Representing and Reformulating Diagonalization Methods

Erica Melis

July 1994
CMU-CS-94-174

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

The author is on leave from University of Saarbrücken

Abstract

Finding an appropriate representation of planning operators is crucial for theorem provers that work with proof planning. We show a new representation of operators and demonstrate how diagonalization can be represented by operators. We explain how a diagonalization operator used in one proof-plan can be analogically transferred to an operator used in another proof-plan. Finally, we find an operator that is common to all the proof-plans and thus might be considered as the *Diagonal Method*.

This research was supported by the Max-Kade Foundation

Keywords: proof planning, analogy, knowledge representation

1 Introduction

As pointed out by Bundy [3] and Bledsoe [1], using proof-plans is often very helpful in automated deduction. In planning, operators are needed and therefore an appropriate representation of these operators is crucial for proof planning. The operators have the same function in proof planning as mathematical methods (in the following referred to as m-methods) have in human theorem proving. Since m-methods can be adapted to different proofs, it is also desirable to have mechanisms for adapting operators. To be employed by a human-oriented theorem prover, these operators should allow for representing logical proof methods, such as `Indirect Proof`, and mathematical methods, such as Cantor's `Diagonal` method.

In this paper we examine whether the presented representation actually covers mathematician's methods and how the methods can be adapted for other proof plans. We do this by analyzing the well-known Diagonal Method which is central and widely applicable in many mathematical proofs concerning computability and decidability, including Gödel's Incompleteness theorem for arithmetic, the Unsolvability of the halting problem, Rice's theorem (see [5]), and the Second Recursion theorem (see [5]). Although this m-method seems to be clearly understood, not all proofs have an obvious common proof schema, and some proofs are difficult to generate in logical detail.

After defining our representation of operators by *methods*, we investigate several proofs in which mathematicians have used diagonalization. We sketch these proofs and show which methods belong to the respective proof-plans. We also discuss, how a method from the proof-plan for Cantor's theorem can be transferred to a method for a proof-plan of the halting problem and of Gödel's First incompleteness Theorem. Finally, a comparison of the methods yields a new method that is common to all the proofs and which might be considered as *the Diagonal Method.*

2 Representation of Methods

First we give a brief definition of methods that allow for reformulation (For more details see [6, 8]).

Sequents P, written as $(ass \vdash concl)$, are pairs of a set ass of formulas and a formula $concl$ in an object language that is extended by meta-variables for formulas, sets of formulas, and terms[1]. As an abbreviation, we shall display formulas F instead of sequents $(\emptyset \vdash F)$.

Methods M are frame-like structures similar to Bundy's methods in [3]. Methods have the slots parameter, preconditions (pre(M)), postcondition (post(M)), constraints, proof schema and procedure. Preconditions is a set of sequents, postcondition is a sequent; both pre- and postconditions are needed in planning. Constraints are formulated in a meta-language and serve to restrict the search during planning and may, e.g., express restrictions of pre(M) and post(M). The proof schema is a declarative schematic representation of incomplete proofs[2]

[1] $(ass \vdash concl)$ expresses "$concl$ is inferred from ass".

[2] Incomplete in the sense that it may have nonaxiomatic preconditions and may be incorrect.

in the object logic, relying on the Natural Deduction (ND) calculus. The lines of the proof schema may contain method names as the justification for an inference, hence methods are recursively defined, with basic methods corresponding to basic ND-inference rules. These inference rules are implemented as schemas with meta-variables and justify any inference that can be obtained by instantiating meta-variables in the schema. Like for a Hoare triple, after applying the proof schema to pre(M), post(M) should result. The program in the slot procedure executes the application of the proof schema by interpreting the proof schema. The structural template for methods is:

method: `name of the method`	
parameter	parameters which can be instantiated
preconditions	preconditions that have to be true for the method to be applicable
postcondition	postconditions that should be fulfilled after the method application, e.g., a derived sequent
constraints	meta-language constraints that may restrict pre- and post-conditions, parameter
proof schema	a declarative proof schema
procedure	procedure that interprets the proof schema

The proof schema of a method M may contain so-called LEMMA-lines in which an element of pre(M) occurs and that have LEMMA as their justification. The proof schema of a method may also have PLAN-lines, that contain a method-variable PLAN_i as the justification, which means that the method to be applied is not specified. A method with a PLAN-line is considered equivalent to the method for which the PLAN-line is replaced by a LEMMA-line and that contains the sequent of the PLAN-line as a precondition.

A method M is *verifiable* if it can recursively be checked that for every instantiation of the meta-variables the method is correct, i.e., it yields a correct proof of post(M) when applied to pre(M) in case the constraints are satisfied.

The methods defined here differ from those in [3] mainly in that the tactic slot is replaced by a declarative proof schema *and* a procedure interpreting this schema. The intention behind this difference is to enable reformulations of methods.

3 The Diagonalization Methods

We want to check whether the definition of methods, as described above, is an appropriate representation for m-methods. To that end we investigate some proofs in which mathematicians have used diagonalization. Usually diagonalizations proceed by
supposing $a_1, a_2 \ldots$ is an enumeration of objects of a certain kind. Then an object a of the same kind is constructed, that is different from every a_n using the following principle: "Make a and a_n differ at n." The interpretation of *differ at n* depends on the kind of object involved.

In the following we examine proofs of

1. **Cantor's Theorem**, which states that for any set M the cardinality of M is smaller than the cardinality of the powerset $\mathcal{P}M$ of M,
   $cardM < card\mathcal{P}M$ (see [2])

2. **Uncountability of the set of real numbers** (actually the interval [0 1]), which means that $card\ \mathbb{N} < card\ [0,1]$ (see [4])

3. **Unsolvability of the Halting Problem** for Turing machines, which means there is no algorithm (no t-computable function) to determine whether an arbitrary Turing machine in an arbitrary configuration with a finite length of nonblank tape symbols will eventually halt (see [7]). That is, no t-computable function c[3] exists with

$$c(t, \text{conf}_y) = \begin{cases} 1 & : \quad t \text{ does not halt with } \text{conf}_y \\ 0 & : \quad t \text{ halts with } \text{conf}_y \end{cases}$$

4. **Gödel's Incompleteness theorem for S**, which states that there is a sentence in the language of arithmetic S with addition and multiplication that is not provable in S neither is its negation(see [10]).

3.1 The Proof Sketches

Throughout the paper $(M \mapsto N)$ denotes the set of functions from M to N.

3.1.1 $cardM < card\mathcal{P}M$

1. **Unfold definition**
   The theorem $cardM < card\mathcal{P}M$ is rerepresented by applying the definition of the partial order of cardinals to the theorem:
   There is a one-to-one correspondence from all elements of M to a subset of $\mathcal{P}M$, but no one-to-one correspondence from all elements of $\mathcal{P}M$ to a subset of M. Since the proof of the first part is trivial, the task can be reduced to prove
   $\neg\exists f \forall x \exists y (f \in (M \mapsto \mathcal{P}M) \wedge (x \in \mathcal{P}M \rightarrow y \in M \wedge f(y) = x))$.

2. **Indirect proof**
   Derives $\neg\exists f \forall x \exists y (f \in (M \mapsto \mathcal{P}M) \wedge (x \in \mathcal{P}M \rightarrow y \in M \wedge f(y) = x))$ via a contradiction. This means, the indirect assumption[4] is
   $\forall x \exists y (x \in \mathcal{P}M \rightarrow y \in M \wedge F(y) = x)$, for a function $F \in (M \mapsto \mathcal{P}M)$.

3. **Method D1** yields the contradiction. In more detail, D1 includes the derivations of

   (a) the existence of a function $G(x)$ with $G(x) = nonF(xx)$ with

$$non(x) = \begin{cases} 0 & : \quad x \neq 0 \\ 1 & : \quad x = 0 \end{cases}$$

[3] c would be the characteristic function of the halting problem.

[4] We call the negations of the theorem to be proved by an indirect proof the *indirect assumption*.

By using the precondition (0): $\forall g(g \in \mathcal{P}M \rightarrow \forall x(x \in M \rightarrow g(x) \in \{0,1\}))$, which is part of the representation lemma for $\mathcal{P}M$, the indirect assumption is expressed by
$\neg\exists f\forall x\exists y(f \in (\mathrm{M} \mapsto \mathrm{M} \mapsto \{0,1\}) \wedge (x \in \mathcal{P}M \rightarrow y \in M \wedge f(y) = x))$. Thus the application of a comprehension axiom becomes possible.

Preconditions for this step are (0), the indirect assumption and the comprehension axioms[5] for *non* and compound functions.

(b) $\forall x(x \in M \rightarrow (F(x)(x) = 0 \rightarrow G(x) = 1) \wedge (F(x)(x) \neq 0 \rightarrow G(x) = 0))$.
Preconditions are (a) and the definitions of *non* and G.

(c) $G(x) \in \mathcal{P}M$. **Preconditions** for this step are
(2): $\forall g\forall x(x \in M \rightarrow g(x) \in \{0,1\}) \rightarrow g \in \mathcal{P}M)$ and $1, 0 \in \{0,1\}$.

(d) $F(x_0)(x_0) = G(x_0)$. **Preconditions** are the indirect assumption and $G \in \mathcal{P}M$.

(e) $\perp$ is proved in two steps from $F(x_0x_0) \vee \neg F(x_0x_0)$:
derive $\perp$ from $F(x_0x_0) = 0$ and
derive $\perp$ from $F(x_0x_0) \neq 0$. **Preconditions** are (c), (b), the definition of F, $1 \neq 0, \forall x(x = x)$.

The precondition $\forall g(g \in \mathcal{P}M \leftrightarrow (\forall x(x \in M \rightarrow g(x) \in \{0,1\})))$, i.e. (0) and (2), is a representation lemma that states that each subset of M can be represented as a function from $(M \mapsto \{0,1\})$. Figure 1 shows the proof structure of the proof of Cantor's theorem as well as of the proof of the uncountability of $\mathbb{R}$.

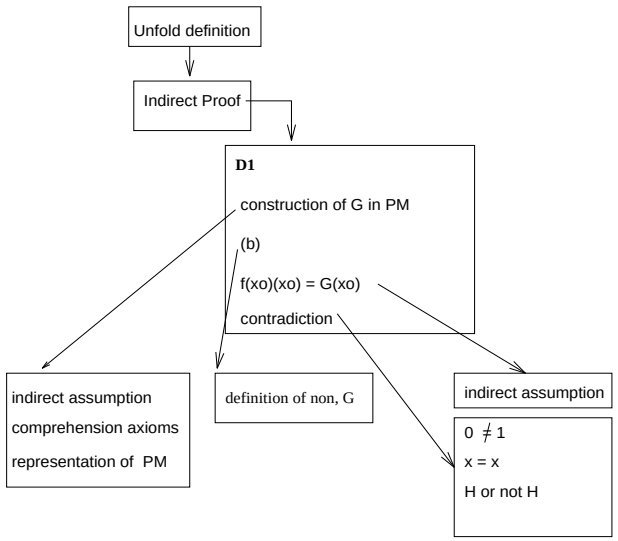

Figure 1: Proof Structure of Cantor's Theorem

[5]This idea is due to X.Huang and M.Kerber who also gave an ND-proof

3.1.2 $card\,\mathbb{N} < card[0,1]$

1. **Unfold definition**
   The theorem $card\,\mathbb{N} < card[0,1]$ is rerepresented by applying the definition of the partial order of cardinals to the theorem:
   There is a one-to-one correspondence from all elements of $\mathbb{N}$ to a subset of $[0,1]$, but no one-to-one correspondence from all elements of $[0,1]$ to a subset of $\mathbb{N}$. Since the proof of the first part is trivial, the task can be reduced to prove
   $\neg\exists f\forall x\exists y(f \in (\mathbb{N} \mapsto [0,1]) \wedge (x \in [0,1] \rightarrow y \in \mathbb{N} \wedge f(y) = x))$.

2. **Indirect proof**
   Derives $\neg\exists f\forall x\exists y(f \in (\mathbb{N} \mapsto [0,1]) \wedge (x \in [0,1] \rightarrow y \in \mathbb{N} \wedge f(y) = x))$ via a contradiction. The indirect assumption is $\forall x\exists y(x \in [0,1] \rightarrow y \in \mathbb{N} \wedge F(y) = x)$ for a function constant $F \in (\mathbb{N} \mapsto [0,1])$.

3. **Method D2** yields a contradiction. In more detail, D2 includes the derivations of

   (a) the existence of a function $G(x)$ with: $G(x) = nonF(xx)$ for

   $$non(x) = \begin{cases} 0 & : \quad x \neq 0 \\ 1 & : \quad x = 0 \end{cases}$$

   By using the precondition (0): $\forall g(g \in [0,1] \rightarrow \forall x(x \in \mathbb{N} \rightarrow g(x) \in \{0\ldots9\}))$, which is part of the representation lemma for $[0,1]$, the indirect assumption can be written as
   $\neg\exists f\forall x\exists y(f \in (\mathbb{N} \mapsto \mathbb{N} \mapsto \{0\ldots9\}) \wedge (x \in [0,1] \rightarrow y \in \mathbb{N} \wedge f(y) = x))$. Thus the application of a comprehension axiom becomes possible. **Preconditions** for this step are (0), the indirect assumption and the comprehension axioms for *non* and compound functions.

   (b) $\exists g\forall x(x \in \mathbb{N} \rightarrow (F(x)(x) = 0 \rightarrow G(x) = 1) \wedge (F(x)(x) \neq 0 \rightarrow G(x) = 0))$.
   Preconditions for this step are (a) and the definitions of *non* and G.

   (c) $G(x) \in [0,1]$. **Preconditions** for this step are
   (2): $\forall g\forall x(x \in \mathbb{N} \rightarrow g(x) \in \{0\ldots9\}) \rightarrow g \in [0,1])$ and $1, 0 \in \{0\ldots9\}$.

   (d) $F(x_0)(x_0) = G(x_0)$. **Preconditions** are the indirect assumption, and $G \in [0,1]$.

   (e) $\perp$ is proved in two steps from $F(x_0x_0) \vee F(x_0x_0)$:
   derive $\perp$ from $Fx_0x_0 = 0$ and
   derive $\perp$ from $Fx_0x_0 \neq 0$. **Preconditions** are (c), (b), the definition of F, $1 \neq 0, \forall x(x = x)$.

