stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 150/80 OKTOBER
(DEPARTMENT OF COMPUTER SCIENCE)

R.J.R. BACK

PROVING TOTAL CORRECTNESS OF NONDETERMINISTIC PROGRAMS IN INFINITARY LOGIC

Preprint

kruislaan 413 1098 SJ amsterdam

1980 Mathematics subject classification: 68B10

ACM-Computing Reviews-category: 5.24, 5.21

Proving total correctness of nondeterministic programs in infinitary logic[*)]

by

R.J.R.Back

ABSTRACT

It is shown how the weakest precondition approach to proving total correctness of nondeterministic programs can be formalized in infinitary logic. The weakest precondition technique is extended to hierarchically structured programs by adding a new primitive statement for operational abstraction, the nondeterministic assignment statement, to the guarded commands of Dijkstra. The infinitary logic $L_{\omega_1\omega}$ is shown to be strong enough to express the weakest preconditions for Dijkstra's guarded commands, but too weak for the extended guarded commands. Two possible solutions are considered: going to the essentially stronger infinitary logic $L_{\omega_1\omega_1}$ and restricting the power of the nondeterministic assignment statement in a way which allows the weakest preconditions to be expressed in $L_{\omega_1\omega}$.

KEY WORDS & PHRASES: Infinitary logic, total correctness, nondeterministic programs, weakest preconditions, operational abstraction, stepwise refinement, unbounded nondeterminism, expressibility

---

*) This report will be submitted for publication elsewhere.

1. INTRODUCTION

The infinitary logic $L_{\omega_1\omega}$, which extends first-order logic by allowing countable disjunctions and conjunctions of formulas, has been proposed by ENGELER[10,11] as a suitable framework in which to reason about termination of programs. He shows how to assign to a given program a formula in $L_{\omega_1\omega}$ which will be satisfied if and only if the program terminates. SALWICKI[21] extended Engeler's approach to total correctness of programs. In Salwicki's system, known as *algorithmic logic*, first-order logic is extended with formulas of the form $S\alpha$, where S is a program and α is a formula. The formula $S\alpha$ expresses that the program S terminates and that upon termination α holds. Infinitary rules of inference are used to prove total correctness of loops, so this logic is also of an infinitary nature. Salwicki's approach has been carried further by a number of people, mostly working in Warsaw (see e.g. BANACHOWSKI[3]). The idea of extending first-order formulas with programs is also exploited in the *dynamic logic* by PRATT[19] and in the *programming logic* by CONSTABLE[6].

Another, less formal approach to reasoning about total correctness has been proposed by DIJKSTRA[8], based on the concept of weakest preconditions. Essentially his idea is to associate with each program a predicate transformer, which for any postcondition of the program gives the weakest precondition which guarantees that the program terminates in a final state satisfying the postcondition. Dijkstra's technique covers both deterministic and nondeterministic programs, whereas the early work on algorithmic logic only was concerned with deterministic programs. Algorithmic logic has later been extended to also cover nondeterministic programs (RASIOWA[20]). A careful analysis of nondeterminism and weakest preconditions in the framework of dynamic logic can be found in HAREL[12].

Our intention here is first to show how Engeler's work on proving termination can be extended to provide a simple formalization of

Dijkstra's weakest precondition technique for proving total correctness of guarded commands. We then go on to consider the effect of extending the guarded commands with a new primitive statement for expressing operational abstraction, the <u>nondeterministic assignment statement</u>. This construct is useful when developing programs by stepwise refinement, permitting a hierarchical decomposition of the correctness proof (BACK[1]). It turns out, however, that the weakest preconditions for the extended guarded commands cannot always be expressed in the logic $L_{\omega_1\omega}$, because of the "unbounded" nondeterminism allowed by the nondeterministic assignment statement.

Recognizing the desirability of hierarchically structured programs and correctness proofs, we consider two different possible solutions. The first one is to find a stronger logic, in which the weakest preconditions for the extended guarded commands are expressible. The second possibility is to restrict the nondeterministic assignment statement in a way which keeps the weakest preconditions expressible in $L_{\omega_1\omega}$.

2. INFINITARY LOGIC

The usual first-order logic is extended to infinitary logic by allowing infinitely long formulas like

$$\forall x(x=0 \vee x=1 \vee x=2 \vee \ldots)$$

or

$$\exists x_0 x_1 x_2 x_3 \ldots (x_0<x_1 \ \& \ x_1<x_2 \ \& \ x_2<x_3 \ \ldots)$$

In the first case we have a disjunction over an infinite set of formulas while in the second case we have a conjunction over an infinite set of formulas together with an existential quantification over an infinite set of variables.

Let L be a set of constant, function and predicate symbols and let α and β be two infinite cardinals ($\alpha \geq \beta$). The infinitary logic $L_{\alpha\beta}$ is like the ordinary first-order logic of L, with the same logical and nonlogical symbols, except that it allows the conjunction and disjunction over a set of fewer than α formulas and the universal and existential quantification on a set of fewer than β variables.

We will mainly be interested in $L_{\alpha\beta}$ when α and β are either ω or ω_1. The cardinal ω is the cardinality of the set of natural numbers and is the smallest infinite cardinal. A set with fewer than ω elements is thus finite. The cardinal ω_1 is the next smallest infinite cardinal; a set with fewer than ω_1 elements is countable. $L_{\omega\omega}$ is thus the usual first-order logic, allowing only finite disjunctions, conjunctions and quantification. $L_{\omega_1\omega}$ extends this by allowing countable disjunctions and conjunctions, but still only permitting finite quantification, while $L_{\omega_1\omega_1}$ also allows countable quantification. Formula (1) above is in $L_{\omega_1\omega}$ and formula (2) is in $L_{\omega_1\omega_1}$.

