The Heine-Borel Challenge Problem
In Honor of Woody Bledsoe

Erica Melis*
Universität des Saarlandes, Fachbereich Informatik,
D-66041 Saarbrücken, Germany,
email: melis@cs.uni-sb.de

Abstract

Woody Bledsoe's last challenge problem is the analogical transfer of the Heine-Borel theorem for real intervals to the two-dimensional case. This could not be solved by the up to then known techniques of analogical theorem proving. The Heine-Borel theorem is a widely known result in mathematics. It is usually stated in the field of real numbers R^1 and similar versions are also true in R^2, in topology, and in metric spaces. This article shows how analogy-driven proof plan construction is applicable to this *genuinely* mathematical problem. Our goal here was to use a *source* proof plan of HB1 (the Heine-Borel theorem in R^1) as a guide to automatically produce a proof plan of HB2 (the Heine-Borel theorem in R^2). We were able to accomplish our goal by generating the *target* proof plan of HB2 by reformulation and analogical replay.[1]

keywords: analogy, challenge problems

1 Woody Bledsoe's Legacy

Being a working mathematician, Woody Bledsoe was an early proponent of non-resolution theorem proving [Ble77]. In particular, he suggested to employ theorem proving by analogy, to generate examples, and to use large mathematical knowledge bases [Ble86]. He considered analogical reasoning a hard task that is, however, among the most important for automated theorem proving and, more generally, crucial for any intelligent behavior. In 1986 he wrote: "*Analogy* is the heart and soul of all intelligent behavior, especially mathematical behavior. Why have we made so little use of analogy in ATP?...We predict

*The research reported in this paper was supported by a DFG grant and HC&M grant CHBICT930806

[1] Article in *Journal of Automated Reasoning*, 1997.

no substantial advance until our provers begin to effectively use Analogy with the help of an adequate MKB" (mathematical knowledge base) [Ble86]. Here, theorem proving by analogy means finding a proof for a target theorem guided by a given proof of a source theorem.

As early as 1986, Woody Bledsoe and his students Bishop Brock, Shaun Cooper, and Bill Pierce had developed important ideas for the analogical replay of real mathematical problems such as the limit theorem Lim+, the theorem from calculus which states that the limit of a sum of two functions is the sum of their limits.

Until 1993 Woody worked on his Precondition Prover (PC) for Analogy [Ble95a, Wan91] that was designed for debugging analogically transferred proofs, for instance, the proof for the theorem of Ground Completeness of Resolution. Its basic idea is to find the weakest preconditions under which a source proof step works in a proof of the new (target) theorem. Given a new theorem, a similar theorem, and its proof (plan), PC normally acts as an ordinary proof checker but it returns a "precondition" when it is unable to prove the given formula. This is similar to the situation where a person is proving a theorem, making a number of decisions "and sometimes makes a mistake in early decisions. He does discover that there is a problem only 'deep' in the proof, and then has to decide which of the early decisions has to be altered and how" [Ble95a].

Woody Bledsoe focused early on proving hard and genuinely mathematical theorems [Ble90, BBH72, Ble95a], in particular in analysis. To attack real mathematical problems was one of his strong convictions and his contributions to automated theorem proving (ATP) were strongly motivated by this. As a consequence, Woody often provided challenge problems that have fertilized the field and are still a valuable source, see, e.g, [Ble90]. His last challenge problem was the analogical transfer of the proof of the Heine-Borel theorem for real intervals (HB1) to a proof of the Heine-Borel theorem for the two dimensional space (HB2) which he suggested in summer 1994 [Ble94]. He wrote: "We give a pair of theorems that we hope to use in our Analogy Theorem Proving experiments. They both are Heine-Borel Theorems ...We do not believe that the PC [1][2] procedure will work for this pair ... Our view, at this time, is that something like the 'Reformulation' methods of Erica Melis [2][3] should be able to provide the help needed, or maybe even do the whole analogy proof... by itself...Also other methods described in [3][4] could be of use" [Ble94].

We had discussed the ideas of reformulation and transferring proof plans by email. In March 1994 I met Woody for the first time when he invited me for a talk about my analogy-driven proof plan construction to the University of Texas at Austin. I met a kind person who embodied all the experience of a scientist and teacher who inspired his students. In fact, I know one of his last students, a German student in mathematics, who decided to change to theorem proving while he was a student of Woody's at Austin. As everybody, I was extremely impressed by Woody's courage not to despair given his illness: In

[2] which is[Ble95a] in the this article
[3] which is [Mel93b] in the this article
[4] which is [Owe90] in this article

spite of his diminishing speech capabilities, we had intensive discussions about my approach and about analogy in general for several days. We both enjoyed them immensely. I felt very encouraged - and I really needed encouragement at that time. Of course, I enthusiastically tried to convince him of my – at that time somewhat premature – account of analogy. It was not difficult to win him over to the idea of transferring *proof plans*, a high-level representation of proofs, because he had already similar ideas himself. It was not as easy to explain the details of the analogical replay and of the reformulation because in a very friendly but firm way, he insisted on clarifying the ideas and on proving that my analogy method can cope with hard mathematical problems. As a result, I agreed to work on a genuinely mathematical problem.

In August 1994 he had sent me the first version of the Heine-Borel challenge problem. In September 1994 I started to look at the example and to adapt it to my analogy framework that needs a proof plan representation. Little adaptation was necessary, though, because Woody had suggested a high-level representation of the HB1 proof. Emails went back and forth for several months to correct and improve the representation, to implement the reformulation, and to check whether the reformulations did what they were expected to do. Admittedly, we argued a great deal – but all the time we stayed good friends and we jointly wanted to find and to publish the best solution for the non-trivial analogy problem. In December 1994 when we were wrapping up the whole thing for a paper, Woody suggested I should go ahead alone and finish and submit the paper. Although I would have liked to include him and all the time worked under the assumption that this would be a joint work, he did not want to co-author anymore (“`we could later co-author an extended version for JAR.`”). The love of his friends and family at home made him feel better and became his most important support for the next months. We kept in touch all the time although it became difficult for him to type his emails “`It seems that Als is gradually winning the battle`”. Eventually I lost a dear friend in October 1995.

I consider myself very fortunate to have collaborated with Woody Bledsoe in solving the Heine-Borel problem and I present a solution in this paper. A great deal of the article was originally written for the joint paper that we planned together. This was almost certainly his final piece of collaborative research and it is a great honor to dedicate this article to his memory.

The rest of this article is organized as follows: Following an introduction and some technical preliminaries, I describe the Heine-Borel problem. Then I introduce the method of analogy-driven proof plan construction and apply it to the HB1 proof plan. This yields an incomplete proof plan for the HB2 problem which can be extended to a proof of HB2. An evaluation of the result and future work are discussed at the end.

2 Introduction

Mathematicians have clearly recognized the power of analogy in mathematical problem solving [Had45, Pol45, Pol54, vdW64]. In automated and interactive theorem proving, analogy is particularly useful in situations where much search or user interaction is necessary, e.g., when many proof assumptions, difficult theorems, or long proof paths are involved.

Computational accounts of theorem proving by analogy [Kli71, Mun81, Owe90, KW94, Cur95] have been dominated by the idea of

- a (second-order) mapping from symbols of the source theorem and lemmata to the target theorem and lemmata and then
- transferring each single proof step from the source proof to a proof step in the target.

Empirical investigations [Mel93a], however, have provided evidence that this idea does not appropriately cover the analogies drawn by mathematicians, not even all the analogies of a standard university-level textbook [Deu71]. The failure of step-by-step transfers at the calculus-level has been experienced more widely. For instance, transformational tactics in Nuprl failed to succeed because formal proofs are often too brittle to apply such a transformation.[5]

This evidence motivated the search for a more high-level representation of proofs that is better suited for the analogical transfer. We have found proof plans an appropriate representation where proof plans consist of methods which are specifications of tactics [GMW79]. Only when executed, a proof plan yields an object-level proof by running the tactics. The use of proof plans in the analogy context is advantageous because proof plans are *abstract* representations of proofs. Therefore, the same proof plan may yield different object-level (i.e. calculus-level) proofs for different problems. Furthermore, a proof plan provides a proof structure, e.g. lemmata to be proved, and more directly displays the essential proof idea. The methods which build a plan represent the *chunks* of proof steps to be transferred analogically. This structured representation also allows for the mere transfer of proof ideas by transferring (hierarchical) proof plans in which the proof ideas can be distinguished from proofs of details. This is mathematically desirable but usually not possible when a sequential calculus proof is to be transferred analogically.

The analogy aims at reducing the search or interaction effort by providing guidance for constructing a target proof plan from a source proof plan. Tedious guidance in automated theorem proving is usually necessary to produce a complicated structured proof,[6] whereas a great deal of an appropriate source plan is likely to be transferred analogically to a target case. We integrate the call of an automated theorem prover (namely OTTER [McC90]) into our methods

[5] Personal communication with Bob Harper.

[6] For instance, 1741 lemmas had to be user supplied in Shankar's proof of Gödel's First Incompleteness theorem [BM88].

and thus obtain another advantage of transferring proof plans, namely the suggestion of subproofs likely to be feasible for an automated prover within a given time limit.

The empirical results also indicated a need for more complicated reformulations in addition to pure symbol mappings. This is in accordance with the well known fact that constructing an analogy in mathematics often amounts to first finding the appropriate representation which brings out the similarity of two problems, i.e., finding the right concepts and the right level of abstraction. Therefore, analogy-driven proof plan construction includes reformulation. In order to show the advantages of analogy-driven proof plan construction let us look at some real mathematical theorems. Our approach makes some complicated proofs feasible, as we will show with the transfer of a Heine-Borel theorem proposed in [Ble95a] as a challenge example for theorem proving by analogy. Our system processes the Heine-Borel theorem and its proof as it was provided by Bledsoe who suggested the formulation, the lemmata, and the subproofs to be found by OTTER.

3 Preliminaries

Proof planning operators, called *methods*, have first been introduced by Bundy [Bun88]. A method is a (partial) specification of a tactic, where a tactic executes a number of logical inferences [GMW79]. The methods used in this article are frame-like structures defined in [HKKR94] with pre- and postconditions just as the common planning operators. More specifically, these methods M have the following slots: parameter, preconditions (pre(M)), postcondition (post(M)), constraints, proof schema, and procedure. pre(M) is a set of sequents[7] from which the application of the method derives post(M) which is a set of sequents as well. pre(M) and post(M) are needed for planning.

