The induction rule of De Bakker and Scott

Wim H. Hesselink, February 1989

On the occasion of the Symposium:
"J. W. de Bakker: 25 years of semantics"

Manna's book [M] contains a theorem that goes back to [BS], an unpublished paper of De Bakker and Scott of 1969. This theorem is thus 20 years old. It is called "stepwise computational induction" ([M] 5.5). Disguised as Scott's induction rule, the theorem can also be found in De Bakker's book [B]. More precisely, the book [B] contains a deterministic version (5.37) and a nondeterministic one (7.16). The latter version heavily relies on continuity with respect to the so-called Egli–Milner ordering. Continuity is roughly the same as finite nondeterminacy, for it implies that every necessarily terminating command has a finite set of potential results.

In this note in honour of De Bakker, I would like to announce a generalisation of his induction rule to cases where infinite nondeterminacy is allowed. Actually, a full generalisation is not possible, for I will show a case with infinite nondeterminacy, where the induction rule is not valid. For the proof of the result and for more details, I refer to [H2] and [H3]. The note [H1] contains a small application.

The rule is stated here in a form that differs considerably from the forms in [M] and [B]. The reason is that I re-invented the wheel, so that I also invented my own formalisms. On the other hand, the formalism to be described is more convenient for the applications that I had in mind. In fact, my aim was program transformation rather than correctness. It was only recently, that C. Hemerik pointed out to me that my result was a version of the induction rule of De Bakker and Scott.

Command algebras

Command algebras are introduced to serve as an abstract syntax with a more flexible concept of equality. They are inspired by De Bakker's treatment of nondeterminacy in [B] chapter 7, and also by the process algebras of Bergstra and Klop [BK].

A command algebra A is defined to be a set with constants *fail* $\in A$, *skip* $\in A$ and with binary operators ";" and "$[\!]$" such that the following axioms are satisfied

(0)
$$\begin{array}{ll} a \,[\!]\, a = a & a \,[\!]\, b = b \,[\!]\, a \,, \\ (a \,[\!]\, b) \,[\!]\, c = a \,[\!]\, (b \,[\!]\, c) & a \,[\!]\, \mathit{fail} = a \,, \\ \mathit{fail}\,; a = \mathit{fail} & (a\,; b)\,; c = a\,; (b\,; c) \,, \\ \mathit{skip}\,; a = a & a\,; \mathit{skip} = a \,, \\ (a \,[\!]\, b)\,; c = a\,; c \,[\!]\, b\,; c & a\,; (b \,[\!]\, c) = a\,; b \,[\!]\, a\,; c \,. \end{array}$$

A command algebra A is equipped with a partial order "$\leq$" given by

$$a \leq b \quad \equiv \quad a = a \,[\!]\, b.$$

It turns out that $a \| b$ is the greatest lower bound of a and b with respect to the order. Therefore, it is natural to use the symbol $\|$ for arbitrary greatest lower bounds in A. So, if $E \subset A$ has a greatest lower bound in A, then that bound is denoted by $(\| x \in E :: x)$. The algebra A is called *complete* if and only if every subset of A has a greatest lower bound. It turns out that a command $(\| x \in E :: x)$ may be regarded as a nondeterminate choice between the commands $x \in E$.

Now we assume that a command algebra B is given. We regard the elements of B as straight-line commands. We assume that the semantics of B is given by relational semantics. So let Σ_0 be the state space and let $\Sigma = \Sigma_0 \cup \{\perp\}$. The meaning of a command c is given as a subset $M.c$ of the cartesian product $\Sigma_0 \times \Sigma$. A pair (σ, τ) belongs to $M.c$ iff τ is a potential result when command c is called in state σ. A pair $(\sigma, \perp)$ belongs to $M.c$ iff command c need not terminate when called in σ. For a boolean function b on Σ_0, let $?b \in B$ be the command such that $M.(?b)$ consists of all pairs (σ, σ) such that $b.\sigma$ holds, cf. [B] definition 7.8.

Procedures and recursion are treated as follows. We introduce a set H of the occurring procedure names. We then form the polynomial algebra $B[H]$, which consists of the command algebra expressions in elements of B and H modulo the equalities induced by the axioms (0) and the identity relations of B and H, see [H2] section 3.1. The next step is to construct an embedding of algebra $B[H]$ into a complete command algebra $B[H]^*$, see [H3]. This completion satisfies the strong distributive law

$$(\| p \in E, q \in F :: p ; q) = (\| p \in E :: p) ; (\| q \in F :: q)$$

for any pair of nonempty subsets E and F of $B[H]^*$.

A declaration of the procedures is a function $d : H \rightarrow B[H]^*$, where the body of procedure $h \in H$ is defined to be the element $d.h \in B[H]^*$. In this way, recursion and even mutual recursion is possible, and procedure bodies may contain unbounded choice. The semantic function M from $B[H]^*$ to subsets of $\Sigma_0 \times \Sigma$ can be defined as the smallest interpretation with respect to the Egli–Milner ordering and there are equivalent definitions by operational means or by means of predicate transformers, cf. [H0]. In [H2], I use predicate transformer semantics.