Figure 1 also shows the proof structure of the IR-proof for $\mathcal{P}M$ is replaced by [0,1]. The precondition $\forall g(g \in [0,1] \leftrightarrow (\forall x(x \in \mathbb{N} \rightarrow g(x) \in \{0,1\})))$ is given by a representation lemma that states that each real number in [0,1] can be represented as a function from $(\mathbb{N} \mapsto \{0\ldots9\})$.

3.1.3 Unsolvability of the Halting Problem

We follow the proof of the halting problem in [2]. Prior to conducting the proof, a Gödel enumeration of the configurations is assumed and the fact that Turing machines are t-computable functions. Consequently "t halts on $config_y$" is equivalent (modulo the presumed theory) to "$t(y)$ is defined" and "t does not halt on $config_y$" is equivalent to "$t(y)$ is undefined" for $y \in \mathbb{N}$, and thus the halting problem can equivalently rerepresented as
There is no t-computable function c such that for $y \in \mathbb{N}$

$$c(t(y)) = \begin{cases} 0 & : \quad t(y) \text{ defined} \\ 1 & : \quad t(y) \text{ undefined} \end{cases}$$

That is, the theorem is then $\exists c \neg (c \in T \wedge \forall t, n (t \in T \wedge n \in \mathbb{N} \rightarrow (c(t(n)) \in \{0,1\})))$, where $c \in T$ means c is t-computable. Assuming a Gödel enumeration F of Turing machines, each Turing machine t_m is an $F(m)$ for $m \in \mathbb{N}$ and $F(mn) = t_m(n)$.

1. **Indirect proof**
   derives the theorem via contradiction, starting with the indirect assumption
   $c \in T \wedge \forall t, n (t \in T \wedge n \in \mathbb{N} \rightarrow (c(t(n)) \in \{0,1\}))$.

2. **Method D3** infers a contradiction from the indirect assumptions $(c \in T)$ and (0): $\forall t, n (t \in T \wedge n \in \mathbb{N} \rightarrow (c(t(n)) \in \{0,1\}))$. In more detail, D3 includes the derivations of

   (a) the existence of $G(x) = non\ F(xx)$, where

   $$non(x) = \begin{cases} 0 & : \quad x \text{ undefined} \\ \text{undefined} & : \quad x \text{ defined} \end{cases}$$

   Preconditions are the definitions of non and F.

   (b) $(G \in T)$. It is shown that since c is t-computable, non is t-computable, and so is the composition of non, F.

   (c) $\forall x (x \in \mathbb{N} \rightarrow (c(Fxx) = 0 \rightarrow c(Gx) = 1) \wedge (c(Fxx) \neq 0 \rightarrow c(Gx) = 0))$.
   Preconditions are (a) and the lemma (8): $\forall x ((cx = 0 \rightarrow c\, non(x) = 1) \wedge (cx \neq 0 \rightarrow c\, non(x) = 0))$ that can be derived from the definitions of c and non.

   (d) $F(x_0)(x_0) = G(x_0)$ for a constant x_0. **Preconditions** are $G \in T$ and the enumerability of Turing machines.

   (e) $\perp$ is proved in two steps:
   derive $\perp$ from $cFx_0x_0 = 0$ and
   derive $\perp$ from $cFx_0x_0 \neq 0$. **Preconditions** are (d), (c), $0 \neq 1, \forall x (x = x)$, $(H \vee \neg H)$[6], and the definition of c.

Figure 2 shows a proof structure of the proof of the halting problem.

[6] Proof sketch: Since $cF(x_0x_0) = cG(x_0)$. Case 1: $cF(x_0x_0) = 0$, then $cG(x_0) = 0$ then $cF(x_0x_0) = 1$ then $0 = 1$. Case 2: $cF(x_0x_0) \neq 0$ then $cG(x_0) = 0$ and $cF(x_0x_0) = 1$ then $cF(x_0x_0) = 0$ then $0 = 1$.

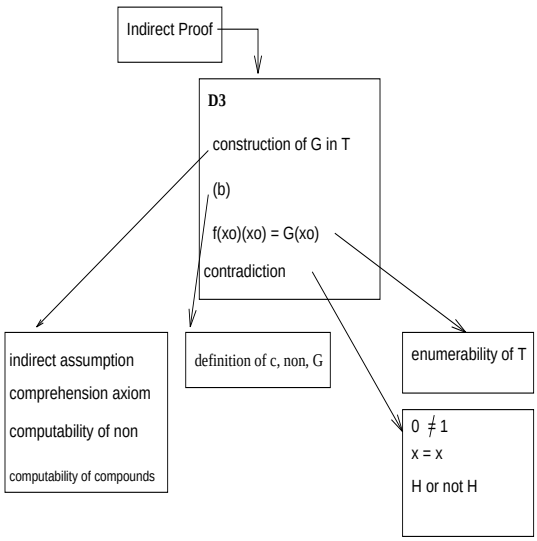

Figure 2: Proof Structure of the Halting Problem

3.1.4 Gödel's Theorem

The following mathematical proof is taken from [11].
A Gödel numbering f of the expressions in S with one variable is assumed.
Lemma: The predicate R(xy) which states that the proof with Gödel number y proves the sentence which is the instantiation $\phi_{\mathrm{x}}(\mathrm{x})$ of an one-variable-expression ϕ_{x} with Gödel number x by the number x is numeralwise expressible by $R(xy)$ in the arithmetic S.

Consider the formula $\forall y \neg R(xy)$. It has a Gödel number p, and thus $\forall y \neg R(xy) = \phi_p(x)$. Now consider the formula $\phi_p(p)$, i.e., $\forall y \neg R(py)$ which contains no variable free.

Theorem: If the number-theoretic formal system S is consistent, then not $\vdash_S \phi_p(p)$; and if the system is ω-consistent, then not $\vdash_S \neg\phi_p(p)$.

To reveal the similarity between the proof of Gödel's theorem and the previous proofs, we introduce a function w which can be defined for sentences H by $w(\mathrm{H})$ = provable_in_S (H), i.e., w is a function from object language sentences to meta-formulas. By the lemma above, we have $w(\phi_{\mathrm{x}}(\mathrm{x})) = w(f(xx)) = \exists y R(xy)$. Thus a logical reconstruction of the proof is:

1. **Indirect proof**
   Derives $\neg\forall g, x(g \in E \rightarrow (x \in \mathbb{N} \rightarrow ((wg(x)) \vee (w\neg g(x)))))$ via contradiction. A weakened[7] indirect assumption is $\forall g, x(g \in E \rightarrow x \in \mathbb{N} \rightarrow ((wg(x)) \vee (w\neg g(x))))$, where E is the set of all expressions in S with one free variable.

2. **Method D-Gödel** yields the contradiction. In more detail, D-Gödel includes the derivations of

[7] Weakened because it is only for sentences $g(x)$ with $g \in E$.

(a) the existence of G with $G(x) = non\, w(f(xx))$, which is substantiated by the definitions of formulas and of w, f and by the definition $non(F) = \neg F$.

(b) $G \in E$. **Preconditions** are the definition of E and the lemma (that provides $G(x) = \neg\exists y R(xy)$).

(c) $f(x_0)(x_0) = G(x_0)$ for a constant x_0. **Preconditions** are (b) and the enumeration of the expressions of S by f.

