MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo 113
(First Draft)

Decidability of Equivalence for Two Classes of Iteration Schemas

by

Joseph E. Qualitz

This work was supported in part by the National Science Foundation under research grant GJ-34671 and in part by funds from International Business Machines.

March 1975

In this paper, we examine a class of computation schemas and consider the problem of deciding when pairs of elements in this class represent equivalent programs. We are able to show that equivalence is decidable for two non-trivial classes of monadic program schemas [2,3], and consider the applicability of these results to more general equivalence problems.

The most general class of schemas we consider, the iteration schemas, consists of those program schemas which may be represented by finite sequences of assignment statements, conditional statements, and iteration statements ("while" or "until" statements). We are able to show that equivalence is decidable for such schemas which are free, i.e. in which tests are not repeated, provided they satisfy either of two sets of conditions:

- They are independent location schemas (the result obtained from the execution of an assignment statement is always stored in the same location from which the argument value for the execution was obtained) and they contain no conditional statements; or
- They are restricted location schemas (their subschemas are independent location) and no predicate symbol appears in more than one statement of the schema.

I. Iteration Schemas

A (monadic) iteration schema is a seven-tuple

$$S = (L, L_I, L_O, \mathcal{F}, \mathcal{P}, \mathcal{S}, P)$$

where:

L is a finite set of location symbols.

L_I L is a finite set of input location symbols.

L_O L is a finite set of output location symbols.

$\mathcal{F}$ is a finite set of function letters.

$\mathcal{P}$ is a finite set of predicate letters.

$\mathcal{S}$ is a finite set of iteration schemas, the subschemas of S.

P is the program of S, a finite sequence of statements of the following types:

i) Assignment statement of the form

$$L_i := f(L_j)$$

or of the form

$$L_i := L_j$$

where L_i and L_j are location symbols and f is a function letter.

ii) Conditional statement of the form

IF $p(L_i)$ THEN S_t ELSE S_f

where p is a predicate letter, L_i is a location symbol, and S_t and S_f are subschemas.

iii) Iteration statement of the form

WHILE $p(L_j)$ DO S_i

or of the form

UNTIL $p(L_j)$ DO S_i

where p is a predicate letter, L_j is a location symbol, and S_i is a subschema.

If $S' = (L', L_I', L_0', \mathcal{F}', \mathcal{P}', \mathcal{S}', P')$ is a subschema of S, then we require that $L' \subset L$, $\mathcal{F}' \subset \mathcal{F}$, $\mathcal{P}' \subset \mathcal{P}$, and $\mathcal{S}' \subset \mathcal{S}$. Also, an iteration schema is not allowed to be recursive, i.e. it may not be a subschema of itself, nor may it have a subschema which is a subschema of itself.

Interpretations, Computations, and Equivalence

Let S be an iteration schema with input location symbols $L_I = \{L_{i_1}, \ldots, L_{i_k}\}$, predicate letters in a set of symbols $\mathcal{P}$, and function letters in a set of symbols $\mathcal{F}$. Then a (free) interpretation for S consists of:

i) The domain D of all strings in $\mathcal{F}^* \cdot K$, where K is a set of symbols $\{\Delta_{i_1}, \ldots, \Delta_{i_k}\}$, such that $K \cap \mathcal{F}$ is empty.

ii) For each j, $1 \leq j \leq k$, the association of the string Δ_{i_j} with the location denoted by L_{i_j}, and the association of ε with all other locations.

iii) The association, for each symbol f in $\mathcal{F}$, of the total function φ_f: $D \to D$ defined by $\varphi_f(\omega) = f \cdot \omega$.

iv) An association, for each symbol p in $\mathcal{P}$, of a total predicate Π_p: $D \to \{\text{true}, \text{false}\}$.

Each interpretation for a schema defines a computation by that schema in a straightforward way: we simply execute the statements in the program of the

schema in the order in which they are encountered, associating appropriate domain individuals with locations as required by assignment statements, and executing the programs of subschemas in accordance with the values of predicates applied during the execution of conditional or iteration statements. When an assignment instruction of the form $L_i := L_j$ is executed, the element of the domain currently associated with L_j is associated with L_i; when an assignment statement of the form $L_i := f(L_j)$ is executed, the value $\varphi_f(\omega)$ is associated with L_i, where ω is the element of the domain associated with L_j at the time of execution. When a conditional statement of the form IF $p(L_i)$ THEN S_t ELSE S_f is executed, the predicate Π_p is evaluated at the element of the domain currently associated with L_i, and the program of S_t or of S_f is executed, according as the outcome of the evaluation is true or false; when the execution of the appropriate program is completed, execution of the main program resumes with the statement immediately following the conditional. Finally, when an iteration statement of the form WHILE(UNTIL) $p(L_i)$ DO S_i is executed, the predicate Π_p is evaluated at the current value of L_i and the program of S_i is executed if the outcome of the evaluation is <u>true</u> (<u>false</u>); upon completion of the program's execution, execution of the main program resumes with the iteration statement. This cycle is repeated until an evaluation of Π_p has outcome <u>false</u> (<u>true</u>), at which point execution proceeds to the statement following the iteration statement. The computation by the schema is complete when and if all statements in the program have been executed.

The <u>value</u> of a computation by a schema S defined by an interpretation I for S (written VAL(S,I)) is the sequence of values associated with the output locations of S when the computation defined by I is complete, and is undefined if the computation fails to terminate.

Let S and S' be iteration schemas with the same set of input location symbols and the same set of output location symbols. Then S and S' are equivalent if and only if, for each interpretation I for S and S', VAL(S,I) = VAL(S',I).

Classes of Iteration Schemas - Definitions

An iteration schema S is an independent location schema if each assignment statement in the program of S is of the form $L_i := f(L_i)$ for some location symbol L_i and some function letter f, and each subschema of S is an independent location schema.

