Lecture Notes in Computer Science, vol. 85: Automata, Languages and Programming-7th Colloquium; Noordwijkerhout, The Netherlands; 14-18 July, 1980. Ed., J.W. de Bakker and J. van Jeeuwen. Springer-Verlag; Berlin, Heidelberg, New York; 1980. pp. 169-181.

CHARACTERIZING CORRECTNESS PROPERTIES OF PARALLEL PROGRAMS USING FIXPOINTS

E. Allen Emerson
Edmund M. Clarke
Aiken Computation Laboratory
Harvard University
Cambridge, Mass. 02138 USA

1. Introduction

1.1 Background. Dijkstra [DI76] proposes the use of weakest precondition predicate transformers to describe correctness properties. A typical example of such a predicate transformer is the *weakest precondition for total correctness* wp(π,Q) which gives a necessary and sufficient condition on initial states to ensure that program π terminates in a final state satisfying predicate Q. Dijkstra defines the wp predicate transformer on a *do-od* program as the least fixpoint of a simple predicate transformer which is derivable directly from the program text. Basu and Yeh [BY75] and Clarke [CL76] extend Dijkstra's fixpoint characterization of weakest preconditions to arbitrary sequential programs with "regular" control structures. In addition, Clarke argues that soundness and (relative) completeness of a Hoare-style [HO69] axiom system are equivalent to the existence and extremality of fixpoints for appropriate predicate transformers. Related results appear in the work of de Bakker [DE77a] and Park [PA69].

Many important properties of parallel programs can also be described using fixpoints. Flon and Suzuki [FS78] give fixpoint characterizations of certain correctness properties including freedom from deadlock, invariance, absence of logical starvation, and inevitability under pure nondeterministic scheduling. They also give proof rules which allow the construction of a sound and (relatively) complete proof system for any correctness property with an appropriate fixpoint characterization (e.g. as the least fixpoint of a continuous predicate transformer or as the greatest fixpoint of a monotonic transformer). This reduces the task of designing a proof system for a specified correctness property to the problem of giving a fixpoint characterization for the property. Developing the initial fixpoint characterization can still be quite difficult, however, for correctness properties such as fair inevitability.

A valuable attribute of a correctness property is the existence of a "continuous" fixpoint characterization (i.e. one in terms of extremal fixpoints of continuous transformers). Such a characterization is desirable because the extremal fixpoints are defined as the limit of a natural sequence of approximations. The approximations can be useful in applications, particularly in mechanical efforts to develop reliable and efficient parallel programs. Sintzoff and Van Lamsweerde [SV76] use continuous fixpoint characterizations of certain correctness properties to show how a given

----

This work was partially supported by NSF Grant MCS-7908365

program can be transformed into another program with the specified correctness properties. Clarke [CL78] shows how a widening operator [CO76] can be used to synthesize resource invariants from a continuous fixpoint characterization of the set of reachable states in a computation. Continuous fixpoint characterizations are also used by Reif [RE79] for parallel program optimization.

1.2 New results of this paper. We investigate the problem of characterizing correctness properties of parallel programs using fixpoints and, in particular, using "continuous" fixpoints. In this paper, a parallel program is treated as a nondeterministic sequential program at an appropriate level of granularity. A program's semantics is defined in terms of computation trees recording all possible execution sequences for the program starting in a particular state. In order to make precise statements about computation trees, we introduce the language of computation tree formulae (CTF). The advantage of this approach is that many correctness properties for parallel programs have a natural description in terms of computation trees. Thus, with CTF's we can define in a straightforward manner most correctness properties of interest for parallel programs, including invariance, deadlock freedom, absence of actual starvation, and inevitability under fair scheduling assumptions. We also define the language of fixpoint formulae, FPF, and give an effective procedure for translating CTF's into FPF's. A practical consequence of this procedure is that we can derive an FPF characterization of a correctness property from its CTF description in a uniform manner. Since it is often easier to give an operational (CTF) characterization than a fixpoint characterization, this can make the Flon and Suzuki proof rule technique considerably easier to apply. We then give conditions on a correctness property's CTF description which ensure that it has an FPF characterization using only continuous transformers. Finally, we show how our results can be interpreted in a modal logic where the computation trees determine Kripke structures.

One consequence of our findings is that, while inevitability under fair scheduling can be characterized as the least fixpoint of a monotonic noncontinuous predicate transformer, it cannot be characterized in terms of fixpoints of continuous transformers (nor can it be meaningfully characterized as the greatest fixpoint of a monotonic transformer). We also show that inevitability under fair scheduling, over the natural numbers, is not expressible by a formula of 1st order arithmetic (see also [CH78]). These facts strongly suggest that it is impossible to formulate a useful, sound and (relatively) complete proof system for this correctness property.

1.3 Outline of paper. The paper is organized as follows: Section 2 gives preliminary information about the model of computation and the lattice of total predicates. Sections 3 and 4 informally discuss the syntax and semantics of computation tree formulae (CTF) and fixpoint formulae (FPF), respectively. Section 5 describes the main results on the existence of fixpoint characterizations for parallel programs. Section 6 discusses the relationship with modal logic. Finally, Section 7 presents some concluding remarks and suggests some remaining open questions.

2. Model of Computation.

We represent parallel programs as nondeterministic sequential programs using Dijkstra's do-od construct:

$$\underline{\text{do}}\ B_1 \rightarrow A_1\ [\,]\ B_2 \rightarrow A_2\ [\,]\ \ldots\ [\,]\ B_k \rightarrow A_k\ \underline{\text{od}}.$$

Let Σ be the set of program states (for this paper, we can assume that $\Sigma = \omega$, the set of natural numbers). Each guard B_i is a total recursive predicate on Σ. Each action A_i is a total recursive function from Σ to Σ. The pair $B_i \rightarrow A_i$ is called a command. Intuitively, we may describe the operation of the do-od construct as follows: repeatedly perform the body of the do-od loop. On each trip through the loop, nondeterministically select a command whose guard B_i evaluates to True and execute the corresponding action A_i. If all guards B_i evaluate to False, execution of the loop halts.