(d) $\bot$. This is shown by deriving a contradiction from $wGx_0 \vee w\neg G(x_0)$[8]. **Preconditions** are (8): $\forall x((w(x) \rightarrow w(non(non(x)))) \wedge (\neg w(x) \rightarrow w(non(x))))$, the second conjunct of which is the indirect assumption, and the definitions of w, R, G, the consistency of S, and the ω-consistency of S.

In order to obtain a version of the method which is better comparable to D1, D2, and D3, we additionally introduce a characteristic function c for sentences H of S by

$$c(H) = \begin{cases} 0 & : \vdash_S H \\ 1 & : \vdash_S \neg H \end{cases}$$

Then (8): $\forall x((c(x) = 0 \rightarrow c(non(x)) = 1) \wedge (c(x) \neq 0 \rightarrow c(non(x)) = 0))$ can be derived. Furthermore the proof of the contradiction can be replaced by a subproof starting with assuming $cw(f(x_0, x_0)) = 0 \vee cw(f(x_0, x_0)) \neq 0$ as an instantiation of $(H \vee \neg H)$. Summarize this new version **D4** consists of the subproofs:

(a) Show the existence of $G(x)$ with $G(x) = non\, wf(xx)$. **Preconditions** are the definition of formulas, the definitions of non, f, and (0): $\forall g(g \in E \rightarrow (x \in \mathbb{N} \rightarrow wg(x) \in set_of_meta_formulas))$.

(b) $G \in E$ which follows from (a) and the lemma.

(c) $f(x_0)(x_0) = G(x_0)$ for a constant x_0. **Preconditions** are (b) and the enumeration of E by f.

(d) $\forall x((cwf(xx) = 0 \rightarrow cG(x) = 1) \wedge (cwf(xx) \neq 0 \rightarrow cG(x) = 0))$ which can be proved using (a) and (8).

(e) $\bot$. As indicated above, $\bot$ is proved starting with $cw(f(x_0, x_0)) = 0 \vee cw(f(x_0, x_0)) \neq 0$. Relying on (d), $cw(f(x_0, x_0)) = 0$ leads to a contradiction, using the ω-consistency of S[9], (c), and $0 \neq 1$ (which encodes the consistency of S), and $cw(f(x_0, x_0)) \neq 0$ leads to a contradiction as well, using (c), the definition of R, and $x = x$.

---

[8]Proof sketch: Assume $wG(x_0)$, then $\vdash G(x_0)$, then exists a proof p_k of $G(x_0)$, then $\vdash R(x_0k)$, then $\vdash \neg G(x_0)$ because of (8), then $w\neg G(x_0)$, then $\bot$ because of consistency of S. Hence, $\neg wG(x_0)$, then not exists a proof of $G(x_0)$, then $R(x_00), R(x_01) \ldots$ are false because of (8), then $\vdash \neg R(x_00) \ldots$ because of the definition of R, then $\nvdash \neg\forall y\neg R(x_0y)$ because of ω-consistency of S, then $\nvdash \neg Gx_0$ because of the definition of G, then $\neg w\neg G(x_0)$ which yields the contradiction with $w\neg G(x_0)$.

[9]The ω-consistency states that If $\neg R(0y), \neg R(1y), \ldots$ then $\nvdash \neg\forall x\neg R(xy)$, which allows to deduce $cG(x_0) = 1 \rightarrow cwG(x_0) = 1$.

This proof version provides a larger commonality of the partial proof schemas for D1, D2, D3, and D4. Figure 3 shows the proof structure of a proof of Gödel's theorem.

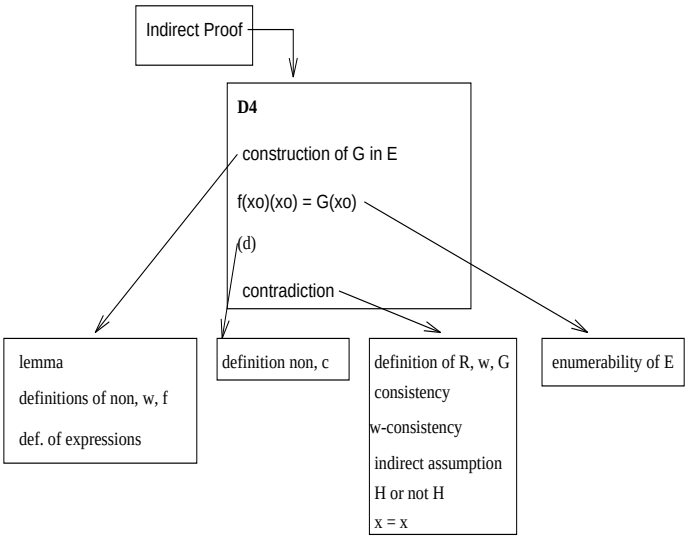

Figure 3: Proof Structure of Gödel's Theorem

3.2 The Methods

One of the methods in all four proof plans is `Indirect Proof`:

method: `Indirect Proof`		
parameter	F: formula, Δ: set of formulae	
preconditions		
postcondition	$\Delta \vdash F$	
constraints		
proof schema	1. $;\neg F \quad \vdash \neg F$	(HYP)
	2. $\Delta;\neg F \quad \vdash \bot$	(PLAN)
	3. $\Delta; \quad \vdash \neg\neg F$	($\neg$ I;2)
	4. $\Delta; \quad \vdash F$	($\neg\neg D$;3)
procedure	schema-interpreter	

A method that jointly represents D1 and D2 is D12, where (1) is the indirect assumption referred to in D1 and D2, equ denotes the application of an equality axiom, and Method$_i$ denotes submethods.

