptions and if the goal is a negation, disjunction, or existentially quantified formula, then try a proof by contradiction. This means that first all possible forward proofs have to be attempted and only when this fails, search for a proof by contradiction. This control is, of course, not particularly efficient and in many cases it may even lead to infinitely many attempts. However, setting a heuristic bound on these attempts it works quite well in practice.

Many domains have domain-specific knowledge, which can be exploited to control the application of the method Contradict . For instance, in the Extended Limit Domain one would *not* try to prove a goal $\neg(a < b)$ by contradiction although it is a negation but rather rewrite this goal to $b \leq a$ using knowledge that is specific for real numbers, or more generally for totally ordered structures.

For the Extended Limit domain we define: an (in)equality is called *simple*, if its lhs and rhs terms are constant expressions. 'Constant expressions' contain only variables (constants as opposed to placeholder variables) that do not depend on the instantiations of other variables (i.e., variables introduced through

\forall -elimination in goals or \exists -elimination in assumptions). For instance, the expression c_ϵ replacing the \forall -quantified variable ϵ in the goal (6) is a constant, whereas the expression replacing \exists -quantified δ is not. Vice versa, the expression introduced for ϵ_1 in (4) is a placeholder variable whereas the expression c_{δ_1} introduced for δ_1 in (4) is a constant.

For proof planning in the Extended Limit domain we found the following two heuristics sufficient to trigger a proof by contradiction

- If the goal is a simple equation or inequality, in which (preferably) at least one constant represents a limit and if the goal is not entailed by (in)equality assumptions, then prefer proof by contradiction.

The mathematical insight for this heuristic is that simple (in)equalities can be satisfied by (finite) forward entailment (realized by the method `AskCS`, which uses the constraint solver of the system to compute entailment). In case the simple (in)equation is not finitely entailed, a proof by contradiction provides an alternative. Mathematically, such an alternative is needed, when a property of the elements of a sequence cannot be finitely transferred to the corresponding property of the limit since this transfer requires an infinitesimal process.

- `Contradict` will be applied at most once in the same branch of a proof plan.

These two heuristics are expressed in the following control rule.

```

IF goal-simple-ineq
  AND not-yet-applied (Contradict)
  AND all-ineqAssumptions-known
THEN prefer (AskCS
             Contradict)

```

This control rule encodes the heuristic that, if the goal is a simple equation or inequality and `Contradict` has not been applied in the current branch of the proof plan already, and all entailment information is available, then the planner should first try to check the entailment of the goal using the method `AskCS` and if this fails, a proof by contradiction should be preferred.

We could test the entailment via a meta-predicate in the preconditions of a control rule but instead this test is done in the application condition of the method `AskCS`. Thereby we avoid duplication of work in case a proof by contradiction is unnecessary.

6 Results

We have collected all the theorems shown by contradiction in the textbook on Real Analysis of Bartle and Sherbert and surprisingly the rather simple heuristic above worked in most cases. Omega’s proof planner finds proofs by contradiction inter alia for the following theorems:⁵

⁵ The numbering is the same as that of the theorems in the textbook [Bartle and Sherbert, 1982].

Theorem 3.1.5:

A sequence of real numbers can have at most one limit.

This 'uniqueness of limit' theorem is shown in [Bartle and Sherbert, 1982] with the following

proof: Suppose on the contrary that x' and x'' are both limits of $X = (x_n)$ and that $x' \neq x''$. We choose $\epsilon > 0$ such that the ϵ -neighborhoods $V_\epsilon(x')$ and $V_\epsilon(x'')$ are disjoint, that is, such that $\epsilon < \frac{1}{2}|x' - x''|$. Now let K' and K'' be natural numbers such that if $n > K'$ then $x_n \in V_\epsilon(x')$, and if $n > K''$ then $x_n \in V_\epsilon(x'')$. However, this contradicts the assumption that these ϵ -neighborhoods are disjoint. (Why?) Consequently, we must have $x' = x''$.

The answer to the corresponding exercise question (Why) as well as the whole proof is found by Omega in a similar way. The formula F that is refuted in the proof of Bartle and Sherbert is 'the ϵ -neighborhoods are disjoint'.

The next theorem 3.2.4 is related to the dominance property mentioned in section 4 and is used in [Bartle and Sherbert, 1982] as the prerequisite for theorem 3.2.5. and theorem 3.2.6. all of which have been shown by contradiction without this prerequisite with the proof planner.

Theorem 3.2.4:

If (x_n) is a convergent sequence and if for all n $x_n \geq 0$, then $\lim(x_n) \geq 0$.

