MIT/LCS/TM-156

DEFINABILITY IN DYNAMIC LOGIC

Albert R. Meyer

Rohit Parikh

February 1980

Definability in Dynamic Logic[1]

Albert R. Meyer[2] and Rohit Parikh[2,3]

Abstract: We study the expressive power of various versions of Dynamic Logic and compare them with each other as well as with standard languages in the logical literature. One version of Dynamic Logic is equivalent to the infinitary logic $L^{CK}_{\omega_1,\omega}$, but regular Dynamic Logic is strictly less expressive.

In particular, the ordinals ω^ω and $\omega^\omega\cdot 2$ are indistinguishable by formulas of regular Dynamic Logic.

0.Introduction. Dynamic logic, a language for expressing properties of programs, was introduced by Pratt in [8] and has, since then, been extensively studied, see [3]. However, as pointed out in [6], dynamic logic (or DL) has various versions depending on the class of programs admitted. These various versions do not all have the same expressive power.

DL resembles the predicate calculus in that its nonlogical symbols are *uninterpreted*. Thus its formulas do not have a full meaning until the interpretations of these symbols together with the range of the universe of discourse, are specified. Such an interpretation is called a *state*. Similarly, the program expressions of DL are program *schemes* and do not determine computations until the state is given. In any given state, the formulas acquire truth values and the program schemes can be executed.

One can think of a program scheme as an experiment on a state, exploring and modifying the state and thereby discovering its properties. We naturally expect that a language with more powerful program schemes can find out more about states than one with simpler program schemes, i.e., we expect it to be more expressive.

Since, as we shall see, the expressive power of a version of DL depends primarily on the class of its program schemes, we expect that there will be a correlation between the power of the language and the ease or difficulty of proving facts about the particular class of programs. Moreover, by measuring the expressive power of these versions of DL we can hope to get some insight into the properties of the classes of programs included in these languages. (See [6] for a discussion of this issue and a comparison of DL with other program logics.)

Definition 1: Let s $\models$A mean that the formula A is true in the state s. Let L be an uninterpreted language (say some version of DL) and P be a property of states, i.e. for every state s, either s has P or lacks P. We shall say that the property P is *expressible* in L if there is formula A of L such that for all states s, s has P iff s $\models$A.

Our basic question is: what properties of states are expressible in various versions of DL? A related question concerns a comparison of expressive power among various versions of DL, as well as standard languages in the literature.

Definition 2: Let L and M be two sets of logical formulas. L is ***no more expressive*** than M (L $\preceq$ M) if for every formula A in L, there is a formula B in M such that for every state s, s $\models$A iff s $\models$B. Similarly we say that L is *strictly less expressive* than M (L $\prec$ M) iff L $\preceq$ M and not M $\preceq$ L. Finally, L and M are *equally expressive* (L ~ M) if L $\preceq$ M and M $\preceq$ L.

We shall assume that the reader has some familiarity with dynamic logic, so we include here only a brief summary. Dynamic logic is an extension of the predicate calculus obtained by allowing the formula construct $\langle\alpha\rangle$A where α is a program and A is a formula that has already been formed at some previous stage. The state s satisfies $\langle\alpha\rangle$A iff there is a computation of (the possibly nondeterministic) program α which begins at the state s and terminates at another state s′ such that s′ satisfies A.

The various versions of DL arise because of choices in the class of programs admitted. These choices can be made at two different points.

(1) In the class of basic *instructions* allowed.

We may or may not allow *random* assignments of the form x←? (which change non-deterministically the value of x, but leave the state otherwise unchanged) and we may or may not allow *array* assignments which change the values of some given function symbol in the language. However we always allow ordinary assignments of the form x←t where t is any term in the language. There is also some choice as to the class of tests allowed. We may allow tests of the form A? for (a) atomic A or (b) arbitrary program free A or, most generously (c) arbitrary A in the language. This last version will be called "rich test", see [3].

(2) In the kind of *program* constructions allowed.

The strictest class of programs will be the class of regular programs, i.e., programs defined by finite flow charts. There is an alternative way to describe this class. Let a *seq* be a finite sequence of assignments (including array assignments and random assignments if these are allowed) and tests. Then a particular program execution consists of the execution of some seq. If we think of a program scheme as the set of all possible seqs which might get performed during any execution, i.e., all the seqs which are provided for in the program, then a regular program is one for which the corresponding set of seqs is regular.