The first to give a completely formal treatment of infinitary logic was KARP[15]. KEISLER[16] gives an extensive survey of the model theory of $L_{\omega_1\omega}$, while DICKMANN[7] treats the model theory of larger infinitary languages. Brief introductory accounts are given by SCOTT[22] and by KEISLER[17]. We will here only be needing the most basic results of infinitary logic, roughly to the extent of the last two references.

The logic $L_{\omega_1\omega}$ is the most useful one of the infinitary logics. The formulas of this logic are constructed like the formulas of first-order logic, with the addition of the following formation rule:

If Φ is a countable set of formulas of $L_{\omega_1\omega}$, then $\&\Phi$ and $V\Phi$ are also formulas of $L_{\omega_1\omega}$

($\&\Phi$ is the conjunction and $V\Phi$ the disjunction of the formulas in Φ). If Φ is given in the form $\Phi = \{\phi_i | i=0,1,2,...\}$, then we write these as

$$\mathop{\&}_{i=0}^{\infty} \phi_i \quad \text{and} \quad \mathop{V}_{i=0}^{\infty} \phi_i .$$

Formula (1) above is thus

$$\forall x(\mathop{\&}_{i=0}^{\infty} x=i)$$

The order of precedence in formulas is assumed to be $\equiv$, $=>$, V, &, $-$, $\forall$ and $\exists$, with $\equiv$ binding weakest and $\exists$ strongest.

The usual model-theoretic notions generalize in a straightforward way to infinitary logic (see e.g. DICKMANN[7] for definitions). We write $M \models A[\sigma]$ when the formula A is satisfied in the structure M for the value assignment σ. A value assignment σ is simply a function $\sigma:Var \to |M|$, where Var is the set of variable symbols and $|M|$ is the domain of M. We write $M \models A$ when A holds in M and $\Phi \models A$ when A holds in every model of Φ (A is a logical consequence of Φ).

$L_{\omega_1\omega}$ is axiomatized by giving the usual axioms and inference rules for first-order logic, together with axioms and inference rules for handling the infinite conjunctions and disjunctions. Thus, for Ψ a countable set of formulas and A a formula in Ψ, we add the axiom schemes

$$\&\Psi => A \quad \text{and} \quad A => V\Psi,$$

We also add the following two rules of inference:

$$\frac{B \Rightarrow A,\ \text{for each } A \in \Psi}{B \Rightarrow \& \Psi} \qquad \frac{A \Rightarrow B,\ \text{for each } A \in \Psi}{\vee \Psi \Rightarrow B}$$

These rules may require an infinite number of premises to be proved. To make them useful, proofs in $L_{\omega_1\omega}$ are allowed to be of infinite (but at most countable) length. As usual, we write $\Phi \vdash A$ when A is provable from the set of formulas Φ.

The logic $L_{\omega_1\omega}$ is similar to first-order logic in that it is complete, in the following sense:

COMPLETENESS THEOREM(KARP[15]): Let A be a formula and let Φ be a countable set of sentences of $L_{\omega_1\omega}$. Then

$$\Phi \models A \text{ if and only if } \Phi \vdash A.$$

3. EXPRESSING WEAKEST PRECONDITIONS

Let L be a set of constant, function and predicate symbols and let M be a structure for L. We will be interested in computations on M, where a computation is a (possibly infinite) sequence of <u>states</u> in M, a state simply being a value assignment in M. Let us denote by Σ_M the set of all states in M. A computation will then be an element in Σ_M^* or in Σ_M^ω, where Σ_M^* is the set of all finite and Σ_M^ω the set of all infinite sequences of states in Σ_M.

Let $Stat_L$ be a set of <u>programs</u> (or <u>statements</u>) of L, i.e. programs which only use constant, function and predicate symbols in L. Let S be a program in $Stat_L$. The <u>interpretation</u> of S in M, denoted S^M, will be a function $S^M:\Sigma_M \to P(\Sigma_M^* \cup \Sigma_M^\omega)$, i.e. S^M assigns to each state in Σ_M a set of computations in M (the notation P(X) is used for the power set of X).

For any initial state $\sigma \in \Sigma_M$, $S^M(\sigma)$ is the set of all possible computations of S starting from the initial state σ. There may be more than one computation of S starting from σ, as the programs in Stat_L are allowed to be nondeterministic.

For any computation c in Σ_M^*, let lt(c) be the last element in c. For $C \subseteq \Sigma_M^*$, define $\mathrm{lt}(C) = \{\mathrm{lt}(c) \mid c \in C\}$. Let $f:\Sigma_M \rightarrow P(\Sigma_M^* \cup \Sigma_M^\omega)$ and let $\Sigma' \subseteq \Sigma_M$. The <u>weakest precondition</u> of f for Σ', denoted $\mathrm{wp}(f,\Sigma')$, is the set of all σ in Σ_M such that

(i) $f(\sigma) \subseteq \Sigma_M^*$ and
(ii) $\mathrm{lt}(f(\sigma)) \subseteq \Sigma'$.

Let R be a formula of $L_{\alpha\beta}$ and let M be a structure for L. The <u>interpretation</u> of R in M is the set $R^M = \{\sigma \in \Sigma_M \mid M \models R[\sigma]\}$. Given a program S in Stat_L, $\mathrm{wp}(S_M,R_M)$ will then be the set of all initial states σ in Σ_M for which the execution of S is guaranteed to terminate in a final state satisfying the condition R. Thus $\mathrm{wp}(S_M,R_M)$ gives the semantical meaning of Dijkstra's weakest preconditions for programs.

DEFINITION: Let S be a statement in Stat_L and let W and R be formulas of $L_{\alpha\beta}$. W is said to <u>express</u> (uniformly for all structures) the weakest precondition of S for R, if $W^M = \mathrm{wp}(S^M,R^M)$ for any structure M of L.

We will say that the weakest preconditions of programs in Stat_L are <u>expressible</u> in $L_{\alpha\beta}$ (or, more briefly, that $L_{\alpha\beta}$ is expressible for Stat_L), if for any S in Stat_L and any R in $L_{\alpha\beta}$ there is a formula W in $L_{\alpha\beta}$ which expresses the weakest precondition of S for R. In Harel's classification [12], the definition of weakest preconditions given here assumes a depth-first execution of nondeterministic programs without any backtracking. Other execution strategies are also possible, but, as shown by Harel, this one is assumed by Dijkstra.