The formula F that gives the contradiction is $F := x_n \geq 0$

Theorem 3.2.5:

If (x_n) and (y_n) are convergent sequences of real numbers and if $x_n \leq y_n$, then $\lim(x_n) \leq \lim(y_n)$.

Theorem 3.2.6:

If (x_n) is a convergent sequence and if $a \leq x_n \leq b$, then $a \leq \lim(x_n) \leq b$.

The next theorem 'assures us that the value L of the limit is uniquely determined, when it exists. As this uniqueness is not part of the definition of limit, it must be deduced.' ([Bartle and Sherbert, 1982], p.113).

Theorem 4.1.5:

If $f : A \mapsto \mathbb{R}$ and if c is a cluster point of A , then f can have only one limit at c .

Here the formula F that gives the contradiction is obtained from 'the ϵ -neighborhoods of L and L' are disjoint (an assumption of a subproof).

The following is taken from the exercises in [Bartle and Sherbert, 1982], chapter 4.1 aimed at the expertise of a freshman.

Theorem 4.1.(3):

Let $f : \mathbb{R} \mapsto \mathbb{R}$ and let $c \in \mathbb{R}$. If $\lim_{x \rightarrow 0} f(x + c) = l_1$ and $\lim_{x \rightarrow c} f(x) = l_2$, then $l_1 = l_2$.

The above sample is taken from many more theorems for which the proof planner found a proof with the fixed set of methods and control rules we have

collected for this branch of mathematics (and reported elsewhere [Melis, 1998b; Melis *et al.*, 2006; ?] except that the above control rule capturing the heuristic for proof by contradiction has been added. As a result, the theorems in the two chapters of the textbook [Bartle and Sherbert, 1982] that require the use the of method `Contradict` are recognized and handled properly.

7 Discussion

The restriction to experiments with theorems from analysis is certainly not a principal one as these results apply in other areas of mathematics as well, where proofs by contradiction are abundant. The Extended Limit Domain is already a very rich one, in particular, when it comes to search spaces and the use of domain-dependent knowledge and this ‘small world’ encapsulates many of the problems from mathematical problem solving in general.

In particular, the search for the formula F that is finally used in the contradiction is an ‘AI-complete’ problem and many proofs by contradiction have in fact become famous, because of an ingenious choice of F . An example is the proof that Euler’s number e is not rational, which was shown by Leonhard Euler in 1737. In fact, e is not only irrational but a transcendental number, which was shown 150 years later by Charles Hermite. Euler’s proof reasons by contradiction and assumes that e is a rational number and hence, can be represented as $e = \frac{p}{q}$ for some integers p and q . The contradiction follows from the fact that $q! \cdot e$ is an integer, whereas its expansion

$$q! + \frac{q!}{1} + \frac{q!}{2} + \frac{q!}{3} + \dots + \frac{q!}{q} + \frac{q!}{(q+1)!} + \frac{q!}{(q+2)!} + \dots \quad (11)$$

is not an integer, hence, e is irrational. So the ingenuity of this proof is to construct the appropriate formula F that states that $q! \cdot e$ is an integer and show that the expansion of $q! \cdot e$ is not an integer.

However, as it is always possible to tune and set the dials of a system to prove a theorem – no matter how famous and difficult – for which a proof is known in the literature, the goal of our game is to take Woody Bledsoe’s criticism of automated theorem proving seriously:

Automated theorem proving is not the beautiful process we know as mathematics. This is ‘cover your eyes with blinders and hunt through a cornfield for a diamond-shaped grain of corn... Mathematicians have given us a great deal of direction over the last three millenia. Let us pay attention to it. (Woody Bledsoe, 1986)

‘Nothing can be explained to a stone’ (McCarthy 1967) and in particular ‘Nothing mathematically interesting can be told to a (resolution-based) automated theorem proving system’ is our mantra capturing this general issue. The Omega system has been ‘told’ one extra control rule and to our own surprise this turned out to be sufficient to find many proofs by contradiction in [Bartle

and Sherbert, 1982]. But no doubt things will not stay this way and we have to uncover stronger means to the end of Reductio ad Absurdum arguments, in particular, more ingenious ways to determine the formula F . This is the subject of ongoing research in the Omega group.

Acknowledgement The reported work was funded by the DFG-project MIPPA (Me 1136/2-1) and the collaborative research center 378 of the German National Science Foundation.