The number of recursive procedures need not be finite. In fact, infinite families of procedures are used to allow value parameters and procedure parameters, cf. [H1]. For example, a procedure p with an input parameter v of type V is regarded as a family of commands $p.v$. A call of procedure p with as actual parameter the state function f is defined as the command $p(f) = (\| v \in V :: ?(f = v) ; p.v)$.

The generalised induction rule of De Bakker and Scott

Instead of the generalised correctness formulae as introduced by De Bakker, cf. [B] 5.25 and 7.11, we use congruences, which are defined as follows. A *congruence* on a complete command algebra is defined to be an equivalence relation $\sim$ such that for all commands p, q, r, and s

$$p \sim q \quad \wedge \quad r \sim s \quad \Rightarrow \quad p\,;r \sim q\,;s$$

and that for all sets of commands E, F

$$(\forall p \in E :: (\exists q \in F :: p \sim q)) \quad \wedge \quad (\forall q \in F :: (\exists p \in E :: p \sim q))$$
$$\Rightarrow \quad ([\!] p \in E :: p) \sim ([\!] q \in F :: q).$$

Let command $\Omega \in B$ be the abortive command with semantics given by

$$\langle \sigma, \tau \rangle \in M.\Omega \quad \equiv \quad \tau = \bot,$$

and let $da : B[H]^* \rightarrow B[H]^*$ be the function such that $da.s$ is obtained from expression s by substituting Ω for every procedure name in expression s. In the same way, we let $d^* : B[H]^* \rightarrow B[H]^*$ be the function such that $d^*.s$ is obtained from s by replacing every procedure name h in expression s by its body $d.h$.

In [H3], we introduce a certain subset *Lia* of $B[H]^*$. Let BU be the set of the commands in B that are of finite nondeterminacy. The results of [H3] imply

(1) $\quad B \subset Lia \quad \wedge \quad (BU\,;H\,;B) \subset Lia$
$\quad \wedge \quad (\forall p, q \in Lia :: p [\!] q \in Lia).$

The generalisation of the induction rule is

Theorem ([H3] 7(14)). Let E be a set of pairs of elements of *Lia* such that

$$(\forall \langle x, y \rangle \in E :: M.(da.x) = M.(da.y)),$$

and that for every congruence $\sim$ on $B[H]^*$ we have

$$(\forall \langle x, y \rangle \in E :: x \sim y) \quad \Rightarrow \quad (\forall \langle x, y \rangle \in E :: d^*.x \sim d^*.y).$$

Then $M.x = M.y$ for all pairs $\langle x, y \rangle \in E$.

In order to show that the condition on *Lia* cannot be omitted, let us consider the following example in which the theorem is not valid. Assume that there is one integer variable i. Let h be the procedure name with the declaration

$$d.h = (?(i > 0)\,; i := i - 1\,; h\,; i := i + 1 [\!] \ ?(i \leq 0)).$$

Here, ";" has higher priority than $[\!]$. Clearly, h is semantically equal to *skip*. Let $p \in B$ be a command that is guaranteed to terminate and that assigns to i an arbitrary positive value. Thus, the composition $(p;h)$ is guaranteed to terminate. This implies that

(2) $\quad M.(p;h) \neq M.(p;h [\!] \Omega).$

On the other hand, let us take E to be the singleton set

$$E = \{\langle (p;h), (p;h [\!] \Omega) \rangle\}.$$

It is possible to prove (cf. [H2] section 5.6) that both formulae of the theorem are satisfied. By (2), however, the consequent of the theorem is false. So, the condition that the commands be element of *Lia* is violated. The first conclusion is that this nasty condition cannot be omitted. Moreover, from formula (1) we get that $(p;h) \notin Lia$ whereas p and h are both element of *Lia*. Therefore, *Lia* is not closed under composition.

Let me conclude with a more positive remark. The note [H1] contains an application of the theorem to a recursive procedure with an input parameter and a procedural parameter. In this case it is important that the set E of the theorem is allowed to be infinite, and that the commands that occur in E can be complicated expressions.

References

[B] J. de Bakker: Mathematical theory of program correctness. Prentice–Hall International, 1980.

[BS] J.W. de Bakker, D. Scott: A theory of programs. IBM Seminar Vienna, Austria (August 1969). Unpublished notes.

[BK] J.A. Bergstra, J.W. Klop: Algebra of communicating processes, in: J.W. de Bakker, M. Hazewinkel and J.K. Lenstra, eds., Proc. CWI Symp. on Mathematics and Computer Science (North–Holland, Amsterdam, 1986)

[H0] W.H. Hesselink: Interpretations of recursion under unbounded nondeterminacy. Theoretical Computer Science **59** (1988) 211–234.

[H1] W.H. Hesselink: Initialisation with a final value, an exercise in program transformation (WHH 16). To appear in the Proceedings of "Mathematics of program construction", June 1989.

[H2] W.H. Hesselink: Command algebras, recursion and program tranformation (WHH 36). Tech. Rep. CS 8812, Groningen University 1988.

[H3] W.H. Hesselink: Command algebras with unbounded choice (WHH 47). Draft, 1989.

[M] Z. Manna: Mathematical theory of computation. McGraw–Hill Book Company 1974.

Address: Rijksuniversiteit Groningen, Department of Computing Science
P.O. Box 800, 9700 AV Groningen, The Netherlands