Programs with recursive calls are the same as context free programs, i.e., the set of seqs is context free. The most general class of programs we shall consider here is the class of *recursively enumerable* programs, where any r.e. set of seqs is allowed. We note the important fact that in an r.e. program an infinite (r.e.) set of distinct tests can occur, whereas in regular or context free programs, the set of distinct tests is finite, though of course each test may occur in infinitely many seqs. *Finite test* DL (denoted DL_{ft} from now on) will be DL with the sole restriction that only finitely many distinct tests have occurrences in (the seqs of) any one program scheme. Of course it is permitted that a particular test has infinitely many occurrences in the seqs of some program scheme. Finite test DL includes both regular DL and context free DL as sublanguages. Moreover it also includes atomic-test r.e. DL. Since most programs considered in the literature use atomic tests, results about finite test DL will have general application. Note that in both cases we shall consider *rich test* versions of DL.

We shall also consider two variations of the well known infinitary language $L_{\omega_1,\omega}$, see [1,10]. The language $L^{CK}_{\omega_1,\omega}$ is like the predicate calculus but certain infinite disjunctions are allowed. (CK stands for "Church-Kleene".) Precisely, if $A_0, A_1, \ldots$ is an r.e. sequence of formulas of $L^{CK}_{\omega_1,\omega}$, then

$$\bigvee_i A_i$$

is also a formula of $L^{CK}_{\omega_1,\omega}$. Of course an infinite disjunction is true iff at least one of the disjuncts is true.

The other language, bounded alternation $L^{CK}_{\omega_1,\omega}$, denoted L_{ba}, is a sublanguage of $L^{CK}_{\omega_1,\omega}$ obtained essentially by restricting formula formation so that there is a fixed finite bound on the number of alternations of existential and universal quantifiers. We show the following results connecting these languages with various versions of DL.

(1) Rich test r.e. DL (from now on denoted DL_{re}) with/without random and/or array assignments is equal in expressive power to $L^{CK}_{\omega_1,\omega}$ (Theorem 1, section 2).

(2) Both regular and context free (rich test) DL are strictly weaker in expressive power than DL_{re} (corollaries to Theorem 2, section 3).

In obtaining result (2) above, we shall show (Theorem 2) that the language L_{ba} cannot distinguish between the ordinal ω^ω and $\omega^\omega \cdot 2$. Since $L^{CK}_{\omega_1,\omega}$ can define any recursive ordinal up to isomorphism, it can certainly define ω^ω, and hence $L_{ba} < L^{CK}_{\omega_1,\omega}$.

We further prove that DL_{ft} is no more expressive than L_{ba}, so it follows from (1) that DL_{ft} is strictly weaker than DL_{re}. We already remarked that regular and context free DL are special cases of DL_{ft}, and result (2) now follows.

The fact that ω^ω is indistinguishable from $\omega^\omega \cdot 2$ by formulas of DL_{ft} provides an explicit example of a limitation on the expressive power of even these powerful logics of programs. We remark that the above ordinals arise naturally in various contexts. For example, consider the set of all polynomials p, q with *integral* coefficients under the ordering $p < q$ iff $p(x) < q(x)$ for all sufficiently large x. This is a well ordering of type ω^ω. If we take two copies of this well ordering and put them end to end we get a well ordering of type $\omega^\omega \cdot 2$.

1.Basic Definitions.

Definition 3: We define the notions: *instruction*, *seq*, *program*, and *formula* of DL_{re} by simultaneous recursion using (A)-(D) below. If (C') is used instead of (C) then we get the corresponding notions for DL_{ft}.

(A)Instructions:

a) If y is a variable and t is a term then

$y \leftarrow t$ is an *assignment.*

For example, $x \leftarrow f(g(y,x),z)$ is an assignment.

(b) If g is an n-ary function symbol, $x_1,x_2,\ldots,x_n$ are variables, and t is a term, then

$g(x_1,\ldots,x_n) \leftarrow t$ is an *array assignment.*

(c) If y is any variable then

$y \leftarrow ?$ is a *random assignment.*

(d) If A is any formula then

A? is a *test.*

An *instruction* is either an assignment, an array assignment, a random assignment, or a test.