References

- [Bartle and Sherbert, 1982] R.G. Bartle and D.R. Sherbert. *Introduction to Real Analysis*. John Wiley& Sons, New York, 1982.
- [Beeson, 1998] M.J. Beeson. Automatic generation of epsilon-delta proofs of continuity. *Artificial Intelligence and Symbolic Computation*, J. Calmet and J. Plaza (eds), pages 67-83. LNAI 1476, Springer Verlag, 1998.
- [Beeson, 2001] M.J. Beeson. Automatic generation of a proof of the irrationality of e . *Journal of Symbolic Computation*, 32(4): 333-349, 2001.
- [Bledsoe, 1977] W.W. Bledsoe. Non-resolution theorem proving. *Artificial Intelligence*, 9:1-35, 1977.
- [Bundy, 1988] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction (CADE-9)*, volume 310 of *Lecture Notes in Computer Science*, pages 111-120, Argonne, 1988. Springer.
- [Church, 1940] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56-68, 1940.
- [Dummett, 2000] M. Dummett. *Elements of Intuitionism*. Oxford, 2nd edition, 2000.
- [Gelernter, 1959] H. Gelernter. Realization of a geometry theorem-proving machine. In *Proceedings of the International Conference on Information Processing, UNESCO*, 1959.
- [Lenat, 1981] D.B. Lenat. AM: an AI approach to discovery in mathematics. In R. Davis and D.B. Lenat, editors, *Knowledge-Based Systems in Artificial Intelligence*. Mc-Graw Hill, New York, 1981.
- [Melis and Meier, 2000] E. Melis and A. Meier. Proof planning with multiple strategies. In J. Loyd, V. Dahl, U. Furbach, M. Kerber, K. Lau, C. Palamidessi, L.M. Pereira, and Y. Sagiv and P. Stuckey, editors, *First International Conference on Computational Logic*, volume 1861 of *Lecture Notes on Artificial Intelligence*, pages 644-659. Springer-Verlag, 2000.
- [Melis and Siekmann, 1999] E. Melis and J.H. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65-105, November 1999.
- [Melis et al., 2006] E. Melis, A. Meier, and J. Siekmann. Proof planning with multiple strategies. *Artificial Intelligence*, submitted, 2006.
- [Melis, 1998a] E. Melis. AI-techniques in proof planning. In *European Conference on Artificial Intelligence*, pages 494-498, Brighton, 1998. Kluwer.
- [Melis, 1998b] E. Melis. The "limit" domain. In R. Simmons, M. Veloso, and S. Smith, editors, *Proceedings of the Fourth International Conference on Artificial Intelligence in Planning Systems*, pages 199-206, 1998.

- [MeierMelis, 2004] A. Meier and E. Melis. Proof Planning Limit Problems with Multiple Strategies. SEKI Technical Report SR-2004-04, FR Informatik, Universitaet des Saarlandes, 2004.
- [Newell, 1981] A. Newell. The Heuristic of George Polya and its Relation to Artificial Intelligence. Technical Report CMU-CS-81-133, Carnegie-Mellon-University, Dept. of Computer Science, Pittsburgh, Pennsylvania, U.S.A., 1981.
- [Polya, 1945] G. Polya. *How to Solve it*. Princeton University Press, Princeton, 1945.
- [Robinson and Voronkov., 2001] A. Robinson and A. Voronkov. *Handbook of Automated Reasoning*, vol 1 and 2. Elsevier, 2001.
- [Robinson, 1965] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12, 1965.
- [Schoenfeld, 1985] A.H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, New York, 1985.
- [Sieg and Byrnes, 1998] W. Sieg and J. Byrnes. Normal natural deduction proofs (in classical logic). *Studia Logica*, 60:67–106, 1998.
- [Siekmann *et al.*, 2003] J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, I. Normann, M. Pollet, Proof Development in OMEGA: The Irrationality of Square Root of 2. In: Kamareddine, F. (Ed.), *Thirty Five Years of Automating Mathematics*. Kluwer Applied Logic series. Kluwer Academic Publishers. 2003.
- [Siekmann *et al.*, 2006] J. Siekmann, C. Benzmüller, and S. Autexier. Computer supported mathematics with Omega. *Journal of Applied Logic*, 2006. in press.
- [Wiedijk , 2006] F. Wiedijk (ed.). *The Seventeen Provers of the World*. LNAI vol. 3600 (the first volume in the AI-Systems subseries), Springer-Verlag, 2006.
- [Zimmer and Melis, 2004] J. Zimmer and E. Melis. Constraint solving for proof planning. *Journal of Automated Reasoning*, 33(1):51–88, July 2004.