An iteration schema S is a restricted location schema if each iteration subschema of S is an independent location schema.

A schema S is a free schema if, for each interpretation I for S, no predicate is ever applied twice to the same value during the computation by S defined by I.

A schema S is a conditional-free schema if the program of S contains no conditional statements, and each of the subschemas of S is a conditional-free schema.

Finally, a schema S satisfies Property A if no predicate letter appears in more than a single statement in the collection of programs belonging to S and the subschemas of S.

Conventions Regarding Input and Output Locations

If our definition of independent location schema is to be reasonable, we must require that each location of such a schema be an input location of the schema, otherwise the location will never have a non-trivial value associated with it during a computation. In keeping with tradition, we shall require each location of an independent location schema to be an output location as well.

For convenience, we adopt the same conventions regarding input and output locations for restricted location schemas.

Terminology

For convenience, we shall often refer to a statement being in a schema S, rather than stating formally that the statement is in the collection of programs belonging to the schema S and its subschemas. If we wish to imply that the statement is in the program of S, as opposed to that of one of its subschemas, we shall do so explicitly.

We shall often want to modify schema interpretations to obtain computations differing in some respect from those defined by the original interpretations. For simplicity, we shall generally define such modifications in terms of the computations themselves, rather than the defining interpretations. In particular, we shall often refer to obtaining a new computation from an original computation by changing the outcome of some test made during the computation, whereas what is meant is that we may obtain the new computation by changing, in an appropriate manner, the interpretation defining the original computation.

II. Additional Definitions

Definition: Let S and S' be iteration schemas. Then computations C by S and C' by S' are consistent computations if there is an interpretation I for S and S' such that C and C' are defined by I.

Definition: Let S be an iteration schema with predicate letters Φ, and let D be the domain of each free interpretation for S. Then the set of tests for S is the set $D \times \Phi$. A test $\tau = (\xi, p)$ is said to be made at statement s of S during computation C if, during the computation C by S, the predicate Π_p assigned by the interpretation defining C to the predicate letter p appearing in s is evaluated at ξ during the execution of s. We denote by TESTS(s,C) the sequence of tests made at statement s during C, and by TESTS*(S,C) the sequence of all tests made at any statement of S during C. A test τ is said to be free with respect to the computation C if $\tau \notin$ TESTS*(S,C).

Definition: Let s and s' be statements in iteration schemas S and S', respectively. Then s and s' are logic equivalent if, whenever C and C' are consistent computations by S and S', TESTS(s,C) = TESTS(s',C').

Definition: Let s be an iteration statement in a schema S. Then location L is the test location of s if L is the location symbol appearing in statement s. In such a case, we say that L is the location of each test τ made at s during a computation by S. A location L is bounded at s if, during any computation by S, the value associated with L prior to any execution of statement s is the same as that associated with L at the conclusion of the execution. A location L is passive in statement s if L is not the test location of s and there is no statement s' in the subschema called in s such that L is the test location of s'. (Intuitively, a locati

is passive in s if its associated values are never tested during an execution of s.)

Definition: Let S be an iteration schema. We denote by MAIN(S) the collection of statements in S which are not in some iteration subschema of S.

Definition: Let I be an interpretation for some schema S containing predicate letter p. Then we denote by (I/p) the partial interpretation for S identical to I, except that p is not interpreted by (I/p), i.e. except that no predicate is assigned by (I/p) to predicate letter p. If p' is a predicate letter not interpreted by some (partial) interpretation I' for S, we denote by ADJ(p',I') the set of all interpretations which can be formed by adjoining to I' an interpretation for p'.

Definition: Let $\mathcal{I}$ be a set of interpretations for a schema S. We denote by PART(S,$\mathcal{I}$) the partition of $\mathcal{I}$ into subsets of interpretations which are equivalent with respect to schema S: I, I' $\in \mathcal{I}$ are in the same equivalence class iff VAL(S,I) = VAL(S,I').

Definition: Let S be an iteration schema. Then the size of S, written SIZE(S), is the number of iteration statements in S. (While this notion of size is, perhaps, not the most intuitive, it shall prove to be quite useful as an argument for induction in subsequent proofs.)

Definition: Let S be an iteration schema, and p a predicate letter of the schema. Then p is a productive letter in S if there exist interpretations I and I' for S, such that I and I' differ only with respect to the predicate assigned to p and such that VAL(S,I) $\neq$ VAL(S,I').

Our final definition pertains to schemas which satisfy Property A.

Definition: Let S be an iteration schema satisfying Property A, and let p and p' be predicate letters appearing in iteration statements in S. Then we write $p > p'$ iff p' appears within the subschema called in statement s, where s is the statement in S in which p appears. (The reader may note that " > " is a partial ordering of the predicate letters of S which appear in iteration statements, and that a specification of the partial ordering describes fully the manner in which the various iteration subschemas of S are "nested" in the schema.)

III. Two Decidability Results

In this section, we prove that equivalence is a decidable property for two classes of free iteration schemas: restricted location schemas with Property A, and independent location schemas which are conditional-free.

We begin with a fundamental lemma:

Lemma 1. Let X, X', Y, Y' and Z' be set of words over some alphabet T, and let γ be a word in T^*. Let $f_1: X \to X'$ and $f_3: X \to X'$ be total functions; let $f_2: Y^* \to (Y')^*$ be a total function such that for each $\psi, \rho \in Y^*$ we have $f_2(\psi) \cdot f_2(\rho) = f_2(\psi\rho)$. Suppose we have, for each $\alpha \in X$ and each $\beta \in Y$, the following equalities:

1. $\gamma \cdot \alpha = f_3(\alpha) \cdot f_1(\alpha)$
2. $\gamma \cdot \beta \cdot \alpha = f_3(\alpha) \cdot f_2(\beta) \cdot f_1(\alpha)$

Then for each $\omega \in Y^*$ we have $\gamma \cdot \omega \cdot \alpha = f_3(\alpha) \cdot f_2(\omega) \cdot f_1(\alpha)$.