(B) A *seq* is a finite sequence of instructions.

(C) A *program* is an r.e. (recursively enumerable) set of seqs.

Note: For DL_{ft} we shall modify (C) to

(C') Let $\mathscr{F}$ be a finite set of tests and let α be an r.e. set of seqs such that for all seqs S in α, *only* the tests in $\mathscr{F}$ occur in S. Then α is a (finite test) program.

(D)Formulas:

(a) An n-ary relation symbol R followed by n terms is an *atomic formula.* (Equality is admitted as a binary relation.)

(b) If A,B are formulas, x is a variable, and α is a program, then

$$\neg A,\ (A \vee B),\ (\exists x)A, \text{ and } \langle\alpha\rangle A$$

are *formulas.*

A formal definition of the semantics of DL_{re} (which includes DL_{ft}) is described in [3] and [7].

Definition 4: We define the formulas of $L^{CK}_{\omega_1,\omega}$.

(a) Every atomic formula is a formula of $L^{CK}_{\omega_1,\omega}$.

(b) If A is in $L^{CK}_{\omega_1,\omega}$ and x is a variable, then

$(\exists x)A$ and $\neg A$ are formulas of $L^{CK}_{\omega_1,\omega}$.

(c) If $\{A_i | i>0\}$ is a recursive enumeration of formulas of $L^{CK}_{\omega_1,\omega}$ then

$\bigvee_i A_i$ is a formula of $L^{CK}_{\omega_1,\omega}$.

(Condition (c), "recursive enumeration", presupposes that we have a uniform way of assigning Godel numbers to the formulas. Such techniques are well known, [9], and we shall not repeat the details here. *Given* such an enumeration, a sequence of formulas is recursive iff there is a recursive function which enumerates the corresponding Godel numbers.)

The semantics of $L^{CK}_{\omega_1,\omega}$ is very much like that of first order logic. A state satisfies the r.e. disjunction $\bigvee_i A_i$ iff it satisfies at least one of the disjuncts. Finite disjunction need not be introduced separately as it is a special case of the r.e. disjunction.

Definition 5: We define L_{ba}, bounded alternation $L^{CK}_{\omega_1,\omega}$ as follows. For each n, let L_n, L'_n be the following sets of formulas:

(a) L_0 = all atomic formulas,

(b) L'_n = closure of L_n under negation and r.e. disjunctions,

(c) L_{n+1} = the closure of L'_n under *existential* quantification.

Then $L_{ba} = \bigcup_{n=1} L_n = \bigcup_{n=1} L'_n$.

2. The expressive power of DL_{re}. In this section we show that DL_{re} with or without array assignments and with or without random assignments, is equivalent in expressive power with $L^{CK}_{\omega_1,\omega}$.

Lemma 1: $L^{CK}_{\omega_1,\omega} \leq DL_{re}$.

Proof: We want to define a map ϕ from $L^{CK}_{\omega_1,\omega}$ to DL_{re} such that for all formulas A of $L^{CK}_{\omega_1,\omega}$, A holds in precisely the same states in which $\phi(A)$ does. We define ϕ by induction on the complexity of A.

(1) If A is atomic, then $\phi(A) = A$,

(2) $\phi(\neg A) = \neg\phi(A)$, and $\phi((\exists x)A) = (\exists x)\phi(A)$,

(3) If $A = \bigvee_i A_i$, let α be the program whose seqs are of length one and consist of precisely the set of tests $\phi(A_i)?$. Then

$$\phi(A) = \langle\alpha\rangle \mathit{true}. \blacksquare$$

Note that the proof of Lemma 1 did not use array assignments or random assignments.

Now we show that DL_{re} *with* random assignments and array assignments is no more expressive than $L^{CK}_{\omega_1,\omega}$. This will show that DL_{re} with/without random and/or array assignments is equally expressive as $L^{CK}_{\omega_1,\omega}$.

Lemma 2: $DL_{re} \leq L^{CK}_{\omega_1,\omega}$.

Proof: We define a map ψ from DL_{re} to $L^{CK}_{\omega_1,\omega}$. We shall also need a map η_α for each program α of DL_{re} from $L^{CK}_{\omega_1,\omega}$ to itself. Since programs and formulas are interdependent, η_α and ψ are defined together. Intuitively, $\eta_\alpha(A')$ provides a translation for $\langle\alpha\rangle A$, if A' provides a translation for A.

