Propositional Logics of Programs: New Directions

Rohit Parikh
Department of Computer Science
Brooklyn College
Brooklyn, NY, 11210

Abstract: We discuss new developments in Propositional Logic of Programs, specifically the μ-calculus, and Game Logic. While this paper is intended largely as a survey, some new results are stated.

Introduction: Since the introduction of Propositional Dynamic Logic (PDL) by Fischer and Ladner, many developments have taken place. Completenss proofs for the new logic have been provided in [P1], [KP] and other places. Extensions of the logic have been considered, with converse operator and the loop construct [P1], [St], and there have been side trips into Process Logic, Temporal Logic, etc. Many of these have already been reported in the survey paper [P2]. In this paper we want to discuss some developments during the last two years, specifically in the μ-calculus and Game Logic.

Most of these propositional logics are decidable, with the exception of CFPDL, the extension of PDL which allows recursive calls. The last has been shown to be π^1_1 complete in [HPS], The decidability of those logics which *are* decidable can always be proved by reduction to SnS, but sometimes there is a finite model property which gives a direct decidability proof and a much better complexity bound. A natural question to ask is: what is the largest propositional logic in this general area which is decidable? The μ-calculus and Game Logic are two quite natural candidates and are studied in sections 3 and 4 respectively. In section 5 we give an application of many-person Game Logic to proving the correctness of a cake cutting algorithm.

§2:*Preliminaries*: We assume that the reader has some familiarity with Dynamic Logic. However, just for the sake of completeness we give a few definitions. We have a state space W and a language L^- which can be used to make statements about individual states. Thus the notion $s \Vdash A$ where s is a state and A is a formula of L^-, is assumed to be defined, and means

intuitively: the formula A is true at the state s. We assume also that we have some actions on W which may take us from one state to another, usually in a non-deterministic way.

Now we augment the language L^- to a larger language L which is closed under the operations of L^- and also under the modalities $\langle\alpha\rangle$ and $[\alpha]$ where α is any action. Thus if A is a formula of L then $s \Vdash \langle\alpha\rangle A$ means that there is a t such that the action α *may* take us from s to t and $t \Vdash A$. Similarly, $s \Vdash [\alpha]A$ iff for all t such that α can take us from s to t, $t \Vdash A$. For example, if a cake is being cut between two people by the action c, F_1 means that the first piece is fair (one person's fair share) and F_2 means that the second piece is fair, Then we have $[c](F_1 \vee F_2)$ and also $\langle c\rangle(F_1 \wedge F_2)$ I.e. no matter how the cake is cut, at least one piece is fair, and it is possible to cut it so that both pieces are fair.

The various actions are closed under certain operations like (We shall use the convention of using letters a,b,c for atomic actions or programs and α, β etc for complex ones):

$(\alpha;\beta)$: first do α and then β.

α^*: do α some finite number of times (maybe 0)

if A then α else β

$\alpha \cup \beta$: Do either α or β non-deterministically.

etc. See [FL] or [P1] or [KP] for a fuller discussion.

§3: *The μ-calculus*: A very natural way to arrive at the μ-calculus is to look at some of the axioms for PDL. The following are easily seen to be valid.

(1) $\langle\alpha;\beta\rangle A \equiv \langle\alpha\rangle\langle\beta\rangle A$

(2) $\langle\alpha \cup \beta\rangle A \equiv \langle\alpha\rangle A \vee \langle\beta\rangle A$

(3) $\langle\alpha^*\rangle A \equiv A \vee \langle\alpha\rangle\langle\alpha^*\rangle A$

Notice that axioms 1 and 2 allow us to eliminate the program connectives ; and ∪. However, the connective * cannot be eliminated so easily since it appears on both sides of its own definition. But if we write X for $\langle\alpha^*\rangle A$, then X satisfies:

(4) $X \equiv A \vee \langle\alpha\rangle X$

Unfortunately, (4) does not have a unique solution. E.g. W (the space of *all* states) may be a solution. What characterises $\langle\alpha^*\rangle A$ among all solutions is that it is the least, i.e. it holds of a state only when it *has* to, by (4). Thus we can write

$\langle\alpha^*\rangle A \equiv \mu X(X \equiv A \vee \langle\alpha\rangle X)$

or, with a small abuse of notation,

$\langle\alpha^*\rangle A \equiv \mu X(A \vee \langle\alpha\rangle X)$

It turns out that the notion of "least set satisfying a condition" is strong enough for us to define all program connectives of PDL and more.

Actually, not every condition defines a least set satisfying it. E.g. there is no least infinite subset of a given set. We need to say that the condition we are looking at is special in some way, and it is sufficient that the condition be monotonic. Specifically, we have the theorem:

Theorem: (Tarski-Knaster) let ϕ be a monotonic operator on some set W. I.e. for $X,Y \subseteq W$, if $X \subseteq Y$ then $\phi(X) \subseteq \phi(Y)$. Then there is a smallest subset X_0 of W such that $\phi(X_0)=X_0$.

This allows us to define the language of the μ-calculus as follows (see [Pr] for a related but slightly different treatment of the μ-calculus. Our version is taken from [K])

atomic formulae: $P_1,\ldots,P_n$

Set parameters: $X_1,\ldots,X_i,\ldots$

atomic programs: $a_1,\ldots,a_m$

Formulae: Each atomic formula is a formula

Each set parameter is a formula

If A, B are formulae, so are $\neg A$ and $A \vee B$

If A is a formula and a is an atomic program, then $\langle a\rangle A$ is a formula

If A is a formula with parameter X and all occurrences of X in A are positive, then $\mu X(A(X))$ is a formula.

Semantics: A model M gives a universe W together with extensions $\pi(P) \subseteq W$ for all atomic formulae P and $\rho(a) \subseteq W \times W$ for all atomic programs a. Suppose that the formula A contains (at most) the parameters $X_1,\ldots,X_k$ and $U_1,\ldots,U_k$ is a sequence of subsets of W. Then we define $\pi((M,U_1,\ldots,U_k), A)$ by induction on the complexity of A as follows:

$\pi((M,U_1,\ldots,U_k), P_i) = \pi(P_i)$

$\pi((M,U_1,\ldots,U_k), X_i) = U_i$

$\pi((M,U_1,\ldots,U_k),\neg A) = W - \pi((M,U_1,\ldots,U_k), A)$

$\pi((M,U_1,\ldots,U_k),\ A\vee B)$

$= \pi((M,U_1,\ldots,U_k),\ A) \cup \pi((M,U_1,\ldots,U_k),\ A)$

$\pi((M,U_1,\ldots,U_k),<a>A)$

$= \{s \mid \exists t,\ t\in \pi((M,U_1,\ldots,U_k),A) \wedge (s,t) \in \rho(a)\}$

$\pi((M,U_1,\ldots,U_{k-1}),\mu X_k(A(X_k)) =$ least U_k such that

$\pi((M,U_1,\ldots,U_k),\ A) = U_k$

If A has no parameters, then A is closed and we write $M,s\models A$ to indicate that $s\in\pi(M,A)$, or A is true at s in M. A is satisfiable if there is an M,s such that $M,s\models A$ and valid if for all M,s, $M,s\models A$.

It is shown in [KP2] that the validity problem for the μ-calculus is decidable. However, the decision procedure is not elementary. It is shown in [K] that there is an exponential time decision procedure for a *restricted* version of the μ-calculus, but the question for the full version is open.

The μ-calculus contains the propositional Game Logic which we shall discuss next. Now the μ-calculus includes Game Logic which includes PDL$^\rhd$ which *properly* includes PDL. Thus all of these logics are proper extensions of PDL. It would be very nice to be able to show that the μ-calculus is the *maximal* logic satisfying certain conditions, rather like Lindstrom's characterisation of first order logic.

§4.*Game Logic*: In this section we describe a Propositional Logic of Games which lies in expressive power between PDL and the μ-calculus. It is stronger than the first, but *might* be equal in expressive power to the second.

If we think of $<\alpha>$ and $[\alpha]$ of PDL as predicate transformers, they are rather similar to the quantifiers $\exists$ and $\forall$ of First Order Logic. Indeed, the quantifiers can be reduced to the two modalities if random assignments are available, see [MP]. Following Ehrenfeucht, we can think of the modalities $<\alpha>$ and $[\alpha]$ as (one or more) moves by players I and II respectively. Thus the modalities of PDL become simple games and just as we can combine programs using the constructs $\cup$, ; and *, games can also be combined in the same way. With this preamble we proceed to a detailed discussion.

We assume that the reader is familiar with the syntax and

semantics of PDL and the μ-calculus. (See [FL], [KP1], [P] for the first and [K] for the second.) However, those for Game Logic *can* be defined independently. We have a finite supply $g_1,\ldots,g_n$ of atomic games and a finite supply $P_1,\ldots,P_m$ of atomic formulae. Then we define games α and formulae A by induction.

1. Each P_i is a formula.
2. If A and B are formulae, then so are $A\vee B$, $\neg A$.
3. If A is a formula and α is a game, then $(\alpha)A$ is a formula.

4. Each g_i is a game.
5. If α and β are games, then so are $\alpha;\beta$ (or simply $\alpha\beta$), $\alpha\vee\beta$, $\alpha\wedge\beta$, $\langle\alpha^*\rangle$, $[\alpha^*]$ and α^d.
6. If A is a formula then $\langle A\rangle$ and $[A]$ are games.

Intuitively, the games can be explained as follows. $\alpha;\beta$ is the game: play α and then β. The game $\alpha\vee\beta$ is: player I decides whether α or β is to be played, and then the chosen game is played. The game $\alpha\wedge\beta$ is similar except that player II makes the decision. In $\langle\alpha^*\rangle$, the game α is played repeatedly (perhaps zero times) until player I decides to stop. If he never says "stop" then he loses. He may not stop in the middle of some play of α. Similarly with $[\alpha^*]$ and player II. If during any of these games, a player whose move it is finds he cannot play, then he loses automatically. Thus a stalemate is not a draw. In α^d, the two players interchange roles. Finally, with $\langle A\rangle$, the formula A is evaluated. If A is false, then I loses, otherwise we go on. (Thus $\langle A\rangle B$ is equivalent to $A\wedge B$.) Similarly with $[A]$ and II.

Formally, a model for game logic consists of a set W of worlds, for each P a subset $\pi(P)$ of W and for each primitive game g a subset $\rho(g)$ of $W\times P(W)$, where $P(W)$ is the power set of W. $\rho(g)$ must satisfy the monotonicity condition: if $(s,X)\in\rho(g)$ and $X\subseteq Y$, then $(s,Y)\in\rho(g)$. In other words, $\rho(g)$ thought of as an operator $\rho(g)(X) = \{s\mid(s,X)\in\rho(g)\}$ is monotonic in X. We define $\pi(A)$ and $\rho(\alpha)$ for more complex formulae and games as follows:

2'. $\pi(A\vee B) = \pi(A)\cup\pi(B)$

$\pi(\neg A) = W-\pi(A)$

3'. $\pi((\alpha)A) = \{s\mid(s,\pi(A))\in\rho(\alpha)\}$

and

5'. $\rho(\alpha;\beta) = \{(s,X)\mid\exists Y\ (s,Y)\in\rho(\alpha)$ and $\forall t\in Y,\ (t,X)\in\rho(\beta)\}$

$\rho(\alpha\vee\beta) = \rho(\alpha)\cup\rho(\beta)$

similarly for conjunction with $\cap$.

$\rho(\langle\alpha^*\rangle) = \{(s,X) \mid s \in \mu Y(X \subseteq Y \wedge (\forall t)((t,Y) \in \rho(\alpha) \Rightarrow t \in Y)\}$

$\rho([\alpha^*]) = \{(s,X) \mid s \in \nu Y(Y \subseteq X \wedge (\forall t)(t \in Y \Rightarrow (t,Y) \in \rho(\alpha))\}$

where νY means: the largest Y such that...

$\rho(\alpha^d) = \{(s,W-X) \mid (s,X) \in / \rho(\alpha)\}$

6'. $\rho(\langle A\rangle) = \{(s,X) \mid s \in \pi(A) \cap X\}$

and $\rho([A]) = \{(s,X) \mid s \in X - \pi(A)\}$

These conditions can be written equivalently as follows:

$\rho(\alpha;\beta)(X) = \rho(\alpha)(\rho(\beta)(X))$

$\rho(\text{skip})(X) = X$

$\rho(\alpha \vee \beta)(X) = \rho(\alpha)(X) \cup \rho(\beta)(X)$

$\rho(\alpha \wedge \beta)(X) = \rho(\alpha)(X) \cap \rho(\beta)(X)$

$\rho(\langle\alpha^*\rangle)(X) = \mu Y(X \subseteq Y \wedge Y \supseteq \rho(\alpha)(Y))$

$\rho([\alpha^*])(X) = \nu Y(Y \subseteq X \wedge Y \subseteq \rho(\alpha)(Y))$

$\rho(\alpha^d)(X) = W - \rho(\alpha)(W-X)$

$\rho(\langle A\rangle)(X) = \pi(A) \cap X$

$\rho([A])(X) = X \cup (W - \pi(A))$

So far we have made no connection with PDL. However, given a language of PDL we can associate with it a Game Logic where to each program a_i of PDL we associate two games $\langle a_i\rangle$ and $[a_i]$. We take

$\rho(\langle a\rangle) = \{(s,X) \mid \exists t\ (s,t) \in R_a$ and $t \in X\}$

and

$\rho([a]) = \{(s,X) \mid \forall t\ (s,t) \in R_a$ implies $t \in X\}$

and the formulae of PDL can be translated easily into those of game logic. However, Game Logic is more expressive. The formula $\langle[b]^*\rangle$false of game logic says that there is no infinite sequence of states $s_1,\ldots,s_i,\ldots$ such that b can take us from s_i to s_{i+1} for all i. This notion can be expressed in Streett's PDL but not in PDL. We suspect that the formula $\langle(\langle a\rangle;[b])^*\rangle$false, which says that player I can make "a" moves to player II's "b" moves in such a way that eventually player II will be deadlocked, cannot be expressed in PDL either. Finally, let us show how well-foundedness can be defined in Game Logic. Given a linear ordering R over a set W, consider the model of Game Logic where $\rho(g)$ denotes [a] and R_a is the inverse relation of R. Then R is well-founded over W iff the formula $\langle g^*\rangle$false is true. Since player I cannot terminate the game without losing, and he also loses if he never terminates, the only way he can win is to keep saying to player II, keep playing, and hope that player II will sooner or later be deadlocked. (the subgame [a] of $\langle[a]^*\rangle$ is a game where player II moves, and in the main game $\langle[a]^*\rangle$, player I is only responsible for deciding how many times is [a] played). Thus I

wins iff there are no infinite descending sequences of R on W.

However, game logic can be easily translated into the μ-calculus of [K] and by the decision procedure of [KP2], is decidable. We do not know if there is an elementary decision procedure.

Completeness: the following axioms and rules are complete for the "dual-free" part of game logic.
The axioms of game logic:
1) All tautologies
2) $(\alpha)(A \Rightarrow B) \Rightarrow ((\alpha)A \Rightarrow (\alpha)B)$
3) $(\alpha;\beta)A \Leftrightarrow (\alpha)(\beta)A$
4) $(\alpha\vee\beta)A \Leftrightarrow (\alpha)A \vee (\beta)A$
5) $(\langle\alpha^*\rangle)A \Leftrightarrow A \vee (\alpha)(\langle\alpha^*\rangle)A$
6) $(\langle A?\rangle)B \Leftrightarrow A\&B$
Rules of Inference:
1) Modus Ponens
2) $$\frac{A \Rightarrow B}{(\alpha)A \Rightarrow (\alpha)B}$$
3) $$\frac{(\alpha)A \Rightarrow A}{(\langle\alpha^*\rangle)A \Rightarrow A}$$ (Bar Induction)

The soundness of these axioms and rules is quite straightforward. The completeness proof is similar to that in [KP1]. The details will appear elsewhere.

§5 *A cake cutting algorithm*: A fairly common method used by children for sharing something fairly is: "you divide and I will pick". The person dividing has then a strong incentive to make the pieces as equal as possible. A more general algorithm that works for n people dividing something, say a cake, goes as follows:

The first person cuts out a piece which he claims is his fair share. Then the piece goes around being inspected, in turn, by persons p_2, p_3, ... etc. Anyone who thinks the piece is not too large just passes it. Anyone who thinks it is too big, may reduce it, putting some back into the main part. After the piece has been inspected by p_n, the *last* person who reduced the piece, takes it. If there is no such person, i.e., no one challenged p_1, then the piece is taken by p_1. In any case, one person now has a piece, and the algorithm continues with n-1 participants. Note incidentally that the algorithm described above for two people is not the special case n=2 of this more general algorithm.

A question recently raised by Even, Paz and Rabin is: what is the minimum number of cuts needed by an algorithm that works? They have shown that the algorithm described above can take $O(n^2)$ cuts, but that there is another algorithm that needs only O(nlog(n)) cuts. However, no good lower bound is known.

We are not here interested in the complexity issue, but just to enter into the spirit of the thing we raise the following question: the algorithm described above works only if all the participants involved are greedy.-i.e. they will take more than their share if we don't watch out. However, adults dividing food often are in a polite mood and try to take *less* than their fair share. Now the algorithm described above can easily be adapted to polite people by changing each reducing action to an increasing action when $p_2,\ldots,p_n$ inspect the piece cut out by p_1. Thus p_3 might say, "no, no, that is too little, let me give you more", etc., and the last person to increase the piece would get it. This modified algorithm will then work for polite people. A similar observation applies to the nlog(n) algorithm. However, it is not clear that this must be true in general, that greedy algorithms and polite algorithms come in pairs. Thus we could ask if the complexities are the same in both cases and also if there are algorithms that work if some people are polite and others are greedy. The solution to this last problem is easy if we know which are which. Then an algorithm that works would be one that gives *nothing* to the polite people and uses a "greedy" algorithm to divide the cake between the rest.

In any case, what we are actually going to be concerned with here is the question: what does it mean to say that either of these two algorithms *works*? And how do we prove its correctness? We show how to solve both these questions in the context of Dynamic Logic and many-person Game Logic.

As the reader may recall, in Dynamic Logic we have a state space W and a language L which can be used to make statements about individual states. Thus the notion $s \Vdash A$ where s is a state and A is a formula of L, is assumed to be defined, and means intuitively: the formula A is true at the state s. We assume also that we have some actions on W which may take us from one state to another, usually in a non-deterministic way. In the example above, a typical action may be a cutting action, or a reducing action, both of which are non-deterministic.

L is closed under the modalities <a> and [a] where a is any action. Thus if A is a formula of L then s⊨<a>A means that there is a t such that the action a *may* take us from s to t and t⊨A. Similarly, s⊨[a]A iff for all t such that a can take us from s to t, t⊨A. For example, if a cake is being cut between two people by the action c, say F_1 means that the first piece is fair (one person's fair share) and F_2 means that the second piece is fair. Then we have [c]($F_1 \vee F_2$) and also <c>($F_1 \wedge F_2$) I.e. no matter how the cake is cut, at least one piece is fair, and it is possible to cut it so that both pieces are fair. Note that both participants must agree to these two statements though they need not agree on which particular pieces are fair.

Now we are ready to state one sense in which the cake cutting algorithm is correct. Suppose that at some point in the algorithm some person p accepts [a]A where a is a formula and action a is just about to be performed. Then afterwards, p must accept A. Also if p accepts <a>B and *p herself performs action a and the algorithm calls for B to be realised* then p must accept B after a is done. However, another person p' who also accepted both [a]A and <a>B before a was done need not accept B after, though he must also accept A.

The correctness of the algorithm then consists in the fact that after the algorithm is finished, each person p_i must accept the proposition F_i which states that the piece received by p_i is fair.

An alternative, game theoretic interpretation will be given later.

The Proof: Now we proceed to a semi-formal proof of the correctness of the algorithm described before.

A state will consist of the values of n+2 variables. The variable m has as its value the main part of the cake. The variable x is the piece under consideration. For i=1 to n, the variable x_i has as its value the piece, if any, assigned to the person p_i. The variables $m, x, x_1, .., x_n$ range over subsets of the cake.

The algorithm uses three basic actions.

c cuts a piece from m and assigns it to x. c works only if x is 0.

r (reduce) transfers some (non-zero) portion from x back to m.

a_i (assign) assigns the piece x to person p_i. Thus a_i is simply, $(x_i,x):=(x,0)$.

The basic predicates are F(u,k) where u is some piece and $k \leqslant n$, and means: the piece u is big enough for k people. F(u) abbreviates F(u,1) and F_i abbreviates $F(x_i)$.

We assume that everyone always accepts the following propositions for all $k \leqslant n$.

(1) $F(m,k) \Rightarrow \langle c\rangle(F(m,k-1)\wedge F(x))$
If the main piece is big enough for k people, it is possible to cut out a fair piece, leaving enough for k-1 people.

(2) $F(m,k) \Rightarrow [r^*]F(m,k)$
If the main piece is big enough for k people, it remains so when more is added to it.

(3) $F(m,k) \Rightarrow [c][r^*](F(m,k-1)\vee\langle r\rangle(F(m,k-1)\wedge F(x)))$
If the main piece was big enough for k people, then after cutting and several reductions, either it is big enough for k-1 people, or else a reduction will make m big enough for k-1 people and leave x fair.

(4) $F(x) \Rightarrow [a_i]F_i$ Obvious.

There are tacit assumptions of relevance, e.g. that r and c can only affect statements in which m or x occurs. We assume moreover that everyone accepts F(m,n) at the beginning.

The main algorithm consists of n cycles during each of which one person is assigned a piece. We show now that after the kth cycle, the people still in the game accept F(m,n-k) and each p_i who is assigned a piece, accepts F_i. This is true at start since k=0, everyone accepts F(m,n) and no one yet has a piece. We now consider the inductive step from k to k+1. We assume that everyone in the game accepts F(m,n-k) at this stage.

Since p_1 (or whoever does the cutting) *does* the cutting, by (1) she must accept $F(m,n-k-1)\wedge F(x)$. If no one does an r, she gets x, and she must accept this as fair since x did not change. If someone does do an r she must (by (2) still accept F(m,n-k-1) and this is OK since she will then be participating at the next stage.

Let us now consider the other people.

The last person to do r (if there is someone who does r) must (by (3) accept F(x) and therefore accept x when it is assigned to him.

A person who refrains from doing r must accept F(m,n-k-1) because of (3) and will be in the next cycle with n-k-1 participants.

A person who does do r but is not the last such person, must accept F(m,n-k-1) at the time of doing r and hence also after more r's are done. Hence he will accept F(m,n-k-1) at the next cycle.

Thus we have shown that if p_i is assigned the piece during that cycle, then p_i accepts F_i and the other people still in the game accept F(m,n-k-1) at the next stage. This finishes the inductive step and we are done.

It is also possible to give a game-theoretic analysis of this problem. Namely, if we think of the algorithm as an n-person game, then every participant p_i has a strategy for ensuring that F_i holds when the game is over.

We notice first that each move of game α consists of two things. First we have an execution of some atomic program a by some person p. Moreover, the game α is replaced by a new game α' which is just the *rest of* α. Let A be some property that some player q wants to have holding at the end of the game (q may or may not equal p) and let $(q,\alpha)A$ mean that player q has a winning strategy, playing α to achieve A. (In two person games we did not need to mention q explicitly, since everything could be seen from player I's point of view.)

Then if q is the same as p then $(q,\alpha)A$ is equivalent to $(p,\alpha)A$ which is equivalent to $\langle a\rangle(p,\alpha')A$. For player p can achieve A after α iff she can achieve A after a and α' iff $\langle a\rangle(p,\alpha')A$ is true.

If q is not p, then he can be sure of achieving A after α iff he can be sure of achieving A after a play of a by p and then α'. And thus $(q,\alpha)A$ is equivalent to $[a](q,\alpha')A$.

Finally, when the game is *over*, then A can be achieved iff A is true.

These three observations taken together allow us to show that the proof we gave before can be put in terms of winning strategies in an n-person game. We leave the details to the reader.

Yet another way to look at this problem is measure theoretically. Given n probability measures $\mu_1,\ldots,\mu_n$ on a space X, the algorithm shows the existence of a partition $\{X_1,\ldots,X_n\}$ such that $\mu_i(X_i) \geq 1/n$. However, it is in fact true that there can be a partition with $\mu_i(X_i) = 1/n$. Can we find an algorithm that will achieve this latter property? Or at least to within ϵ?

Finally, we point out that there is a clear sense in which none of the algorithms described above are fair since they allow something very like gerrymandering. Suppose, for example, that there are two participants p and q. p loves icing, but q likes both cake and icing equally. Thus if we represent the cake by the interval $[0,2]$ where $[0,1]$ is the cake, and $[1,2]$ is the icing, then we might have $\mu_1([0,1]) = 1/10$ and $\mu_1([1,2]) = 9/10$. On the other hand, $\mu_2([0,1]) = \mu_2([1,2]) = 1/2$. Now the wily player p divides the cake into two pieces X and Y where $X = [0,1.1]$ and $Y = [1.1,2]$. Since $\mu_2(X) = .55$, q will choose X and leave Y to p. However $\mu_1(Y) = .81$ so that player p has got much the better bargain. Is it possible to define a notion of fairness that will get around this problem and find an algorithm that is fairer than the ones we have described?

References

[CKS] Chandra, A., Kozen, D., and Stockmeyer, L.,"Alternation", *J. ACM* 28, (1981) 114-133.

[E] Ehrenfeucht, A., " An Application of Games to to the Completeness Problem for Formalised Theories", *Fund. Math.*, 49 (1961) 129-141.

[EPR] S. Even, A. Paz and M. Rabin, oral communication.

[FL] M. Fischer and R. Ladner, "Propositional Dynamic Logic of Regular Programs", *J. Comp. and System Science* 18 (1979) 194-211.

[GH] Gurevich, Y., and Harrington, L., "Trees, Automata, and Games", *Proc. 14th STOC Symposium*, 1982, 60-65.

[K] Kozen, D., "Results on the Propositional μ-calculus", *Proc 9th ICALP* (1982) Springer LNCS vol 348-359

[KP1] D. Kozen and R. Parikh, "An Elementary Proof of the Completeness of PDL", *Theor. Comp. Sci.* 14 (1981) 113-118.

[KP2] Kozen, D., and Parikh, R., "A Decision Procedure for the Propositional μ-calculus", Typescript, April 1983.

[MP] A. Meyer and R. Parikh, "Definability in Dynamic Logic", *JCSS* 23 (1981) 271-298.

[MS] Mycielski, J., and Steinhaus, H., "A Mathematical Axiom Contradicting the Axiom of Choice", *Bull. Acad. Pol. Sci.*, 10, (1962) 1-3.

[P1] R. Parikh, "The Completeness of Propositional Dynamic Logic", *Proc. 7th Symp. on Math. Found. Comp. Sci*, Springer LNCS 64 403-415.

[P2] R. Parikh, "Propositional Dynamic Logics of Programs: A Survey", *Logics of Programs* Ed. E. Engeler, Springer LNCS 125 102-144.

[P3] R. Parikh, "A Completeness Result for a Propositional Game Logic", Manuscript, Nov. 1982.

[Pr] V. Pratt, "A Decidable mu-calculus (Preliminary report)" IEEE-FOCS 22 (1981) 421-428.

[Ra] Rabin, M., "Decidability of Second Order Theories and Automata on Infinite Trees", *Tran. AMS*, 141 (1969) 1-35.

[St] Streett, R., "Propositional Dynamic Logic of Looping and Converse", *Proc. 13th STOC* (1981) 375-383.

[vNM] von Neumann, J., and Morgenstern, O.,*the Theory of Games and Economic Behavior*, Princeton Univ. Press, 1944.