The constraints are formulated in a meta-language and serve to restrict the search during planning, e.g., restrict the instantiations of pre(M), post(M), or of the parameters. The program in the slot procedure executes the application of the proof schema and/or applies tactics occurring in constraints or in the proof schema [Seh95]. The proof schema is a declarative schematic representation of proofs in the object logic, that relies on the Natural Deduction (ND) calculus and on invoking automated theorem provers such as OTTER [McC90]. The proof schema lines contain a label, a sequent, and a line-justification. The line-justification consists of the name of an ND-rule, the name of a prover or tactic, a meta-variable, or OPEN in case the sequent is to be planned for. Additionally, the line-justification may include supporting lines. For instance,
3. $\Delta \vdash F$ $\quad(\neg$E;2)
describes that the sequent $\Delta \vdash F$ is derived from the sequent in line 2 of the proof schema by the ND-rule $\neg$E. An example of a method is `Refutation`:

[7] *Sequents* $\mathsf{P} = (\Delta \vdash F)$, are represented as pairs of a set Δ of formulas and a formula F in an object language that is extended by meta-variables for functions, relations, formulas, sets of formulas, and terms.

method: `Refutation`	
parameter	F: formula, Δ: set of formulas
preconditions	$(\Delta \cup \{\neg F\} \vdash \bot)$
postcondition	$(\Delta \vdash F)$
constraints	
proof schema	1. $\Delta, \neg F \quad \vdash \quad \bot$ (OPEN) 2. $\Delta, \quad \vdash \quad \neg\neg F$ ($\neg$I,1) 3. $\Delta \quad \vdash \quad F$ ($\neg$E;2)
procedure	standard schema-interpreter

Essentially, these methods differ from those in [Bun88] in that the tactic slot is replaced by a declarative proof schema *and* a procedure interpreting this schema.[8] The intention behind this difference has been to enable more reformulations of methods. This is not too much of a difference because purely procedural line-justifications, such as OTTER, are allowed and this is equivalent to a tactic application. The described representation of methods works for the Heine-Borel example whose source was originally given as a sequence of subplans consisting of sequences of proof lines justified by OTTER [Ble94]. The original representation has been translated into a proof plan consisting of methods.

A method is *correct* if, given pre(M), the method yields a correct proof of post(M) in case the constraints are satisfied. For verifying a line with an OTTER-call, a time limit is set for OTTER to prove the sequent. In OMEGA, a verifier does this job as described in section 6.

Goals and assumptions are sequents. A *proof plan* is an abstract representation of a proof that consists of trees of method nodes which are connected by sequent nodes that satisfy the "link condition": A method node M follows a sequent node g and precedes the sequent nodes $\mathsf{g}_1, \ldots, \mathsf{g}_n$ if $\mathsf{g} \in \sigma(\mathrm{post(M)})$ and $\sigma(\mathrm{pre(M)}) = \{\mathsf{g}_1, \ldots, \mathsf{g}_n\}$ for a substitution σ of parameters. For an illustration see Figure 1. A proof plan is a high-level representation of a proof because the methods represent large chunks of calculus-level proof steps.

Proof planning starts with a goal g and assumptions $(\emptyset \vdash F_i)$ and $(H_j \vdash H_j)$, for axioms, definitions, or lemmata F_i and proof-assumptions H_j. Since mathematical proofs are constructed top down and bottom up, we consider backward *and* forward search in proof planning too. The proof planning inserts methods and sequents into a proof plan: Backward proof planning inserts a method following a goal and yielding new (sub)goals as its successors. Forward planning inserts a method preceding some assumptions and yielding a new assumption. Planning aims at reducing the gap between leaf goals and assumptions. Leaf goals that are not equal to an assumption are called *open goals.* As soon as a goal g_i equals an assumption, the two nodes collapse. Then g_i is no longer an open goal but is *satisfied.* The planning terminates if there are no more open

[8] Besides, the slots are renamed, e.g., our preconditions are named input there.

goals.

A source proof plan is a tree with the source problem at the root, with no open goals, and with correct methods. As in [Vel94], justification structures, used to encode justifications for the decision made, annotate the plan nodes. These *justifications* capture the subgoaling structure of a plan and point to reasons for the choice of a method or of a goal, such as application conditions of a method, user-given guidance, or pre-programmed control knowledge. The correctness of a method is one of the justifications and it is used in the Heine-Borel example.

4 The Heine-Borel Examples

Woody Bledsoe [Ble95a] suggested the analogical transfer of the proof of the theorem HB1 to a proof of HB2 as a challenge problem. First, let us look at the source theorem HB1 and the target theorem HB2. (Some abbreviations are listed in section 4.1. The lemmata used can be found in appendix A.)

THEOREM: *(Heine-Borel-1* HB1*) If a closed interval* $[a,b]$ *of* R^1 *is covered by a family* G *of open sets (in* R^1*), then there is a finite subfamily* H *of* G *which covers* $[a,b]$*.*[9] ■

THEOREM: *(Heine-Borel-2* HB2*) If a closed rectangle* $[a,b,c,d]$ *of* R^2 *is covered by a family* G *of open sets (in* R^2*), then there is a finite subfamily* H *of* G *which covers* $[a,b,c,d]$*.* ■

For readers who are interested, a mathematical proof of HB1 by contradiction may go like this:
A sequence $g(i)$ is defined by
$g(0) = [a, b]$

$$g(i{+}1) = \begin{cases} [lf(g(i)), mid(g(i))] & : \text{if no finite subset of } G \text{ covers} [lf(g(i)), mid(g(i))] \\ [mid(g(i)), rt(g(i))] & : \text{otherwise} \end{cases}$$

for $mid(g(i)) = lf(g(i)) + (lf(g(i)) - rt(g(i)))/2$.[10] By induction on i it is shown that $g(i)$ is a nested sequence of closed intervals and that for each $i \in \mathbb{N}$ no finite subset of G covers $g(i)$ and $(rt(g(i)) - lf(g(i))) = (b-a)/2^i$. Since $g(i)$ is a nested sequence of non-empty closed intervals, by the Nested Interval Lemma there is a z for which $z \in g(i)$ for each $i \in \mathbb{N}$. Thus, $z \in [a, b]$ and since $[a, b] \subset \cup G$, it follows by Lemma6[11] that $z \in B$ for some $B \in G$. Since B is open, Lemma7 yields that there is a closed interval $[u, v]$ for which $u < v, [u, v] \subset B$, and $z = mid([u, v])$. By Lemma8 choose i such that $(b-a)/2^i < (u-v)/2$.

[9] R^1 denotes the set of real numbers and R^2 denotes the set of ordered pairs (x,y) of real numbers x and y.

[10] lf, rt, and mid denote the lower and the upper bound and the middle of an interval, respectively

[11] The numbering of the lemmata follows the listing in appendix A.

The result $g(i) \subset [u, v] \subset B$ follows by Lemma9 and thus $g(i) \subset \cup(\{B\})$ by Lemma12, where $\{B\}$ is a finite subset of G.

In principle, a mathematical proof of HB2 could follow the same line. However, the sequence of intervals has to be defined differently and several lemmata and their use have to be adapted to the two dimensional case.

I would like to make clear that essentially, the first-order (unsorted) formalization of HB1 and of its proof are due to Bledsoe. For a discussion of the formalization and the proof see section 7.

4.1 Some Abbreviations

This subsection provides abbreviations that will be used in the following proof plans and methods. Δ, Δ' are the proof assumptions from HB1 and HB2 respectively. Th, Th' are the two theorem's conclusions. F_i are the intermediate results in the proof plans, and P, Q are abbreviations introduced in the original problem specification [Ble94]. Bledsoe proposed the lemmata listed in appendix A.[12]. In particular, he substantiated the use of CIA as follows. "Our first version of the plan was to define g recursively .... But the induction steps require large FOL theorems to be proved which are simple and obvious to people, and 'impossible' for our 'helper-prover' OTTER. So we changed to another version, to avoid the problems with OTTER, and to make use of the more natural method used by mathematicians. The basic idea is this induction-definition-axiom" [Ble94].

The constant, relation, and function symbols have the following meaning: *clsdint* denotes the relation "is a closed interval" $rt, lf, [x, y], [x, y, s, t]$ denote the functions "right bound of", "left bound of", "closed interval from x to y", "closed rectangular with vertices $(x, y), (s, y), (s, t), (x, t)$, respectively. $\mathbb{N}, R$ denote the natural numbers and real numbers respectively. The abbreviations are:

Δ : The set of proof assumptions of HB1
$\{a \in R \wedge b \in R \wedge a \leq b, \forall B(B \in G \rightarrow open(B)), [a, b] \subset \cup G\}$

Th : $\exists H(H \subseteq G \wedge finite(H) \wedge [a, b] \subset \cup H)$

Δ' : The set of proof assumptions of HB2
$\{a \in R \wedge b \in R \wedge c \in R \wedge d \in R \wedge a \leq b \wedge c \leq d,$
$\forall B(B \in G \rightarrow open(B)), [a, b, c, d] \subset \cup G\}$

Th' : $\exists H(H \subseteq G \wedge finite(H) \wedge [a, b, c, d] \subset \cup H)$

HB1 : $\Delta \vdash Th$

HB2 : $\Delta' \vdash Th'$

$P(D)$: $clsdint(D) \wedge \neg\exists H(H \subseteq G \wedge finite(H) \wedge D \subset \cup H) \wedge lf(D) \leq rt(D)$

$Q(D, D')$: $D' \subseteq D \wedge (rt(D') - lf(D')) = (rt(D) - lf(D))/2$

[12] The original lemmata had to be slightly changed in order to be correct.