The importance of the weakest preconditions stems from the fact that they can be used to express total correctness of programs. Given a program S in $Stat_L$, a precondition P and a postcondition Q in $L_{\alpha\beta}$, S is totally correct with respect to P and Q in a structure M of L, denoted $M \models P[S]Q$, if $P^M \subseteq wp(S^M, Q^M)$. Termination of a program S is again a special case of total correctness, i.e. S is guaranteed to terminate for precondition P if $M \models P[S]true$, where true is an identically true sentence (i.e. $true^M = |M|$ for any M). We write $\Phi \models P[S]Q$, when $M \models P[S]Q$ holds for any model M of Φ. If the logic $L_{\alpha\beta}$ is expressible for $Stat_L$, then proving $\Phi \models P[S]Q$ can be reduced to proving $\Phi \models P \Rightarrow W$ for some formula W of $L_{\alpha\beta}$ which expresses the weakest precondition of S for Q. We will write $\Phi \vdash P[S]Q$ when $\Phi \vdash P \Rightarrow W$ for some such formula W.

We will be interested in the weakest logic $L_{\alpha\beta}$ which is expressible for a specific choice of $Stat_L$. This amounts to asking for the weakest infinitary logic in which one can reason about total correctness of programs in the manner described above.

4. WEAKEST PRECONDITIONS OF GUARDED COMMANDS

Let us first consider the case when $Stat_L$ is taken to be the guarded commands of DIJKSTRA[8]. That is, the statements of $Stat_L$ are defined by

$$\begin{aligned} S ::= {} & skip \mid abort \mid x := t \mid S_1; S_2 \mid \\ & if\ b_1 \rightarrow S_1\ [\,]\ \dots\ [\,]\ b_n \rightarrow S_n\ fi \mid \\ & do\ b_1 \rightarrow S_1\ [\,]\ \dots\ [\,]\ b_n \rightarrow S_n\ od \quad , \end{aligned}$$

$n \geq 1$. Here x is a list of distinct variables, t is a list of terms of L (x and t must be of equal length), $b_1, \dots, b_n$ are boolean expressions (quantifier free formulas) of L and S, $S_1, \dots, S_n$ are statements of L.

The interpretation S^M of a guarded command S in a structure M of L can be defined in a straightforward way and will not be given here. Because the guards in the conditional and iteration statements are not required to be mutually exclusive, the guarded commands may be nondeterministic, i.e. $S^M(\sigma)$ may contain more than one computation.

We will now show that the logic $L_{\omega_1\omega}$ is the weakest infinitary logic $L_{\alpha\beta}$ which is expressible for the guarded commands. As a first step, we show that the weakest preconditions of guarded commands cannot be expressed in ordinary first-order logic $L_{\omega\omega}$.

PROPOSITION 1. $L_{\omega\omega}$ is not expressible for the guarded commands.

Proof: Let L be the language of groups, i.e. L consists of the nonlogical symbols 1 (unit element), $^{-1}$ (inverse) and $\times$ (multiplication). Let Φ be the set of group axioms (a finite set of first-order sentences, see e.g. BARWISE[4]). Let S in Stat_L be the statement

```
x:= y;
do x ≠ 1 → x:= x×y od.
```

If $L_{\omega\omega}$ would be expressible, then there would be a first-order sentence W expressing the weakest precondition for this program to terminate . The set of first-order formulas $\Phi \cup \{W\}$ would then characterize the class of <u>torsion groups</u>, i.e. those groups in which for any element a in the group, $a^n = 1$ for some $n \geq 1$. However, it is known that this class cannot be characterized by a finite set of first-order sentences [4]. Thus W cannot be a formula of $L_{\omega\omega}$.[]

MANNA[18] has shown that weakest preconditions of nondeterministic programs can be expressed in first-order logic, if one is allowed to use predicate variables. Our definition of expressibility prohibits the use of predicate variables, so this result is not in conflict with Manna's result. The use of predicate variables makes Manna's formulation of the weakest preconditions complicated and difficult to use in reasoning about program correctness, as compared to Dijkstra's formulation.

In order to simplify notation, let us introduce the following abbreviations for the iteration and the conditional statement: DO denotes the iteration statement

$$\text{do } b_1 \rightarrow S_1 \text{ [] } \ldots \text{ [] } b_k \rightarrow S_k \text{ od},$$

IF denotes the corresponding conditional statement

$$\text{if } b_1 \rightarrow S_1 \text{ [] } \ldots \text{ [] } b_k \rightarrow S_k \text{ fi}.$$

and bb denotes the condition $b_1 \vee \ldots \vee b_k$.

PROPOSITION 2. $L_{\omega_1 \omega}$ is expressible for the guarded commands.

Proof: DIJKSTRA[8] defines for each guarded command S and formula Q a formula WP(S,Q) and shows that this formula expresses the weakest precondition of S for Q. If Q is a formula of $L_{\omega_1 \omega}$, then WP(S,Q) is easily seen to be a formula of $L_{\omega_1 \omega}$. Actually it is only necessary to change the definition of WP(S,Q) in the case when S is an iteration statement. Dijkstra defines WP(DO,R) by

$$WP(DO,R) = \exists n(n \geq 0).H_n$$

where H_0, H_1, H_2, ... is a sequence of formulas defined in terms of IF. If we write this as

$$WP(DO,R) = \bigvee_{n=0}^{\infty} H_n,$$

we get a formula of $L_{\omega_1 \omega}$. []

The completeness of $L_{\omega_1 \omega}$ gives us the following result.