Proof: We know from (1) and (2) that the assertion is valid for $\omega = \varepsilon$ and for $\omega \in Y$. Assume the assertion is valid for all $\omega \in Y^i$, $0 \le i < i_0$. Then the assertion is valid for all $\omega \in Y^{i_0}$ as follows:

Let $\omega = \omega_1 \cdot \omega_2$, $\omega_1 \in Y^{i_0-1}$, $\omega_2 \in Y$. We consider three cases:

Case 1. $\alpha = f_1(\alpha)$, $\gamma = f_3(\alpha)$. We have by assumption $\gamma \cdot \omega_1 \cdot \alpha = \gamma \cdot f_2(\omega_1) \cdot \alpha$ and $\gamma \cdot \omega_2 \cdot \alpha = \gamma \cdot f_2(\omega_2) \cdot \alpha$, from which we have $\omega_1 \cdot \omega_2 = f_2(\omega_1 \cdot \omega_2)$ and thus $\gamma \cdot \omega_1 \cdot \omega_2 \cdot \alpha = f_3(\alpha) \cdot f_2(\omega_1 \cdot \omega_2) \cdot f_1(\alpha)$.

Case 2. $\alpha = \alpha \cdot f_1(\alpha)$, $\alpha \neq \Lambda$, $\gamma \cdot \alpha = f_3(\alpha)$. Then:

$\forall \sigma [(\gamma \cdot \sigma \cdot \alpha = f_3(\alpha) \cdot f_2(\sigma) \cdot f_1(\alpha)) =$
$(\gamma \cdot \sigma \cdot \alpha \cdot f_1(\alpha) = f_3(\alpha) \cdot f_2(\sigma) \cdot f_1(\alpha)) \Rightarrow$
$(\gamma \cdot \sigma \cdot \alpha = f_3(\alpha) \cdot f_2(\sigma)) \Rightarrow$
$(\gamma \cdot \sigma \cdot \mu = \gamma \cdot \mu \cdot f_2(\sigma)) \Rightarrow$
$(\sigma \cdot \mu = \mu \cdot f_2(\sigma))]$

Substituting ω_1 and ω_2 for σ yields:

$\omega_1 \cdot \mu = \mu \cdot f_2(\omega_1)$
$\omega_2 \cdot \mu = \mu \cdot f_2(\omega_2)$

Then:

$$\begin{aligned}\gamma \cdot \omega \cdot \alpha &= \gamma \cdot \omega \cdot \mu \cdot f_1(\alpha) \\ &= \gamma \cdot \omega_1 \cdot \omega_2 \cdot \mu \cdot f_1(\alpha) \\ &= \gamma \cdot \omega_1 \cdot \mu \cdot f_2(\omega_2) \cdot f_1(\alpha) \\ &= \gamma \cdot \mu \cdot f_2(\omega_1) \cdot f_2(\omega_2) \cdot f_1(\alpha) \\ &= f_3(\alpha) \cdot f_2(\omega_1 \cdot \omega_2) \cdot f_1(\alpha) \\ &= f_3(\alpha) \cdot f_2(\omega) \cdot f_1(\alpha)\end{aligned}$$

Case 3. $\mu \cdot \alpha = f_1(\alpha)$, $\mu \neq \lambda$, $\gamma = f_3(\alpha) \cdot \mu$.

The proof for Case 3 is similar to that for Case 2 and is left to the reader.

□

We consider first the problem of deciding equivalence of free, independent location schemas (FIL schemas) which are conditional-free. The proof of the following lemma is rather tedious, and is deferred to Section IV of this paper:

Lemma 2. Let S and S' be equivalent FIL schemas such that each is conditional-free and each contains some iteration statements. Let s be the last iteration statement in the program of S. Then there exists a statement s' in the program of S' such that s and s' are logic equivalent; moreover, s and s' have the same unbounded locations, and if the ith location of schema S is unbounded at s, then the ith location of schema S' is passive in all statements following s'.

Proof: See Section IV.

Corollary 2.1. Let S and S' be FIL schemas such that each is conditional-free and each contains some iteration statements. Let s be the last iteration statement in the program of S and let L_i be the test location of s. Then S and S' are equivalent only if s is logic equivalent to statement s', where s' is the last iteration statement in the program of S' with test location L_i.

Lemma 3. Let S and S' be conditional-free FIL schemas. Let s be the last iteration statement in the program of S, and let s' be an iteration statement in the program of S' such that s and s' are logic equivalent, each has the same unbounded locations, and each location unbounded at s is passive in all statements following s'. Then S and S' are equivalent if and only if they appear equivalent for all pairs of consistent computations C by S and C' by S' such that TESTS(s,C) and TESTS(s',C') each contain no more than two elements.

Proof: (Only If) Immediate.

(If) Suppose that S and S' appear equivalent for each pair of computations C and C' as above. Let C_o and C_o' denote arbitrary consistent computations by S and S', respectively, and let L_i be any location unbounded at s. (The reader will note that if no such location exists, other than the test location of s, the lemma is trivial.)

We may write the value associated with location L_i of S at the conclusion of computation C_o as $\gamma \cdot \beta \cdot \alpha$, where α is the value associated with L_i just prior to the execution of statement s, $\beta \cdot \alpha$ is the value associated with L_i immediately following the execution of statement s, and γ is the fixed portion of the final value due to the assignment statements involving L_i which follow statement s. In a similar manner, we may write the final value associated with location L_i of S' at the conclusion of computation C_o' as $\gamma' \cdot \beta' \cdot \alpha'$, where α' is the value associated with L_i just prior to the execution of statement s', $\beta' \cdot \alpha'$ is the value associated with L_i immediately following the execution of statement s', and γ' is the portion of the final value due to the execution of statements following s'.