The intermediate results (goals, assumptions) in the proof plan are

$F1 \,:\, g(0) = [a,b] \land \forall i(i \in \mathbb{N} \to clsdint(g(i)) \land g(i+1) \subset g(i) \land$
$(rt(g(i)) - lf(g(i))) = (rt([a,b]) - lf([a,b]))/2^i \land lf(g(i)) \leq rt(g(i)) \land$
$\neg\exists H(H \subseteq G \land finite(H) \land g(i) \subset \cup H)).$

$F2 \,:\, \exists i \exists H(i \in \mathbb{N} \land H \subseteq G \land finite(H) \land g(i) \subset \cup H)$

$F3 \,:\, g(0) = [a,b] \land \forall i(i \in \mathbb{N} \to P(g(i)) \land Q(g(i), g(i+1)))$

$F4 \,:\, P([a,b])$

4.2 A Proof Plan for HB1

Figure 1 shows the proof plan for HB1 which, in fact, yields a proof of HB1 when executed. The numbers annotating the methods in the figure indicate the sequence in which the methods have been introduced into the proof plan. The plan's methods shown below have been tested to be correct. Their empty constraints slots are due to the way the plan was built, namely completely interactively and without using any constraining meta-knowledge. Our analogy system works, however, for automatically produced proof plans and for proof plans with more general methods as well. For a discussion see section 7.

Since the overall proof is by refutation the first method is `Refutation` as given above. method-111 up to method-1 prove the existence of a sequence $g(i)$ that satisfies the intermediate assumption F1 which describes certain properties of the intervals $g(i)$. F1 is then used by method-2 to prove that for some i, $g(i)$ is finitely covered. This result together with F1 can be used by method-0 to infer a contradiction. method-111 proves F4, which states properties of $g(0)$ and essentially is the proof obligation for $i = 0$ of F1; method-111 uses the lemmata L3 and L4. For clarity reasons the lemmata are not mentioned explicitly in the LHS of the sequents in proof schema lines. They logically belong to the hypotheses, however.[13] method-111 looks as follows.

[13] $\forall E^*$ iterates over the ND-rule $\forall E$.

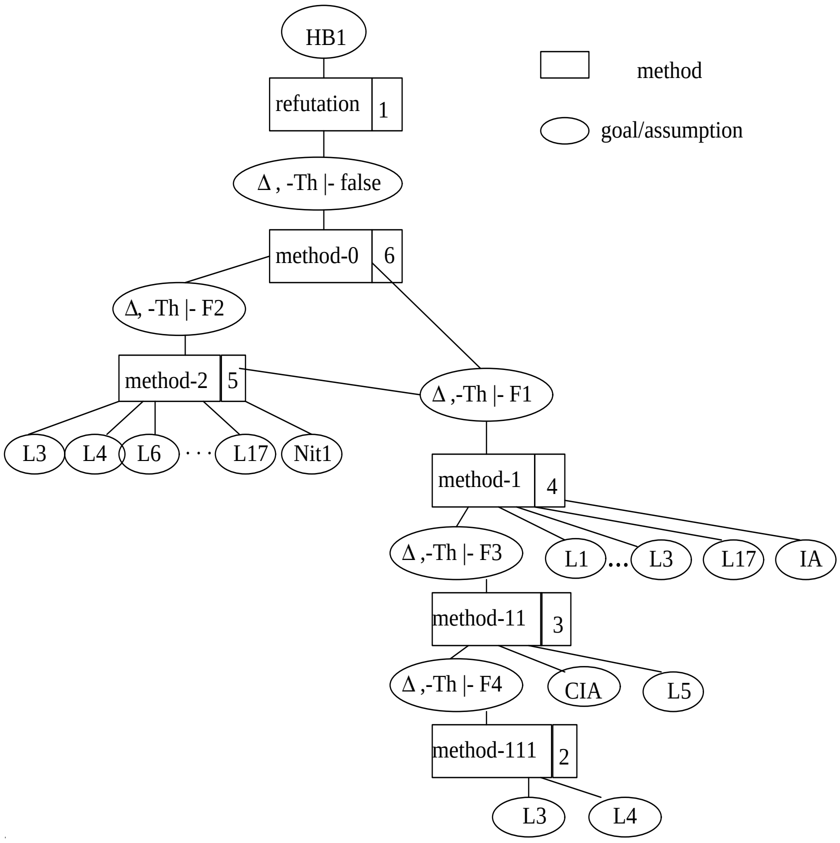

Figure 1: The proof plan of HB1 with the annotated sequence of method nodes

method: `-111`	
parameter	$a, b, G, clsdint, rt, lf, [-,-]$
preconditions	L3,L4
postcondition	$(\Delta, \neg Th \vdash F4)$
constraints	
proof schema	0-1. $\Delta \vdash a \in R \wedge b \in R \wedge a \leq b$ (HYP) 1. $\Delta \vdash a \in R \wedge b \in R$ ($\wedge$E,0-1) 2. $\Delta \vdash a \in R \wedge b \in R \rightarrow clsdint([a,b])$ ($\forall$E*;L3) 3. $\Delta \vdash clsdint([a,b])$ ($\rightarrow$E,1,2) 4. $\{\neg Th\} \vdash \neg\exists H(H \subseteq G \wedge finite(H) \wedge [a,b] \subset \cup H)$ (HYP) 5. $\Delta \vdash lf([a,b]) \leq rt([a,b])$ (OTTER;L4, 0-1) 6. $\Delta, \neg Th \vdash clsdint([a,b]) \wedge \neg\exists H(H \subseteq G \wedge finite(H) \wedge [a,b] \subset \cup H)$ ($\wedge$I,3,4) 7. $\Delta, \neg Th \vdash F4$ ($\wedge$I,6,5)
procedure	schema-interpreter

method-11 yields the intermediate result F3 using the lemmata L5, CIA and the previous intermediate result F4:

method: `-11`	
parameter	$a, b, G, clsdint, rt, lf, [-,-], g$
preconditions	$(\Delta, \neg Th \vdash F4)$,L5,CIA
postcondition	$(\Delta, \neg Th \vdash F3)$
constraints	
proof schema	1. $\Delta, \neg Th \quad \vdash \quad F4$ (OPEN) 2. $\Delta, \neg Th \quad \vdash \quad \exists g(g(0) = [a,b] \land \forall i(i \in \mathbb{N} \to P(g(i)) \land Q(g(i), g(i+1))))$ (OTTER;L5, CIA,1) 3. $\Delta, \neg Th \quad \vdash \quad F3$ ($\exists$E;2)
procedure	schema-interpreter

method-1 adds the subformula $\forall i(rt(g(i)) - lf(g(i))) = (rt([a,b]) - lf([a,b]))/2^i$ about the length of the interval $g(i)$ w.r.t. to the length of the interval $[a,b]$ to F3. method-1 uses the lemmata L1, L2 ,L3, L17, and IA to prove F1.

method: `-1`	
parameter	$a, b, G, clsdint, rt, lf, [-,-], g$
preconditions	$(\Delta, \neg Th \vdash F3)$,L1,L2,L3,L17,IA
postcondition	$(\Delta, \neg Th \vdash F1)$
constraints	
proof schema	0-1.$\Delta \quad \vdash \quad a \in R \land b \in R \land a \leq b$ (HYP) 1. $\Delta, \neg Th \quad \vdash \quad F3$ (OPEN) 2. $\Delta, \neg Th \quad \vdash \quad (rt(g(0)) - lf(g(0)) = (rt([a,b]) - lf([a,b]))/2^0$ (OTTER;L1, L17,L3,01,1) 3. $\Delta, \neg Th \quad \vdash \quad \forall i(i \in \mathbb{N} \land (rt(g(i)) - lf(g(i))) = (rt([a,b]) - lf([a,b]))/2^i \to ((rt(g(i+1)) - lf(g(i+1))) = (rt([a,b]) - lf([a,b]))/2^{i+1}))$ (OTTER;L2, L17,1) 4. $\Delta, \neg Th \quad \vdash \quad \forall i(i \in \mathbb{N} \to (rt(g(i)) - lf(g(i))) = (rt([a,b]) - lf([a,b]))/2^i)$ (OTTER;IA, 2,3) 5. $\Delta, \neg Th \quad \vdash \quad F1$ (OTTER;1,4)
procedure	schema-interpreter

method-2 provides the intermediate result F2 by finding an interval $g(i)$ that is finitely covered. As indicated in the brief mathematical proof, method-2 employs the Nested Interval Lemma and several other lemmata.

<table>
<tr><td colspan="2">method: -2</td></tr>
<tr><td>parameter</td><td>$a, b, G, clsdint, rt, lf, [-,-], g, center$</td></tr>
<tr><td>preconditions</td><td>$(\Delta, \neg Th \vdash F1)$,Nit1,L3,L4,L6,... L17</td></tr>
<tr><td>postcondition</td><td>$(\Delta, \neg Th \vdash F2)$</td></tr>
<tr><td>constraints</td><td></td></tr>
<tr><td>proof schema</td><td>
<table>
<tr><td>0-1.</td><td>Δ</td><td>$\vdash$</td><td>$a \in R \wedge b \in R \wedge a \leq b$</td><td>(HYP)</td></tr>
<tr><td>0-2.</td><td>Δ</td><td>$\vdash$</td><td>$\forall B(B \in G \to open(B))$</td><td>(HYP)</td></tr>
<tr><td>0-3.</td><td>Δ</td><td>$\vdash$</td><td>$[a, b] \subset \cup G$</td><td>(HYP)</td></tr>
<tr><td>1.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$F1$</td><td>(OPEN)</td></tr>
<tr><td>2.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\exists z \forall i(i \in \mathbb{N} \to z \in g(i))$</td><td>(OTTER; Nit1,1)</td></tr>
<tr><td>2a.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\forall i(i \in \mathbb{N} \to z \in g(i))$</td><td>($\exists$E;2)</td></tr>
<tr><td>3.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\exists B(B \in G \wedge z \in B)$</td><td>(OTTER;L6, 1,2a,0-3)</td></tr>
<tr><td>3a.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$B \in G \wedge z \in B$</td><td>($\exists$E;3)</td></tr>
<tr><td>4.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\exists u \exists v(u \in R \wedge v \in R \wedge u < v \wedge [u, v] \subset B \wedge center(z, [u, v]))$</td><td>(OTTER;L7, 3a,0-2)</td></tr>
<tr><td>4a.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$(u \in R \wedge v \in R \wedge u < v \wedge [u, v] \subset B \wedge center(z, [u, v]))$</td><td>($\exists$E,4)</td></tr>
<tr><td>5.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\exists i(i \in \mathbb{N} \wedge (rt([a, b]) - lf([a, b]))/2^i \leq (rt([u, v]) - lf([u, v]))/2)$</td><td>(OTTER;L8, L15,L16,L17, L3,L4, 4a,0-1)</td></tr>
<tr><td>5a.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$(i \in \mathbb{N} \wedge (rt([a, b]) - lf([a, b]))/2^i \leq (rt([u, v]) - rt([u, v]))/2)$</td><td>($\exists$E;5)</td></tr>
<tr><td>6.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\exists c \exists d(c \leq d \wedge g(i) = [c, d] \wedge c \in R \wedge d \in R)$</td><td>(OTTER; L11,1,5a)</td></tr>
<tr><td>7.</td><td>$\Delta \neg Th$</td><td>$\vdash$</td><td>$g(i) \subset [u, v] \wedge i \in \mathbb{N}$</td><td>(OTTER;L9, 1,2a,4a,5a,6)</td></tr>
<tr><td>8.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$g(i) \subset \cup\{B\} \wedge i \in \mathbb{N}$</td><td>(OTTER;L12 L13,3a,4a,7)</td></tr>
<tr><td>9.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$F2$</td><td>(OTTER;L10 L14,3a,8)</td></tr>
</table>
</td></tr>
<tr><td>procedure</td><td>schema-interpreter</td></tr>
</table>