PROPOSITION 3. Let Φ be a countable set of sentences of $L_{\omega_1 \omega}$ and S a guarded command in $Stat_L$. Then

$$\Phi \models P[S]Q \text{ if and only if } \Phi \vdash P \Rightarrow WP(S,Q),$$

for any formulas P and Q of $L_{\omega_1\omega}$.

This means that the formalization of the weakest precondition in $L_{\omega_1\omega}$ provides a complete technique for proving total correctness of guarded commands, in the sense that if a guarded command is totally correct in a theory Φ, the corresponding formula is provable from Φ, and vice versa.

The weakest precondition of the iteration statement can be expressed in an alternative way as

$$WP(DO,R) = \bigvee_{n=0}^{\infty} WP(DO^n,R)$$

where DO^0, DO^1, DO^2, ... is a sequence of approximations of DO, defined by

$$DO^0 = \text{abort}$$

and for $n \geq 0$,

```
DO^{n+1} =  if bb → IF;DO^n
            []¬bb → skip
            fi.
```

This definition can be extended to parameterless recursive procedures as follows. Consider the procedure declaration

```
procedure p; S[p]
```

where S[p] indicates that the body S of p contains possible recursive calls on p itself. The weakest precondition of the call p is easily seen to be

$$WP(p,R) = \bigvee_{n=0}^{\infty} WP(S^n,R),$$

where S^n is defined by

$$S^0 = \text{abort}$$

and for $n \geq 0$,

$$S^{n+1} = S[S^n/p].$$

Here $S[S^n/p]$ denotes the result of substituting S^n for each call p in S. (This definition of the weakest precondition for parameterless recursive procedures is essentially due to HEHNER[13].)

5. REASONING IN INFINITARY LOGIC

The logic $L_{\omega_1\omega}$ is an essentially stronger logic than $L_{\omega\omega}$. Thus one can give a categorical characterization of the standard model of arithmetic by a single sentence of this logic. It is sufficient to take the sentence ϕ which is the conjunction of all first-order instances of the Peano axioms, together with the sentence

$$\forall x[\bigvee_{n=0}^{\infty} (x=s^n0)]$$

Here s is the successor function and s^nx is defined by $s^0x = x$ and $s^{n+1}x = s(s^nx)$, for $n = 0,1,2,\ldots$. This construction can in fact be generalized, in that it is possible to characterize in $L_{\omega_1\omega}$ the isomorphism type of any countable algebra of a given signature (Scott's isomorphism theorem, see e.g. KEISLER[16]).

Let L be the language and N the standard model of first-order arithmetic. Let S be a statement in Stat_L. Then the completeness result of the preceding section gives that

$$N \models P[S]Q \text{ if and only if } \phi \vdash P[S]Q,$$

for any P and Q in $L_{\omega_1\omega}$. In other words, S is totally correct in the standard model of arithmetic if and only if it can be proved to be correct from the axiom ϕ. The trade-off between using $L_{\omega_1\omega}$ and $L_{\omega\omega}$ should be evident here. On the one hand we get rid of the nonstandard models of arithmetic (which usually undermine the faithfulness of first-order axioms) if we use $L_{\omega_1\omega}$, on the other hand we also loose the finitary nature of proofs in $L_{\omega\omega}$.

DIJKSTRA[8, ch. 4 and 9] gives five basic properties of weakest preconditions for guarded commands. These properties are all valid, i.e. they will hold in any structure of L. By the completeness of $L_{\omega_1\omega}$, this means that they are all theorems in $L_{\omega_1\omega}$, and may thus be used in proofs in this logic. The first property e.g.,

$$WP(S,\text{false}) \equiv \text{false},$$

asserts that in any structure M of L, $wp(S^M,\emptyset) = \emptyset$, a fact which is easily seen to be true by the definition of wp (false is an identically false sentence, i.e. $\text{false}^M = \emptyset$ for any structure M). Similarly for the other four properties.

The fifth property, <u>continuity</u>, can be formulated as follows: Let C_0, C_1, C_2, ... be formulas of $L_{\omega_1\omega}$ and let C be the set

$$C = \{C_i \Rightarrow C_{i+1} \mid i = 0,1,2, \ldots \}.$$

Then

$$C \models WP(S, \bigvee_{i=0}^{\infty} C_i) \equiv \bigvee_{i=0}^{\infty} WP(S,C_i).$$

The continuity property rests on the assumption of <u>bounded nondeterminism</u>. The nondeterminism of a statement S in $Stat_L$ is bounded in a structure M of L, if for any $\sigma \in \Sigma_M$,

$$S^M(\sigma) \subseteq \Sigma_M^* \Rightarrow S^M(\sigma) \text{ is finite.}$$

The nondeterminism of the guarded commands is easily seen to be bounded in any structure M, so the continuity property holds for the weakest preconditions of guarded commands.

Weakest precondition are used by Dijkstra both as a tool for developing programs and as a framework in which to establish the soundness of more practical proof techniques for program correctness. The formalization of weakest preconditions in infinitary logic, as described here, is primarily intended to support this second goal. As an example of this, we show how to prove a theorem which establishes the correctness of the invariant assertion technique for proving partial correctness of loops. A proof of this theorem is given by DIJKSTRA[8], who refers to it as the "fundamental invariance theorem". Our purpose here is to show how Dijkstra's proof is translated into a proof in $L_{\omega_1\omega}$.