By definition, the values associated with locations bounded at statement s are the same prior to and immediately following the execution of s during C_o. Since the locations unbounded at s are passive in all statements following statement s', we have that $\gamma \cdot \alpha = \gamma' \cdot \alpha'$, and hence that α' and γ' are completely determined by α. Also, it must be the case that β' is completely determined by β:

Suppose otherwise. Then there must be a test τ made during C_o prior to the execution of statement s such that β' is dependent on the outcome of τ, i.e. such that τ is made in C_o' during the execution of statement s'. Let L_j be the location of τ. Since S' is free, L_j must be unbounded at s' and hence, by hypothesis, passive in all statements following s'.

Let C_1' be the computation by S' consistent with C_o' except that the outcome of the first test made at s' during C_1' is <u>true</u> or <u>false</u>, according as s' is an UNTIL or WHILE statement. Clearly, C_1' is finite.

Let s" denote the statement in S at which test τ is made during computation C_o. Let C_1 be the computation by S consistent with C_o except that the outcome of test τ and each subsequent test made at statement s" is <u>true</u> or <u>false</u>, according as s" is a WHILE or UNTIL statement. C_1 is clearly infinite and is consistent with C_1'; also, TESTS(s,C_1) is empty, while TESTS(s',C_1') has but a single member, contradicting our hypothesis regarding the appearance of equivalence of S and S' for such computations.

From the preceeding arguments, it is clear that there exist functions f_1, f_2, and f_3 such that for all computations C_o and C_o' as above, $\alpha' = f_1(\alpha)$, $\beta' = f_2(\beta)$, and $\gamma' = f_3(\alpha)$. The reader may verify, moreover, that concatenation "distributes" over f_2. The lemma then follows from Lemma 1.

=

The following lemma provides us with the remaining result for our proof of the decidability of equivalence for conditional-free FIL schemas:

Lemma 4. Let S and S' be conditional-free FIL schemas such that each contains at least one iteration statement. Let k be the maximum of the sizes of S and S'. Then the problem of deciding the equivalence of S and S' reduces to the problem of deciding the equivalence of four pairs of conditional-free FIL schemas such that each schema in each pair is of size no greater than k-1.

Proof: Let s be the last iteration statement in the program of schema S, and let L_i be the test location of s. Let s' be the last iteration statement in the program of schema S' with test location L_i. We may assume that the statements s and s' have the same unbounded locations, and that the locations unbounded at s are passive in the statements following s'; otherwise we may immediately conclude that S and S' are not equivalent.

Let S_o be the schema obtained from S by removing statement s from the program of S, as well as all assignment statements involving location L_i which follow statement s. Let S_1 be the schema obtained from S by replacing statement s with the program of the subschema called in s, and then deleting the following assignment statements involving location L_i, as above. Let S_o' and S_1' be the schemas obtained in a similar manner from S', with statement s' in place of statement s. By Corollary 2.1, S is equivalent to S' only if S_o is equivalent to S_o', and S_1 is equivalent to S_1'. From Lemma 1, s is logic equivalent to s' if S_o is equivalent to S_o', and S_1 is equivalent to S_1'.

Let S_2 be the schema obtained from S by removing statement s from the program of S, and let S_3 be the schema obtained from S by replacing statement s with the program of the schema called in s. Let S_2' be the schema obtained from S' by deleting statement s' from the program of S', and let S_3' be the schema obtained from S' by replacing s' with the program of the schema called in s'. Assuming that s and s' are logic equivalent, S is equivalent to S' (by the previous lemma) if and only if S_2 is equivalent to S_2', and S_3 is equivalent to S_3'. From the remarks in the previous paragraph, then, S is equivalent to S' if and

only if S_i is equivalent to S_i', $0 \leq i \leq 3$. The lemma follows from the observation that $SIZE(S_i)$ is less than $SIZE(S)$, and $SIZE(S_i')$ is less than $SIZE(S')$, $0 \leq i \leq 3$.

□

We are now in a position to state our first decidability result:

Theorem 1: Let S and S' be conditional-free FIL schemas. Then it is decidable whether or not S and S' are equivalent.

Proof: By induction on the maximum of SIZE(S) and SIZE(S'), and the observation that equivalence is trivially decidable for schemas of size 0.

□

Our second decidability result concerns free restricted location schemas (FRL schemas) satisfying Property A. The proof of each of the following two lemmas is quite easy and is left to the reader:

Lemma 5. Let S and S' be equivalent FRL schemas satisfying Property A. Let $\mathcal{P}_W$ and $\mathcal{P}_U$ be the sets of predicate letters appearing in WHILE and UNTIL statements, respectively, in schema S, and let $\mathcal{P}_W'$ and $\mathcal{P}_U'$ be the sets of predicate letters appearing in WHILE and UNTIL statements in schema S'. Then $\mathcal{P}_W = \mathcal{P}_W'$ and $\mathcal{P}_U = \mathcal{P}_U'$.

Lemma 6. Let S and S' be equivalent FRL schemas satisfying Property A. Let p and p' be predicate letters appearing in iteration statements in schema S such that $p \succ p'$. Then $p \succ p'$ in schema S'.

The proof of the following lemma is also straightforward, but the arguments employed are of some interest:

Lemma 7. Let S and S' be equivalent FRL schemas satisfying Property A. Let p be a predicate letter appearing in a conditional statement in MAIN(S). Then either p is a non-productive predicate letter in S, or p appears in a conditional statement in MAIN(S').