method: D12	
parameter	F: function, $M1, M2$: structures, U: set, H: formula
preconditions	(0)$\forall g \forall x(g \in M2 \rightarrow (x \in M1 \rightarrow g(x) \in U))$ (1) $\forall x \exists y(x \in M2 \rightarrow y \in M1 \wedge F(y) = x)$, (6) $1 \neq 0$, (2) $\forall g \forall x((x \in M1 \rightarrow g(x) \in U) \rightarrow g \in M2)$,(5)$\forall x(x = x)$, (3) $U \subseteq domain(non)$, (4) $0 \in U \wedge 1 \in U$, (7) $(H \vee \neg H)$, (8) $\forall x((x \neq 0 \rightarrow non(x) = 0) \wedge (x = 0 \rightarrow non(x) = 1))$
postcondition	$\perp$
constraints	
proof schema	1. ; $\vdash$ $\exists g \forall x(x \in M1 \rightarrow g(x) = nonF(xx))$ (comprehension F, non,(0)) 2. ; $\vdash$ $\forall x(x \in M1 \rightarrow G(x) = nonF(xx))$ ($\exists$D 1) 3. ; $\vdash$ $\forall x(x \in M1 \rightarrow (F(xx) = 0 \rightarrow G(x) = 1) \wedge (F(xx) \neq 0 \rightarrow G(x) = 0))$ (Method$_1$,(8) 2) 4. ; 4 $\vdash$ $a \in M1$ (HYP) 5. ; $\vdash$ $F(aa) = 0 \vee F(aa) \neq 0$ ((7)) 6. ; 6 $\vdash$ $F(aa) = 0$ (HYP) 7. ; 4, 6 $\vdash$ $G(a) = 1$ ($\forall D, \wedge D, \rightarrow$ D 4 6 3) 8. ; 4, 6 $\vdash$ $G(a) \in U$ (equ,(3)) 9. ; 9 $\vdash$ $F(aa) \neq 0$ (HYP) 10. ; 4, 9 $\vdash$ $G(a) = 0$ ($\forall D, \wedge D, \rightarrow$ D 4 9 3) 11. ; 4, 9 $\vdash$ $G(a) \in U$ (equ (4)) 12. ; 4 $\vdash$ $G(a) \in U$ ($\vee$D 5 8 11) 13. ; $\vdash$ $a \in M1 \rightarrow G(a) \in U$ ($\rightarrow$D 4 12) 14. ; $\vdash$ $G \in M2$ ($\forall$D,$\leftrightarrow$D,(2) 13) 15. ; $\vdash$ $\exists x(x \in M1 \wedge F(x) = G)$ ($\forall$D(1) 14) 16. ; $\vdash$ $F(x_0) = G$ ($\exists$D 15) 17. ; $\vdash$ $F(x_0, x_0) = G(x_0)$ (equ 16) 18. ; $\vdash$ $(x_0 \in M1 \rightarrow (F(x_0x_0) = 0 \rightarrow G(x_0) = 1) \wedge (F(x_0x_0) \neq 0 \rightarrow G(x_0) = 0))$ ($\forall$D 3) 19. ; $\vdash$ $F(x_0x_0) = 0 \vee F(x_0x_0) \neq 0$ (LEMMA) 20. ; 20 $\vdash$ $F(x_0x_0) = 0$ (HYP) 21. ; 20 $\vdash$ $G(x_0) = 0$ (eq 20 17) 22. ; 20 $\vdash$ $G(x_0) = 1$ ($\wedge D, \rightarrow$ D 18 20) 23. ; 20 $\vdash$ $0 = 1$ ($\wedge$I,equ,(5) 22 21) 24. ; 20 $\vdash$ $\perp$ ($\wedge$I,$\perp$I,(6)) 25. ; 25 $\vdash$ $F(x_0x_0) \neq 0$ (HYP) 26. ; 25 $\vdash$ $G(x_0) = 0$ ($\wedge$D,$\rightarrow$D 25 18) 27. ; 25 $\vdash$ $F(x_0x_0) = 0$ (equ 26 17) 28. ; 25 $\vdash$ $F(x_0x_0) = 0 \wedge \neg F(x_0x_0) = 0$ ($\wedge$I 27 25) 29. ; 25 $\vdash$ $\perp$ 10 ($\perp$I 28) 30. ; $\vdash$ $\perp$ ($\vee$D 29 24 19)
procedure	schema-interpreter

Some Definitions
In the following, goals and assumptions are intended to be sequents. Since in general proofs are constructed top down and bottom up, we construct proof-plans with forward-methods used in forward search and backward-methods employed in backward search. For instance, the method $\wedge$-deletion is typically employed in forward search, whereas the method $\wedge$-introduction is typically employed in backward search[10].

A proof-plan is a forest. Its trees consist of sequent nodes and verifiable method nodes, where the successor of a sequent node $\mathtt{g}$ is a method node M and the successors of a method node are sequent nodes $\mathtt{g}_1, \ldots, \mathtt{g}_n = \overline{\mathtt{g}}$, such that the following "link condition" is satisfied: $\sigma(\text{post(M)}) = \mathtt{g}$ and $\sigma(\text{pre(M)}) = \overline{\mathtt{g}}$ for a substitution σ (using the obvious extension of σ to sequents and sequences of sequents).

A proof-plan may contain forward-trees the sequents of which are assumptions and which are constructed by forward search. It may contain a backward-tree with a goal root node and goals as sequent nodes which is constructed by backward search or a tree which combines the backward-tree with forward-trees.

The planning starts with a root goal and assumption leaves $(\emptyset \vdash \top)$ and $(\emptyset \vdash F_i)$, where F_i is a proof-assumption, or $F_i \in KB$, where KB is the knowledge base of axioms, definitions, and lemmas. The planning proceeds by inserting methods and sequents satisfying the link condition, always aiming at closing 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 $\mathtt{g}_i$ equals an assumption, the two nodes collapse and thus the backward- and a forward-tree are combined. Then $\mathtt{g}_i$ is no longer an open goal but *satisfied*. The planning terminates if there are no open goals anymore.

In a proof-plan $\mathcal{P}$ a node N is termed *dependent* on a node N', if N is an ancestor of N' or N' is an ancestor of N in $\mathcal{P}$. A *sketch* differs from a proof-plan in that it may contain methods which are not verifiable. Sketches and proof-plans may be *summarized* by methods.

3.3 Transferring Methods Analogically

The general idea of analogy-driven proof-plan construction detailed in [8] is to use a source proof-plan as a guide for constructing an analogous target proof-plan. Specifically, we employ the structure and the methods belonging to the source plan or somewhat reformulated and restructured methods, for the target proof-plan.

We shall utilize the analogy-driven proof-plan construction for analogically transferring the method D12 of Cantor's proof-plan to a method for the proof-plan of the halting problem. Actually we shall transfer a proof-plan summarized by D12 to a proof-plan summarized by `Transferred D3`. This transfer is cognitively substantiated by the fact that mathematicians often describe their proving by analogy as *applying a method* used in another proof if this method is named, such as the Diagonalization method. If the method is not named, then they state that the target proof is done *analogously* to the source proof.

The analogy-driven proof-plan construction includes two different kinds of mapping proof-plans or sketches, reformulation and restructuring as characterized below. *Reformula-*

[10] In case the planner searches only backward, the proof-plan definition and the analogy procedure are simplified considerably.

tion denotes a sequence of reformulations which are triggered by a source sequent or method. Reformulation aims at matching a source goal with a target goal or as many preconditions of a source method with target assumptions respectively. *Restructuring* aims at providing submethods of the source plan the postcondition (or preconditions) of which eventually matches an open target goal (or assumptions) or at splitting a method that cannot be verified in order to find a verifiable submethod. By finding a verifiable submethod the amount of work left for the base-level planning is diminished.

Reformulations ρ map a proof-plan/sketch to another plan/sketch while preserving the proof-plan/sketch structure. A set of admissible reformulations is stored in a data base. A reformulation ρ consists of a mapping of methods M to methods, written as ρ applied to M, and mappings of sequents P to sequents, written as ρ applied to P. The mappings of methods are executed by so-called meta-methods that may change all slots of methods but procedure. Some of the stored meta-methods are `Term-Mapping`, `Homomorphy-Abstraction`, and `Introduce-Function-Parameter`. The changes of the pre- and postconditions by the meta-method establish the mapping of the sequents. If ρ is applied to a sequent g (or to a method M) of a plan $\mathcal{P}$, then ρ has to be applied to the nodes in the $\mathcal{P}$ that are dependent on g (or M). (Imagine, e.g., that a symbol is replaced by another one in M, then it has to be replaced in the same way in all methods and sequents dependent on M.) Thus a reformulation in analogy-driven proof-plan construction is not only dependent on ρ but also on the node in $\mathcal{P}$ that *triggers* the reformulation.