The theorem to be proved is the following. Let DO be the iteration statement, as defined in the previous section, and let IF be the corresponding conditional statement. The fundamental invariance theorem states that

$$\{P \; \& \; bb \Rightarrow WP(IF,P)\} \vdash$$

$$P \; \& \; WP(DO,true) \Rightarrow WP(DO, P \; \& \; \neg \, bb).$$

First one has to show that

$$P \; \& \; WP(DO^n,true) \Rightarrow WP(DO^n,P \; \& \; \neg bb)$$

is provable under the given hypothesis, for $n \geq 0$. This part of the proof requires no infinitary reasoning and is therefore omitted here. We may then infer that

$$P \; \& \; WP(DO^n,true) \Rightarrow \bigvee_{i=0}^{\infty} WP(DO^i,P \; \& \; \neg bb)$$

for $n \geq 0$, by the axiom of infinite disjunction and the transitivity of implication. This is again equivalent to

$$P \Rightarrow [WP(DO^n,true) \Rightarrow \bigvee_{i=0}^{\infty} WP(DO^i,P \; \& \; \neg bb)].$$

Let us assume P. By modus ponens, we then have that

$$WP(DO^n,true) \Rightarrow \bigvee_{i=0}^{\infty} WP(DO^i,P \; \& \; \neg bb)$$

for $n \geq 0$. We may now use the inference rule for infinite disjunctions, giving

$$\bigvee_{i=0}^{\infty} WP(DO^i,true) \Rightarrow \bigvee_{i=0}^{\infty} WP(DO^i,P \;\&\; \neg bb)$$

Applying the deduction theorem (which holds in $L_{\omega_1\omega}$), we then get

$$P \Rightarrow [\bigvee_{i=0}^{\infty} WP(DO^i,true) \Rightarrow \bigvee_{i=0}^{\infty} WP(DO^i,P \;\&\; \neg bb)].$$

Using the definition of WP for loops, this is finally equivalent to

$$P \;\&\; WP(DO,true) \Rightarrow WP(DO, P \;\&\; \neg\, bb),$$

which is the required result.

6. NONDETERMINISTIC ASSIGNMENT STATEMENTS

Let us now replace the assignment statement x:=t in $Stat_L$ by a more general construct, called a nondeterministic assignment statement. This has the form

$$x{:=}\ y.Q,$$

where x and y are lists of distinct variables (of equal length) and Q is a formula of $L_{\omega\omega}$. The effect of this statement is to assign to x some new value y such that the condition Q is satisfied (Q will usually contain free occurrences of the variables in x and y). If no such y exists, then the effect of the statement is considered to be undefined. If there is more than one possible choice of y making Q true, one of these is chosen nondeterministically and assigned to x. (The nondeterministic assignment statement can be seen as a more sophisticated and usable version of the "random assignment" in HAREL[12]).

The assignment statement can be expressed by this new construct. The effect of

$$x_1,\ldots,x_n := t_1,\ldots,t_n$$

is the same as the effect of

$$x_1,\ldots,x_n := y_1,\ldots,y_n.(y_1 = t_1 \ \& \ \ldots \ \& \ y_n = t_n)$$

Thus e.g. $u := u+1$ is equivalent to $u := v.(v = u+1)$.

The weakest precondition for the nondeterministic assignment statement is given by

$$WP(x:=y.Q,\ R) = \ \exists y.Q \ \& \ \forall y.(Q \Rightarrow R[y/x]).$$

The first conjunct expresses the requirement that there must exist an y satisfying the condition Q, otherwise the effect is undefined. The second conjunct expresses the requirement that any choice of y satisfying Q must result in a new state in which R holds. The second conjunct here is very similar to the corresponding conjunct occurring in the rule of adaptation in HOARE[14].

Let us exemplify this rule by computing the weakest precondition of the assignment statement. We have

$$\begin{aligned} WP(x:=t,\ R) &= WP(x:=y.(y=t),\ R) \\ &= \exists y.(y=t) \ \& \ \forall y.(y=t \Rightarrow R[y/x]) \\ &\equiv \text{true} \ \& \ (R[y/x])[t/y] \\ &\equiv R[t/x], \end{aligned}$$

as was to be expected.

The purpose of the nondeterministic assignment statement is to extend the weakest precondition technique to hierarchically structured programs. Consider a statement S containing a call on the parameterless procedure p. Assume that only the entry condition P and the exit condition Q for p is known, together with the fact that p only can change the variables x. This information should be sufficient to enable one to prove the correctness of the statement S, i.e. knowledge of how p is actually implemented should not be required.

We can achieve this by replacing each call on p in S by the nondeterministic assignment statement

$$x := y.(P \& Q),$$

where we assume that x in P and Q refer to the initial value of x, while y in Q refers to the value of x after the call. We assume here that the pre- and postconditions are consistent, i.e. that $P \Rightarrow \exists y.Q$. The correctness of S with respect to some given specification can then be established, as we know how to compute the weakest preconditions of S.

The correctness of an implementation S´ of p can then be proved as a separate step, by proving that $P \Rightarrow WP(S´,Q)$, at the same time checking that only variables in x are updated in S´ (this can be guaranteed by syntactic restrictions on S´). In general, S´ will be a correct implementation of $x := y.Q$, if

$$\exists y.Q \Rightarrow (WP(S´, Q[z/x,x/y]))[x/z]$$

holds, where z is a list of fresh variables, not used in S´ or Q before. A more thorough discussion of correctness of implementations along these lines is presented in BACK[1].

7. STRONG AND WEAK TERMINATION

The previous discussion should be sufficient to indicate that the nondeterministic assignment statement would be very convenient to have, allowing correctness proofs of hierarchical programs (e.g. developed by stepwise refinement) using the weakest precondition technique. However, simply extending the guarded commands with a construct like this does not work, as observed by DIJKSTRA[8]. To see this, consider the following statement S:

```
do x ≠ 0 → if x > 0 → x:= x-1            (1)
           [] x < 0 → x:= y.(y > 0)
           fi
od.
```

We assume that this program works on the standard model of integers.

Computing the weakest precondition for S to terminate gives

$$WP(S,true) = x \geq 0,$$

i.e. the loop is only guaranteed to terminate for non-negative initial values of x. On the other hand, any possible execution of S for negative initial values of x must obviously also terminate, so we would expect WP(S,true) = true.