I. (1) If A is atomic, then $\psi(A) = A$,

(2) $\psi(\neg A) = \neg\psi(A)$, and $\psi((\exists x)A) = (\exists x)\psi(A)$,

(3) $\psi(\langle\alpha\rangle A) = \eta_\alpha(\psi(A))$ where η_α is defined below.

II. (1) If β is an instruction then

(a) If β is $x \leftarrow t$ then $\eta_\beta(A)$ is the formula obtained by replacing x by t in A. Bound variables are renamed if necessary to avoid conflicts.

(b) If β is B? , then $\eta_\beta(A)$ is $\psi(B) \rightarrow A$,

(c) If β is $x \leftarrow ?$, then $\eta_\beta(A)$ is $(\exists x)A$,

(d) If β is $g(u) \leftarrow t$

(β could be a more complicated array assignment; we use this case for illustration), then first rewrite A so that all terms are of height at most one, and terms of height one occur *only* in equalities. For example, $R(g(f(x)))$ becomes

$$(\exists y)(\exists z)(R(y) \wedge y = g(z) \wedge z = f(x)).$$

Note that this converts atomic formulas into more complex formulas and maps A to an equivalent formula A′.

Now every subformula $y = g(v)$ of A′ is replaced by

$$(v = u \wedge y = t) \vee (v \neq u \wedge y = g(v)).$$

The resulting formula is $\eta_\beta(A)$.

(2) If β is a seq $\beta_1;\ldots;\beta_m$, then

$$\eta_\beta(A) = \eta_{\beta_1}(\eta_{\beta_2}(\ldots\eta_{\beta_m}(A)\ldots)).$$

(3) If β is a program whose seqs are $\{\beta_i\}$, then

$$\eta_\beta(A) = \bigvee_i \eta_{\beta_i}(A).$$

The proof that A and $\psi(A)$ are equivalent is straightforward and we omit it.∎

We have now proved

Theorem 1: Rich test r.e. DL with or without array assignments and/or random assignments is equally expressive as $L^{CK}_{\omega_1,\omega}$.

Remark: We point out that regular DL with random assignments and the operator loops_α (see [4],[7]) cannot be reduced to $L^{CK}_{\omega_1,\omega}$. This is because well-ordering cannot be defined in $L^{CK}_{\omega_1,\omega}$ [5].
But if $<$ is a linear order on some set, and if α is the program

$$x \leftarrow ?;\ (y \leftarrow ?;\ y < x?;\ x \leftarrow y)^*$$

then loops_α holds iff $<$ is not a well order.

3.Expressive power of DL_{ft}. In this section we show that DL_{ft} is no richer than L_{ba} and that L_{ba} is strictly less expressive than $L^{CK}_{\omega_1,\omega}$. Thus,

$$DL_{ft} \leq L_{ba} < L^{CK}_{\omega_1,\omega} \sim DL_{re}.$$

This shows that DL_{ft} is strictly less expressive than DL_{re}. Since both regular and context free DL are included in DL_{ft}, we get as a corollary that regular DL and context free DL are strictly poorer in expressive power than DL_{re}.

Lemma 3: $DL_{re} \leq L_{ba}$. *Proof*: The proof is quite like that of Lemma 2 of Section 2 and we omit it.

See [9] for the definition of recursive ordinals.

Lemma 4: For each recursive ordinal α there is a formula $A_\alpha(c,d)$ of $L^{CK}_{\omega_1,\omega}$ such that the structure $(D, <, c, d)$ satisfies $A_\alpha(c,d)$ iff

$<$ is a linear order on D,
$c < d$, and
the open segment (c,d) has order type α.

Proof: First define A'_α by recursion on α:

(1) If $\alpha = 0$, then $A'_\alpha(c,d) = c<d \wedge \neg(\exists x)(c<x<d)$,

(2) if $\alpha = \beta+1$, then $A'_\alpha(c,d) = (\exists z)(A'_\beta(c,z) \wedge A'_0(z,d))$,

(3) if $\alpha = \lim \{\beta_i\}$ where $\{\beta_i\}$ is a recursive sequence of the ordinals $< \alpha$, and A'_{β_i} is the corresponding sequence of formulas, then $A'_\alpha(c,d) = (\forall z)(c<z<d \rightarrow \bigvee_i A_{\beta_i}(c,z)) \wedge \neg(\bigvee_i (\forall z)(c<z<d \rightarrow \neg A_{\beta_i}(c,z)))$.