Proof: Suppose that p is a productive letter. By definition of productivity, there exist interpretations for S which differ only with respect to the predicate assigned to p, and which are not equivalent with respect to S. Clearly, then, predicate letter p must appear in any schema equivalent to S, and hence must be in S'. From Lemma 5 and Property A, it must be the case that p labels a conditional statement in S'. Moreover, this conditional statement must be in MAIN(S'):

Suppose otherwise. Since p is productive in S, and S' and S are equivalent, p must be productive in S'. But, by hypothesis, p appears within some iteration subschema R of S'; hence, for each $n \geq 0$, there

exists an interpretation I_n for S' such that PART(S',ADJ(p,(I_n/p))) contains precisely 2^n equivalence classes. (We may choose for I_n any interpretation in which schema R is executed n times.) But this implies that S and S' are non-equivalent, since PART(S,ADJ(p,(I/p))) contains no more than two equivalence classes for any interpretation I for S, a contradiction. Hence, our hypothesis that p is not in MAIN(S) must be incorrect.

□

Arguments similar to those employed in the proof of Lemma 7 may be used to prove the following:

Lemma 8. Let S and S' be equivalent FRL schemas satisfying Property A. Let p be a predicate letter appearing in an iteration statement in schema S, and let p' be a predicate letter appearing in a conditional statement in S such that $p \succ p'$. Then either p' is non-productive in S, or $p \succ p'$ in schema S'.

Proof: Left to the reader.

The proof of the following result is extremely tedious and is left as an exercise for the ambitious reader:

Proposition. Let S, S' be FRL schemas satisfying Property A, such that the programs of S and S' are each free of conditional statements. Let I_i be an interpretation for the schemas such that the ith elements of VAL(S,I_i) and VAL(S',I_i) differ, and let I_j be an interpretation such that the jth elements of VAL(S,I_j) and VAL(S',I_j) differ. Then there exists an interpretation $I_{i,j}$ for the schemas such that both the ith elements and the jth elements of VAL(S,$I_{i,j}$) and VAL(S',$I_{i,j}$) differ.

Proof: See Section 5.

We now prove a result similar to that of Lemma 3 for a class of FRL schemas:

Lemma 10. Let S and S' be FRL schemas satisfying Property A, such that MAIN(S) and MAIN(S') are each free of conditional statements. Let s be the last iteration statement in the program of S, and let s' be an iteration statement in the program of S' such that s and s' are logic equivalent. Then S and S' are equivalent if and only if they appear equivalent for all pairs of consistent computations C by S and C' by S' such that TESTS(s,C) and TESTS(s,C') each contain no more than two elements.

Proof: (Only If) Immediate.

(If) Suppose that S and S' appear equivalent for each pair of computations C and C' as above. Let C_o and C_o' denote arbitrary consistent computations by S and S', respectively, let p be the predicate letter appearing in s and s', and let L_i be any location unbounded at s.

We may write the value associated with L_i of S at the conclusion of computation C_o as $\gamma \cdot \beta \cdot \alpha$, where α is the value associated with L_i just prior to the execution of statement s, $\beta \cdot \alpha$ is the value associated with L_i immediately following the execution of statement s, and γ is the fixed portion of the final value due to the assignment statements involving L_i which follow statement s. Similarly, we may write the final value associated with location L_i of S' at the conclusion of the computation C_o' as $\gamma' \cdot \beta' \cdot \alpha'$, where α' is the value associated with L_i just prior to the execution of statement s', $\beta' \cdot \alpha'$ is the value associated with L_i immediately following the execution of statement s', and γ' is the portion of the final value due to the execution of statements following s'.

Since S and S' satisfy Property A, it must be the case that each location unbounded at s is unbounded at s', and each location unbounded at s' is passive in all statements following s'. Also, since S is a FRL schema, the values associated with the locations bounded at s are the same immediately prior to and immediately following the execution

of s during C_o. Hence, we must have both α' and γ' completely determined by α. Also, β' must be completely determined by β:

Suppose otherwise. Then there must be some test made during the portion of C_o prior to the execution of statement s such that β' is dependent on the outcome of the test. Let p' be the predicate letter of this test. Clearly, p' is a productive letter in S' and $p \succ p'$ in S'. This implies that, for some interpretation I for S' such that precisely two tests are made at s' during the computation defined by I, PART(S',ADJ(p',(I,p'))) contains two equivalence classes. Let C_1 be a computation by S' from the first class, and C_2 a computation by S' from the second. Then any computation by S consistent with C_1 must also be consistent with C_2, since, by Lemma 8, p' cannot be productive in S, and thus PART(S,ADJ(p',(I/p')) has a single equivalence class, for any I. But this contradicts our assumption about the appearance of equivalence for S and S', since TESTS(s',C_1) and TESTS(s',C_2) each contain two elements, as must TESTS(s,C) for any computation C consistent with C_1 or C_2 (since s and s' are logic equivalent).

We must again, therefore, have functions f_1, f_2, and f_3 as in Lemma 3, and the result then follows from Lemma 1.

□

The proof of the following corollary is virtually identical to that of Lemma 4, and is left to the reader:

Corollary 10.1: Let S and S' be FRL schemas such that each satisfies Property A, and the programs of each are free of conditional statements. Let k be the maximum of SIZE(S) and SIZE(S'). Then the problem of deciding the equivalence of S and S' reduces to that of deciding the equivalence of four pairs of FRL schemas such that each schema in each pair is of size no greater than k-1 and satisfies Property A.

We need only show, to conclude that equivalence is decidable for FRL

schemas satisfying Property A, that the problem of deciding equivalence for a pair of such schemas reduces to the problem of deciding equivalence for a finite number of effectively constructed pairs of FRL schemas which satisfy Property A, have no conditional statements in their programs, and are each of size no greater than the maximum of the sizes of the schemas in the original pair. To that end, we have:

Lemma 11: Let S and S' be FRL schemas, let k be the maximum of SIZE(S) and SIZE(S'), and let ℓ and ℓ' be the number of conditional statements in MAIN(S) and MAIN(S'), respectively. Then we may construct from S and S' η_1 pairs of schemas such that:

i) Each schema in each pair is of size no greater than k.

ii) Each schema in each pair has a program devoid of conditional statements.

iii) S and S' are equivalent if and only if each constructed pair consists of equivalent schemas.

where η_1 is a constant bounded by $(n+\max(\ell,\ell'))\cdot 2^{\ell+\ell'}$.