The problem here is that the weakest precondition for the iteration statement, as it is defined in section 4, formalizes a stronger notion of termination of loops than the ordinary one, which requires that any possible execution of the loop must eventually terminate. A loop is said to <u>terminate strongly</u> if for any initial state σ there is an integer N_σ such that the loop is guaranteed to terminate in less than N_σ iterations. Termination which is not strong is called <u>weak termination</u>. (The notion of strong and weak termination is due to DIJKSTRA[9]). The nondeterministic assignment statement in S has the effect that no upper bound can be given for the number of iterations required for the loop to terminate when x is initially negative, although the loop is guaranteed to terminate for such initial values also.

Without nondeterministic assignment statements, termination of guarded commands is always strong. The reason for this is that the nondeterministic choices which arise during execution of a guarded command are always made from a finite number of alternatives. Thus, if each execution of such a command terminates for a given initial state, there can only be a finite number of possible different executions (this follows by König´s lemma), and consequently there must be an execution requiring the greatest number of iterations. If we allow nondeterministic assignment statements in guarded commands, the nondeterministic choice can be made from an infinite number of alternatives. König´s lemma does not then apply any more and the existence of a maximum number of iterations is therefore not guaranteed.

Let us refer to the guarded commands of L in which nondeterministic assignment statements are allowed, together with arbitrary first-order formulas of L as guards, as <u>extended guarded commands</u> of L. We then have the following result.

PROPOSITION 4. $L_{\omega_1\omega}$ is not expressible for the extended guarded commands of L.

Proof: Choose L = {P}, where P is a binary predicate symbol and let S be the extended guarded command

$$\text{do } \exists y.P(x,y) \rightarrow x:= y.P(x,y) \text{ od.} \qquad (2)$$

Let M be a structure for L and let $\sigma \in \Sigma_M$. S will be guaranteed to terminate in M for the initial state σ if and only if there does not exist an infinite sequence $d_0, d_1, d_2, \ldots$ of elements in $|M|$ such that $\sigma(x) = d_0$ and

$$d_0 > d_1 > d_2 > \ldots$$

where $>$ is the interpretation of the predicate P in the structure M. If there was a formula W in $L_{\omega_1\omega}$ expressing the weakest precondition of S to terminate, then the formula $\forall x.W$ would also be a formula of $L_{\omega_1\omega}$, and would hold in M if and only if $>$ is a well-founded relation in M. Thus the formula $\forall x.W$ would characterize well-foundedness. However, well-foundedness cannot be characterized by a formula of $L_{\omega_1\omega}$ (see e.g. KEISLER[16]). Consequently, no such formula W exists in $L_{\omega_1\omega}$, i.e. this logic is not expressible for the extended guarded commands. []

In fact, well-foundedness cannot be expressed in $L_{\alpha\omega}$ for any infinite cardinal α, nor can it be expressed in the logic $L_{\infty\omega}$, which is the union of all these logics $L_{\alpha\omega}$. Thus none of these logics is expressible for the extended guarded commands. (If one would allow disjunctions over the class of <u>all</u> ordinals, then the weakest precondition of loops with nondeterministic assignments could be expressed in a manner similar to the one Dijkstra uses, as shown by BOOM[5]. Such a formula is not, however, a formula of any infinitary logic $L_{\alpha\beta}$).

It should be remarked that the use of arbitrary first-order formulas as guards in the extended guarded commands is not essential. If we allow boolean values in our programs, then program (2) above can also be expressed in the form

```
set b;
do b → x:= y.P(x,y); set b od
```

where "set b" is the statement

```
b:= c.(∃y.P(x,y) & c=true  v
      ¬∃y.P(x,y) & c=false)
```

which has the same effect.

Allowing nondeterministic assignment statements in guarded commands also affects the proof rule for total correctness of loops given in DIJKSTRA[8]. According to this rule, to prove

$$P \Rightarrow WP(DO, P \,\&\, \neg bb),$$

it is sufficient to show that the following three conditions are satisfied for some suitably chosen integer valued function t on the program variables:

(1) $P \,\&\, bb \Rightarrow WP(IF,P)$,
(2) $P \,\&\, bb \Rightarrow t > 0$ and
(3) $P \,\&\, bb \,\&\, t \leq t_0+1 \Rightarrow WP(IF, t \leq t_0)$.

The first condition guarantees that P is preserved by the body of the loop, the second condition guarantees that the value of t is bounded from below, while the third condition guarantees that each iteration of the loop decreases the value of t with at least one.

A suitable integer function t can always be found for a loop containing no nondeterministic assignment statements, provided the loop does in fact terminate. It is sufficient to choose t such that in any initial state σ for which the loop is guaranteed to terminate, $t(\sigma)$ is the maximum number of iterations required for termination. This choice of t is easily seen to satisfy both condition (2) and (3).

If we allow nondeterministic assignment statements within a loop, the existence of a suitable integer function t is not guaranteed any more. Program (1) above provides an example of this. Assume that there is an integer valued function t on the program variables which satisfies condition (2) and (3). Suppose $t(x_0) = n$ for some $x_0 < 0$, and the program, given the input x_0, happens to compute $x := m$ with $m > n$. Then there are m further iterations, in each of which t is decreased, so $m \leq n$ must hold. This is a contradiction, hence there can be no integer function t which satisfies both (2) and (3).

The unbounded nondeterminism of the nondeterministic assignment statement also causes problems with the semantics of programs. In BACK[2] it is shown that the simple Egli-Milner ordering on the power set domain is not sufficient for defining the denotational semantics of such programs, but that one is forced into a much more complicated, essentially operational semantics.

The fact that $L_{\omega_1\omega}$ is not expressible for the extended guarded commands puts us into something of a dilemma. On the one hand the logic $L_{\omega_1\omega}$ is a very convenient one to reason in, on the other hand we also would like to be able to prove the total correctness of programs in a hierarchical fashion. There are essentially two different ways in which this dilemma can be resolved. We can either try to use a logic more powerful than $L_{\omega_1\omega}$, or we can restrict the power of the nondeterministic assignment statement in a way which restores the expressiveness of $L_{\omega_1\omega}$. We will consider both possibilities in turn, the first one in the next section and the second one in the section after.