Finally $A_\alpha(c,d)$ is $A'_\alpha(c,d) \wedge B$ where B is the conjunction of the axioms for a linear ordering.∎

We now adapt the technique of Ehrenfeucht-Fraisse games [2] to L_{ba} to show that Lemma 4 fails for L_{ba}.

Definition 6: Let D be the set of all ordinals less than ω^ω and G be the set of all ordinals less than $\omega^\omega \cdot 2$. We define for $i \in \mathbb{N}$ the relation $\equiv_i$ between n-tuples from D and n-tuples from G by recursion on i. Let $\delta_1,\dots,\delta_n$ be in D and $\gamma_1,\dots,\gamma_n$ be in G.

(1) $(\delta_1,\dots,\delta_n) \equiv_0 (\gamma_1,\dots,\gamma_n)$

iff for all j, k $\leq$ n,

$$\delta_j < \delta_k \text{ iff } \gamma_j < \gamma_k.$$

(2) $(\delta_1,\dots,\delta_n) \equiv_{i+1} (\gamma_1,\dots,\gamma_n)$

iff for all $m \geq 1$ and all $\delta'_1,\dots,\delta'_m$ in D there exist $\gamma'_1,\dots,\gamma'_m$ in G such that $(\delta_1,\dots,\delta_n, \delta'_1,\dots,\delta'_m) \equiv_i (\gamma_1,\dots,\gamma_n,\gamma'_1,\dots,\gamma'_m)$ and vice-versa with D, G interchanged.

Lemma 5: Let $A(x_1,\dots,x_n) \in L'_i$ and $(\delta_1,\dots,\delta_n) \equiv_i (\gamma_1,\dots,\gamma_n)$. Then $A(\delta_1,\dots,\delta_n)$ holds in $(D, <)$ iff $A(\gamma_1,\dots,\gamma_n)$ holds in $(G, <)$.

Proof: Clearly true if i = 0.

For the inductive step notice that the transition from L_{i+1} to L'_{i+1} is purely truth functional. Thus if $(\delta_1,\dots,\delta_n)$ and

$(\gamma_1,\dots,\gamma_n)$ satisfy the same formulas of L_{i+1}, then they satisfy the same formulas of L'_{i+1}.

Hence to prove the result for L'_{i+1} it is enough to show it for L_{i+1}.

So suppose $(\delta_1,\dots,\delta_n) \equiv_{i+1} (\gamma_1,\dots,\gamma_n)$, A is of the form $(\exists y_1\dots\exists y_m)B(x_1,\dots,x_n,y_1,\dots,y_m)$ where $B \in L'_i$. Suppose now that $A(\delta_1,\dots,\delta_n)$ holds. Then there exist $\delta'_1,\dots,\delta'_n \in D$ such that $B(\delta_1,\dots,\delta_n,\delta'_1,\dots,\delta'_m)$ holds. Now by definition of $\equiv_{i+1}$, there exist $\gamma'_1,\dots,\gamma'_m$ such that $(\delta_1,\dots,\delta_n,\delta'_1,\dots,\delta'_m)$ $\equiv_i (\gamma_1,\dots,\gamma_n,\gamma'_1,\dots,\gamma'_m)$.

Hence, by induction hypothesis, $B(\gamma_1,\dots,\gamma_n,\gamma'_1,\dots,\gamma'_m)$ holds, and hence $(\exists y_1\dots\exists y_m)B(\gamma_1,\dots,\gamma_n,y_1\dots y_m)$ holds.∎

To proceed further we use certain basic facts about ordinal addition. If $m < n$ then $\omega^m+\omega^n=\omega^n$. For example, $(\omega^3+\omega^2+\omega+1) + (\omega^2+\omega\cdot 2+7) = \omega^3+\omega^2.2+\omega\cdot 2+7$; the $\omega+1$ is absorbed by the following ω^2. Nonetheless, given ordinals $\alpha < \beta$, there is a *unique* γ such that $\alpha+\gamma=\beta$. We shall write γ as $\beta-\alpha$. For convenience, define $\beta-\alpha$ to be 0 if $\alpha \geq \beta$.