Proof: We may assume that S and S' have the same number of locations, otherwise they cannot be equivalent. Let $L_1, \ldots, L_n$ be the symbols denoting these locations.

Let $\mathcal{P} = \{p_1, \ldots, p_m\}$ be the set of predicate letters which appear in conditional statements in both MAIN(S) and MAIN(S'). Let S_o and S_o' be the schemas constructed from S and S', respectively, by including m new location symbols $L_{n+1}, \ldots, L_{n+m}$ in the set of location symbols for the schemas, and adding, for each i, $1 \le i \le m$, the statement

$$L_{n+i} := L_j$$

to the beginning of the programs of each of the alternative subschemas called in the conditional statement in which predicate letter p_i appears, where L_j is the test location of that statement. (Intuitively, the added

locations are used to remember the value, if any, which is tested at the corresponding conditional statement during a computation by the schema.)

Let $p_1, \ldots, p_m, p_{m+1}, \ldots, p_\ell$ be an enumeration of the predicate letters appearing in conditional statements in MAIN(S), and let T be an arbitrary element of $\{\underline{\text{true}},\underline{\text{false}}\}^\ell$. Then the T-expansion of schema S_o, written $\langle T, S_o \rangle$, is the schema formed from S_o by replacing, for each i such that p_i appears in a conditional statement in the program of S_o, the statement containing p_i with the program of its true or false alternative subschema, according as the ith element of T is true or false, and repeating this recursively until the program of the resultant schema is devoid of conditional statements. In a similar manner, we may define $\langle T', S_o' \rangle$ for any T' in $\{\underline{\text{true}},\underline{\text{false}}\}^{\ell'}$.

A necessary and sufficient condition for the non-equivalence of S and S' is that there exist $T \in \{\underline{\text{true}},\underline{\text{false}}\}^\ell$ and $T' \in \{\underline{\text{true}},\underline{\text{false}}\}^{\ell'}$ such that, for some computation C by $\langle T, S_o \rangle$ and some consistent computation C' by $\langle T', S_o' \rangle$, the ith element of VAL(C, $\langle T, S_o \rangle$) differs from that of VAL(C', $\langle T', S_o' \rangle$) for some i, $1 \leq i \leq n$, and the jth element of VAL(C, $\langle T, S_o \rangle$) differs from that of VAL(C', $\langle T', S_o' \rangle$) for all j, $n+1 \leq j \leq n+m$, such that the (j-n)th element of T differs from that of T'. (If this condition holds, it indicates that during the computations by S and S' corresponding to C and C', tests made at main conditional statements containing the same predicate letters had opposite outcomes only if the predicate was evaluated at a different value in each case (and thus that the computations are consistent), and that the values of the computations were different, this in turn implying that S and S' are non-equivalent.)

We are thus faced with the problem of deciding, for a pair of FRL schemas R and R' whose programs are each devoid of conditional statements,

whether or not a pair of consistent computations for the schemas exists such that for some set of output locations $\{L_{i_1}, \ldots, L_{i_{n_o}}\}$, the value associated with L_{i_j} at the conclusion of the computation by R differs from the value associated with L_{i_j} at the conclusion of the computation by R', $1 \leq j \leq n_o$. For each such j, let R_j and R_j' be the schemas obtained from R and R', respectively, by appending to their programs the statement $L := L_{i_j}$ for each location symbol L in the schemas. From the Proposition, suitable computations for R and R' exist if and only if R_j and R_j' are not equivalent, $1 \leq j \leq n_o$.

We note that these last pairs of schemas are FRL schemas, are each of size no greater than k, and each have programs devoid of conditional statements. Verification of the bound on m is left as an exercise for the reader.

□

Finally, we have:

Theorem 2: Let S and S' be FRL schemas satisfying Property A. Then it is decidable whether or not S and S' are equivalent.

Proof: Follows immediately from Corollary 10.1 and Lemma 11, by induction on the maximum of SIZE(S) and SIZE(S').

□

IV. Discussion

i) Considering the application of Lemma 1 in the proofs of Theorems 1 and 2, the reader might be tempted to conclude that a pair of FIL schemas which are conditional-free, or a pair of FRL schemas which satisfy Property A, are equivalent if and only if they are equivalent for all consistent pairs of base computations, i.e. computations in which each instance of a subschema is executed no more than once. This conclusion cannot be justified on the basis of the previous proofs, however, since the relationships among the output locations do not remain unaffected during the procedure by which each pair of corresponding iteration statements is removed from two schemas being compared. Moreover, it can be demonstrated that the conclusion would be erroneous in each case. This is the object of the following two theorems:

Theorem 3. There exist conditional-free FIL schemas S and S' such that S and S' appear equivalent for all consistent pairs of base computations by the schemas, but are not equivalent schemas.

Proof: Let S and S' be the FIL, conditional-free schemas with programs P and P' as given below:

P:		P':	
	L := F(L)		L := F(L)
	WHILE p(L) DO S_1		L := F(L)
	L := F(L)		UNTIL p(L) DO S_1
	L := F(L)		L := F(L)
	WHILE p(L) DO S_1		WHILE p(L) DO S_1

where S_1 is the schema with program

L := F(L)

Clearly, S and S' are not equivalent. In particular, S converges and and S' diverges under the free interpretation which assigns to p the predicate Π_p defined as follows:

$$\Pi_p(F^i.\ \Delta) = \underline{\text{false}}, \quad 0 \le i \le 4$$
$$\Pi_p(F^i.\ \Delta) = \underline{\text{true}}, \quad i > 4$$

The reader may verify that S and S' appear equivalent for all pairs of consistent computations in which no instance of S_1 is executed more than once.