Restructuring maps a proof-plan/sketch to another proof-plan/sketch by replacing a sub-proof-plan $\mathcal{P}$ with one method by a proof-plan with several methods while preserving the root and leaves of $\mathcal{P}$. We refer to restructurings that are executed by restructuring meta-methods. Some of these meta-methods are `Deduction-Theorem-Splitting`, `Conjunctive-Decomposition`, and `Apply-Axiom-Splitting`. For a more detailed motivation, description and examples see [9].

Table 1 shows the top-level procedure of the analogy-driven proof-plan construction. The actual analogy procedure is embedded into the planning by a basic planner. Starting with a given source proof-plan, target assumptions, and a target goal, the output of the procedure is a proof-plan for the target goal.

1. Terminate if there are no open goals.
2. If the source plan is exhausted, then base-level plan for the open goals. For open goals that are not establishable, base-level plan for the closest preceding goal or assumption found by restructuring.
3. Get next sequent P from the source plan. The sequent is either an assumption or a goal.
4. If there is a reformulation ρ, such that $\rho\mathsf{P}$ matches an open target goal or a current target assumption respectively, then go to step 7.
5. If restructuring possible, then
   - Restructure and update source plan.
   - Go to step 3.
6. Go to step 2.
7. Reformulate the source plan by ρ and triggered by P.
8. Select for the target the method M chosen in the source. If in the source M was a forward-method, then go to step 13.
9. If M is not verifiable, then go to step 5.
10. Update open target goals and current target assumptions.
11. Link the new P and M to the source plan.
12. Go to step 1.
13. If there is a reformulation ρ' such that $|missing| < \theta$, for $missing :=$ set of preconditions of $\rho' M$ not matching a current target assumption, then
    - Reformulate the source plan by ρ' and triggered by M.
    - Go to step 9.
14. Go to step 5.

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

The following specifics are to note in Table 1: Base-level plan in step 2 denotes the basic planner activity. In step 13, θ is a threshold usually set to 2. A goal g is considered *not establishable* in step 2, if neither g not a reformulation of g holds for the target. This judgement is to be provided by the user and may reduce the search notably.

3.4 Transferring D12 to D3

Let D12 be decomposed into submethods according to the dividing lines in D12. Since we consider only a transfer from method D12 to a method D3, we start with the rerepresented halting problem and the assumptions given in the text.

The postcondition is $(\emptyset \vdash \bot)$ in both D12 and D3 and thus it does not require any reformulation. The analogical transfer of D12 to D3 replaces the submethod with postcondition $(G \in M2)$ by another method with postcondition $(G \in M2)$ found by base-level planning because a precondition similar to that of (2) in D12 is not establishable in D3. Thus the justification for $(G \in M2)$ becomes a new method Method_3 that provides $(G(x) = non\ F(xx))$, has also to be found by base-level planning.

Eventually matching the assumptions (8) in D12 and D3 requires an `Introduce-Function-Parameter` reformulation for a parameter c that includes the map

$(x \Rightarrow cx)$ for all x. This reformulation has to be applied to all sequents and methods dependent upon (8) in the source proof plan. Figure 4 shows the dependencies in the plan summarized by D12, where the numbers represent proof schema lines and assumptions as is D12.

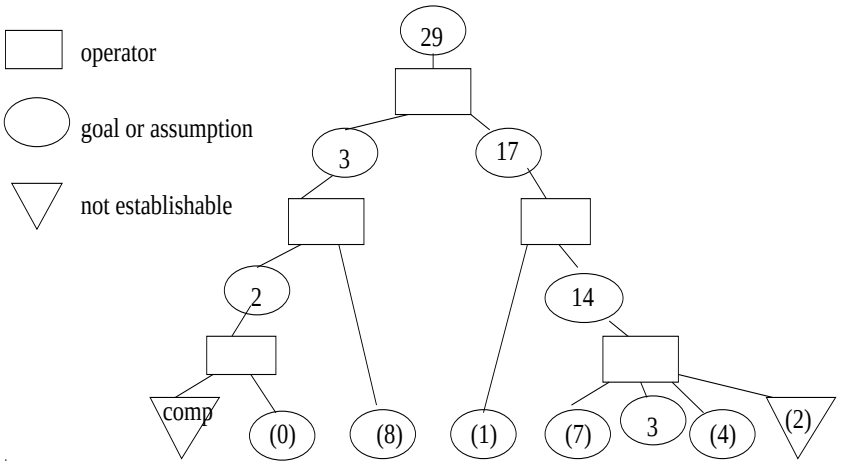

Figure 4: Dependencies in the proof-plan summarized by D12

Examining the current proof-plan shows that the submethods of D12 represented by line 4 - 14 and line 15 -17 are not dependent on (8) anymore. The analogy-driven proof-plan construction has produced a proof-plan that is summarized by the method `Transferred D3` shown below.

method: `Transferred D3`	
parameter	F, c: function, non: (partial) function, $M1, M2$: structure, H: formula
preconditions	$(1)\forall x \exists y(x \in M2 \rightarrow y \in M1 \wedge F(y) = x)$, $(5)\forall x(x = x)$ $(6)1 \neq 0$ $(7)H \vee \neg H$, $(8)\ \forall x((cx \neq 0 \rightarrow c\ non(x) = 0) \wedge (cx = 0 \rightarrow c\ non(x) = 1))$
postcondition	$\perp$
constraints	
proof schema	1. ; $\vdash$ $\forall x(x \in M1 \rightarrow G(x) = non\ F(xx))$ (Method$_2$) 2. ; $\vdash$ $G \in M2$ (Method$_3$) 3. ; $\vdash$ $\exists x(x \in M1 \wedge F(x) = G)$ ($\wedge D(1)$ 1) 4. ; $\vdash$ $F(x_0) = G$ ($\exists$D 3) 5. ; $\vdash$ $F(x_0, x_0) = G(x_0)$ (equ 4) 6. ; $\vdash$ $\forall x(x \in M1 \rightarrow (cF(xx) = 0 \rightarrow cG(x) = 1) \wedge (cF(xx) \neq 0 \rightarrow cG(x) = 0))$ (Method$_1$(8) 1) 7. ; $\vdash$ $(x_0 \in M1 \rightarrow (F(x_0x_0) = 0 \rightarrow G(x_0) = 1) \wedge (F(x_0x_0) \neq 0 \rightarrow G(x_0) = 0))$ ($\forall$D 6) 8. ; $\vdash$ $cF(x_0x_0) = 0 \vee cF(x_0x_0) \neq 0$ (LEMMA) 9. ; 9 $\vdash$ $cF(x_0x_0) = 0$ (HYP) 10. ; 9 $\vdash$ $cG(x_0) = 0$ (equ. 9 5) 11. ; 9 $\vdash$ $cG(x_0) = 1$ ($\wedge D \rightarrow$D 7 9) 12. ; 9 $\vdash$ $0 = 1$ ($\wedge$I,equ,(5) 11 10) 13. ; 9 $\vdash$ $\perp$ ($\wedge$I,$\perp$I,(6)) 14. ; 14 $\vdash$ $cF(x_0x_0) \neq 0$ (HYP) 15. ; 14 $\vdash$ $cG(x_0) = 0$ ($\wedge D, \rightarrow D$ 14 7) 16. ; 14 $\vdash$ $cF(x_0x_0) = 0$ (equ 15 5) 17. ; 14 $\vdash$ $cF(x_0x_0) = 0 \wedge \neg cF(x_0x_0) = 0$ ($\wedge$I 16 14) 18. ; 14 $\vdash$ $\perp$ ($\perp$I 17) 19. ; $\vdash$ $\perp$ ($\vee$D 18 13 8)
procedure	schema-interpreter

Replacing Method$_2$ and Method$_3$ by method variables in `Transferred D3` yields the method `D123` which can be used in proof-plans of Cantor's theorem, of the uncountability of $\mathbb{R}$, and of the halting problem.