Moreover, given an ordinal α, and $i > 0$, we can write α uniquely in the form

$$\alpha = \omega^i.\beta + \omega^{i-1}.n_{i-1} + \cdots + n_0$$

where β is an ordinal and $n_{i-1},\dots,n_0$ are natural numbers.

Definition 7: For any ordinal α and $i > 0$, the i-normal form of α, written $\|\alpha\|_i$, is

$$\omega^i + \omega^{i-1}.n_{i-1} + \cdots + n_0, \qquad \text{if } \beta \neq 0, \text{ and}$$

$$\omega^{i-1}.n_{i-1} + \cdots + n_0, \qquad \text{if } \beta = 0.$$

Let $\|\alpha\|_0$ be zero if $\alpha = 0$ and one if $\alpha > 0$.

Definition 8: Let $\delta_1,\dots,\delta_n \in D$ and $\gamma_1,\dots,\gamma_n \in G$. Then for $i \in \mathbb{N}$,

$$(\delta_1,\ldots,\delta_n) \approx_i (\gamma_1,\ldots,\gamma_n)$$

iff

(1) $\|\delta_j-\delta_k\|_i = \|\gamma_j-\gamma_k\|_i$ for all $j,k \le n$, and

(2) if δ_j, γ_j are the least elements of their respective n-tuples then $\|\delta_j\|_i = \|\gamma_j\|_i$.

Lemma 6: If $(\delta_1,\ldots,\delta_n) \approx_i (\gamma_1,\ldots,\gamma_n)$, then $(\delta_1,\ldots,\delta_n) \equiv_i (\gamma_1,\ldots,\gamma_n)$.

Proof: By induction on i.

(1) Def.8(1) implies Def.6(1) in the case i=0, since $\alpha < \beta$ iff $\|\beta-\alpha\|_0 = 1$.

(2) Suppose true at i. To show at i+1, let us simplify notation by looking at the case n=2 and say that $(\delta_1,\delta_2) \approx_{i+1} (\gamma_1,\gamma_2)$, $\delta_1 < \delta_2$, and $\gamma_1 < \gamma_2$. Suppose that we are given additional elements $(\gamma'_1,\ldots,\gamma'_m)$ in *G*; we shall find $(\delta'_1,\ldots,\delta'_m)$ in *D* such that $(\delta_1,\delta_2,\delta'_1,\ldots,\delta'_m) \approx_i (\gamma_1,\gamma_2,\gamma'_1,\ldots,\gamma'_m)$. Then we will have, by induction hypothesis, that $(\delta_1,\delta_2,\delta'_1,\ldots,\delta'_m) \equiv_i (\gamma_1,\gamma_2,\gamma'_1,\ldots,\gamma'_m)$,so that $(\delta_1,\delta_2) \equiv_{i+1} (\gamma_1,\gamma_2)$ as required.

Now the new γ' fall into three groups. Those less than γ_1, those greater than γ_2, and those between γ_1 and γ_2. Consider for instance the last group.

We know that $\gamma_2 - \gamma_1$ has the same i+1 form as $\delta_2 - \delta_1$. Either it begins with ω^{i+1} or it does not. If not, $\delta_2 - \delta_1$ is exactly equal to $\gamma_2 - \gamma_1$ and we can find exactly matching elements in *D*. If so, recall that ω^{i+1} contains ω copies of ω^i. This enables us to choose the δ' as follows. Suppose that γ'_3 and γ'_5, in that order, are the new γ' between γ_1 and γ_2. To find the corresponding δ', let δ'_3 be $\delta_1+\|\gamma'_3-\gamma_1\|_i$ and let δ'_5 be $\delta'_3+\|\gamma'_5-\gamma'_3\|_i$. It is readily checked that in that case we will still have $\delta'_5<\delta_2$ and that $\|\delta_2-\delta'_5\|_i = \|\gamma_2-\gamma'_5\|_i$.

