Cantor's Diagonalisation Principle

0. Diagonalisation

With $P.x.y$ a predicate in which x and y are free variables having the same range, the following step may occur in calculational proofs:

$$\begin{array}{ll} & (\forall x :: (\exists y :: P.x.y)) \\ \Leftarrow & \quad \{ \text{ instantiation } y := x \} \\ & (\forall x :: P.x.x) \quad . \end{array}$$

Such a step is not at all far-fetched: instantiation is one of the few ways to eliminate a quantor, and within the scope of x the instantiation $y := x$ is the only possibility as long as we do not have or wish to use further knowledge about the range of x and y.

The above kind of step hardly deserves a special name, although it is the crux of what is known as "Cantor's Diagonalisation Principle". To make occurrences of this step visible we shall use the hint "diagonalisation" in the following examples. These examples serve to show that diagonalisation certainly is effective, but that otherwise there is nothing deep about it.

1. Cantor's theorem

In this section surjective functions play a

major role. Surjectivity can be defined as follows.

definition: for function f: $f \in X \leftarrow Y$:

(0) $\quad sur.f \equiv (\forall x : x \in X : (\exists y : y \in Y : x = f.y))$

□

The following is a generalisation of a theorem by Cantor.

theorem 0: for non-singleton set X and any set U:

$$\neg(\exists F : F \in (X \leftarrow U) \leftarrow U : sur.F) .$$

proof: By de Morgan's rule the theorem is equivalent to:

$$(\forall F : F \in (X \leftarrow U) \leftarrow U : \neg sur.F) ,$$

which we prove by deriving for any F in $(X \leftarrow U) \leftarrow U$:

$$\begin{array}{ll}
 & \neg\ sur.F \\
\equiv & \quad \{\ (0)\ \} \\
 & \neg(\forall g : g \in X \leftarrow U : (\exists u : u \in U : g = F.u)) \\
\equiv & \quad \{\ \text{de Morgan (twice)}\ \} \\
 & (\exists g : g \in X \leftarrow U : (\forall u : u \in U : g \neq F.u)) \\
\Leftarrow & \quad \{\ \text{Leibniz ; leaving the ranges of } g,u,v \text{ implicit}\ \} \\
 & (\exists g :: (\forall u :: (\exists v :: g.v \neq F.u.v))) \\
\Leftarrow & \quad \{\ \text{diagonalisation } (u \in U \text{ and } v \in U)\ \} \\
 & (\exists g :: (\forall u :: g.u \neq F.u.u)) \\
\Leftarrow & \quad \{\ \text{instantiation } g.u := h.(F.u.u)\ \} \\
 & (\exists h : h \in X \leftarrow X : (\forall u :: h.(F.u.u) \neq F.u.u)) \\
\Leftarrow & \quad \{\ \text{generalisation , to eliminate } F\ \} \\
 & (\exists h : h \in X \leftarrow X : (\forall x : x \in X : h.x \neq x)) .
\end{array}$$

The formula $(\exists h : h \in X \leftarrow X : (\forall x : x \in X : h.x \neq x))$ is equivalent to "X is not a singleton" —which we shall not prove here— . Thus we have proved the theorem. The above proof is rather straightforward; the step labelled "Leibniz" is unavoidable: the only way to establish that functions, such as g and $F.u$, are different is by applying them to arguments. The instantiation $g.u := h.(F.u.u)$ is a bit ad hoc; an other possible continuation is:

$$
\begin{array}{ll}
 & (\exists g :: (\forall u :: g.u \neq F.u.u)) \\
\Leftarrow & \quad \{ \text{ Axiom of Choice } \} \\
 & (\forall u :: (\exists x :: x \neq F.u.u)) \\
\Leftarrow & \quad \{ \text{ generalisation, to eliminate } F \} \\
 & (\forall y :: (\exists x :: x \neq y)) \ ,
\end{array}
$$

and the formula thus obtained is a more obvious characterisation of X being not a singleton. The latter proof is simpler than the former, but the former contains no appeal to the Axiom of Choice.

□

The <u>cardinality</u> of a nonempty set X is <u>at most</u> the cardinality of a set Y if a surjective function in $X \leftarrow Y$ exists. If no such function exists then X's cardinality is greater than Y's. This explains the relevance of the following theorem, because it implies

that $\mathbb{P}.\mathcal{U}$ has greater cardinality than $\mathcal{U}$, for all sets $\mathcal{U}$.

theorem 1: every set $\mathcal{U}$ satisfies:

$$\neg(\exists F: F \in \mathbb{P}.\mathcal{U} \leftarrow \mathcal{U}: \mathrm{sur}.F) \ .$$

proof:

$$\begin{array}{ll} & (\exists F: F \in \mathbb{P}.\mathcal{U} \leftarrow \mathcal{U}: \mathrm{sur}.F) \\ \equiv & \quad \{ \ \mathbb{P}.\mathcal{U} \text{ is isomorphic with } \{0,1\} \leftarrow \mathcal{U} \ \} \\ & (\exists F: F \in (\{0,1\} \leftarrow \mathcal{U}) \leftarrow \mathcal{U}: \mathrm{sur}.F) \\ \equiv & \quad \{ \text{ theorem 0 , with } X := \{0,1\} \ \} \\ & \text{false} \ . \end{array}$$

□

As a matter of fact, the generalisation embodied by theorem 0 followed from the observation that the only relevant property of $\{0,1\}$ is its being a non-singleton set. We now invite the reader to compare the proof of theorem 0 with the following standard proof, taken from "Naive Set Theory", by Paul R. Halmos.