Given a state σ in Σ and a do-od program π we define the computation tree $\mathscr{T}(\pi,\sigma)$. Each node of the tree is labelled with the state it represents, and each arc out of a node is labelled with the guard indicating which nondeterministic choice is taken, i.e., which command having a true guard is executed next. The root is labelled with the start state σ. Thus, a path from the root through the tree represents a possible computation sequence of program π starting in state σ. A fullpath of $\mathscr{T}(\pi,\sigma)$ is a path which starts at the root and which is not a proper "subpath" of any other path. Any infinite path starting at the root is a fullpath. A finite path is a fullpath only when its last node is labelled with a state in which all guards are false. A segment of $\mathscr{T}(\pi,\sigma)$ is a (finite or infinite) contiguous initial portion of a fullpath. Similar uses of computation trees appear in the work of de Bakker [DE77b], Meyer and Winklmann [MW79], and Flon and Suzuki [FS78].

We now describe how to transform cobegin-coend programs with conditional critical regions into do-od programs. We let the underlying domain of states Σ be the set of all tuples of the form

$$(pc_1,\ldots,pc_n,v_1,\ldots,v_m)$$

where $pc_1,\ldots,pc_n$ are explicit location counters and $v_1,\ldots,v_m$ are all the variables which appear in the cobegin-coend program. The transformation is essentially the same as that used by Flon and Suzuki [FS78] and will only be illustrated by example:

```
x:=0;
cobegin
   repeat produce; when true do x:=x+1 end forever
   //
   repeat when x > 0 do x:=x-1 end; consume forever
coend
```

is transformed into

```
pc1:=pc2:=0; x:=0;
do
    pc1 = 0          -> produce; pc1:=(pc1+1) mod 2 []
    pc1 = 1 and TRUE -> x:=x+1; pc1:=(pc1+1) mod 2 []
    pc2 = 0 and x > 0 -> x:=x-1;pc2:=(pc2+1) mod 2 []
    pc2 = 1          -> consume; pc2:=(pc2+1) mod 2
od
```

Finally, we use PRED(Σ) to denote the lattice of total predicates where each predicate is identified with the set of states in Σ which make it true and the ordering is set inclusion.

2.1 Definition

Let τ: PRED(Σ) -> PRED(Σ) be given; then

(1) τ is monotonic provided that $P \subseteq Q$ implies $\tau[P] \subseteq \tau[Q]$

(2) τ is $\cup$-continuous provided that $P_1 \subseteq P_2 \subseteq \ldots$ implies $\tau[\bigcup_i P_i] = \bigcup_i \tau[P_i]$

(3) τ is $\cap$-continous provided that $P_1 \supseteq P_2 \supseteq \ldots$ implies $\tau[\bigcap_i P_i] = \bigcap_i \tau[P_i]$. []

A monotonic functional τ on PRED(Σ) always has both a least fixpoint, lfpX.τ[X], and a greatest fixpoint, gfpX.τ[X] (see Tarski [TA55]): lfpX.τ[X] = $\cap$\{X:τ[X]=X\} whenever τ is monotonic, and lfpX.τ[X] = $\bigcup_i \tau^i$[False] whenever τ is also $\cup$-continuous; gfpX.τ[X] = $\cap$\{X:τ[X]=X\} whenever τ is monotonic, and gfpX.τ[X] = $\bigcap_i \tau^i$[True] whenever τ is also $\cap$-continuous.

3. Computation Tree Formulae

Given a program π and an initial state σ, a computation tree formula (CTF) makes a statement about the occurence of nodes and arcs satisfying certain correctness predicates in the computation tree $\mathcal{T}(\pi,\sigma)$. When presenting the syntax of CTF's we will use the notation:

$$\begin{bmatrix} \text{choice 1} \\ \cdot \\ \cdot \\ \text{choice n} \end{bmatrix}$$

to indicate that one item may be selected from among n alternatives. A fixed but arbitrary k > 1 is chosen. Our discussion of CTF semantics will assume a fixed interpretation I = (π,<R_i>,σ) consisting of: 1) a do-od program π with k commands over domain of states Σ, 2) an assignment of predicates $R_i \subseteq \Sigma$ to each predicate symbol X_i, and 3) an initial state σ.

A CTF is a boolean combination of predicate symbols, guard symbols, and constructs of the following form:

$$\begin{bmatrix} \exists \\ \forall \end{bmatrix} \begin{bmatrix} \text{fullpath} \\ \text{segment} \end{bmatrix} \quad \langle\text{body}\rangle$$

The body is a boolean combination of one or more terms. Each term makes a statement

about a particular path p of $\mathcal{T}(\pi,\sigma)$ and has the form:

$$\begin{bmatrix} \exists \\ \forall \\ \overset{\infty}{\exists} \\ \overset{\infty}{\forall} \end{bmatrix} \begin{bmatrix} \text{node } \langle\text{CTF}\rangle \\ \text{arc } \langle\text{guard symbol set}\rangle \end{bmatrix}$$

$\exists$ and $\forall$ have their usual meanings. $\overset{\infty}{\exists}$ means "there exist infinitely many" and $\overset{\infty}{\forall}$ means "for all but a finite number". Each predicate symbol is interpreted by one of the predicates R_i from I. Similarly, each guard symbol is interpreted by one of the guards B_i in the program π. A guard symbol set corresponds to a subset $\{B_{i_1},\ldots,B_{i_n}\}$ of the actual guards of π. Examples of CTF's together with their intuitive meanings are given below:

1) " $\forall$fullpath $\exists$ node R_1" which means "for every fullpath p of $\mathcal{T}(\pi,\sigma)$, there is a node v on p labelled with a state satisfying predicate R_1" (this describes the correctness property inevitability of R_1 [under pure nondeterministic scheduling]).

2) "$R_1 \wedge \exists$ segment $\exists$ arc $\{B_1,B_3\}$" which means "σ satisfies predicate R_1 and there is a segment of $\mathcal{T}(\pi,\sigma)$ having an arc labelled with guard B_1 or guard B_3".

3) " $\forall$fullpath ($\overset{\infty}{\exists}$ node B_1 => $\overset{\infty}{\exists}$ arc $\{B_1\}$)" which means "for every fullpath p of $\mathcal{T}(\pi,\sigma)$ if there are infinitely many states along p at which the guard B_1 is true, then there are infinitely many arcs labelled with guard B_1 (i.e., if process 1 is enabled infinitely often, it is executed infinitely often)".

4) " $\exists$fullpath $\exists$ node ($\forall$ fullpath $\overset{\infty}{\forall}$ node R_1)" which means "there exists a fullpath p of $\mathcal{T}(\pi,\sigma)$ and there is a node v on p labelled with state σ' such that for every fullpath q of $\mathcal{T}(\pi,\sigma')$ all but a finite number of nodes on q are labelled with a state satisfying R_1".

Each of the above CTF's will be either true or false depending on the particular interpretation I that is chosen.

CTF's define predicate transformers for correctness properties of parallel programs. Let $e(X_1,\ldots,X_n)$ be a CTF involving predicate symbols $X_1,\ldots,X_n$. Then e defines the mapping e': PROG(Σ) x (PRED(Σ))^n -> PRED(Σ) such that $e'(\pi,R_1,\ldots,R_n)$ = $\{\sigma \in \Sigma : e(X_1,\ldots,X_n)$ is true when interpreted with program π, start state σ, and X_i assigned R_i for $i \in [1:n]$ \}.

4. The Language of Fixpoint Formulae

A fixpoint formula (FPF) is interpreted with respect to a program π (with k commands) and domain Σ. Each FPF is built up from predicates $R_1,R_2,R_3,\ldots$ over Σ, guards $B_1,B_2,B_3,\ldots$ of the program π, and "sub-FPF's" using

(1) the logical connectives ($\wedge,\vee,\sim$),

(2) the weakest preconditions for the actions A_i of π (A_1^{-1}, A_2^{-1},$A_3^{-1}\ldots$), and

(3) the least fixpoint and greatest fixpoint operators (lfp, gfp).

We write $E[X_i]$ to indicate that the predicate R_i in the fixpoint formula E is viewed as a variable ranging over PRED(Σ). $E[X_i]$ defines a mapping E': PRED(Σ) -> PRED(Σ) in the obvious way; for example, $(\sim B_1 \wedge R_2) \vee (B_3 \wedge A_3^{-1}X_1)$ sends each predicate $R \subseteq \Sigma$ to $((\Sigma \setminus B_1) \cap R_2) \cup (B_3 \cap wp(A_3,R))$. We use $lfpX_i.E[X_i]$ and $gfpX_i.E[X_i]$ to denote the least fixpoint and the greatest fixpoint, respectively, of E' where $E[X_i]$ is required to be (formally) monotonic; i.e. all occurrences of X_i in E must be in an even number of distinct negated "sub-FPF's". Examples of FPF's are

$$gfpX_1.[(B_1 \wedge R_2) \vee (B_3 \wedge A_3^{-1}X_1)]$$

$$lfpX_1.[(R_1 \wedge \sim B_1 \wedge \sim B_2) \vee ((B_1 \wedge A_1^{-1}X_1) \vee (B_2 \wedge A_2^{-1}X_1))].$$

We shall be interested in "Continuous FPF's" where we only allow least fixpoint formation on (formally) $\cup$-continuous transformers and greatest fixpoint formation on (formally) $\cap$-continuous transformers. We say that an FPF E is formally $\cup$-continuous ($\cap$-continuous) in X provided that its normal form E' (obtained by driving all negations "inward" using DeMorgan's laws and the fact that $\sim gfpX.t[X,Y] = lfpX.\sim t[\sim X,Y]$) satisfies two conditions: (i) E' is monotonic in X and (ii) E' contains no free occurrence of X inside a subFPF of the form $gfpY.E''[\ldots Y,\ldots,X\ldots]$ ($lfpY.E''[\ldots Y,\ldots,X\ldots]$). Given that E is in normal form, we can say the following: if E contains no occurrences of lfp or gfp, then E is a Continuous FPF. If E contains no occurrences of lfp or gfp, and no negated occurrences of X, then $lfpX.E[X]$ and $gfpX.E[X]$ are Continuous FPF's. On the other hand, if E involves an "alternation" of lfp and gfp operators (entailing the situation disallowed in (ii)), then E is not a Continuous FPF.