Finally, method-0 collects the results of method-1 and method-2 in order to prove a contradiction with the help of OTTER

<table>
<tr><td colspan="2">method: -0</td></tr>
<tr><td>parameter</td><td>$a, b, G, clsdint, rt, lf, [-,-], g$</td></tr>
<tr><td>preconditions</td><td>$(\Delta, \neg Th \vdash F1)$, $(\Delta, \neg Th \vdash F2)$</td></tr>
<tr><td>postcondition</td><td>$(\Delta, \neg Th \vdash \bot)$</td></tr>
<tr><td>constraints</td><td></td></tr>
<tr><td>proof schema</td><td>
<table>
<tr><td>1.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$F1$</td><td>(OPEN)</td></tr>
<tr><td>2.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\neg F2$</td><td>(OTTER;1)</td></tr>
<tr><td>3.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$F2$</td><td>(OPEN)</td></tr>
<tr><td>4.</td><td>$\Delta, \neg Th$</td><td>$\vdash$</td><td>$\bot$</td><td>($\bot$I;2,3)</td></tr>
</table>
</td></tr>
<tr><td>procedure</td><td>schema-interpreter</td></tr>
</table>

5 Analogy-Driven Proof Plan Construction

The analogy model, developed in my earlier papers [Mel93b, Mel95b], views analogy as a control strategy of proof planning that suggests a method for the proof plan rather than searching for an appropriate method or asking the user. The analogy-driven proof plan construction has the following characteristics.

- Proof *plans* are transferred analogically,
- Proof plans can be changed by *reformulations* that are more powerful than mere symbol mapping,
- *Derivational* analogy [Car86] is used for the transfer. That is, the analogy guides the search for a target solution by replaying decisions of the problem solving *process* for the source problem if the reasons for the decisions (*justifications*) hold in the target.

5.1 Reformulation

Reformulations are mappings of proof plans to proof plans which usually (but not necessarily) preserve the correctness of methods. They encode mathematical heuristics on how the changes in a proof plan depend on certain changes of the problem to be proved. Reformulations may insert methods or replace methods and may change methods, sequents, and justifications of methods.

For instance, the reformulation `Add-Argument` is applicable if a n-ary function f is to be mapped to a 2n-ary function f'.[14] The program `Add-Argument` replaces the function f by f' with duplicated arguments in goals, assumptions, justifications, and methods. It yields additional related changes in the proof schema of effected methods such as replacing certain subformulas by conjunctions and duplicating certain lines of the proof schema which may also cause the duplication of certain preconditions of the method (see appendix B for more details). If a method's precondition P became duplicated, then `Add-Argument` modifies the structure of the proof plan by duplicating the subplan that yields the sequent P in the original proof plan.

For instance, the function $[-,-]$ that returns a real interval $[x,y]$ when given a pair of real numbers x,y is mapped to a function $[-,-,-,-]$ that returns an interval $[x,y,z,w]$ in R^2 when given four real arguments x,y,z,w. Then a proof-schema line
$\emptyset \vdash a \in R \wedge b \in R \rightarrow clsdint([a,b])$
would be changed to the line
$\emptyset \vdash a \in R \wedge b \in R \wedge c \in R \wedge d \in R \rightarrow clsdint([a,b,c,d])$

Even though it might seem that `Add-Argument` was designed exactly for the Heine-Borel example or that this particular example was chosen to fit the reformulation, this is *not* the case. `Add-Argument` proved to be a fairly frequent

[14]It is the simplest version of a more general reformulation `Add-Arguments` that is applicable if an n-ary function f is to be mapped to an (n+m)-ary function f', where arguments x_i of f are mapped to arguments x_{ij} of f'.

and general reformulation that has been used in many examples documented in [Mel93b, MV94, HKKR94]. Moreover, the particular Heine-Borel example was chosen independently in [Ble95a].

The objection that every new example would need a new reformulation did not turn out to be true in our experiments with the analogy procedure.

5.2 Analogical Replay

Table 1 shows the top-level procedure of analogy-driven proof plan construction. This description is general enough to cover analogy for different proof planners.

---

input: source plan, source theorem and assumptions, target theorem and assumptions
output: target plan

Attempt to second-order match triggers reformulations of the source plan.
source plan := reformulated source plan.

if setting `suggest-lemmata`, **then** suggest target lemma for each source lemma with no image in the target. **if** lemma not rejected, **then** introduce it as a *suggested-target-lemma*.

while source plan not exhausted **do** replay:
    Get next method M from source plan.
    Check M's justifications.
    if justifications hold, **then** transfer M to target and advance source,
    else choose suitable action.

Table 1: Outline of the analogy-driven proof plan construction

In fact, the analogy procedure has been implemented and successfully used in the proof planner $CL^{A}M$[BvHS91], see [Mel95a]. It is also being implemented in the OMEGA system [CLD^{+}97]. Of course, the concrete implementation depends on the respective planner and its methods.

Given a source theorem, source assumptions (lemmata, proof assumptions, definitions, axioms), a source proof plan, a target theorem, and target assumptions, the procedure returns a target proof plan.

First, the procedure attempts to second-order match the source and target theorems and then the source and target assumptions. The second-order matching attempts trigger reformulations by analyzing success and partial failure of the matching attempts. These reformulations are applied to the source plan. The matchings can also introduce suggested target lemmata if `suggest-lemmata` is set (we have different possible settings for suggesting lemmata, user interaction, and amount of output) and if the lemma is not rejected by the user. (This introduction of target lemmata could as well be a specific re-action to a failed justification about the existence of a lemma.) Then the analogical replay follows the methods of the source plan in the sequence in which they have been introduced into the plan. For each method M of the reformulated source plan, M is transferred if its justifications hold in the target. If a justification of M does not hold in the target, different actions can be taken depending on the kind of violated justification. For instance,

- If the current goal is satisfied in the target already, then M is superfluous in the target and can be skipped.
- If a lemma needed for a method M is not available in the target (because it has been rejected by the user), then an action introduces a gap into the proof plan by making the postcondition of M an open goal that has to be planned for later.
- If the correctness-justification does not hold for M, then an action tries to *modify* M to a correct method M'. One possible modification technique decomposes M such that a correct submethod M' results with post(M)= post(M') and M' replaces M. This technique creates new open goals.

When all methods of the source plan have been visited, then the remaining open goals have to be proved by base-level planning, that is, planning that is not guided by analogy. This situation is similar to human theorem proving by analogy that usually leaves details to be proved.

6 Proving HB2 by Analogy

The attempt to second-order match the (parametrized) source theorem HB1 to the target theorem HB2 triggers the reformulation `Symbol-Mapping` for sending a, b, G to a, b, G and the reformulation `Add-Argument` which inter alia replaces the binary interval function $[\_,\_]$ by the 4-ary interval function $[\_,\_,\_,\_]$. Then the second-order matching of the lemmata suggests `Symbol-Mapping` for sending $lf, rt, clsdint$ to $lf, rt, clsdrec$.[15] L1, L2, L6, L8, L10, L12, L13, L14, L15, L16 are potential lemmata for HB2 because they contain only symbols not specific for R^1. With the setting `suggest lemmata`, the remaining lemmata of the HB1 problem are reformulated to suggested lemmata for the HB2 problem. In the following, we present a variant of the replay in which none of the lemmata suggested by the system has been rejected by the user.

A connection table `CT`, as originally suggested by Bledsoe, has not been used for the current analogical transfer of HB1. `CT` can, however, be defined additionally by the user. A connection table that contains associations such as ($clsdint$ $clsdrec$) and ($length, diam$), provides semantic information. It restricts the search for reformulations and reduces the number of symbols that have to be interpreted in order to prove target lemmata suggested by analogy. `CT` is a *permanent* connection table, which is not designed for a particular analogy problem, but is supposed to contain information for a large area of mathematics.

[15] $clsdrec$ denotes the relation “is closed rectangle”and , $length$ denotes the “length of an interval”.

According to the analogy outline, the HB1 proof plan is reformulated. Then the methods for which the justifications hold are transferred step by step. The justifications given for the HB1 methods that have to be checked in the target are the correctness of the method and the existence of lemmata used by the method. For the former purpose, OMEGA's verifier is invoked that checks the correctness of methods. The verifier processes lines with OTTER line-justifications by normalizing the problem contained in the line to clausal form and producing an input file for OTTER with auto modus and with a default time limit of five minutes, in case no limit is given in the method's line.

Now let us look at the process that is going on step by step:

1. The refutation method is unchanged but all other methods are changed by the reformulation. The refutation method stays correct and is transferred to the target plan.

2. The next method is the (reformulated) method-111'. The function symbols lf, rt and lf', rt' in method-111' that have been introduced by `Add-Argument` remain variables because L4 has a *suggested* image lemma L4' in the target only.

   The suggested and accepted target lemmata L3' and L4' satisfy the lemma-justification. The correctness check of method-111' yields a minor incorrectness in line 1 of the reformulated method which invokes an action: The correctness can be established by our extended verifier which allows for very simple modifications.[16] The need for those modifications is due to the unsatisfying low level (ND-)representation in proof schemas. See the discussion in section 7.