8. STRENGTHENING THE LOGIC

As shown in the preceding section, none of the logics $L_{\alpha\omega}$ is expressible for the extended guarded commands, because well-foundedness cannot be characterized in them. The logic $L_{\omega_1\omega_1}$ is again essentially stronger than these logics, in that well-foundedness of a binary predicate P can be characterized in it, by the formula

$$\neg\exists x_0 x_1 x_2 \ldots (\mathop{\&}_{i=0}^{\infty} P(x_i, x_{i+1})).$$

The proof of proposition 4 does therefore not apply to this logic. In fact, we can show that this logic is expressible for the extended guarded commands.

To show the expressibility of $L_{\omega_1\omega_1}$, it is sufficient to show that the weakest precondition for the iteration statement DO is expressible in $L_{\omega_1\omega_1}$. The construction of a formula WP(DO,R) which expresses the weakest precondition of DO for the formula R in $L_{\omega_1\omega_1}$ proceeds as follows.

Assume that we already know how to express WP(IF,P) for any P. Consider the formula $\neg$ WP(IF,$\neg$ P). For a given structure M and a given state $\sigma \in \Sigma_M$,

$$M \models \neg\, WP(IF, \neg\, P)[\sigma]$$

if and only if there is a nonterminating computation of IF in M starting from σ, or there is a terminating computation which ends in a final state σ' satisfying P. In other words, if IF is guaranteed to terminate for initial state σ, then one of the possible final states must satisfy P.

Let now x be the list of all variables occurring in DO. Define the formula

$$K(x,y) = \neg\, WP(IF,\ x \neq y),$$

which says that if IF is guaranteed to terminate, then y is one of the possible final values of x. We then define a sequence H_0, H_1, H_2, ... of formulas by

$$H_0 = \text{true}$$

and

$$H_{n+1} = H_n \ \&\ bb(x^n) \ \&\ K(x^n, x^{n+1}),$$

for $n = 0,1,2,\ldots$.

Let $A_1(x^0)$ be the formula

$$\forall x_1 x_2 x_3 \ldots [\mathop{\&}_{n=0}^{\infty} (H_n \;\&\; bb(x^n) \Rightarrow T(x^n))],$$

where T(x) = WP(IF,true). $A_1(x^0)$ will be true in a structure if and only if execution of DO from initial state x^0 cannot lead to nontermination of IF after a finite number of iterations. Let $A_2(x^0)$ be the formula

$$\forall x^1 x^2 x^3 \ldots [\mathop{\&}_{n=0}^{\infty} (H_n \;\&\; -bb(x^n) \Rightarrow R(x^n))].$$

This again says that if an execution of DO ever terminates, R will hold for the final state. Finally, define $A_3(x^0)$ to be the formula

$$\neg \exists x^1 x^2 x^3 \ldots [\mathop{\&}_{n=0}^{\infty} H_n].$$

This formula says that it is not possible to have an infinite number of iterations of DO, when initially $x = x^0$. Obviously we now have that

$$WP(DO,R) = A_1(x) \;\&\; A_2(x) \;\&\; A_3(x).$$

This gives us the following result:

PROPOSITION 5. $L_{\omega_1\omega_1}$ is expressible for the extended guarded commands.

The result of this section thus shows that admitting unrestricted nondeterministic assignment statements forces us into a stronger logic, with a resulting essentially operational definition of the weakest precondition for loops.

9. FINITE ASSIGNMENT STATEMENTS

We now consider the other possibility left open, that of restricting the power of the nondeterministic assignment statement. The problems with weak termination only turn up when the nondeterministic choice is made from an infinite number of different alternatives. An obvious solution is therefore to restrict the nondeterministic assignment to finite choices only.

Consider the assignment x:= y.Q, where x and y are simple variables. Given a specific structure M, the nondeterministic choice in executing this statement for initial state $\sigma \in \Sigma_M$ will be finite, if the set

$$\{d \mid M \models Q[\sigma\langle d/y\rangle]\}$$

is finite, where $\sigma\langle d/y\rangle$ denotes a state which agrees with $\sigma \in \Sigma_M$ on all other variables except on y, where it has value $d \in |M|$. If this is the case for each $\sigma \in \Sigma_M$, then this assignment statement is said to be <u>finite</u> in M.

Let us define the formula

$$\psi(Q,y) = \bigvee_{n=0}^{\infty} \forall y_0 y_1 \ldots y_n [\mathop{\&}_{i=0}^{n} Q[y_i/y] \Rightarrow \bigvee_{0 \leq i < j \leq n} y_i = y_j].$$

For any structure M, $M \models \psi(Q,y)$ if and only if x:= y.Q is finite in M. An extended guarded command S is said to be <u>finitary</u> in Φ, Φ a set of sentences, if each nondeterministic assignment statement x:= y.Q in S is finite in any model of Φ. A sufficient condition which guarantees that S is finitary in Φ is that $\Phi \models \psi(Q,y)$ holds for any assignment x:= y.Q in S.

The notion of expressiveness can be relativized to a set of sentences Φ as follows. We say that the logic $L_{\alpha\beta}$ is expressive in Φ for the set of statements $Stat_L$, if for any R in $L_{\alpha\beta}$ and any S in $Stat_L$ there is a formula W in $L_{\alpha\beta}$ such that $W^M = wp(S^M,R^M)$ for any model M of Φ. If we assume that Φ is an axiomatization of the theory which we are working in, then we can restrict ourselves to extended guarded commands which are finitary in Φ. We then have the following result, as an immediate consequence of the observations above:

PROPOSITION 6. $L_{\omega_1\omega}$ is expressive in Φ for the set of extended guarded commands which are finitary in Φ.

In practice it would be better to have a standard collection of formulas B for which $\psi(B,y)$ is known to hold in the underlying theory Φ. Any nondeterministic assignment statement used would then have to be of the form x:= y.(B & Q), where Q can be any first-order formula of L. This will guarantee that all statements constructed are finitary in Φ.