□

Theorem 4. There exist FRL schemas S and S' satisfying Property A such that S and S' appear equivalent for all consistent pairs of base computations, but are not equivalent schemas.

Proof: S and S' are the schemas with programs P and P' as given below:

P: WHILE $p(L_1)$ DO S_1
$L_3 := F(L_3)$
IF $p'(L_1)$ THEN S_2 ELSE S_3

P': WHILE $p(L_1)$ DO S_1'
$L_3 := F(L_3)$
IF $p'(L_1)$ THEN S_2 ELSE S_3

where S_1 has program $L_1 := F(L_1)$
$L_2 := F(L_2)$

S_1' has program $L_1 := G(L_1)$
$L_2 := F(L_2)$

S_2 has program $L_4 := L_2$
$L_1 := L_4$

and S_3 has program $L_4 := L_3$
$L_1 := L_4$

The reader may verify that S and S' are not equivalent, but appear to be so for all consistent pairs of base computations.

□

In each of the above proofs, S and S' demonstrate their non-equivalence for pairs of consistent computations in which no instance of a subschema is

executed more than twice. It is easy to construct, however, non-equivalent schemas which appear equivalent for all pairs of consistent computations in which no instance of a subschema is executed more than k times, for any $k > 0$.

ii) Extensions of the results:

We are able to show that equivalence is decidable for restricted location schemas satisfying Property A, even if they are not free.

We are also able to show that equivalence is decidable for conditional-free FRL schemas.

We conjecture that equivalence is decidable for conditional-free independent location schemas which are not free, although we are able to show that equivalence is <u>not</u> decidable for the class of such schemas which are restricted location, rather than independent location [1].

iii) Open questions:

Is equivalence decidable for FIL schemas in general?

Does the equivalence problem for independent location iteration schemas reduce to that for FIL schemas?

An answer to either question would be quite interesting; an affirmative answer to both would indicate that equivalence is decidable for multi-tape finite automata [5] and hence for independent location program schemas [2], via the results in [4].

V. Proof of Lemma 2.

For convenience, the Lemma is reproduced below:

Lemma 2. Let S and S' be equivalent FIL schemas such that each is conditional-free and each contains some iteration statements. Let s be the last iteration statement in the program of S. Then there exists a statement s' in the program of S' such that s and s' are logic equivalent; moreover, s and s' have the same unbounded locations, and if the ith location of schema S is unbounded at s, then the ith location of schema S' is passive in all statements following s'.

Proof: Let C be a computation by S such that TESTS(s,C) contains infinitely many elements, and let C' be a consistent computation by S'.

Clearly, every test in TESTS(s,C) must be in TESTS*(S',C'), for if some test $\tau \in$ TESTS(s,C) were not in TESTS*(S',C'), we could change the outcome of τ in C to obtain a finite computation C" by S consistent with computation C', contradicting the equivalence of S and S'.

Also, all but finitely many elements of TESTS(s,C) must be in TESTS(s',C') for some statement s' in schema S':

Suppose otherwise. Then there exist statements s_1 and s_2 in schema S' such that TESTS(s_1,C') and TESTS(s_2,C') each contain infinitely many elements of TESTS(s,C). (For simplicity, we shall assume that s_1 and s_2 are the only such statements.) We must have one of the statements, say s_2, nested within the iteration subschema called at the other statement. Let τ be any test in TESTS(s,C) $\cap$ TESTS(s_2,C'). We may change the outcome of τ in computation C to obtain a finite computation C" by S. But we can easily change the outcome of τ in C' without resulting in a finite computation: the first test made at statements s_1 after τ is made at s_2 must be free with respect to C" since the expression tested must be longer than any similar expression

tested during C". If s_1 is a WHILE(UNTIL) statement, we choose the outcome of this test to be true(false), as we do for each subsequent test made at s_1. The resultant computation is consistent with C" but is infinite, again contradicting the equivalence of S and S'. Hence, there must exist a single statement s' in schema S' such that all but finitely many elements of TESTS(s,C) are in TESTS(s',C').

We now show that the statement s' must be in the program of schema S':

Let L_j be the test location of statements s and s', and suppose that s' is not in the program of S', i.e. suppose that there is an iteration statement s_o in the program of S' such that s' is within the subschema called in s_o. Clearly, no statement following s_o in the program of S' can be an iteration statement with test location L_j, else we can obtain a finite computation by S and a consistent, infinite computation by S', by changing the outcome of any test $\tau \in$ TESTS(s,C) $\cap$ TESTS(s',C'), as above. Similarly, statement s_o itself cannot have test location L_j. Let τ' be the first test made at s_o with the property that, during the corresponding execution of the subschema called in s_o, s' makes some test $\tau'' \in$ TESTS(s,C), let C_1 be any computation by S consistent with C except for the outcome of test τ'' (if it is made during C), and let C_1' be any computation consistent with C' except for the outcome of test τ''. (We note that C_1 and C_1' may be chosen such that C_1 and C_1' are consistent, TESTS(s,C_1) is infinite, and C_1' is consistent with C' prior to the execution of statement s_o.) If we then apply the arguments of the previous paragraph, we have that there exists a statement s_1' in schema S' such that all but finitely many elements of TESTS(s,C_1) are in TESTS(s_1'',C_1'); moreover, s_1' must be within the iteration subschema called in some statement s_1 which follows s_o in the program of S' (since C' and C_1' are the same prior to the execution of s_o and, as noted, s_1' itself cannot be in the program of S'). But then by repeating this argument, we can generate an infinite sequence of state-

ments s_o, s_1, ... , s_k, s_{k+1}, ... which must follow one another in the program of S', violating the finiteness of its program. Hence, s' must be in the program of S', and must be the last iteration statement in the program with test location L_j. Moreover, from the preceeding argument, we may conclude that L_j is passive in all statements following s' in the program of S'.