Some additional remarks on changes not directly belonging to the transfer of D12 to D3: If the decomposition of D12 was not given at the beginning, then it could be established in the cycles of the analogy procedure. If we were to transfer the whole proof plan, then we would need to find an appropriate change of the representation of the halting problem by a reformulation employing equivalences modulo the theory. The employment of (8) requires a word of explanation. Actually, only the definitions of *non* and c are given as assumptions of the halting problem proof, and (8) is a lemma derived from these definitions. If (8) was

not provided as a lemma, it would be a goal that is not in the current state and to be found by the analogy procedure: In step 13 a possible reformulation of (8) of D12 could be created by `Introduce-Function-Parameter`, since instantiating the new parameter by the characteristic function c yields (8): $\forall x((cx \neq 0 \rightarrow c\,non(x) = 0) \wedge (cx = 0 \rightarrow c\,non(x) = 1))$ which can be established as an assumption for the halting problem proof.

3.5 Transferring D12 to D4

In the analogical transfer of D12 to a method D4 for the Gödel theorem proof-plan, a submethod Method_{31} for $(G \in M2)$ has to be found by base-level planning because a pre-condition similar to (2) to in D12 is not establishable for D4. The assumptions (0) in D12 and D4 require an `Introduce-Function-Parameter` reformulation (for the parameter w) that includes the mapping $(F \Rightarrow wF)$. The assumptions (8) of D12 and D4 require an `Introduce-Function-Parameter` reformulation that includes the mapping $(x \Rightarrow cx)$ for all x. The reformulations have to be applied to the submethods dependent on (8) and (0).

Examining the current sketch shows that the submethods of D12 represented by lines 4 - 14 and lines 15 -17 are not dependent on (8) and (0) anymore. Thus the respective reformulations do not affect these submethods. Up to this point the analogical transfer has produced a sketch that is summarized by the method (`Transferred D4`) shown below, which has a submethod represented by line 7 - 18 of the proof schema that cannot be verified. According to the analogy procedure, this submethod has to be restructured at the places indicated by "gap" in order to transfer as much as possible to a target plan. Two additional submethods[11] have to be inserted there, (which are superfluous in the case of D12) for obtaining verifiable methods.

The submethod Method_4, to be produced in order to create a proof plan makes use of the ω-completeness of S.

[11] E.g., for $cG(x_0) = 1 \vdash cwG(x_0) = 1$ and $cG(x_0) = 0 \vdash cwG(x_0) = 0$.

method: `Transferred D4`	
parameter	F, non, c, w: function, $M1, M2$: structure, U: set, H: formula
preconditions	$(0)\forall g(g \in M2 \rightarrow (x \in M1 \rightarrow wg(x) \in U))$, $(1)\forall x \exists y(x \in M2 \rightarrow y \in M1 \wedge F(y) = x)$, $(3)U \subseteq domain(non)$ $(5)\forall x(x = x)$ $(6)1 \neq 0$ $(7)(H \vee \neg H)$, $(8)\forall x((c(x) \neq 0 \rightarrow c(non(x)) = 0) \wedge (c(x) = 0 \rightarrow c(non(x)) = 1))$
postcondition	$\perp$
constraints	
proof schema	1. ; $\vdash$ $\forall x(x \in M1 \rightarrow G(x) = non\ wF(xx))$ (comprehension) 2. ; $\vdash$ $G \in M2$ (Method_{31}) 3. ; $\vdash$ $\forall x(x \in M1 \rightarrow (cwF(xx) = 0 \rightarrow cG(x) = 1) \wedge (cwF(xx) \neq 0 \rightarrow cG(x) = 0))$ (Method_1(8) 1) 4. ; $\vdash$ $\exists x(x \in M1 \wedge F(x) = G)$ ($\wedge D$(1) 1) 5. ; $\vdash$ $F(x_0) = G$ ($\exists$D 4) 6. ; $\vdash$ $F(x_0, x_0) = G(x_0)$ (equ 5) not verified submethod 7. ; $\vdash$ $(x_0 \in M1 \rightarrow (F(x_0x_0) = 0 \rightarrow G(x_0) = 1) \wedge (F(x_0x_0) \neq 0 \rightarrow G(x_0) = 0))$ ($\forall$D 3) 8. ; $\vdash$ $cwF(x_0x_0) = 0 \vee cwF(x_0x_0) \neq 0$ (LEMMA) 9. ; 9 $\vdash$ $cwF(x_0x_0) = 0$ (HYP) 10. ; 9 $\vdash$ $cG(x_0) = 1$ ($\wedge D \rightarrow$D 7 9) gap to be filled by (Method_4) 11. ; 9 $\vdash$ $0 = 1$ ($\wedge$I,equ,(5),?) 12. ; 9 $\vdash$ $\perp$ ($\wedge$I,$\perp$I,(6)) 13. ; 13 $\vdash$ $cwF(x_0x_0) \neq 0$ (HYP) 14. ; 13 $\vdash$ $cG(x_0) = 0$ ($\wedge D, \rightarrow$ D 13 7) gap to be filled by (Method_5) 15. ; 13 $\vdash$ $cwF(x_0x_0) = 0$ (equ,? 6) 16. ; 13 $\vdash$ $cwF(x_0x_0) = 0 \wedge \neg cwF(x_0x_0) = 0$ ($\wedge$I 15 13) 17. ; 13 $\vdash$ $\perp$ ($\perp$I 16) 18. ; $\vdash$ $\perp$ ($\vee$D 17 12 8)
procedure	schema-interpreter

Again, we do not detail the rerepresentation of D-Gödel to D4 that does not belong to the very transfer of D12 to D4. This rerepresentation is carried out by the introduction of the characteristic function c and that is eventually due to the aim of matching the assumption (8) of D12 with (8) of D-Gödel. It is caused by an `Introduce-Function-Parameter` reformulation substantiated by introducing a characteristic function c into (8). The other changes (introducing the new step (d) and changing the last step in D-Gödel) appear automatically by the analogical procedure. The parameters have to be instantiated for a proof of Gödel's theorem in the following way: non, c, w as defined in the text, F by the enumeration of E, $M1$ by $\mathbb{N}$, $M2$ by E and U by the set_of_meta_formulas.