(In general we requuire the following easily established fact connecting the function $\|-\|_i$ and ordinal addition: suppose $\|\beta\|_{i+1} = \|\alpha\|_{i+1}$ and $\beta = \beta_1+\ldots+\beta_p$, then there exist $\alpha_1,\ldots,\alpha_p$ such that $\alpha = \alpha_1+\ldots+\alpha_p$ and for all $j<p$, $\|\alpha_j\|_i = \|\beta_j\|_i$.)

We can deal with the other two groups, the γ' that are less than γ_1, and those that are greater than γ_2 in a similar way. One extra property is needed in the last case, namely, that ω^ω-α and $\omega^\omega \cdot 2$-α can only equal $\omega^\omega \cdot 2$, ω^ω, or 0 as α ranges over all ordinals. We omit the details. ∎

Theorem 2: Any closed formula of L_{ba} that holds in $(\omega^\omega, <)$ holds in $(\omega^\omega \cdot 2, <)$ and vice versa.

Proof: For all i, the zero-tuple of ω^ω is $\approx_i$ to the zero-tuple of $\omega^\omega \cdot 2$. This is because Definition 8 of $\approx_i$ holds vacuously. Now apply Lemma 6. ∎

Corollary: Both rich test regular DL and rich test context free DL are strictly weaker in expressive power than rich test r.e. DL.

Proof: This follows immediately from Theorem 2 above together with Theorem 1 and Lemmas 3 and 4. ∎

The fact that $L_{ba} < L^{CK}_{\omega_1,\omega}$ could also be proved by standard model theoretic methods [1] if we notice that L_{ba} only contains formulas of $L^{CK}_{\omega_1,\omega}$ whose ordinal height is less than ω^2. However we have preferred to give a proof that proceeds directly and yields an explicit example of the difference in their expressive power.

4. Further Results

We briefly note a few further results which we do not have time to elaborate here.

We saw in Section 2 that DL_{re} does not change in expressive power if array assignments or random assignments are included. However, this is not true for regular DL. In particular there is a formula A of regular DL with array assignments and random assignments such that A holds in a state s iff the domain of s is finite.

Such a formula is constructed by the following method: we construct, by means of array assignments and random choices, a function which, by repeated compositions, forms a loop that covers *all* the elements in the universe. Clearly such an attempt can succeed iff the domain is finite. The formula A merely says that such an attempt can succeed.

The same technique which allows constructon of the formula which defines finiteness can be extended to show that any ordinal less than ω^{ω} *is* definable up to isomorphism in regular DL with random and array assignments. In fact, it allows us to show that DL with array and random assignments is actually equivalent to weak second order predicate calculus. We conjecture that the latter is strictly less expressive than L_{ba}.

However a formula defining finiteness *cannot* be obtained if either array assignments or random assignments are left out. This is a joint result with K. Winklmann.

These and related results will be elaborated in a later paper.

Notes

1. This research was supported in part by NSF grants numbered MCS77 19754 and MCS79 10261.

2. Laboratory for Computer Science, MIT, Cambridge, Mass.

3. Mathematics Department, Boston University, Boston, Mass.

References

[1] J. Barwise, Back and forth through Infinitary Logic, *Studies in Model Theory*, Mathematical Association of America, 1973.

[2] A. Ehrenfeucht, An Application of Games to the Completeness Problem for Formalised Theories, *Fundamenta Mathematicae* 49, pp. 129-141.

[3] D. Harel, First Order Dynamic Logic, *Lecture Notes in Computer Science* 68, Springer, New York, 1979.

[4] D. Harel and V. R. Pratt. Nondeterminism in Logics of Programs, *5th Annual Symposium on Principles of Programming Languages*, January 1978, 203-213.

[5] J. Keisler, *Model Theory for Infinitary Logic*, North Holland 1971.

[6] A. Meyer, Ten Thousand and One Logics of Programming, *EATCS Bulletin*, to appear 1980.

[7] Meyer, A.R. and K. Winklmann, On the Expressive Power of Dynamic Logic, *Proc. of the 11th Annual ACM Conf. on* Theory *of Computing*, Atlanta, Ga., May 1979, 36 pp.

[8] V. Pratt, Semantical Considerations in Floyd-Hoare Logic, *Proc. 17th IEEE Symp. on FOCS*, 1976, pp. 109-121.

[9] H. Rogers, *The Theory of Recursive Functions*, McGraw-Hill, 1967.