The logic equivalence of s and s' is now easily demonstrated:

We note first that, for any pair of consistent computations C_o by S and C_o' by S', the last element of TESTS(s,C_o) must be the same as that of TESTS(s',C_o'). (Otherwise we may change the outcome of whichever test involves the longer expression, or either test if the expressions tested are of equal length, without causing a conflict in the computations. This will cause another test to be made at the corresponding statement, and since this test, and all subsequent tests made at the statement, is guaranteed to be free with respect to the computation by the other schema, we may permit one computation to diverge while still remaining consistent with the other, contradicting the equivalence of S and S'.) Since the schemas are free, however, the last elements of TESTS(s,C_o) and TESTS(s',C_o') will be the same for all consistent C_o and C_o' if and only if TESTS(s,C_o) and TESTS(s',C_o') are the same for all such computations, i.e. if and only if s and s' are logic equivalent.

It thus remains only to demonstrate that each location unbounded at s must be unbounded at s' (it is clearly the case that each location unbounded at s' must be unbounded at s if S and S' are to be equivalent) and passive in all statements following s'.

Suppose that this is not the case:

We note that if all locations unbounded at s are passive in all statements following s', then each such unbounded location must be unbounded at s' also. (Otherwise, there must exist some location L_i such that, for some computation by S, the length of the value

associated with L_i grows in proportion to the number of times the subschema of s is executed, while, for any computation by S', the length of the value associated with L_i is independent of the number of times the subschema of s' is executed, contradicting the equivalence of S and S'.) Hence, there must be some location which is unbounded at s and is active in some statement which follows s' in the program of S'. Let s" be this statement (with no loss of generality we may assume that there is only one such statement), and let L be the location. We shall assume for simplicity that L is the test location of s" - the generalization is straightforward. Clearly, L must be unbounded at s':

Suppose otherwise. We may construct a computation by S (and a consistent computation by S') in which the length of the value associated with L grows in proportion to the number of times the subschema of s is executed, and in which each test performed during the execution of s on a value associated with L has outcome <u>true</u> or <u>false</u>, according as s" is an UNTIL or WHILE statement. Let us denote by k the length of the value associated with L just prior to the execution of statement, and let us assume that we have terminated the execution of s after its subschema has been executed some number of times.

During the consistent computation by S', we know that if a test is made at s" on a value of length greater than k, then the test will either be free with respect to the computation by S (in which case we may terminate the execution of s"), or it will have outcome <u>true</u> (if s" is an UNTIL statement) or outcome <u>false</u> (if s" is a WHILE statement), in which case we <u>must</u> terminate the execution of s". In either case, it is clear that we may choose the computation by S' in such a way that the length of the value associated with L at the conclusion of the computation depends only on the portion of the

computation by S prior to the execution of statement s, and not on the number of times the subschema of s is executed. But then S and S' cannot be equivalent, since we have chosen the computation by S in such a way that the length of the final value associated with L is proportional to the number of times that the subschema of s has been executed.

We have, therefore, that L is unbounded at statement s'. An argument similar to that above can be used to show that if L is active in s, then L is active in s' (left as an exercise for the reader). But in fact, L must be active in statement s: otherwise, we choose a pair of consistent computations by the schemas satisfying the property that, during the execution of statement s', the value associated with location L in S' grows longer than that associated with L in S prior to the execution of statement s. Once this property is satisfied, we may terminate the execution of s' (and hence terminate the computation by S) and we are assured that the first test made at s" is free with respect to the computation by S. We assign to this test the outcome true if s" is a WHILE statement, or false if it is an UNTIL statement, and assign the same outcome to every subsequent test made at s". The resultant computation is clearly infinite and is consistent with the finite computation by S, contradicting the equivalence of S and S'. Thus L must be active in both s and s'.

Let s_L and s_L' be statements in the subschemas called in s and s', respectively, such that each has test location L. For simplicity, we assume that s_L and s_L' are the only such statements - again the generalization is straightforward. Let C_L' be a computation by S' such that infinitely many tests are made at s_L' during C_L' and no more than a single test is made at any other statement in S' during the computation; let C_L be a computation by S consistent with C_L'. No element of $TESTS(s_L, C_L)$ may be free with respect to C_L', otherwise we may clearly terminate the computation by S without conflicting with

computation C_L', thereby contradicting the equivalence of S and S'. In particular, there is some element τ which is in both $TESTS(s_L, C_L)$ and $TESTS(s_L', C_L')$. Let C_τ be the finite computation by S obtained by changing the outcome of τ in C_L and providing each subsequent test in the computation with outcomes appropriate for its termination. Let C_τ' be a computation by S' consistent with C_τ. We merely note that the first test made at s" during C ' must be free with respect to C_τ, as must each subsequent test made at s", and hence that C_τ' can be chosen so that it is infinite, again contradicting the equivalence of S and S'.

We have, therefore, that no such location L can exist, and thus that each location unbounded at s is unbounded at s' and is passive in all statements following s' in the program of S'.

□

REFERENCES

1. Leung, C., and Qualitz, J.
   Undecidability of Equivalence for Restricted Location Schemas.
   Computation Structures Group Note 21, MIT, Feb. 1975.

2. Luckham, D., Park, D., and Paterson, M.S.
   On Formalized Computer Programs.
   Journal of Computer and Systems Sciences, Vol. 4, No. 3, 1970.

3. Paterson, M.S.
   Equivalence Problems in a Model for Computation.
   Ph.D. Thesis, University of Cambridge, 1967.

4. Qualitz, J.E.
   Reducibility of the Equivalence Problem for Multi-tape Finite Automata to that for Independent Location Iteration Schemas.
   Computation Structures Group Memo 118, Mar. 1975.

5. Rabin, M., and Scott, D.
   Finite Automata and Their Decision Problems.
   IBM Journal of Research and Development 3,2 1959.