   The modified method-111' is correct and is transferred to the target.

method: `-111'`	
parameter	rt, lf, rt', lf'
preconditions	L3',L4'
postcondition	$(\Delta', \neg Th' \vdash F4')$
constraints	type(rt)=type(rt'), type(lf)=type(lf')
proof schema	0-1. Δ' $\vdash$ $a \in R \land b \in R \land a \leq b \land c \in R \land d \in R \land c \leq d$ (HYP) 1. Δ' $\vdash$ $a \in R \land b \in R \land c \in R \land d \in R$ ($\land$EM,0-1) 2. Δ' $\vdash$ $a \in R \land b \in R \land c \in R \land d \in R \rightarrow clsdrec([a,b,c,d])$ ($\forall$E*;L3') 3. Δ' $\vdash$ $clsdrec([a,b,c,d])$ ($\rightarrow$E,1,2) 4. $\{\neg Th'\}$ $\vdash$ $\neg\exists H(H \subseteq G \land finite(H) \land [a,b,c,d] \subset \cup H)$ (HYP) 5. Δ' $\vdash$ $lf([a,b,c,d]) \leq rt([a,b,c,d]) \land lf'([a,b,c,d]) \leq rt'([a,b,c,d])$ (OTTER;L4', 1) 6. $\Delta', \neg Th'$ $\vdash$ $clsdint([a,b,c,d]) \land \neg\exists H(H \subseteq G \land finite(H) \land [a,b,c,d] \subset \cup H)$ ($\land$I,3,4) 7. $\Delta', \neg Th'$ $\vdash$ $F4'$ ($\land$I,5,6)
procedure	schema-interpreter

[16] It modifies $\land$E to $\land$EM, where $\land$EM is a tactic that can apply $\land$E and $\land$I.

3. After method-111' is transferred to the target, the next method is method-11'. method-11' is correct. The suggested and accepted target lemmata L5' and CIA' satisfy the lemma-justification. Therefore, method-11' is transferred to the target.

method: `-11'`	
parameter	rt, lf, rt', lf'
preconditions	$(\Delta', \neg Th' \vdash F4')$,L5',CIA'
postcondition	$(\Delta', \neg Th' \vdash F3')$
constraints	type(rt)=type(rt'), type(lf)=type(lf')
proof schema	1. $\Delta', \neg Th' \vdash F4'$ (OPEN) 2. $\Delta', \neg Th' \vdash \exists g(g(0) = [a,b,c,d] \wedge \forall i(i \in \mathbb{N} \rightarrow P'(g(i)) \wedge Q'(g(i), g(i+1))))$ (OTTER;L5', CIA',1) 3. $\Delta', \neg Th' \vdash F3'$ ($\exists$E;2)
procedure	schema-interpreter

4. The next method in the source plan, method-1', is correct. The suggested target lemmata L3', L17', and IA' satisfy the lemma-justification. Hence, method-1' is transferred to the target.

method: `-1'`	
parameter	rt, lf, rt', lf'
preconditions	$(\Delta', \neg Th' \vdash F3')$,L1,L2,L3',L17',IA'
postcondition	$(\Delta', \neg Th' \vdash F1')$
constraints	type(rt)=type(rt'), type(lf)=type(lf')
proof schema	0-1. $\Delta' \vdash a \in R \wedge b \in R \wedge a \leq b \wedge c \in R \wedge d \in R \wedge c \leq d$ (HYP) 1. $\Delta', \neg Th' \vdash F3'$ (OPEN) 2. $\Delta', \neg Th' \vdash (rt(g(0)) - lf(g(0)) = (rt([a,b,c,d]) - lf([a,b,c,d]))/2^0 \wedge (rt'(g(0)) - lf'(g(0)) = (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^0$ (OTTER;L1, L17',L3',01,1) 3. $\Delta', \neg Th' \vdash \forall i(i \in \mathbb{N} \wedge (rt(g(i)) - lf(g(i))) = (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i \wedge (rt'(g(i)) - lf'(g(i))) = (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i \rightarrow ((rt(g(i+1)) - lf(g(i+1))) = (rt([a,b]) - lf([a,b,c,d]))/2^{i+1}) \wedge (rt'(g(i+1)) - lf'(g(i+1))) = (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^{i+1}))$ (OTTER;L2, L17',1) 4. $\Delta', \neg Th' \vdash \forall i(i \in \mathbb{N} \rightarrow (rt(g(i)) - lf(g(i))) = (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i) \wedge (rt'(g(i)) - lf'(g(i))) = (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i)$ (OTTER;IA', 2,3) 5. $\Delta', \neg Th' \vdash F1'$ (OTTER;1,4)
procedure	schema-interpreter

5. The next method to consider is method-2'. The variable *center* remains in the preconditions of method-2' since no original target lemma to match L7 or L9 is given. The suggested target lemmata Nit2, L7', L9', L11' satisfy the lemma-justification of the candidate method-2'. Unfortunately, method-2' is not correct because line-5a of method-2' cannot be verified. The reason is that

$$\exists i (i \in \mathbb{N} \wedge (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i \leq (rt([u,v,s,t]) - lf([u,v,s,t]))/2$$
$$\wedge\ (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i \leq (rt'([u,v,s,t]) - lf'([u,v,s,t]))/2)$$

cannot be proved from L8 etc. For a proof, of the above formula, the additional knowledge would be needed that for $max = max(i,j)$ in R^1 holds
$z/2^i \leq x \wedge z/2^j \leq y \rightarrow z/2^{max} \leq x \wedge z/2^{max} \leq y$. A modification action corresponding to the failure of the correctness decomposes method-2' into a plan consisting of a correct method-21', a subgoal $\mathtt{g'}_{5a}$, and a method-22':

- method-21'
- subgoal $\mathtt{g'}_{5a}$ is $\Delta', \neg Th' \vdash (i \in \mathbb{N} \wedge (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i \leq (rt([u,v,s,t]) - lf([u,v,s,t]))/2 \wedge (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i \leq (rt'([u,v,s,t]) - lf'([u,v,s,t]))/2)$.
- method-22' is built as the complement of method-21' w.r.t. method-2'.

method: `-21'`			
parameter	$rt, lf, rt', lf', center$		
preconditions	$(\Delta', \neg Th' \vdash F1')$, $\mathtt{g}_{5a}$, Nit2,L6,L7',L9',L10,L11',L12,. . .,L14		
postcondition	$(\Delta', \neg Th' \vdash F2)$		
constraints	type(rt)=type(rt'), type(lf)=type(lf')		
proof schema	0-2. Δ'	$\vdash \forall B(B \in G \rightarrow open(B))$	(HYP)
	0-3. Δ'	$\vdash [a,b,c,d] \subset \cup G$	(HYP)
	1. $\Delta', \neg Th'$	$\vdash F1'$	(OPEN)
	2. $\Delta', \neg Th'$	$\vdash \exists z \forall i(i \in \mathbb{N} \rightarrow z \in g(i))$	(OTTER; Nit2,1)
	2a. $\Delta', \neg Th'$	$\vdash \forall i(i \in \mathbb{N} \rightarrow z \in g(i))$	($\exists$E;2)
	3. $\Delta', \neg Th'$	$\vdash \exists B(B \in G \wedge z \in B)$	(OTTER;L6, 1,2a,0-3)
	3a. $\Delta', \neg Th'$	$\vdash B \in G \wedge z \in B$	($\exists$E;3)
	4. $\Delta', \neg Th'$	$\vdash \exists u \exists v \exists s \exists t(u \in R \wedge v \in R \wedge s \in R \wedge t \in R \wedge u < v \wedge s < t \wedge [u,v,s,t] \subset B \wedge center(z,[u,v,s,t]))$	(OTTER;L7', 3a,0-2)
	4a. $\Delta', \neg Th'$	$\vdash (u \in R \wedge v \in R \wedge u < v \wedge s \in R \wedge t \in R \wedge s < t \wedge [u,v,s,t] \subset B \wedge center(z,[u,v,s,t]))$	($\exists$E,4)
	5a. $\Delta', \neg Th'$	$\vdash (i \in \mathbb{N} \wedge (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i \leq (rt([u,v,s,t]) - rt([u,v,s,t]))/2 \wedge (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i \leq (rt'([u,v,s,t]) - lf'([u,v,s,t]))/2)$	(OPEN)
	6. $\Delta', \neg Th'$	$\vdash \exists c \exists d \exists e \exists f(c \leq d \wedge e \leq f \wedge g(i) = [c,d,e,f] \wedge c \in R \wedge d \in R \wedge e \in R \wedge f \in R)$	(OTTER; L11',1,5a)
	7. $\Delta' \neg Th'$	$\vdash g(i) \subset [u,v,s,t] \wedge i \in \mathbb{N}$	(OTTER;L9', 1,2a,4a,5a,6)
	8. $\Delta', \neg Th'$	$\vdash g(i) \subset \cup\{B\} \wedge i \in \mathbb{N}$	(OTTER;L12 L13,4a,7)
	9. $\Delta', \neg Th'$	$\vdash F2$	(OTTER;L10 L14,3a,8)
procedure	schema-interpreter		

The modified method-21' is correct and, hence, is transferred to the target, while the method-22' is not. The subgoal $\mathtt{g'}_{5a}$ becomes an open goal which remains open until the source plan is exhausted. That means, it has to be proved by base-level planning.

6. After method-21' is transferred to the target plan, the next method is method-0' which is correct and can, therefore, be transferred to the target.

6.1 The Resulting Proof Plan for HB2

The analogy procedure as implemented in OMEGA yields the target proof plan, shown in Figure 2, with reformulated subgoals and methods that have been proved to be correct.

The proof plan for HB2 looks like that for HB1 except that the new lemmata L3' ... replace L3 ... and all methods, but method-2, are replaced by the corresponding reformulated methods. method-2 is replaced by its reformulated submethod method-21' and an additional subgoal. The analogy procedure suggests as open goals the following lemmata which are left to be proved.