In order to obtain a method that can replace D1, D2, and D4 in the three plans and that closer corresponds to the principle: "Make a and a_n differ in n", the additional submethods are moved up to Method$_1$[12]. A new Method$_6$ results with the postcondition $(x_0 \in M1 \rightarrow (cwF(x_0x_0) = 0 \rightarrow cwG(x_0) = 1) \wedge (cwF(x_0x_0) \neq 0 \rightarrow cwG(x_0) = 0))$

method: `Method`$_6$				
parameter	F, c, w: function, $M1$: structure			
preconditions	$(9)\forall x(x \in M1 \rightarrow G(x) = non\ wF(xx))$, $(8)\forall x((c(x) \neq 0 \rightarrow c(non(x)) = 0) \wedge (c(x) = 0 \rightarrow c(non(x)) = 1))$			
postcondition	$(x_0 \in M1 \rightarrow (cwF(x_0x_0) = 0 \rightarrow cwG(x_0) = 1) \wedge (cwF(x_0x_0) \neq 0 \rightarrow cwG(x_0) = 0))$			
constraints				
proof schema	1.	;	$\vdash$	$\forall x(x \in M1 \rightarrow (cwF(xx) = 0 \rightarrow cG(x) = 1) \wedge (cwF(xx) \neq 0 \rightarrow cG(x) = 0))$	(Method$_1$,(8),(9))
	2.	;	$\vdash$	$(x_0 \in M1 \rightarrow (F(x_0x_0) = 0 \rightarrow G(x_0) = 1) \wedge (F(x_0x_0) \neq 0 \rightarrow G(x_0) = 0))$	($\forall$D 1)
	2.	$_{cG(x_0)};= 1$	$\vdash$	$cwG(x_0) = 1$	(PLAN4)
	3.	$_{cG(x_0)};= 0$	$\vdash$	$cwG(x_0) = 0$	(PLAN5)
	4.	;	$\vdash$	$(x_0 \in M1 \rightarrow (cwF(x_0x_0) = 0 \rightarrow cwG(x_0) = 1) \wedge (cwF(x_0x_0) \neq 0 \rightarrow cwG(x_0) = 0))$	(Method$_7$;1 2 3)
procedure					

Note that the method variables PLAN4 and PLAN5 indicate that the respective methods differ for the proof-plans of Cantor's and Gödel's theorem. Because of the instantiations of the parameter w by the identity function, the methods to be inserted for PLAN4 and PLAN5 are trivial in all cases but in the Gödel proof-plan.

Employing Method$_6$ and its postcondition instead of Method$_1$, replacing the method represented by line 4 - 14 in D12 by the method variable PLAN3, and omitting preconditions different in D12 and `Transferred D4` yields the method `D124` which can be used in proof-plans of Cantor's theorem, of the uncountability of $\mathbb{R}$, and of Gödel's theorem.

3.6 The Diagonal Method

Comparing `D123` and `D124` results in the method `Diagonal` that can be used in the proof-plans of Cantor's theorem, of the uncountability of $\mathbb{R}$, of Gödel's theorem, as well as of the halting problem.

[12] A means for moving submethods in a proof-plan is Bledsoe's precondition prover [1].

method: `Diagonal`	
parameter	F, c, w: function, non: (partial) function, $M1, M2$: structure, H: formula
preconditions	$(1)\forall x \exists y(x \in M2 \rightarrow y \in M1 \wedge F(y) = x)$, $(5)\forall x(x = x)$ $(6)0 \neq 1$, $(7)H \vee \neg H$, $(8)\forall x((c(x) \neq 0 \rightarrow c(non(x)) = 0) \wedge (c(x) = 0 \rightarrow c(non(x)) = 1))$
postcondition	$\perp$
constraints	
proof schema	1. ; $\vdash$ $\forall x(x \in M1 \rightarrow (G(x) = non\ wF(xx)))$ (PLAN2) 2. ; $\vdash$ $G \in M2$ (PLAN3) 3. ; $\vdash$ $\exists x(x \in M1 \rightarrow F(x) = G)$ ($\forall$D (1) 1) 4. ; $\vdash$ $F(x_0) = G$ ($\exists$D 3) 5. ; $\vdash$ $F(x_0x_0) = G(x_0)$ (equ 4) 6. ; $\vdash$ $(cwF(x_0x_0) = 0 \rightarrow cwG(x_0) = 1) \wedge (cwF(x_0x_0) \neq 0 \rightarrow cwG(x_0) = 0)$ (Method$_6$) 7. ; $\vdash$ $cwF(x_0x_0) = 0 \vee cwF(x_0x_0) \neq 0$ (LEMMA) 8. ; 8 $\vdash$ $cwF(x_0x_0) = 0$ (HYP) 9. ; 8 $\vdash$ $cwG(x_0) = 0$ (equ. 8 5) 10. ; 8 $\vdash$ $cwG(x_0) = 1$ ($\wedge D \rightarrow$D 6 8) 11. ; 8 $\vdash$ $0 = 1$ ($\wedge$I,equ,(5) 9 10) 12. ; 8 $\vdash$ $\perp$ ($\wedge$I,$\perp$I (6) 11) 13. ; 13 $\vdash$ $cwF(x_0x_0) \neq 0$ (HYP) 14. ; 13 $\vdash$ $cwG(x_0) = 0$ ($\wedge$D,$\rightarrow$D 13 6) 15. ; 13 $\vdash$ $cwF(x_0x_0) = 0$ (equ 14 5) 16. ; 13 $\vdash$ $cwF(x_0x_0) = 0 \wedge \neg cwF(x_0x_0) = 0$ ($\wedge$I 15 13) 17. ; 13 $\vdash$ $\perp$ ($\perp$I 16) 18. ; $\vdash$ $\perp$ ($\vee$D 17 12 7)
procedure	schema-interpreter

`Diagonal` is a top-level method, which means it represents a proof idea and leaves some methods unspecified which prove details. The method variables PLAN$_i$ have to be instantiated for particular applications of the `Diagonal` method. Basically, PLAN1, PLAN2, and Method$_6$ together denote the unspecified proof of what is often called the "diagonalization lemma" in mathematical diagonalization proofs.

4 Acknowledgement

The first two presented cases are based upon investigations and Natural Deduction proofs provided by my colleagues Xiaorong Huang and Manfred Kerber, who also did most of the

work towards the representational framework. I would also like to thank Jörn Richts for thoroughly reading a draft of this paper.

References

[1] W. Bledsoe. A precondition prover for analogy. In *Biosystems*. 1994. in press.

[2] G.S. Boolos and R.C. Jeffrey. *Computability and Logic*. Cambidge University Press, Cambridge, third edition, 1989.

[3] 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)*, volume 310 of *Lecture Notes in Computer Science*, pages 111–120, Argonne, 1988. Springer.

[4] G. Cantor. *Contributions to the Foundation of the Theory of Transfinite Numbers*. Dover (Reprint), New York, 1955.

[5] N. Cutland. *Computability, an Introduction to Recursive Function Theory*. Cambridge University Press, Cambridge, New York, 1980.

[6] X. Huang, M. Kerber, and M. Kohlhase. Methods - the basic units for planning and verifying proofs. SEKI-Report SR-92-20 (SFB), Universität des Saarlandes, 1992.

[7] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1981.

[8] E. Melis. Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, Universität des Saarlandes, Saarbrücken, 1993.

[9] E. Melis. Decomposition techniques and their applications. In *Proc. AIMSA'94*, Sofia, Bulgaria, 1994.

[10] E. Mendelson. *Introduction to Mathematical Logic*. Van Nostrand Company Inc., Princeton, 1964.

[11] S.C.Kleene. *Introduction to Metamethematics*. North Holland Publ Co., Amsterdam, 1952.