5. Results

Our main results are summarized below (and proved in the appendix):

5.1 Theorem: There is an effective procedure to translate CTF definitions of correctness properties into FPF definitions. Any correctness property defined in CTF without use of the $\overset{\infty}{\exists}$ and $\overset{\infty}{\forall}$ quantifiers is translated into a Continuous FPF. []

5.2 Theorem: Any correctness property definable as a Continuous FPF is Δ_1^1 over the natural numbers. []

5.3 Theorem: The correctness property definable in CTF as "$\forall$ fullpath $\overset{\infty}{\forall}$ node R" is Π_1^1-complete on the domain of natural numbers. []

5.4 Corollary: "$\forall$ fullpath $\overset{\infty}{\forall}$ node R" is not definable as a Continuous FPF, nor as a CTF without use of the $\overset{\infty}{\exists}$ or $\overset{\infty}{\forall}$ quantifiers. []

A formula $F(x_1,\ldots,x_n)$ of 1st order arithmetic with free variables $x_1,\ldots,x_n$ defines, in a natural way, an n-ary relation over ω. For example, the formula $F(x) \equiv \exists y\ x=2y$ defines the set of even natural numbers. A relation definable by a formula of 1st order arithmetic is called an arithmetical relation. The class of all arithmetical relations can be organized in a hierarchy based on the number of alternations of

existential and universal quantifiers required in the defining formulae: Let $\Sigma_0^0 = \Pi_0^0 =$ the class of all recursive relations over ω. For all $m > 0$, let Σ_m^0 be the class of all relations R definable as $R(x_1,\ldots,x_n) \equiv \exists y\ S(y,x_1,\ldots,x_n)$ where S belongs to Π_{m-1}^0, and let Π_m^0 be the class of all relations whose complements belongs to Σ_m^0. By induction, we can show that each relation in Σ_m^0 can be defined by a formula of the form: $\exists x_1 \forall x_2 \exists x_3 \ldots Qx_m R(z,x_1,\ldots,x_m)$ where R is recursive and Q denotes $\exists$ for odd m, $\forall$ for even m. Similarly, each relation in Π_n^0 can be defined by a formula of the form: $\forall x_1 \exists x_2 \ldots Qx_m R(z,x_1,\ldots,x_m)$ where R is recursive and Q denotes $\forall$ for odd m, $\exists$ for even m. It can be shown that the arithmetical hierarchy is indeed a hierarchy ($\Sigma_m^0 \cup \Pi_m^0 \subseteq \Sigma_{m+1}^0 \cap \Pi_{m+1}^0$ for all m) which covers the arithmetical relations. Also, for each class in the hierarchy there are "complete" relations ("hardest" relations in the class). For example, the class Σ_1^0 (which coincides with the class of all recursively enumerable relations) has a complete relation K = "the set of encodings of Turing machines which halt on their own encodings".

There are also relations whose arguments include "2nd order" objects such as predicates (i.e. total functions $\omega \to \{0,1\}$). For instance, if P is a variable ranging over 2^ω, then we could have a relation $R(x_1,\ldots,x_n,P) \subseteq \omega^n \times 2^\omega$. To say that such a relation R is recursive means that there is an "oracle Turing machine" which (i) takes as input $x_1,\ldots,x_n$, (ii) uses a read-only, one-way infinite oracle tape encoding the graph of P, and (iii) always halts (for all inputs and all possible oracle tape contents). This notion leads to the analytical hierarchy which is a classification for relations definable in 2nd order arithmetic based on the alternation of 2nd order quantifiers. We are interested in two classes at the bottom of the analytical hierarchy: Π_1^1 and Σ_1^1. Π_1^1 consists of all relations $R(x_1,\ldots,x_n,P_1,\ldots,P_p)$ over the natural numbers which are definable by a formula of 2nd order arithmetic of the form $\forall F\ \exists y\ \forall z\ Q(x_1,\ldots,x_n,y,z,F,P_1,\ldots P_p)$ where $F,P_1,\ldots,P_p$ range over 2^ω, $x_1,\ldots,x_n,y,z$ range over ω, and Q is a recursive relation. Σ_1^1 = the class of all relations whose complements are in Π_1^1. $\Delta_1^1 = \Sigma_1^1 \cap \Pi_1^1$ is called the class of hyperarithmetical relations and contains the class of arithmetical relations. (See [RO67] and [HI78] for discussions of hierarchy theory.)

When we say that a correctness property such as "$\forall$ fullpath $\overset{\infty}{\forall}$ node R" is, e.g., Π_1^1, we mean that the representing relation $\{(\pi,\sigma,R): \forall$ fullpath of $\mathcal{F}(\pi,\sigma)\ \overset{\infty}{\forall}$ node $R\} \subseteq \omega^2 \times 2^\omega$ is Π_1^1. Since a Π_1^1-complete relation cannot be hyperarithemetical, it follows that "$\forall$ fullpath $\overset{\infty}{\forall}$ node R" cannot be characterized in Continuous FPF. Using some additional machinery from recursive function theory, it can also be shown that "$\forall$ fullpath $\overset{\infty}{\forall}$ node R" cannot be characterized as the greatest fixpoint of any monotonic transformer of any degree of complexity lower than Π_1^1. These conclusions hold for inevitability under fair scheduling as well. (Here we use "weak eventual fairness": there is no process which is active almost everywhere yet executed only finitely often; "strong eventual fairness" would also work.) Inevitability under fair scheduling can be characterized in CTF as

"$\forall$ fullpath [path_is_unfair $\vee$ $\exists$ node R]"

where path_is_unfair abbreviates

"$\forall$ node ($\underset{i}{\vee} B_i$) $\wedge$ [($\overset{\infty}{\forall}$ node B_1 $\wedge$ $\sim\overset{\infty}{\exists}$ arc $\{B_1\}$) $\vee$ ... $\vee$ ($\overset{\infty}{\forall}$ node B_k $\wedge$ $\sim\overset{\infty}{\exists}$ arc $\{B_k\}$)]".

Since this has the form " $\forall$ fullpath (... $\overset{\infty}{\forall}$ node B_1 ...)" it can be shown that it is at least as hard to describe as "$\forall$ fullpath $\overset{\infty}{\forall}$ node R". In the appendix we show how to apply the effective translation procedure to derive a monotonic, noncontinuous FPF characterization for fair inevitability from the above CTF formula.