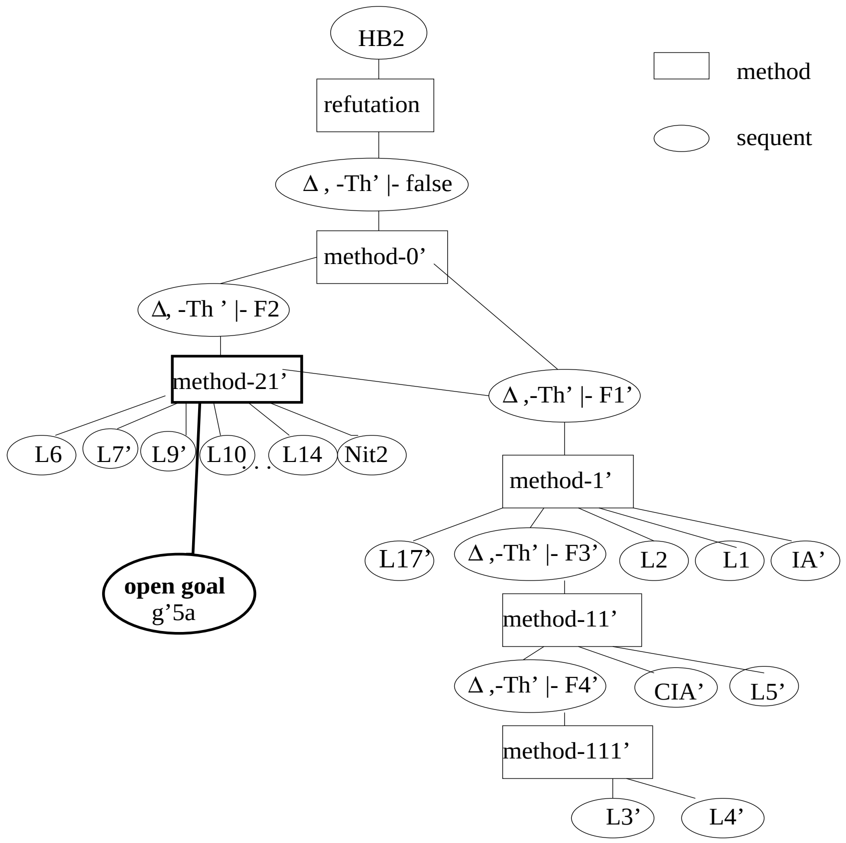

Figure 2: The proof plan of HB2

L3' : $\emptyset \vdash \forall x \forall y \forall z \forall w(x \in R \wedge y \in R \wedge z \in R \wedge w \in R \rightarrow clsdrec([x,y,z,w]))$

L4' : $\emptyset \vdash \forall x \forall y \forall z \forall w(x \leq y \wedge z \leq w \rightarrow lf([x,y,z,w]) = x \wedge rt([x,y,z,w]) = y \wedge lf'([x,y,z,w]) = z \wedge rt'([x,y,z,w]) = w)$

L5' : $\emptyset \vdash \forall D \forall G(clsdrec(D) \wedge \neg\exists H(H \subset G \wedge finite(H) \wedge D \subset \cup H) \wedge lf(D) \leq rt(D) \wedge lf'(D) \leq rt'(D) \rightarrow \exists D'(clsdrec(D') \wedge \neg\exists H(H \subseteq G \wedge finite(H) \wedge D' \subset \cup H) \wedge lf(D') \leq rt(D') \wedge lf'(D') \leq rt'(D') \wedge D' \subseteq D \wedge (rt(D') - lf(D')) = (rt(D) - lf(D))/2 \wedge (rt'(D') - lf'(D')) = (rt'(D) - lf'(D))/2))$

L7' : $\emptyset \vdash \forall x \forall B(x \in B \wedge open(B) \rightarrow \exists u \exists v \exists s \exists t(u \in R \wedge v \in R \wedge s \in R \wedge t \in R \wedge u < v \wedge s < t \wedge [u,v,s,t] \subset B \wedge center(x,[u,v,s,t])))$

L9' : $\emptyset \vdash \forall x \forall u \forall v \forall s \forall t \forall c \forall d \forall e \forall f(u \in R \wedge v \in R \wedge s \in R \wedge t \in R \wedge c \in R \wedge d \in R \wedge e \in R \wedge f \in R \wedge u \leq v \wedge s \leq t \wedge c \leq d \wedge e \leq f \rightarrow (center(x,[u,v,s,t]) \wedge x \in [c,d,e,f] \wedge$
$(rt([c,d,e,f]) - lf([c,d,e,f])) \leq (rt([u,v,s,t]) - lf([u,v,s,t]))/2 \wedge$
$(rt'([c,d,e,f]) - lf'([c,d,e,f])) \leq (rt'([u,v,s,t]) - lf'([u,v,s,t]))/2 \rightarrow$
$[c,d,e,f] \subset [u,v,s,t]))$

L11' : $\emptyset \vdash \forall D(clsdrec(D) \rightarrow D = [lf(D), rt(D), lf'(D), rt'(D)] \wedge lf(D) \in R \wedge lf'(D) \in R \wedge rt'(D) \in R \wedge rt(D) \in R)$
L17' : $\forall D(clsdrec(D) \rightarrow (rt(D) - lf(D)) \in R \wedge (rt'(D) - lf'(D)) \in R)$
Nit2 : $\emptyset \vdash \forall n(n \in \mathbb{N} \rightarrow clsdrec(g(n)) \wedge g(n+1) \subseteq g(n)) \rightarrow \exists z \forall i(i \in \mathbb{N} \rightarrow z \in g(i))$
CIA' : $P'([a,b,c,d]) \wedge \forall D(P'(D) \rightarrow \exists D'(P'(D') \wedge Q'(D,D'))) \rightarrow \exists g(g(0) = [a,b,c,d] \wedge \forall i(i \in \mathbb{N} \rightarrow P'(g(i)) \wedge Q'(g(i), g(i+1))))$
IA' : induction axiom for $((rt(g(i)) - lf(g(i))) = (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i \wedge (rt'(g(i)) - lf'(g(i))) = (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i)$
g_{5a} : $\Delta', \neg Th' \vdash (i \in \mathbb{N} \wedge (rt([a,b,c,d]) - lf([a,b,c,d]))/2^i \leq (rt([u,v,s,t]) - rt([u,v,s,t]))/2 \wedge (rt'([a,b,c,d]) - lf'([a,b,c,d]))/2^i \leq (rt'([u,v,s,t]) - lf'([u,v,s,t]))/2)$.

In order to find proofs for these open goals, one has to find an appropriate interpretation of the variables $rt, rt', lf, lf', center$, i.e., of the symbols not occurring in the target theorem or target assumptions, nor in CT. Assuming the open goals as proof assumptions of HB2, the proof plan actually yields a proof for the theorem HB2 when executed.

Straightforwardly, all of these subgoals but g_{5a} could be subject to proofs by analogy to proofs of the source lemmata again. Currently, our system leaves these goals open, however. That is, the target plan produced currently by our system may be incomplete. We feel that this is not a major omission because in mathematics analogy suggests target lemmata whose proofs – subject to generic or analogical theorem proving – may fail.

7 Evaluation and Future Work

As stated above, the first-order (unsorted) formalization of HB1 and of its proof are due to Bledsoe. One could argue about this formalizaton, e.g., about the explicit introduction of lemmata such as L17 (together with L11) because general facts about R could be inbuilt into a particular automated theorem prover or into the methods. However, we did not make any such prerequisites about the used theorem prover or methods. In fact, Bledsoe produced the proof of HB1 by manually providing the subgoals and assumptions and by calling OTTER on them. In general, this is not the way source proof plans have to be produced that are then transferred analogically by our system. In particular, the basic proof planner should have methods at its disposal that belong to a whole class of proof plans.

If we had more general methods to be used by the proof planner to plan several Heine-Borel theorems, the plans would be hierarchically structured. Presumably, then the top-level methods could be analogically transferred to target proof plans whereas some of the lower-level methods could not. Finding appropriate methods is subject to research in proof planning.

The low-level (ND)-representation in the proof schemas of some presented methods is unsatisfactory because those methods bear the danger of falling back to a level of abstraction that is too low. As the modifications of method-111' show, this can be unhealthy for reformulations as well. However, a more abstract method representation is a matter of the proof planner and its methods rather than of the analogy. As shown with the implementation of analogy-driven proof plan construction in CL^AM [Mel95a], reformulations and analogical replay work well with methods that do not rely on one particular proof schema.

To repeat, the presented version of the HB1 problem and its proof were taken from Bledsoe's challenge. The described analogical transfer of the HB1 plan has been accomplished automatically by OMEGA/Analogy. The argument that maybe the HB1 problem and proof plan was specifically tailored for the transfer to a known HB2 proof by our system does not hit the bull's eye because the Heine-Borel challenge problem and the described system grew **independently**.

However, there is no doubt that our analogy system does not succeed for arbitrary formalizations and proofs of HB1. For instance, with Bledsoe's first version of the HB1 plan, method-2 is not transferred analogically as discussed below. A better robustness of the analogy largely depends on the generality of the methods in the source proof plan. Note also that originally Bledsoe thought that the HB1 plan would be fully transferable. Only by running the system we discovered that method-2 would not be transferable and that the additional subgoal g_{5a} had to be proved.

One of the problems right now is the robustness of the analogy-driven proof plan construction with regard to different formulations of theorems. Possible improvements have to be investigated; so far, we have looked at three different formulations of the Heine-Borel theorems:

1. Suppose the two theorems HB1 and HB2 are formalized as:
   $\ldots \vdash clsdint(I) \rightarrow \exists H(H \subseteq G \wedge finite(H) \wedge I \subset \cup H)$ and:
   $\ldots \vdash clsdrec(I) \rightarrow \exists H(H \subseteq G \wedge finite(H) \wedge I \subset \cup H)$.
   Let a source lemma
   $\emptyset \vdash \forall I(clsdint(I) \rightarrow \exists x, y(I = [x, y] \wedge x \in R \wedge y \in R))$ and a target lemma
   $\emptyset \vdash \forall I(clsdrec(I) \rightarrow \exists x, y, z, w(I = [x, y, z, w] \wedge x \in R \wedge y \in R)) \wedge z \in R \wedge w \in R$ be given. Then the reformulation `Add-Argument` is triggered by the attempt to match the lemmata rather than by matching the source to the target theorem.