A special notation might be introduced for such assignments, e.g. writing the above assignment in the form x:= y[B].Q. If we are working with the integers, it would be natural to choose the finite intervals as the finiteness conditions. We would then only allow assignments of the form x:= y[$m\leq y\leq n$].Q(x,y), m and n integers.

10. CONCLUSIONS

We have tried to show that the infinitary logic $L_{\omega_1\omega}$ is a natural one in which to formalize Dijkstra's weakest precondition technique. We have shown that this logic is sufficiently strong when one is interested in proving the total correctness of guarded commands, but that it cannot handle nondeterministic assignment statements. These would permit a hierarchical decomposition of the correctness proofs, and would also be quite handy when developing programs by stepwise refinement. One would therefore like to allow this kind of constructs in guarded commands.

Two solutions were offered to this problem. The first one consisted in going to the essentially stronger logic $L_{\omega_1\omega_1}$, while the other was to restrict the nondeterministic assignment statement so that the nondeterministic choice is always made from a finite set of alternatives. We do not want to take any definite stand on which of these solutions is to be chosen, as this depends on the objectives one tries to achieve by the formalization. The second solution, restricting the power of the nondeterministic assignment statement, fits best into the framework of Dijkstra´s book. It preserves the simplicity of the underlying logic, yet does not restrict the applicability of the nondeterministic assignment statement too much. On the other hand, from a theoretical point of view, the need to restrict oneself to finite choices only seems somewhat artificial, so this would again favor the first solution.

ACKNOWLEDGEMENT: I would like to thank J.W. de Bakker, E.W. Dijkstra, M. Karttunen and D. Park for the very helpful discussions I had with them on the topics treated here during the early stages of this work, and J.V. Tucker for his keen interest and help during the later stages of this work.

REFERENCES

1. BACK,R.J.R.: On the correctness of refinement steps in program development. Mathematical Center Tracts (to appear). Amsterdam: Mathematisch Centrum 1980.

2. BACK,R.J.R.: Semantics of unbounded nondeterminism. In: Proc. 7th Coll. Automata, Languages and Programming (J.W. de Bakker & J. van Leeuwen, eds), Lecture Notes in Computer Science, Vol. 85, pp.51-63. Berlin-Heidelberg-New York:Springer 1980.

3. BANACHOWSKI,L., A. KRECZMAR, G. MIRKOWSKA, H. RASIOWA & A. SALWICKI: An introduction to algorithmic logic; metamathematical investigations in the theory of programs. In: Mathematical Foundations of Computer Science (A. Mazurkiewicz & Z. Pawlak, eds.), Banach Center Publications, Vol. 2, pp. 7-99. Warsaw: PWN-Polish Scientific Publishers 1977.

4. BARWISE,J.: An introduction to first-order logic. In: Handbook of Mathematical Logic (J.Barwise,ed.), pp. 5-46. Amsterdam: North-Holland 1977.

5. BOOM, H.J.: A weaker precondition for loops. Mathematisch Centrum Amsterdam, report IW 104/78, 1978.

6. CONSTABLE,R.L.: On the theory of programming logic. In: 9th ACM Symposium on Theory of Computing, Boulder, Colorado 1977, pp. 269-285. New York: ACM 1977.

7. DICKMANN,M.A.: Large Infinitary Languages. Amsterdam: North-Holland 1975.

8. DIJKSTRA, E.W.: A Discipline of Programming. Engelwood-Cliffs: Prentice-Hall 1976.

9. DIJKSTRA, E.W.: Private communication,1978.

10. ENGELER, E.: Remarks on the theory of geometrical constructions. In: The Syntax and Semantics of Infinitary Languages (J. Barwise, ed.), Lecture Notes in Mathematics 72. Berlin-Heidelberg-New York: Springer 1968.

11. ENGELER, E.: Algorithmic logic. In: Foundations of Computer Science (J.W. de Bakker, ed.), Mathematical Center Tracts 63, pp. 57-85. Amsterdam: Mathematisch Centrum 1975.

12. HAREL, D.: First-Order Dynamic Logic. Lecture Notes in Computer Science 68. Berlin-Heidelberg-New York: Springer 1979.

13. HEHNER,E.: Do considered od: a contribution to the programming calculus. Acta Informatica 11, 287-304 (1979).

14. HOARE, C.A.R.: Procedures and parameters: An axiomatic approach. In: Symposium on Semantics of Algorithmic Languages (E. Engeler, ed.), Lecture Notes in Mathematics, Vol. 188, pp. 102-116. Berlin-Heidelberg-New York: Springer 1971.

15. KARP, C.R.: Languages with Expressions of Infinite Length. Amsterdam: North-Holland 1964.

16. KEISLER, H.J.: Model Theory for Infinitary Logic. Amsterdam: North-Holland 1971.

17. KEISLER,H.J.: Fundamentals of model theory. In: Handbook of Mathematical Logic (J.Barwise,ed.), pp.47-104. Amsterdam: North-Holland 1977.

18. MANNA, Z.: Mathematical Theory of Computing. New York: MGraw-Hill 1974.

19. PRATT, V.R.: Semantic considerations of Floyd-Hoare logic. In: Proc. 17th IEEE Symp. on Foundations of Computer Science, Houston, Texas 1976, pp. 109-121. Long Beach: IEEE 1976.

20. RASIOWA,H.: Algorithmic logic and its extensions, a survey. In: 5th Scandinavian Logic Symposium, Aalborg 1979, pp. 163-174. Aalborg University Press 1979.

21. SALWICKI, A.: Formalized algorithmic languages. Bull. Acad. Polon. Sci., Ser. Math. Vol. 18, 227-232 (1970).

22. SCOTT, D.: Logic with denumerably long formulas and finite strings of quantifiers. In: Symp. on the Theory of Models (J. Addison, L. Henkin & A. Tarski, eds.), pp.329-341. Amsterdam: North-Holland 1965.