6. Relationship to Modal Logic

CTF may be viewed as a modal logic. The computation trees determine Kripke structures where the accessibility relation R between states is given by σ_1 R σ_2 iff there is a path in the computation tree from a node labelled with state σ_1 to a node labelled with state α_2. Since each CTF formula $E(R_1,...,R_n)$ defines a modality, there are an infinite number of modalities in this logic. However, in the proof that CTF is translatable into FPF, we show that all these modalities can be expressed in terms of four basic types of modalities: α , ξ , ι , and ν . Thus, these modalities are expressively complete for CTF.

7. Conclusion

We have shown that correctness properties of parallel programs can be described using computation trees and that from these descriptions fixpoint characterizations can be generated. We have also given conditions on the form of computation tree descriptions to ensure that a correctness property can be characterized using continuous fixpoints. A consequence is that a correctness property such as inevitability under fair scheduling can be characterized as the least fixpoint of a monotonic, noncontinuous transformer, but cannot be characterized using fixpoints of continuous transformers (nor as the greatest fixpoint of a monotonic transformer of any degree of complexity lower than fair inevitability itself). Hence, currently known proof rules are not applicable (see however [FS80]). We are now investigating whether useful proof rules can exist for correctness properties having only a monotonic, noncontinuous least fixpoint characterization. In addition, we are examining alternate notions of fairness which do have continuous fixpoint characterizations.

8. References

[BY75] Basu, S.K. and Yeh, R.T., Strong Verification of Programs. IEEE Trans. on Software Engineering, v. SE-1. no. 1, pp.339-354, September 1975.

[CH78] Chandra, A. K., Computable Nondeterministic Functions. 19th Annual Symp. on Foundations of Computer Science, 1978.

[CL77] Clarke, E. M., Program Invariants as Fixpoints. 18th Annual Symp. on Foundations of Computer Science, 1977.

[CL79] Clarke, E. M., Synthesis of Resource Invariants for Concurrent Programs, 6th POPL Conference, January, 1979.

[CO76] Cousot, P. and Cousot R., Static Determination of Dynamic Properties of Programs. Proc. 2nd Int. Symp. on Programming, (B. Robinet, ed.), Dunod, Paris, April 1976.

[DE77a] de Bakker, J. W., Recursive Programs as Predicate Transformers. Mathematical Centre, Amsterdam, 1977.

[DE77b] de Bakker. J. W., Semantics of Infinite Processes Using Generalized Trees. Mathematical Centre, Amsterdam, 1977.

[DI76] Dijkstra, E. W., A Discipline of Programming, Prentice-Hall, 1976.

[FS78] Flon, L. and Suzuki, N., Consistent and Complete Proof Rules for the Total Correctness of Parallel Programs. 19th Annual Symp. on Foundations of Computer Science, 1978.

[FS80] Flon, L. and Suzuki, N., The Total Correctness of Parallel Programs. SIAM J. Comp., to appear

[HI78] Hinman, P. G., Recursion-Theoretic Hierarchies, Springer-Verlag,Berlin, 1978.

[HO69] Hoare, C. A. R., An Axiomatic Approach to Computer Programming CACM, v.10. no 12., pp.322-329, October 1969.

[MW79] Meyer, A. R. and Winklmann, K., On the Expressive Power of Dynamic Logic. Proceedings of the 11th Annual ACM Symp. on Theory of Computing, 1979.

[PA69] Park, D., Fixpoint Induction and Proofs of Program Properties, in Machine Intelligence 5, (D. Mitchie, ed.) Edinburgh University Press, 1970.

[RE79] Reif, J. H., Data Flow Analysis of Communicating Processes. 6th POPL Conference, January 1979.

[RO67] Rogers, H. R., Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.

[SV76] Sintzoff, M. and Van Lamsweerde, A., Formal Derivation of Strongly Correct Parallel Programs, M. B. L. E. Research Lab., Brussels, Report R338, October 1976.

[TA55] Tarski, A., A Lattice-Theoretical Fixpoint Theorem and Its Applications. Pacific J. Math.,5, pp.285-309 (1955).

9. Appendix

Proof outline for 5.1

We outline the main steps of the effective procedure for translating CTF into FPF. Note that if there are no path quantifiers present in the CTF, then it is already a legitimate FPF. (To simplify notation we adopt these conventions: symbols such as G, G_1, G_2^3, etc. denote guardsets chosen from $\{B_1,..., B_k\}$ and the corresponding lower case symbols g, g_1, g_2^3, etc. denote the corresponding set of indices; e.g., if G_2^3 represents $\{B_1,B_3,B_5\}$ then g_2^3 denotes $\{1,3,5\}$. We use,e.g., $\bar{R}^2$ and $\bar{G}^2$ to denote vectors $(R_1^2,...,R_n^2)$ and $(G_1^2,...,G_n^2)$, respectively. Finally, we use BB to abbreviate $B_1 \vee ... \vee B_k$.)

(1) Reduce to translating CTF's with at most one path quantifier by recursively applying the translation procedure to nested CTF sub-expressions. For example:

Let e(R,S) = "$\exists$ fullpath $\forall$ node [$\forall$ fullpath $\exists$ node R $\wedge$ $\exists$ fullpath $\overset{\infty}{\exists}$ node S]" be the CTF to be translated.

Let f1(R), f2(S), f3(T) denote the FPF translations of " $\forall$ fullpath $\exists$ node R", "$\exists$ fullpath $\overset{\infty}{\exists}$ node S", and "$\exists$ fullpath $\forall$ node T", respectively.

Then the FPF translation is $f3(f1(R) \wedge f2(S))$.

(2) Reduce to translating CTF's with one <u>existential</u> path quantifier by use of duality. The <u>dual</u> of a correctness property $C(R_1,\ldots,R_n)$ is $C^{\#}(R_1,\ldots,R_n) = \sim C(\sim R_1,\ldots, \sim R_n)$. The following facts about fixpoints of duals are useful (see [PA69]):

$$\mathrm{lfp}R_1.C^{\#}(R_1,\ldots,R_n) = \sim \mathrm{gfp}R_1.C(R_1,\sim R_2,\ldots,\sim R_n)$$
$$\mathrm{gfp}R_1.C^{\#}(R_1,\ldots,R_n) = \sim \mathrm{lfp}R_1.C(R_1,\sim R_2,\ldots,\sim R_n)$$

For example, let $C(R)$ = "$\forall$ fullpath $\exists$ node R" be the CTF to be translated. Its dual $C^{\#}(R)$ = " $\sim\forall$ fullpath $\exists$ node $\sim$R" = "$\exists$ fullpath $\forall$ node R" is in the desired form with a single existential quantifier. The remaining steps of the procedure will show that the FPF translation of $C^{\#}(R)$ is

$$\mathrm{gfp}X.D[X,R] \text{ where } D[X,R] = R \wedge [\, \sim(\bigvee_i B_i) \vee \bigvee_i (B_i \wedge A_i^{-1}X)\,]$$

To get the FPF translation of $C(R) = (C^{\#})^{\#}(R)$ we again dualize to get

$$\begin{aligned} &\sim \mathrm{gfp}X.D[X,\sim R] \\ &= \mathrm{lfp}X.D^{\#}[X,R] \\ &= \mathrm{lfp}X.\sim(\sim R \wedge [\sim(\vee\ B_i) \vee \bigvee_i (B_i \wedge A_i^{-1}(\sim X))]) \\ &= \mathrm{lfp}X.R \vee [\bigvee_i B_i \wedge \bigwedge_i (\sim B_i \vee \sim A_i(\sim X))] \\ &= \mathrm{lfp}X.R \vee [\bigvee_i B_i \wedge \bigwedge_i (B_i \Rightarrow A_i^{-1}X)] \end{aligned}$$

(3) Reduce to translating a <u>disjunction of CTF's</u>. First, place the body in the following general form:

$$\bigvee(\forall\text{-PART} \wedge \exists\text{-PART} \wedge \overset{\infty}{\forall}\text{-PART} \wedge \overset{\infty}{\exists}\text{-PART})$$

where each $\forall$-PART has the form " $\bigwedge \forall$ node P $\wedge$ $\bigwedge \forall$ arc G ",

each $\exists$-PART has the form " $\bigwedge(\bigvee \exists$ node P $\vee$ $\bigvee$ $\exists$ arc G)",

each $\overset{\infty}{\forall}$-PART has the form " $\bigwedge \overset{\infty}{\forall}$ node P $\wedge$ $\bigwedge$ $\overset{\infty}{\forall}$ arc G", and

each $\overset{\infty}{\exists}$-PART has the form " $\bigwedge(\bigvee \overset{\infty}{\exists}$ node P $\vee \bigvee$ $\overset{\infty}{\exists}$ arc G)".

There may be more than one specific form for the body consistent with the above general form, but there is always at least one: Disjunctive Normal Form (DNF may be obtained by having certain appropriate inner disjunctions of $\exists$-PART and $\overset{\infty}{\exists}$-PART be vacuous). However, more concise FPF translations often result from keeping $\exists$-PART and $\overset{\infty}{\exists}$-PART in true "product of sums" form with the sums as large as possible. We use the fact that $\vee$ "commutes" with $\exists$ to separate into a disjunction of CTF's, e.g.:

$$\exists \text{ fullpath } (\bigvee_i (\forall\text{-PART}_i \vee \exists\text{-PART}_i \vee \overset{\infty}{\forall}\text{-PART}_i \vee \overset{\infty}{\exists}\text{-PART}_i))$$
$$\equiv \bigvee_i \exists \text{ fullpath}(\forall\text{-PART}_i \vee \exists\text{-PART}_i \vee \overset{\infty}{\forall}\text{-PART}_i \vee \overset{\infty}{\exists}\text{-PART}_i\ )$$

Each of the disjuncts is then translated separately by the remaining step.

(4) Put the CTF in a <u>standard form</u> which can be translated via the tables below. Note that $\wedge$ and $\cap$ "commute" with $\forall$ and $\overset{\infty}{\forall}$ and that $\vee$ and $\cup$ "commute" with $\exists$ and $\overset{\infty}{\exists}$. For example, $\bigwedge_i \overset{\infty}{\forall}$ node $R_i \equiv \overset{\infty}{\forall}$ node $\bigwedge_i R_i$ and $\bigvee_i$ $\exists$ arc $G_i \equiv \exists$ arc $\bigcup_i$ G_i. Using these facts we obtain the standard form:

$$\exists \begin{bmatrix} \text{fullpath} \\ \text{segment} \end{bmatrix} \left(\begin{array}{l} \overbrace{(\ \forall \text{ node } R^1 \wedge \forall \ \text{ arc } G^1)}^{(1)} \wedge \overbrace{(\bigwedge_i\ (\ \exists \text{ node } R^2 \vee \exists \ \text{ arc } G^2))}^{(2)} \wedge \\ \underbrace{(\ \overset{\infty}{\forall} \text{ node } R^3 \wedge \overset{\infty}{\forall} \ \text{ arc } G^3)}_{(3)} \wedge \underbrace{(\bigwedge_i\ (\ \overset{\infty}{\exists} \text{ node } R_i^4 \vee \overset{\infty}{\exists} \ \text{ arc } R_i^4))}_{(4)} \end{array} \right)$$

Any (nonempty) combination of clauses (1) - (4) may be "present" (i.e., have at least one term for nodes or at least one term for arcs present). The tables below show how to translate any combination of clauses (1) - (4) by an appropriate composition of the basic correctness properties (a) - (d) below:

(a) $\alpha(R,G)$ which means "along some fullpath, R holds of all nodes and G holds of all arcs."

(b) $\xi(R,G,S)$ which means "along some (finite) segment, R holds of all nodes and G holds of all arcs upto (and including) the last node at which S holds"

(c) $\iota(R^1,G^1,\bar{R}^2,\bar{G}^2)$ which means "along some (infinite) fullpath, R_1 holds of all nodes, G_1 holds of all arcs, and for each $i \in [1{:}m]$ there are infinitely many occurrences of a node where R_i^2 holds or infinitely many occurrences of an arc where G_i^2 holds."

(d) $\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,S)$ which means " along some (finite) segment, R^1 holds of all nodes, G^1 holds of all arcs, for each $i \in [1{:}m]$ there is on the segment either a node satisfying R_i^2 or an arc satisfying G_i^2, and then at the last node of the segment , S holds.

These basic correctness properties may be defined directly in FPF:

(a) $\alpha(R,G) = \mathrm{gfp}X.R \wedge (\sim \bigvee_j B_j \vee \bigvee_{i\in g} (B_i \wedge A_i^{-1}X)\ )$

(b) $\xi(R,G,S) = \mathrm{lfp}X.R \wedge (\ S \vee \bigvee_{i\in g} (B_i \wedge A_i^{-1}X)\ )$

(c) $\iota(R^1,G^1,\bar{R}^2,\bar{G}^2) =$

$$\mathrm{gfp}X. \bigwedge_{j=1}^{m} \xi(R^1,G^1,\ (R_j^2 \wedge \bigvee_{i\in g^1}(B_i \wedge A_i^{-1}X)) \ \vee \bigvee_{i\in g^1\cap g_j^2} (B_i \wedge A_i^{-1}X)\ )$$

(d) $\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,S) =$

$$\bigvee\{ \gamma_{i_1} \circ \cdots \circ \gamma_{i_m} \circ \varepsilon(R^1,G^1,S) : (i_1,\ldots,i_m) \text{ is a permutation of } \{1,2,\ldots,m\}\ \}$$
$$\text{where for } j \in [1{:}m]\ \gamma_j(P) = \xi(R^1,G^1,\ \ (R_j^2 \wedge P) \vee \bigvee_{i\in g^1\cap g_j^2} (B_i \wedge A_i^{-1}P)\ \ )$$

Note that while the notation looks formidable, the idea in (4) is simple. Each permutation records a possible order of occurrence of nodes satisfying each of the R_i^2 (or arcs satisfying the G_i^2). It is necessary to consider all permutations since the definition of ν does not specify an order. Verification of the correctness of the above FPF characterizations and the table entries below is straightforward and is left to the reader.

Table	for fullpath	for segment
$1\bar{2}\bar{3}\bar{4}$:	$\alpha(R^1,G^1)$	$R^1 \wedge G_1^1 \wedge \ldots \wedge G_n^1$ where $G^1 = \{G_1^1,\ldots,G_n^1\}$
$1\bar{2}\bar{3}4$:	$\iota(R^1,G^1,\bar{R}^4,\bar{G}^4)$	$\iota(R^1,G^1,\bar{R}^4,\bar{G}^4)$
$1\bar{2}3\bar{4}$:	$\xi(R^1,G^1,\sim BB \vee \alpha(R^1 \wedge R^3,G^1 \cap G^3))$	$R^1 \wedge G_1^1 \wedge \ldots \wedge G_n^1$
$1\bar{2}34$:	$\xi(R^1,G^1,\iota(R^1 \wedge R^3,G^1 \cap G^3,\bar{R}^4,\bar{G}^4))$	$\xi(R^1,G^1,\iota(R^1 \wedge R^3,G^1 \cap G^3,\bar{R}^4,\bar{G}^4))$
$12\bar{3}\bar{4}$:	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,\alpha(R^1,G^1))$	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,True)$
$12\bar{3}4$:	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,\iota(R^1,G^1,\bar{R}^4,\bar{G}^4))$	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,\iota(R^1,G^1,\bar{R}^4,\bar{G}^4))$
$123\bar{4}$:	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,\sim BB \vee \alpha(R^1 \wedge R^3,G^1 \cap G^3))$	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,True)$
1234:	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,\iota(R^1 \wedge R^3,G^1 \cap G^3,\bar{R}^4,\bar{G}^4))$	$\nu(R^1,G^1,\bar{R}^2,\bar{G}^2,\iota(R^1 \wedge R^3,G^1 \cap G^3,\bar{R}^4,\bar{G}^4))$

Note that 2 indicates clause 2 is present, and $\bar{2}$ indicates clause 2 is absent, etc. If clause (1) is absent, we translate just as when it is present but let R^1 = True, $G^1 = \{B_1,\ldots,B_k\}$. If clause (i) is present but there are no node conditions, we let R^i = True. If clause (i) is present but there are no arc conditions, let $G^i = \{B_1,\ldots,B_k\}$. For example, the segment table entry for $1\bar{2}\bar{3}4$ indicates that $\exists$ segment ($\forall$ node $R^1 \wedge \forall$ arc $G^1 \wedge \bigwedge_i$ ($\overset{\infty}{\exists}$ node $R_i^4 \vee \overset{\infty}{\exists}$ arc G_i^4) = $\iota(R^1,G^1,\bar{R}^4,\bar{G}^4)$, which follows directly from the definition of ι and the fact that any infinite segment is a fullpath.

We now derive an FPF characterization of fair inevitability starting with the CTF description $\forall$ fullpath [path_is_unfair $\vee$ $\exists$ node R]:

1. Dualize to obtain
   $\exists$ fullpath [$\forall$ node R $\wedge$~path_is_unfair]
2. Use the definition of path_is_unfair to obtain
   $\exists$ fullpath [$\forall$ node R $\wedge$ [$\exists$ node (~BB) $\vee$ $\bigwedge_j$ ($\overset{\infty}{\exists}$ node ~B_j $\vee$ $\overset{\infty}{\exists}$ arc $\{B_j\}$)]]
3. Use the distributive law and then split apart disjuncts to obtain
   $\exists$ fullpath [$\forall$ node R $\wedge$ $\exists$ node (~BB)] $\vee$
   $\exists$ fullpath [$\forall$ node R $\wedge$ $\bigwedge_j$ ($\overset{\infty}{\exists}$ node B_j $\vee$ $\overset{\infty}{\exists}$ arc $\{B_j\}$)]]
4. Use the translation procedure to obtain:
   $\xi(R,\{B_1,\ldots,B_k\},\sim BB) \vee \iota(R,\{B_1,\ldots,B_k\}, B_1,\ldots, B_k,\{B_1\},\ldots,\{B_k\})$
5. Dualize again to obtain
   $\xi^*(R,\emptyset,BB) \wedge \iota^*(R,\emptyset,\{B_2,\ldots,B_k\},\ldots,\{B_1,\ldots,B_{k-1}\})$

Finally, observe that any CTF defined without using the quantifiers $\overset{\infty}{\exists}$ or $\overset{\infty}{\forall}$ is translated using only α and ξ which are Continuous FPF's. []

<u>Proof outline for 5.2</u>

We outline a proof by induction on the structure of correctness properties definable

in Continuous FPF that any such correctness property is hyperarithmetical (i.e. Δ_1^1): Since gfpX.t[X] = ~lfpX.t^*[X] and {$\vee$,~} is a complete set of boolean connectives, the following steps suffice:

<u>Basis</u>: $t[R_1,\ldots,R_n] = R_i$ $(1 \le i \le n)$ is Δ_1^1 immediately.
$t[R_1,\ldots,R_n] = B_i$ $(1 \le i \le k)$ is Δ_1^1 since guard B_i is Δ_1^0.

<u>Induction</u>: $t[R_1,\ldots,R_n] = t_1[R_1,\ldots,R_n] \vee t_2[R_1,\ldots,R_n]$ is Δ_1^1 since Δ_1^1 is closed under finite union.
$t[R_1,\ldots,R_n] = \sim t_1[R_1,\ldots,R_n]$ is Δ_1^1 since Δ_1^1 is closed under complementation.
$t[R_1,\ldots,R_n] = A_i^{-1} t_1[R_1,\ldots,R_n]$ $(1 \le i \le k)$ is Δ_1^1 since action A_i is Δ_1^0.
$t[R_1,\ldots,R_n]$ = lfpR.$t_1[R,R_1,\ldots,R_n]$ is Δ_1^1 since Δ_1^1 is closed under inductive definitions with closure ordinals $\le \omega$. (See [HI78]). []

<u>Proof outline of 5.3</u>

We show that Q = $\{(\pi,\sigma,R)$: $\exists$ fullpath $\overset{\infty}{\exists}$node R is true of $\mathscr{T}(\pi,\sigma)\}$ is Σ_1^1-complete. It follows that (the reprepresenting relation of the) dual correctness property "$\forall$ fullpath $\overset{\infty}{\forall}$node R" is Π_1^1-complete.

<u>Q is Σ_1^1-hard</u>: Let S be an arbitrary Σ_1^1 set. then for some recursive relation $P \subseteq \omega^3 \times 2^\omega$, $x \in S \iff \exists F \forall y \exists z\, P(x,y,z,F)$ (see [RO67]). We can construct a <u>do-od</u> program π_S which on input x, guesses F and attempts to find for each y some z so that P holds. We design π_S so that for each distinct y = 1,2,3,... when (and if) the appropriate z is found, π_S sets (for one step) a special flag q_0 to true before proceeding to check the next y value. There is a possible computation of π_S starting in state x for which q_0 becomes true infinitely often iff $x \in S$. Thus, the question "Is $x \in S$?" has been effectively reduced to "Is (π_S,x,{q_0=True}) $\in$ Q?". So Q is Σ_1^1-hard.

<u>Q is in Σ_1^1</u>: We use <x,y> to indicate a (recursive) pairing function establishing a bijection $\omega^2 \to \omega.(\langle x,y\rangle_0 = x$ and $\langle x,y\rangle_1 = y)$. We define the recursive functional <u>state</u>: $\omega^3 \times 2^\omega \to \omega$ by <u>state</u>$(\pi,\sigma,n,F) = \langle 1, A_{F(n)} \circ A_{F(n-1)} \circ \cdots \circ A_{F(1)}\ (\sigma)\rangle$ if F(1),...,F(n) encodes a legitimate initial segment in $\mathscr{T}(\pi,\sigma)$ where π = <u>do</u> $A_1 \to B_1$ []...[] $A_k \to B_k$ <u>od</u> and each $F(i) \in [1:k]$ and gives the index of the command chosen on the ith trip through the loop. Otherwise, <u>state</u>$(\pi,\sigma,n,F) = \langle 0,0\rangle$.

Then Q is defined by the Σ_1^1 formula of 2nd order arithmetic:

$\exists F \forall i \exists j\ (j > i \wedge (\underline{state}(\pi,\sigma,j,F))_0 = 1 \wedge P(\ (\underline{state}(\pi,\sigma,j,F))_1\ ) = True)$. []

<u>proof outline for 5.4</u>

If "$\forall$fullpath $\overset{\infty}{\forall}$ node R" were definable as a Continuous FPF, it would be Δ_1^1 by Theorem 5.2. But this contradicts Theorem 5.3 since no Δ_1^1 relation can be Π_1^1-complete. If it were definable as a CTF without using the $\overset{\infty}{\exists}$ or $\overset{\infty}{\forall}$ quantifiers, it would be definable as a Continuous FPF by Theorem 5.1. But, we just saw that this is impossible. []