2. In the first version of a proof plan for HB1, Bledsoe had used the formula $(b-a)$ to represent the *length* of the closed interval $[a, b]$. In [Ble94] Woody Bledsoe suggested a formalization of the HB1 lemmata with $length[a, b]$ instead, in order to make the analogical transfer simpler. We ran, however, into trouble when we tried to use Bledsoe's formalization of lemma L9 with $length(D)$ because the target lemma:

   L9'$= \emptyset \vdash \forall x \forall u \forall v \forall s \forall t \forall c \forall d \forall e \forall f(u \in R \wedge v \in R \wedge s \in R \wedge t \in R \wedge c \in R \wedge d \in R \wedge e \in R \wedge f \in R \wedge u \leq v \wedge s \leq t \wedge c \leq d \wedge e \leq f \rightarrow (center(x, [u, v, s, t]) \wedge x \in [c, d, e, f] \wedge length([c, d, e, f]) \leq length([u, v, s, t])/2 \wedge length([c, d, e, f]) \leq length([u, v, s, t])/2 \rightarrow [c, d, e, f] \subset [u, v, s, t]))$

is suggested by the analogy procedure but had not been accepted because we did not know how to interpret *length* in order to satisfy L9' in R^2. The rejection of the suggested L9' prohibits to transfer the reformulated method-2' but rather produces it postcondition as an open subgoal. This is not particularly satisfying since a large part of the plan is not transferred at all.

Currently we are conducting more experiments with different formalizations of several Heine-Borel theorems in order to make the analogy application more robust. In particular, we allow for different sequences of proof assumptions in source and target.

Future directions for this work are being pursued along several fronts. The first targets more robustness of the analogical transfer and generating whole proofs that include the proofs of the suggested lemmata. This is, at least partly, a task for proof planning. A second front is the investigation of frequent reformulations,[17] where we have gained additional experience [Mel95a] in the proof planner *CLAM* that mainly works for proofs by induction. Meta-planning with meta-methods that represent reformulations will be investigated to automate the choice of *sequences* of reformulations. Another front of future research is to include more semantic information. One of Bledsoe's concerns was how to use semantic information more elaborately than just using a predefined `CT`, when proving a theorem by analogy. We shall encode semantic information into the control knowledge that is available to the planner. The control information that is actually used in the source planning process is stored as justifications and can be used in the analogy process. This appears to be a very promising way to employ semantic information in the analogy process.

Techniques for *automatically* patching the target methods are required such as abduction or "debugging" [Ble95a] which automatically provide additional preconditions in order to verify target methods. A more elaborated modification could use abduction. E.g., abduction performed on line5a of method2' which cannot be verified replaces L8 by L8'=

$$\forall v,w,x,y(0 \leq v \wedge 0 \leq x \wedge 0 \leq w \wedge 0 \leq y \rightarrow (f(v,w) \in \mathbb{N} \wedge x/2^{f(v,w)} \leq v \wedge y/2^{f(v,w)} \leq w))$$

as a new precondition for the proof of line5a, where f is a Skolem function [Ble95b]. L8' becomes an open subgoal which is eventually to be proved. Such a procedure may yield subgoals which are easier to prove and may avoid restructuring. However it can be misleading in cases where not just one additional precondition is necessary for accomplishing the target proof and where a whole method (e.g., yielding line5a of method-2' in our example) had to be replaced.

8 Conclusion

Analogy is a fertile technique for proving difficult theorems by reusing information from a source proof plan. The presented analogy method was developed in-

[17] In [KW95] a reformulation very similar to `Add-Argument` is used

dependently from the Heine-Borel example. It works at the proof plan level and includes reformulations. Solving the Heine-Borel analogy shows the substantial progress that has been made by the analogy-driven proof plan construction, since no previous approach was able to perform this analogical transfer.

Our approach advances ideas of previous systems for theorem proving by analogy and of derivational analogy. The power of the analogy procedure partially stems from the supplied source proof plan which guides the search for the target methods. Besides, the reformulation contributes to the strength of the analogy procedure. The remaining open goals have to be satisfied by base-level proof planning. Hence, in order for the analogy procedure to succeed, it has to be embedded into a proof planner.

9 Acknowledgment

My sincere gratitude goes to Woody Bledsoe for helping with this paper. He first suggested that we collaborate on a non-trivial analogy problem, in which a well known mathematical theorem is involved. He recommended the HB1, HB2 pair. He proposed a proof plan and lemma list of HB1 [Ble95a]. He checked the validity of the plans for HB1 and HB2 by "running" them on a prover which calls OTTER as a subroutine.

Personally, Woody encouraged me to continue my research on analogy-driven proof plan construction. I regret that Woody has not been able to continue our joint work on analogy. His courage, inspiration, and vision could have helped to push the horizon much further.

I would like to thank my student Wolf Schaarschmidt who implemented the analogy procedure in OMEGA. Last but not least, I thank Jörg Siekmann for his moral support and the suggestions that helped to improve this paper.

References

[BBH72] W. W. Bledsoe, Robert S. Boyer, and W. H. Henneman. Computer proofs of limit theorems. *Artificial Intelligence*, 3(1):27–60, 1972.

[Ble77] W. Bledsoe. Non-resolution theorem proving. *Artificial Intelligence*, 9:1–35, 1977.

[Ble86] W.W. Bledsoe. The use of analogy in automatic proof discovery. Tech.Rep. AI-158-86, Microelectronics and Computer Technology Corporation, Austin, TX, 1986.

[Ble90] W. Bledsoe. Challenge problems in elementary analysis. *Journal of Automated Reasoning*, 6:341–359, 1990.

[Ble94] W.W. Bledsoe. Heine-Borel theorem analogy example. Technical Report Memo ATP 124, University of Texas Computer Science Dept, Austin, TX, 1994.

[Ble95a] W.W. Bledsoe. A precondition prover for analogy. *BioSystems*, 34:225–247, 1995.

[Ble95b] W.W. Bledsoe. Abduction rules. Technical Report Memo ATP 126, University of Texas Computer Science Dept, Austin, TX, August 1995.

[BM88] R.S. Boyer and J.S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. *Machine Intelligence*, 11:83–124, 1988.

[Bun88] 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.

[BvHS91] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324, 1991.

[Car86] J.G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalsky, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. Morgan Kaufmann Publ., Los Altos, 1986.

[CLD+97] C.Benzmueller, L.Cheikhrouhou, D.Fehrer, A.Fiedler, X.Huang, M.Kerber, M.Kohlhase, K.Konrad, A.Meier, E. Melis, W.Schaarschmidt, J.Siekmann, and V.Sorge. OMEGA: Towards a mathematical assistant. In *Proc. 14th International Conference on Automated Deduction (CADE)*, pages –, Townsville, 1997. Springer.

[Cur95] R. Curien. *Outils pour la Preuve par Analogie*. PhD thesis, Universite Henri Poincare - Nancy, 1995.

[Deu71] P. Deussen. *Halbgruppen und Automaten*, volume 99 of *Heidelberger Taschenbücher*. Springer, 1971.

[GMW79] M. Gordon, R. Milner, and C.P. Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation*. Lecture Notes in Computer Science 78. Springer, Berlin, 1979.

[Had45] J. Hadamard. *The Psychology of Invention in the Mathematical Field*. Princeton Univ. Press, Princeton, 1945.

[HKKR94] X. Huang, M. Kerber, M. Kohlhase, and J. Richts. Methods - the basic units for planning and verifying proofs. In *Proceedings of Jahrestagung für Künstliche Intelligenz KI-94*, Saarbrücken, 1994. Springer.

[Kli71] R.E. Kling. A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178, 1971.

[KW94] Th. Kolbe and Ch. Walther. Reusing proofs. In *Proceedings of 11th European Conference on Artificial Intelligence (ECAI-94)*, Amsterdam, 1994. http://kirmes.inferenzsysteme.informatik.th-darmstadt.de/ kolbe/.

[KW95] Th. Kolbe and Ch. Walther. Patching proofs for reuse. In N. Lavrac and S. Wrobel, editors, *Proceedings of the 8th European Conference on Machine Learning 1995*, Kreta, 1995.

[McC90] W.W. McCune. Otter 2.0 users guide. Technical Report ANL-90/9, Argonne National Laboratory, Maths and CS Division, Argonne, Illinois, 1990.

[Mel93a] E. Melis. Analogies between proofs – a case study. SEKI-Report SR-93-12, University of Saarbrücken, Saarbrücken, 1993. available from http://jswww.cs.uni-sb.de/pub/seki/index.html.

[Mel93b] E. Melis. Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, University of Saarbrücken, Saarbrücken, 1993. available from http://jswww.cs.uni-sb.de/pub/seki/index.html.

[Mel95a] E. Melis. Analogy in CLAM. Technical Report DAI Research Paper No 766, University of Edinburgh, AI Dept, Dept. of Artificial Intelligence, Edinburgh, 1995. available from http://jswww.cs.uni-sb.de/˜melis/.

[Mel95b] E. Melis. A model of analogy-driven proof-plan construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 182–189, Montreal, 1995.

[Mun81] J.C. Munyer. *Analogy as a Means of Discovery in Problem Solving and Learning.* PhD thesis, University of California, Santa Cruz, 1981.

[MV94] E. Melis and M. Veloso. Analogy makes proofs feasible. In D. Aha, editor, *AAAI-workshop on Case-Based Reasoning*, pages 13–17, Seattle, 1994.

[Owe90] S. Owen. *Analogy for Automated Reasoning.* Academic Press, 1990.

[Pol45] G. Polya. *How to Solve it.* Princeton University Press, Princeton, 1945.

[Pol54] G. Polya. *Mathematics and Plausible Reasoning.* Princeton University Press, NJ, 1954.

[Seh95] A.C. Sehn. DECLAME – eine deklarative Sprache zur Repräsentation von Methoden. Master's thesis, Universität des Saarlandes, 1995. published as SEKI Working Paper SWP-95-02.

[vdW64] B.L. van der Waerden. Wie der Beweis der Vermutung von Baudet gefunden wurde. *Abh.Math.Sem.Univ.Hamburg*, 28, 1964.

[Vel94] M.M. Veloso. Flexible strategy learning: Analogical replay of problem solving episodes. In *Proc. of the twelfth National Conference on Artificial Intelligence 1994*, Seattle, WA, 1994.

[Wan91] R. Wang. The use of analogy in the PC system. CS Department Memo, Univerity of Texas at Austin, May 1991.

A HB1 Lemmata

The formalizations of the lemmata used in the proof of HB1 are:

L1 : $\emptyset \vdash \forall x(x \in R \to x = x/2^0)$

L2 : $\emptyset \vdash \forall x \forall y \forall d \forall n(d \in R \wedge n \in \mathbb{N} \to (x = d/2^n \wedge y = x/2 \to y = d/2^{n+1}))$

L3 : $\emptyset \vdash \forall x \forall y(x \in R \wedge y \in R \to clsdint([x,y]))$

L4 : $\emptyset \vdash \forall x \forall y(x \leq y \to lf([x,y]) = x \wedge rt([x,y]) = y)$

L5 : $\emptyset \vdash \forall D \forall G(clsdint(D) \wedge \neg\exists H(H \subseteq G \wedge finite(H) \wedge D \subset \cup H) \wedge lf(D) \leq rt(D) \to \exists D'(clsdint(D') \wedge \neg\exists H(H \subseteq G \wedge finite(H) \wedge D' \subset \cup H) \wedge lf(D') \leq rt(D') \wedge D' \subseteq D \wedge (rt(D') - lf(D')) = (rt(D) - lf(D))/2))$

L6 : $\emptyset \vdash \forall A \forall H \forall x(A \subset \cup H \wedge x \in A \to \exists B(B \in H \wedge x \in B))$

L7 : $\emptyset \vdash \forall x \forall B(x \in B \wedge open(B) \to \exists u \exists v(u \in R \wedge v \in R \wedge u < v \wedge [u,v] \subset B \wedge center(x,[u,v]))$

L8 : $\emptyset \vdash \forall d \forall b(d \in R \wedge b \in R \wedge 0 < d \wedge 0 \leq b \to \exists i(i \in \mathbb{N} \wedge b/2^i \leq d/2))$

L9 : $\emptyset \vdash \forall x \forall u \forall v \forall c \forall d(u \in R \wedge v \in R \wedge c \in R \wedge d \in R \to center(x,[u,v]) \wedge x \in [c,d] \wedge (rt([c,d]) - lf([c,d])) \leq (rt([u,v]) - lf([u,v]))/2 \to [c,d] \subset [u,v])$

L10 : $\emptyset \vdash \forall B finite(\{B\})$

L11 : $\emptyset \vdash \forall D(clsdint(D) \to D = [lf(D), rt(D)] \wedge lf(D) \in R \wedge rt(D) \in R)$

L12 : $\emptyset \vdash \forall B(B = \cup\{B\})$

L13 : $\emptyset \vdash \forall A \forall B \forall C(A \subset B \wedge B \subset C \to A \subset C)$

L14 : $\emptyset \vdash \forall B \forall G(B \in G \to \{B\} \subseteq G)$

L15 : $\emptyset \vdash \forall u \forall v(u \in R \wedge v \in R \wedge u < v \to 0 < (v - u))$

L16 : $\emptyset \vdash \forall u \forall v(u \in R \wedge v \in R \wedge u \leq v \to 0 \leq (v - u) \wedge 0 \leq (v - u)/2)$

L17 : $\forall D(clsdint(D) \to (rt(D) - lf(D)) \in R)$

CIA : $P([a,b]) \wedge \forall D(P(D) \to \exists D'(P(D') \wedge Q(D,D'))) \to \exists g(g(0) = [a,b] \wedge \forall i(i \in \mathbb{N} \to P(g(i)) \wedge Q(g(i), g(i+1))))$

IA : induction axiom for $((rt(g(i)) - lf(g(i))) = (rt([a,b]) - lf([a,b]))/2^i)$, i.e., $(rt(g(0)) - lf(g(0))) = (rt([a,b]) - lf([a,b]))/2^0 \wedge \forall k(k \in \mathbb{N} \wedge (rt(g(k)) - lf(g(k))) = (rt([a,b]) - lf([a,b]))/2^k \to (rt(g(k+1)) - lf(g(k+1))) = (rt([a,b]) - lf([a,b]))/2^{k+1})) \to \forall i(i \in \mathbb{N} \to ((rt(g(i)) - lf(g(i))) = (rt([a,b]) - lf([a,b]))/2^i)$

Another key lemma is the *Nested Interval Lemma*: If g is a sequence of nested closed intervals $I(n)$, then there is a point z in each of the $I(n)$'s.
Nit1: $\emptyset \vdash \forall n(n \in \mathbb{N} \to clsdint(g(n)) \wedge g(n+1) \subseteq g(n)) \to \exists z \forall i(i \in \mathbb{N} \to z \in g(i))$

B Details of `Add-Argument`

Terms of the form $f(term_1, \ldots, term_n)$ that occur in the source, where $term_i$ are terms, are called *f-terms*. The $term_i$ are called *reference terms*. `Add-Argument` modifies the source plan as follows:

1. Replace in all sequents and methods of the source plan the occurrences of f-terms $f(term_1, \ldots, term_n)$ by $f'(term_1, \ldots, term_n, term'_1, \ldots, term'_n)$ where $term'_i$ corresponding to reference term $term_i$ is obtained from $term_i$ by a syntactic replacement of all reference variables x_{ij} and constants c_{jk} by corresponding new variables y_{ij} and corresponding new constants d_{jk} of the same types respectively (captured in the constraints of the method) (See special case in 4.).
Add parameters d_{jk} to parameters c_{jk}. Modify the quantification accordingly.
Example: Let f =[-,-] be a function that maps two real numbers to an interval in R^1 and let f' =[-,-,-,-] be a function that maps four real numbers to a rectangle in R^2 Replace $[a, b]$ by $[a, b, c, d]$ in $\exists H(H \subseteq G \wedge finite(H) \wedge [a, b] \subset \cup H)$ resulting in $\exists H(H \subseteq G \wedge finite(H) \wedge [a, b, c, d] \subset \cup H)$ with new constants c, d.

2. After 1., replace in all those formulae which were affected by the previous replacement, the occurrences of all smallest subformulae (not considering quantification)
$\phi(x_1, \ldots, x_m, c_1, \ldots, c_p, x_{i1}, \ldots, x_{ik}, c_{j1}, \ldots, c_{jl})$, in which reference terms occur and which do not contain f' by
$(\phi(x_1, \ldots, x_m, c_1, \ldots, c_p, x_{i1}, \ldots, x_{ik}, c_{j1}, \ldots, c_{jl}) \wedge$
$\phi(x_1, \ldots, x_m, c_1, \ldots, c_p, y_{i1}, \ldots, y_{ik}, d_{j1}, \ldots, d_{jl}))$, where x_{ij} and c_{jk} are reference variables and constants and y_{ij} and d_{jk} are the corresponding new variables and constants.

   For instance, change
$\forall x \forall B(x \in B \wedge open(B) \rightarrow \exists u \exists v \exists s \exists t(u < v \wedge [u, v, s, t] \subset B \wedge center(x, [u, v, s, t]))$
to
$\forall x \forall B(x \in B \wedge open(B) \rightarrow \exists u \exists v \exists s \exists t(u < v \wedge s < t \wedge [u, v, s, t] \subset B \wedge center(x, [u, v, s, t]))$.

3. If some of the reference constants c occur in a line l_c of the method, but no f'-term is in this line, a copy l_d of l_c has to be inserted into the proof schema. This copy l_d is obtained from l_c by renaming the reference constants by the corresponding new constants (introduced in 1.).

   Then insert a line l_{cd} that contains the conjunctions of the formulae in l_c and l_d. Also, replace in all method lines the reference label l_c in justifications by l_{cd}.

   Example: A method with lines
$(\ldots \vdash symm(inv(a)) \ldots)$ and
$(l_c \ldots \vdash symm(a)$ (OPEN)) for the unary function inv is changed to a

method with lines
$(\ldots \vdash symm(a \cap b)) \ldots)$
$(l_c \ldots \vdash symm(a)$ (OPEN)),
insert l_d and l_{cd}
$(l_d \ldots \vdash symm(b)$ (OPEN)), for the constant b introduce before, and
$(l_{cd} \ldots \vdash symm(a) \wedge symm(b))(\wedge I l_c, l_d))$ for the binary function $\cap$.

4. If for destructor functions $f_1, f_2 ... f_n$ the formula
$\forall x(f(f_1(x), \ldots, f_n(x)) = x)$ or
$\forall x_1 \ldots x_n(f_1(f(x_1, \ldots, x_n)) = x_1 \wedge f_n(f(x_1, \ldots, x_n)) = x_n)$ is a proof assumption or axiom, then $f_i(t)$ are (additional) reference terms and $f_i'(t)$ are the corresponding terms, where the f_i' are new function symbols with the same type as f_i (new constraints). If f_i is parameter, then add parameter f_i'. Thus,

   (a) replace $f(f_1(t), \ldots, f_n(t))$ by $f'(f_1(t), \ldots, f_n(t), f_1'(t), .., f_n'(t))$, and then

   (b) in all formulae in which some $f_i(t_0), \ldots, f_k(t_0)$ occur replace those smallest subformulae $\phi(f_i(t_0), \ldots, f_k(t_0), x_{i1} \ldots, x_{il}, c_{j1}, \ldots, c_{jm})$, in which $f_i(t_0), \ldots, f_k(t_0)$ occur but not as arguments of f', by $\phi \wedge \phi'$ where we obtain the second conjunct by replacing in ϕ the reference terms by their corresponding terms at all occurrences that are *not* arguments of f' (i.e., $f_i(t_0)$ is treated as a non-decomposable term). (That is, for those hidden occurrences of f in $\phi(f_i(t), \ldots, f_k(t))$ treat the formula $\phi(f_i(t), \ldots, f_k(t))$ in the same way as `Add-Argument` treats the formula $f(f_1(t), \ldots, f_n(t)) = t \wedge \phi(f_i(t), \ldots, f_k(t))$ and treat the terms $f_i(t)$ in the same way as reference constants by introducing corresponding new terms $f_i'(t)$.)
   Example: Let $\forall x([lf(x), rt(x)] = x)$ be a proof assumption, then $lf(D) \leq rt(D)$ is replaced by $lf(D) \leq rt(D) \wedge lf'(D) \leq rt'(D)$ for new function symbols lf', rt' and
   $\forall x \forall y \forall z \forall w(lf([x, y, z, w]) = x \wedge rt([x, y, z, w]) = y)$ (z corresponding to x and w corresponding to y) is replaced by $\forall x \forall y \forall z \forall w(lf([x, y, z, w]) = x \wedge lf'([x, y, z, w]) = z \wedge rt([x, y, z, w]) = y \wedge rt'([x, y, z, w]) = w)$