Cantor's theorem. *Every set is strictly dominated by its power set, or, in other words,*

$$X \prec \mathcal{P}(X)$$

for all X.

PROOF. There is a natural one-to-one mapping from X into $\mathcal{P}(X)$, namely, the mapping that associates with each element x of X the singleton $\{x\}$. The existence of this mapping proves that $X \preceq \mathcal{P}(X)$; it remains to prove that X is not equivalent to $\mathcal{P}(X)$.

Assume that f is a one-to-one mapping from X onto $\mathcal{P}(X)$; our purpose is to show that this assumption leads to a contradiction. Write $A = \{x \in X: x \notin f(x)\}$; in words, A consists of those elements of X that are not contained in the corresponding set. Since $A \in \mathcal{P}(X)$ and since f maps X onto $\mathcal{P}(X)$, there exists an element a in X such that $f(a) = A$. The element a either belongs to the set A or it does not. If $a \in A$, then, by the definition of A, we must have $a \notin f(a)$, and since $f(a) = A$ this is impossible. If $a \notin A$, then, again by the definition of A, we must have $a \in f(a)$, and this too is impossible. The contradiction has arrived and the proof of Cantor's theorem is complete.

Apart from its verbosity, this proof is ugly for 3 reasons:

- the unnecessary use of reductio ad absurdum,
- the pulling out of the hat of the definition of set A,
- the unnecessary case analysis $a \in A$ and $a \notin A$, which is probably caused by the natural-deduction style of reasoning where implication is the only tool.

2. The Halting Problem

In this section u, x, y denote so-called strings. To understand the theorem presented here we need not know what strings are, but for the interpretation it is useful to know that

i) the contents of the tape of a Turing machine is a string,
ii) every string is a "description" of a Turing machine and every Turing machine is described by some string.

By ii) we may speak about Turing machine x, for every string x, and all Turing machines are thus identified.

Starting point for our discussion is a predicate H on pairs of strings, with the following interpretation:

$H.x.y \equiv$ "when started with y on its tape, Turing machine x will eventually halt".

In what follows, variable L ranges over the sets of

strings —also called "languages"—. We define Ta and Td, which are predicates on languages, as follows.

definition: Ta and Td satisfy, for all L:

(0) $\quad Ta.L \equiv (\exists x :: (\forall y :: y \in L \equiv H.x.y))$

(1) $\quad Td.L \equiv Ta.L \wedge Ta.(\neg L)$

□

We are now able to prove the following theorem with, as we will see, very little knowledge about H.

theorem ("not every acceptable language is decidable"):

$$(\exists L :: Ta.L \wedge \neg Td.L) \ .$$

proof: by construction of a witness; first, we eliminate Td by rewriting:

$$\begin{array}{ll} & Ta.L \wedge \neg Td.L \\ \equiv & \quad \{ \ (1) \ ; \text{ predicate calculus} \ \} \\ & Ta.L \wedge \neg Ta.(\neg L) \ . \end{array}$$

We now proceed by constructing an L satisfying one conjunct and then showing that it satisfies the other conjunct. This can be done in two ways, which happen to yield the same result; therefore, we pursue one possibility only:

$$\begin{array}{ll} & Ta.L \\ \equiv & \quad \{ \ (0) \ \} \end{array}$$

$$
\begin{array}{ll}
 & (\exists x :: (\forall y :: y \in L \equiv H.x.y)) \\
\Leftarrow & \quad \{ \text{ instantiation } x := u \text{ , } u \text{ yet to be determined } \} \\
 & (\forall y :: y \in L \equiv H.u.y) \ .
\end{array}
$$

Taking this as L's definition we now prove:

$$
\begin{array}{ll}
 & \neg Ta.(\neg L) \\
\equiv & \quad \{ (0) \ ; \ y \in \neg L \equiv y \notin L \} \\
 & \neg (\exists x :: (\forall y :: y \notin L \equiv H.x.y)) \\
\equiv & \quad \{ \text{ de Morgan (twice) ; cancellation of } \neg \text{'s } \} \\
 & (\forall x :: (\exists y :: y \in L \equiv H.x.y)) \\
\Leftarrow & \quad \{ \text{ diagonalisation } \} \\
 & (\forall x :: x \in L \equiv H.x.x) \\
\equiv & \quad \{ \text{ above definition of } L \ \} \\
 & (\forall x :: H.u.x \equiv H.x.x) \ .
\end{array}
$$

Thus, we have proved the theorem provided that a string u exists satisfying (2) (see below).

□

specification: string u satisfies

(2) $\quad (\forall x :: H.u.x \equiv H.x.x)$

□

String u represents a restricted form of the Universal Turing Machine; string u exists indeed, and for the validity of the theorem this is all we need to know about H. Notice that in the above proof both the definition of L and the specification of u emerge from the calculation without any difficulty. Again,

the reader is invited to compare this with the presentation of the same theorem in, for example, "Elements of the Theory of Computation", by Harry R. Lewis and Christos H. Papadimitriou.

Eindhoven, 18 june 1993
Rob R. Hoogerwoord
department of mathematics and computing science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven