Annals of Discrete Mathematics 24 (1985) 111-140

The Logic of Games and its Applications

Rohit Parikh

Department of Computer Science
Brooklyn College of CUNY
and
Mathematics Department
CUNY Graduate Center

Abstract: We develop a Logic in which the basic objects of concern are games, or equivalently, monotone predicate transforms. We give completeness and decision results and extend to certain kinds of many-person games. Applications to a cake cutting algorithm and to a protocol for exchanging secrets, are given.

---

§1. *Introduction* Two person games have traditionally played an important part in Mathematical Economics and in Logic. In Economics the classic work of von Neumann and Morgenstern [vNM] has played a fundamental role and the influence of this approach has spread to other areas like political science. In Logic we have the work of Ehrenfeucht [E] who has given a characterisation of First Order Logic in terms of games. We also have in Set Theory the axiom of determinacy (AD), a game theoretic axiom made famous by Mycielski and Steinhaus [MS], and which has played an important role in recent research.

The appearance of games in Computer Science is more recent, but has been quite fruitful. The alternating Turing machines of Chandra, Kozen and Stockmeyer [CKS] give connections between time and space complexities. But ATMs are clearly game theoretic objects with disjunctive states and conjunctive states corresponding, respectively, to the moves of player I and player II. Gurevich and Harrington [GH] have recently used games to give

a new proof of Rabin's famous result [Ra] of the decidability of SnS. Proving the correctness of a concurrent algorithm can also be seen as a game theoretic situation with the scheduler in the role of the adversary player (player II from now on).

In this paper we develop a Propositional Logic of Games and study decision procedures as well as completeness results. Our emphasis will be on games in abstract universes, the way in which games can be combined to yield other games, and the axiomatic properties of such games.

The principal observation that makes such an investigation possible is the realisation that program connectives like "if then else" and "while do", which are used in various programming languages, extend immediately to games and that the axiomatic properties that relate compound programs with their components, properties studied in Hoare Logic and in Dynamic Logic, are shared, to a large extent, by games. Indeed, it is possible to think of programs as games of a special kind. The Logic of games which we develop lies in expressive power between the PDL of Fischer and Ladner [FL] and the μ-calculus of Kozen [K].

To be specific, we develop two Logics of games. We give a complete axiomatisation of one version of Game Logic but leave open the question of axiomatising an apparently stronger logic with the dual operator. The dual operator is one which converts a game into its dual where the two players interchange roles; the moves of player I and his winning positions become the moves and winning positions of player II and vice versa. For example, the two quantifiers $\exists$ and $\forall$ are duals of each other, for consider the formula $(Qx)A(x)$ where Q is an, as yet unspecified, quantifier. An element d is to be chosen and substituted for x, dropping the quantifier. If A(d) is true, then player I wins and if false, then player II wins. Clearly, player I having a winning strategy when *he* chooses d, is equivalent to $(\exists x)A(x)$ and if player II chooses, then $(\forall x)A(x)$ has to be true for player I to win even

then.

In certain contexts the dual operator can be replaced by two negations. (More or less the way the two quantifiers, existential and universal, can be defined from each other. For, as we saw, the quantifiers are simple one move games which are duals of each other.) However, this device fails for example when the dual operator is allowed to fall within the scope of a * (or equivalently, a while do) and then duality appears to make the logic much stronger.

We also give an extension of this two person logic to certain kinds of many person games, and an application of it to a well known "cake cutting algorithm". Finally, we show how the correctness of a bit swapping protocol is best viewed in game theoretic terms.

We shall now give a slightly more detailed but informal account of our games.

A normal (human) game, like chess, consists of *three* parts, the initial position, the legal moves, and the positions that are winning for player I. In our version of games, we shall have no draws, and the player who is deadlocked is automatically a loser. Thus the winning positions for player II are automatically defined from those for player I. Moreover, we shall take the game itself to consist only of the *middle* part, i.e. the legal moves, and that *will* include a determination of who loses if the game does not terminate properly. However, the initial position and the winning states for player I will be, as it were, supplied at the last moment when we convert a game into a statement.

Thus a typical statement of our logic will look like: $s \Vdash (\alpha)A$ where s is a state of some model, α is a game, and A is some formula. Here the state s plays the role of the initial position, and the set of states satisfying A plays the role of the winning states for player I. The whole statement means: player I has a

winning strategy, starting in state s, to play the game α in such a way, that either the game ends with A true, or else it fails to terminate in such a way, that player II will get the blame.

Notice now that (α) is also a predicate transform, converting the predicate A into the new predicate (α)A. This predicate transform is monotonic in that if A implies B then (α)A implies (α)B. For clearly if we increase the set of positions where, just after the game has ended, player I *has* won, then we will also increase the set of positions where the game has yet to begin, but player I *can* win. In this the operator (α) resembles the modalities $\langle\alpha\rangle$ and $[\alpha]$ of PDL which are also monotonic. However, these modalities of PDL are disjunctive and conjunctive respectively. $\langle\alpha\rangle$ commutes with disjunction (i.e. $\langle\alpha\rangle(A\vee B)$ is equivalent to $\langle\alpha\rangle A\vee\langle\alpha\rangle B$) and $[\alpha]$ with conjunction. This can be explained in game theoretic terms by saying that $\langle\alpha\rangle$, like $\vee$, is a game where player I is the only one who moves; and player II is the only one who moves in $[\alpha]$, and in $\wedge$. More general game modalities arising from games where both players move, share the monotonicity of $\langle\alpha\rangle$ and $[\alpha]$, but cannot be either disjunctive or conjunctive. It turns out that this is the only property that (α) lacks, and dropping the corresponding axiom from a complete axiomatisation of PDL yields a complete axiomatisation of Game Logic.

§2. *Syntax and Semantics:* We assume that the reader is familiar with the syntax and semantics of PDL and the μ-calculus. (See appendix I for a brief exposition.) 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 (α)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 $\alpha\beta$), $\alpha\cup\beta$, $\langle\alpha^*\rangle$, and α^d. Here α^d is the *dual* of α.
6. If A is a formula then $\langle A\rangle$ is a game.

We shall write $\alpha\cap\beta$, $[\alpha^*]$ and $[A]$ respectively for the duals of $\alpha\cup\beta$, $\langle\alpha^*\rangle$ and $\langle A\rangle$. If confusion will not result then we shall write αA for $(\alpha)A$. E.g. $\langle g_i^*\rangle A$ instead of $(\langle g_i^*\rangle)A$.

Intuitively, the games can be explained as follows. $\alpha;\beta$ is the game: play α and then β. The game $\alpha\cup\beta$ is: player I has the first move and in it he decides whether α or β is to be played, and then the chosen game is played. The game $\alpha\cap\beta$ 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. He need not declare *in advance* how many times is α to be played, but he is required to eventually stop and player II may use this fact as part of his strategy. Player I may not stop in the middle of some play of α. Similarly with $[\alpha^*]$ and player II. 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 atomic 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 $(s,Y)\in\rho(g)$. We shall find it convenient to think of $\rho(g)$ as an operator from $P(W)$ to itself, given by the formula $\rho(g)(X)=\{s\mid(s,X)\in\rho(g)\}$. It is then monotonic in X. We define $\pi(A)$ and $\rho(\alpha)$ for more complex formulae and games as follows:

2'. $\pi(A\cup B)=\pi(A)\cup\pi(B)$
$\pi(\neg A)=W-\pi(A)$

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

and

5'. $\rho(\alpha;\beta)(X) = \rho(\alpha)(\rho(\beta)(X))$

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

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

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

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

It is easily checked that $\rho(\alpha\wedge\beta)(X) = \rho(\alpha)(X)\cap\rho(\beta)(X)$, $\rho([A])(X) = (W-\pi(A))\cup X$, and $\rho([\alpha^*])(X) = \nu Y((Y\subseteq X)\wedge(Y\subseteq\rho(\alpha)(Y)))$ where νY means "the largest Y such that". This is easily seen by noticing that $\rho([\alpha^*])(X) = W-\rho(\langle\alpha^*\rangle)(W-X)$ = W-smallest Z such that $(W-X)\subseteq Z$ and $\rho(\alpha)(Z)\subseteq Z$

We shall have occasion to use both ways of thinking of ρ, as a map from P(W) to itself, and also as a subset of WXP(W). In particular we shall need the (easily checked) fact that $(s,X)\in\rho(\beta;\gamma)$ iff there is a Y such that $(s,Y)\in\rho(\beta)$ and for all $t\in Y$, $(t,X)\in\rho(\gamma)$. Similarly, $(s,X)\in\rho(\beta\vee\gamma)$ iff $(s,X)\in\rho(\beta)$ or $(s,X)\in\rho(\gamma)$.

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 $\rho(\langle a\rangle)(X) = \{s\mid\exists t\ (s,t)\in R_a$ and $t\in X\}$ and $\rho([a])(X) = \{s\mid\forall t\ (s,t)\in R_a$ implies $t\in X\}$ and the formulae of PDL can be translated easily into those of game logic. Note that if the program a is to be run and player I wants to have A true after, then if *he* runs a, only $\langle a\rangle A$ needs to be true. However, if player II is going to run the program a then $[a]A$ needs to be true for I to win in any case. Note that if there are no a-computations beginning at the state s, then player II is unable to move, $[a]A$ is true and player I wins. In other words, unlike the situation in chess, a situation where a player is unable to move is regarded as a loss for that player in both PDL and Game Logic. However, Game Logic is more expressive than PDL. The formula $\langle[b]^*\rangle$false of game logic says that there is no infinite computation of the program b, a notion that can be

expressed in Streett's PDL^{Δ} 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^{Δ} either. (NB. Moshe Vardi has shown recently that this is indeed the case.) 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 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. Player I cannot terminate the game without losing, But he is *required to terminate the game sometime.* 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 translated into the μ-calculus of [K] (see appendix II) and by the decision procedure of [KP2], is decidable. This translation is not as obvious as might appear at first sight. The μ-calculus, like PDL, is based on binary relations and while every binary relation R can be regarded as a game in two ways, as $\langle R\rangle$ and [R], not every game can be written in one of these two forms. However, it *is* true that every game g over W can be written as $\langle a\rangle[b]$ over W' where W' is a superset of W and its size is exponential in that of W. This allows us to give a translation as we claimed above. (See appendix II for details.) We do not know if there is an elementary decision procedure for the full Game Logic with the dual operator. An elementary decision procedure for dual-free Game Logic will follow as the consequence of the completeness result, given below.

§3 *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) $(\alpha;\beta)A \leftrightarrow (\alpha)(\beta)A$

3) $(\alpha \cup \beta)A \leftrightarrow (\alpha)A \vee (\beta)A$

4) $(\langle\alpha^*\rangle)A \leftrightarrow A \vee (\alpha)(\langle\alpha^*\rangle)A$

5) $(\langle A\rangle)B \leftrightarrow A \wedge B$

Rules of Inference

1) Modus Ponens

$$\frac{A \quad A \Rightarrow B}{B}$$

2) Monotonicity

$$\frac{A \Rightarrow B}{(\alpha)A \Rightarrow (\alpha)B}$$

3) Bar Induction

$$\frac{(\alpha)A \Rightarrow A}{(\langle\alpha^*\rangle)A \Rightarrow A}$$

The soundness of these axioms and rules is quite straightforward. The completeness proof is similar to that in [KP1] and proceeds through four lemmas.

Lemma 1: (Substitutivity of Equivalents) If $\vdash A \leftrightarrow B$ and D is obtained from C by replacing some occurrences of A by B, then $\vdash C \leftrightarrow D$.

Proof: Quite straightforward, using tautologies and rule 2).

To show that every consistent formula is satisfiable, we construct a model for a given consistent C_0 as follows. Let FL be the Fischer-Ladner closure of C_0. This is quite similar to the Fischer-Ladner closure as in PDL. Let W be the set of all consistent conjunctions of elements of FL and their negations (as in [KP1]). We shall use the convention that the letters A,B stand for atoms (elements of W) and C,D for arbitrary formulae. For all formulas C and D, $C \leq D$ means $\vdash C \Rightarrow D$. We let $\pi(P_i)$ be as before, the set of those atoms $A \leq P_i$. Finally we take $\rho(g_i)$ as $\{(A,X) \mid A \wedge \langle g_i \rangle X^{\#}$ is consistent$\}$ where $X \subseteq W$ and $X^{\#}$ is the disjunction of all the elements of X. We give first the proof for the case where all tests in all games are atomic.

Lemma 2: For all $X \subseteq W$, $A \in W$, all atomic test games α, if $A \wedge \langle \alpha \rangle X^{\#}$ is consistent, then $(A,X) \in \rho(\alpha)$.

Proof: By induction on the complexity of α.

1) If α is some g_i then the lemma holds by definition of $\rho(g_i)$.

2) If α is $\langle P \rangle$ for some P, then by hypothesis and axiom 5 $A \wedge X^{\#} \wedge P$ is consistent, so $A \in X$ and $A \vdash P$ by definition of the model. So $(A,X) \in \rho(\alpha)$.

3) α is $\beta \vee \gamma$. Then, by hypothesis and axiom 4, $(A \wedge \langle \beta \rangle X^{\#}) \vee (A \wedge \langle \gamma \rangle X^{\#})$ is consistent. Hence one of the disjuncts, say $(A \wedge \langle \beta \rangle X^{\#})$ is consistent. Hence $(A,X) \in \rho(\beta)$ so that $(A,X) \in \rho(\alpha)$.

4) α is $\beta;\gamma$. Then by axiom 3 $A \wedge \langle \beta \rangle \langle \gamma \rangle X^{\#}$ is consistent. Hence $A \wedge \langle \beta \rangle (T \wedge \langle \gamma \rangle X^{\#})$ is consistent, by lemma 1 where T is *true*. So $A \wedge \langle \beta \rangle (\bigvee (B \wedge \langle \gamma \rangle X^{\#} : B \in W)$ is consistent. Let $Y = \{B \mid B \wedge \langle \gamma \rangle X^{\#}$ is consistent$\}$. Then we can disregard B not in Y above. So $A \wedge \langle \beta \rangle ((B \wedge \langle \gamma \rangle X^{\#} : B \in Y)$ is consistent. Hence $A \wedge \langle \beta \rangle Y^{\#}$ is consistent. It is easily seen, using induction hypothesis that $(A,Y) \in \rho(\beta)$ and for all $B \in Y$, $(B,X) \in \rho(\gamma)$. Hence $(A,X) \in \rho(\alpha)$.

5) α is $\langle \beta^* \rangle$. Let Z be the smallest set such that $X \subseteq Z$ and for all B, if $B \wedge \langle \beta \rangle Z^{\#}$ is consistent, then $B \in Z$. Then note that for all $B \in Z$, $(B,X) \in \rho(\langle \beta^* \rangle) = \rho(\alpha)$.

Then it is immediate that $\vdash X^{\#} \Rightarrow Z^{\#}$ and also, by induction hypothesis $\vdash (\beta)Z^{\#} \Rightarrow Z^{\#}$. Hence, by bar induction, $\vdash (<\beta^*>)Z^{\#} \Rightarrow Z^{\#}$. Now, $A \wedge (\alpha)X^{\#}$ is consistent, so $A \wedge (\alpha)Z^{\#}$ is also consistent, so $A \wedge Z^{\#}$ is consistent, by above, and so A must be in Z. Thus $(A,X) \in \rho(\alpha)$.

Given C in FL let C^+ denote the set of all atoms B such that $C \lesssim B$. Then $C^{+\#}$ will be a formula equivalent to C.

Lemma 3: If $(\alpha)C \in$ FL then (i) $(A,C^+) \in \rho(\alpha)$ iff (ii) $A \wedge (\alpha)C$ is consistent iff (iii) $A \lesssim (\alpha)C$

Proof: Clearly the last two cases are equivalent since A is an atom and either $(\alpha)C$ or its negation occurs in A. Also (ii)=>(i) by lemma 2. So assume (i) and use induction on the complexity of α. The cases where α is $<P_i>$ or g_j or $\beta \vee \gamma$ are all easy. We write X for C^+. Then $X^{\#}$ is a formula C_1 equivalent to C.

2) Suppose α is $\beta;\gamma$. Since $(A,X) \in \rho(\alpha)$ there must exist Y such that $(A,Y) \in \rho(\beta)$ and for all $B \in Y$, $(B,X) \in \rho(\gamma)$. By induction hypothesis, for all $B \in Y$, $B \lesssim (\gamma)X^{\#}$. Hence $Y^{\#} \lesssim (\gamma)X^{\#}$. Also $A \lesssim (\beta)Y^{\#}$. Hence $A \lesssim (\beta)(\gamma)X^{\#}$. Thus $A \lesssim (\alpha)X^{\#}$. Hence $A \lesssim (\alpha)C$.

3) Suppose α is $<\beta^*>$. If $A \in X$ then $A \lesssim X^{\#}$ so $A \lesssim (\alpha)X^{\#}$ by axiom 5. Otherwise $(A,X) \in \rho(\beta;\beta^*)$. Define $Z_0 = X$ and $Z_{n+1} = Z_n \cup \{B : (B,Z_n) \in \rho(\beta)\}$. Since A is in the union of the Z_n, take the least m such that $A \in Z_m$. By induction hypothesis (on n), for all $B \in Z_{m-1}$, $B \lesssim <\alpha>X^{\#}$. Also $(A,Z_{m-1}) \in \rho(\beta)$. By an argument like that of case 2 above, we get $A \lesssim (\beta)(<\beta^*>)X^{\#}$ so $A \lesssim (<\beta^*>)C$. QED.

Lemma 4: For all A in W, all C in FL, $A \lesssim C$ iff $A \models C$.

Proof: By induction on the complexity of C. The atomic case and the cases for negation and disjunction are easy.

Suppose C is $(\alpha)D$.

Then $A \models C$ iff $(A,\pi(D)) \in \rho(\alpha)$

iff $(A,D^+) \in \rho(\alpha)$ (IH)

iff $A \lesssim (\alpha)D^{+\#}$

iff $A \lesssim (\alpha)D$

iff $A \lesssim C$. QED

The completenes result follows at once. For if a formula C_0 is consistent then there is an atom A such that $A \leq C_0$ and then C_0 is satisfiable. QED

To see the case for the "rich-test" version of Game Logic, suppose we define, by induction, forumlae of level n to be formulae whose programs have tests all of level n-1. Then notice that the argument will go through for level n assuming that all lemmas were true at level n-1. For the only fact we used in the proof above for *atomic* P is that $A \leq P$ iff $A \vdash P$. This fact will now be the induction hypothesis for formulae of level n-1 rather than part of the definition of the model, and the proof goes through.

It follows also that dual-free Game Logic is decidable in nondeterministic double exponential time. For given a formula C_0, if C_0 has a model, then it has one where W is at most exponential in the size of C_0. We can guess such a model and a state where C_0 is to hold and proceed to verify this guess in time which is polynomial in the size of W. The only novel feature here that does not appear in PDL is the presence of the μX operator used in defining the extension of $\langle \alpha^* \rangle$. However, if W has k elements than $\mu X \phi(X)$ is merely X_k where X_0 is the empty set and X_{i+1} is $\phi(X_i)$. Thus if ϕ can be calculated in time polynomial in the size of W, so can $\mu X \phi(X)$. We suspect that the decision procedure can be made to run in *deterministic* exponential time using the familiar techniques due to Pratt. (NB. [VW] have recently shown that this is indeed the case.)

We conjecture that the addition of the following axiom scheme results in a system complete for the dual operator.

$$7)\ (\alpha^d)A \iff \neg(\alpha)\neg A$$

§4 *Many person games*: We now consider many person games which are, in general, more complex than two person games. Suppose we have certain actions a,b,c,... and certain persons $p_1,\ldots,p_m$. Suppose they are carrying out a procedure (playing a many person game) which involves certain persons doing certain actions at certain stages. E.g. the game may be, (p_1 does a);(p_2 does b);(p_3 decides whether p_1 or p_2 does c). Now suppose at a certain stage, an action a is to be performed and person p would like the formula A to be true afterwards. Clearly, if p himself is to do a, then the condition $\langle a\rangle A$ is sufficient. However, if someone else is to do a, then p needs $[a]A$ to be true to be sure that A will be true afterwards. Is it possible then, given persons $p_1,\ldots,p_m$ and goals $A_1,\ldots,A_m$ to rig up a procedure so that p_i can be sure of achieving A_i at the end of the procedure, *regardless* of what the others do? In the next section we shall give an example of this. However, right now we just wish to discuss the syntax and semantics of the logic involved.

We assume, as in PDL, a collection of atomic actions $a_1,\ldots,a_n$, and a supply $P_1,\ldots,P_k$ of atomic predicates. Then the m-person games and formulae are defined as follows:

(i) Each P_i is a formula.
(ii) If A,B are formulae, then so are $\neg A, A\supset B$.
(iii) If A is a formula, $i\leq m$, and α is an m-person game, then $(\alpha,i)A$ is a formula. (Person i has a winning strategy, playing α, to ensure that A is true if and when the game ends.)

(iv) For each $i\leq n$, $j\leq m$, (a_i,j) is a game. (The person j does action i)
(v) If α,β are games, then $\alpha;\beta$ is a game and for each $i\leq m$, $(\alpha\vee_i\beta)$ is a game. (The person i chooses which of games α and β is to be played.)
(vi) If α is a game, then for each $i\leq m$, $(\alpha,i,*)$ is a game. (The game α is played repeatedly until player i says to stop and go on to the next step if there is one.)

(vii) If α β are games and A is a formula then "if A then α else β" is a game and so is "while A play α".

A game will be called *-free if the * operator does not occur. Clearly, every *-free game is bounded in length.

A model for this logic will be just a PDL model. The semantics for m-person games can be obtained analogously to that of two person games. The only inductive clause of interest is defining the truth values of the formula $(\alpha,i)A$ from those of the formula A. If we can define the predicate transform $\rho(\alpha,i)$ then we can inductively define $\pi((\alpha,i)A)$ to be $\rho(\alpha,i)\pi(A)$. However, $\rho(\alpha,i)$ is easily seen to be $\rho(\alpha')$ where α' is the *two* person game with player i as player 1 and all other players as player 2. In particular if $j\neq i$, then (a,i) is $\langle a\rangle$, (a,j) is $[a]$. $\vee_i$ is $\vee$ and $\vee_j$ is $\wedge$. We skip the details.

The following axioms are obvious (i and j are *distinct*).

$(\alpha;\beta,i)A \iff (\alpha,i)(\beta,i)A$

$(\alpha\vee_i\beta,i)A \iff (\alpha,i)A\vee(\beta,i)A$

$(\alpha\vee_j\beta,i)A \iff (\alpha,i)A\wedge(\beta,i)A$

$((\alpha,i,*),i)A \iff A\vee(\alpha,i)((\alpha,i,*),i)A$

$((\alpha,j,*),i)A \iff A\wedge(\alpha,i)((\alpha,j,*),i)A$

(if B then α else β,i)A $\iff$ $(B\wedge(\alpha,i)A)\vee(\neg B\wedge(\beta,i)A)$

(while B do α,i)A $\iff$ $(\neg B\vee A)\wedge(B\wedge(\alpha,i)$(while B do α,i)A)

The following axioms connect many person games with PDL.

$((a,i),i)A \iff \langle a\rangle A$

$((a,j),i)A \iff [a]A$

And it is immediate that *-free many person game logic can be translated to PDL, thereby giving us a decision procedure. We can also use PDL notation and many person Game Logic notation interchangably in certain restricted circumstances.

Also the following rules hold.

$$\frac{A \Rightarrow B}{(\alpha,i)A \Rightarrow (\alpha,i)B}$$

$$\frac{A \wedge B \Rightarrow (\alpha,i)A}{A \Rightarrow (\text{while } B \text{ do } \alpha,i)(A \wedge \neg B)}$$

$$\frac{A \Rightarrow (\alpha,i)A}{A \Rightarrow ((\alpha,j,*),i)A}$$

$$\frac{(\alpha,i)A \Rightarrow A}{((\alpha,i,*),i)A \Rightarrow A}$$

These last two rules, of course, are the game theoretic generalisations of the corresponding rules for [ ] and < >.

§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 $O(n\log(n))$ 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 dc we prove its correctness? We shall take the point of view that the algorithm of this sort is really an m-person *game* and that the algorithm works iff each player has a winning strategy for achieving a fair outcome. We then show how to prove the correctness of such an

algorithm in many-person Game Logic.

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 $k \leq n$, 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 the following propositions for all $k \leq n$ and all players $i \leq n$.

(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.

Note that (1) is equivalent to the following game theoretic proposition.

(1') $F(m,k) \Rightarrow (c,i)(F(m,k-1) \wedge F(x))$

The following also holds:

(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 F(m,n) is true at the beginning.

The main algorithm consists of n cycles during each of which one person is assigned a piece. We show now that each person p_i has a winning strategy so that if, after the kth cycle, (s)he is still in the game then F(m,n-k) and if (s)he is assigned a piece, then F_i is true. This is true at start since k=0, F(m,n) holds and no one yet has a piece. We now consider the inductive step from k to k+1. We assume by induction hypothesis that F(m,n-k) holds at this stage.

If i=1 then since p_1 (or whoever does the cutting) *does* the cutting, by (1) and (1') she can achieve $F(m,n-k-1) \wedge F(x)$. If no one does an r, she gets x and F_1 will hold since x did not change. If someone does do an r, then by (2), F(m,n-k-1) will still hold and this is OK since she will then be participating at the next stage.

Let us now consider just one of the other people.

The last person p_1 to do r (if there is someone who does r) could (by (3) achieve F(x) and therefore when x is assigned to him, F_1 will hold.

All the other cases are quite analogous, and the induction step goes through. By taking k=n we see that every p_i has the ability to achieve F_i.

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 $\mu_i(X_i) \geqslant 1/n$. 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?

§5. *A protocol for exchanging secrets:* At STOC-83, Halpern and Rabin introduced a logic of likelihood (LL) to analyse the situation that arises when two parties called Alice and Bob, who do not trust each other, want to exchange secrets without making themselves liable to betrayal.

In the problem discussed by Halpern and Rabin, Alice and Bob are two people who have files that they access with a password. For simplicity the password is assumed to be one bit long. They would like to exchange passwords, but want to do it in such a way that there is no incentive for cheating. A protocol using oblivious transfer is described which succeeds with probability 3/4. A fact which is crucial in the working of the algorithm is

that both files are destroyed if either party accesses the other's file with an incorrect password.

It is clear from the very nature of the problem that the notions involved include not only cryptographic protocols, but also the notions of belief, capacity (ability to take some action), preference, and, of course, truth. Thus the Halpern-Rabin algorithm succeeds only because the two parties *prefer* to exchange secrets, rather than letting things be. Cheating will not happen, because a belief in the cheatee that (s)he *has* the cheater's secret when (s)he *does not*, is dangerous to the cheater. For the cheatee can and will try the incorrect password, thereby destroying both files. It is also clear that we are dealing here with a two-person *non zero sum* game in that while the interests of the parties are *somewhat* opposed (each would prefer to cheat if (s)he could do it in safety), they are not entirely opposed, since each prefers a co-operative exchange of secrets to no exchange.

Let us start by considering the following propositions.

BPA: Bob has Alice's password.
APB: Alice has Bob's password.
Bel(B,BPA): Bob believes that he has Alice's password.
Bel(A,APB): Alice believes that she has Bob's password.
AFD: Alice's file is destroyed.
BFD: Bob's file is destroyed.

Let us now consider Alice's preferences. Her first preference, presumably is APB$\wedge\neg$BPA. Were it not for a fear of cheating, they could just exchange passwords with trust. Her preferences after that are, in order, APB$\wedge$BPA, $\neg$APB$\wedge\neg$BPA, $\neg$APB$\wedge$BPA and finally, AFD. Bob's preferences can be obtained by exchanging the letters A and B. thus we get the little table:

	Alice		Bob
(i)	APB∧¬BPA	(iv)	BPA∧¬APB
(ii)	APB∧BPA	(ii)	BPA∧APB
(iii)	¬APB∧¬BPA	(iii)	¬APB∧¬BPA
(iv)	¬APB∧BPA	(i)	¬BPA∧APB
(v)	AFD	(vi)	BFD

It is clear that the preferencs are *not* exactly the same, and hence the need for caution. On the other hand, *both* prefer alternative (ii) to (iv) and since the facts of the case (and not logic) imply AFD<=>BFD, (v) and (vi) coincide effectively. Thus their preferences coincide to a fair extent and we have the possibility of co-operation. From now on we shall abbreviate the propositions AFD and BFD as just FD and (vi) will be renumbered (v). Thus we get a new table:

	Alice		Bob
(i)	APB∧¬BPA	(iv)	BPA∧¬APB
(ii)	APB∧BPA	(ii)	BPA∧APB
(iii)	¬APB∧¬BPA	(iii)	¬APB∧¬BPA
(iv)	¬APB∧BPA	(i)	¬BPA∧APB
(v)	FD	(v)	FD

Table I

We now propose the following quite simple protocol.

Step 1: Each of Alice and Bob sends the other a random bit.
It is part of the protocol that step 2 is not done until step 1 has been completed.

step 2: *If* the random bit sent by either Alice or Bob does not coincide with the password then (s)he sends a correction
(within some stipulated time period.)

End of protocol.
Note that we did not need an oblivious transfer.

We now have to show that this protocol works. Before step 1, the situation is: $\neg APB \wedge \neg BPA \wedge \neg Bel(A,APB) \wedge \neg Bel(B,BPA)$.

This situation persists through step 1. Note that neither party has an advantage yet since only a random bit has been received and the protocol cannot proceed unless the first step is completed. After step 2, we will have $Bel(A,APB) \wedge Bel(B,BPA)$.

Suppose Alice decides to cheat at step 2. She can cheat in one of two ways. She can fail to send a correction even if one is needed, or she can send one when it is not. It is clear that in either case she is risking the loss of her file since Bob believes that he has her password and will use it. Since loss of her file is the least preferred of her alternatives, she has to be honest and, for similar reasons, so does Bob.

This is the informal proof. Now we proceed to make it a bit more precise.

We consider a state space consisting of a finite number of states divided into five levels. The first level has only the start state s_0 which satisfies the formula $\neg APB \wedge \neg BPA \wedge \neg Bel(A,APB) \wedge \neg Bel(B,BPA)$. I.e. neither knows the other's password, nor believes that (s)he knows it. At the second level, there are two states, the state s_1 which is reached if step 1 is successfully completed and the state s_t which is a terminating state and is reached if someone cheats at the first stage.

After s_1 is reached, step 2a brings us to one of two states s_{0x} and s_{1x} when Alice has either cheated or not, (0 stands for no cheating, the x for Bob's yet undetermined choice). Bob's choice brings us then to one of the four states s_{00}, s_{01}, s_{10}, s_{11} which represent the possibilities: neither cheats, Bob cheats, Alice cheats, and both cheat respectively. (A subscript 0 means no cheat, and Alice's behaviour is mentioned first.) these are the states of level 4 and $Bel(A,APB)$ and $Bel(B,BPA)$ are both true at all these. APB is true at s_{00} and at s_{10} and similarly BPA is true

at s_{00} and at s_{01}. I.e. at s_{00} and at s_{01} Bob actually knows Alice's password and similarly, Alice knows Bob's at s_{00} and at s_{10}. Finally, each of these level four states has four level five successors depending on whether Alice or Bob attempts to enter the other's file. Leaving out the details of this fifth level, we get the following picture:

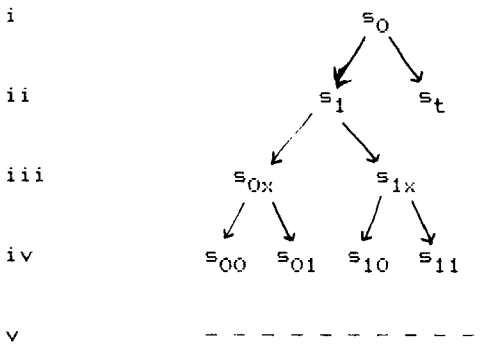

Table II

where the fifth level indicates possible file entry.
Now Alice assumes that the second subscript at the fourth level is 0, because of Bel(A,BPA) and similarly Bob assumes that the first subscript is 0. At s_{00} and s_{10} Alice may safely enter Bob's file and at s_{00} and s_{01} Bob may safely enter hers.

However, Bob, when he is at s_{10}, is likely to assume that he is at s_{00} and attempt file entry. Similarly for s_{11} and s_{01}. Similar remarks hold for Alice. Hence, if files are to be saved, the only acceptable state at level iv is s_{00}. This shows that neither Bob nor Alice can cheat *provided* they are rational.

This brings us to a rather odd fact. Suppose that Alice and Bob are convinced by the argument above. Then the protocol would be safe for them to use, even if it had a bug in it. We will not pursue this line of thought. In this particular case, however, there is a stable strategy that either party can use safely. Namely, send the random bit as agreed, act honestly at step 2 and

then proceed to access the file using the password assuming it is correct. If Bob, for instance, uses this strategy, Alice cannot benefit from dishonesty. Similarly for Alice. Moreover, by the nature of the problem (which is symmetric), this strategy is optimal for both parties, since the only alternative which would be better for either, is unsymmetric.

We can summarise the situation in standard game theoretic terms by using a table. Let the strategies for each of Bob and Alice be, c (cheat at step 1 by not sending in a bit) hc (cheat at the second step *after* first sending in a random bit) and hh (honesty at both steps.) There is no strategy cc or ch since the protocol terminates if bits are not sent at the first step by *both* parties. The payoffs are given in terms of the alternatives i-v of table I before.

		Bob		
		c	hc	hh
Alice	c	iii	iii	iii
	hc	iii	v	v
	hh	iii	v	ii

Table III

It is clear that the strategy hh is stable for both Alice and Bob since the other party then cannot gain by cheating. Thus the bottom right hand square is an equilibrium point.

As the foregoing shows, an analysis of a basically game theoretic situation can be carried out successfully without having to introduce utility values as is usually done. After, all, the decision ultimately is of the sort "shall I do a rather than b?", requiring only a yes/no answer, and hence even if the reals are introduced as utilities, their function should be reducible to certain logical questions. How far this can be carried and whether one gets nice decidable theories like PDL and game Logic remains to be seen.

Acknowledgements: We have benefited from conversations with Moshe Vardi, Ken McAloon, Albert Meyer and David Mumford.

Appendix I

PDL and the μ-calculus: We give a brief review of both PDL and the μ-calculus. A detailed presentation will be found in the literature, Cf. [FL], [Pa1], [KP1], [St].

PDL deals with abstract, regular programs in a state space W. There are no individual variables or functions, and hence the role played by assignments in the usual, imperative programming languages, is taken by abstract atomic programs a_i. These are not assumed to be deterministic, though there is a version (DPDL) of PDL in which this asumption is made. Programs can be combined by means of program connectives ;, U and *. The usual connectives like "while do" and "if then else" can be defined from these.

Specifically, we have a language with a finite number of basic symbols a_i for atomic programs, and P_j for atomic predicates. We then define *programs* and *formulae* as follows.

Each P_i is a formula
If A, B are formulae, then so are ¬A and A∨B
If A is a formula and α is a program, then $\langle\alpha\rangle$A is a formula
(For PDL^{Δ} only) If α is a program, then $\alpha)$ is a formula.

Each a_i is a program.
If α, β are programs, then so are $\alpha;\beta$, $\alpha\cup\beta$, and α^*.
If A is a formula, then A? is a program.

Abbreviations: A B stands for $\neg(\neg A\cup\neg B)$. $[\alpha]A$ stands for $\neg\langle\alpha\rangle\neg B$. If A then α else β for $(A?;\alpha)\cup(\neg A?;\beta)$. While A do α stands for $(A?;\alpha)^*;(\neg A?)$.

Semantics: We assume given a state space W. Elements of W will be called *states* and referred to by letters s,t etc. We assume given for each P_i a subset $\pi(P_i)$ of W and for each a_i a subset $\rho(a_i)$ of WxW. A *model* M consists of such a W and the assignment of the values of π and ρ to the atomic symbols. Then $\pi(A)$ and $\rho(\alpha)$ are defined for arbitrary formulae A and arbitrary programs α as follows:

$\pi(\neg A)=W-\pi(A)$. $\quad\pi(A\cup B)=\pi(A)\cup\pi(B)$
$\pi(\langle\alpha\rangle A)=\{s\mid(\exists t)((s,t)\in\rho(\alpha)$ and $t\in\pi(A)\}$
$\pi(\Delta\alpha))=\{s\mid\exists$ infinite sequence $s=t_1,t_2,\ldots$ such that for all i, $(t_i,t_{i+1})\in\rho(\alpha)\}$

$\rho(\alpha;\beta)=\rho(\alpha)\circ\rho(\beta)$. $\quad\rho(\alpha\cup\beta)=\rho(\alpha)\cup\rho(\beta)$
$\rho(\alpha^*)$=reflexive, transitive closure of $\rho(\alpha)$.
$\rho(A?)=\{(s,s)\mid s\in\pi(A)\}$

A formula A is *true* at s iff $s\in\pi(A)$. It is *valid* iff it is true at all states of all models. It is *satisfiable* iff it is true at some state of some model.

The principal results are that validity in PDL is decidable in DEXPTIME and that the axioms given for Game Logic together with the axiom $\langle\alpha\rangle(A\cup B)\langle=\rangle\langle\alpha\rangle A\cup\langle\alpha\rangle B$ are complete.. In other words, PDL is the logic of disjunctive games. PDL^{Δ} is also decidable in DEXPTIME, [VW] but no nice set of axioms is known.

The μ-calculus takes advantage of the fact that our primary interest in programs is as predicate transforms, and that the

effect of the * in α^* can be obtained through a least fixed point operator. Thus the μ-calculus does away with non-atomic programs and the definition of formulae goes as follows. Note that we are introducing predicate variables X_j which will be bound by the least fixed point operator μX_j.

Each P_i is a formula. Each variable X_j is also a formula. If A,B are formulae then so are A B, $\neg A$ and $\langle a_i \rangle A$ for each a_i. If A is a formula in which all occurrences of X_j are positive, then $\mu X_j.A$ is also a formula.

Given a model M and a formula A, to evaluate $\pi(A)$ we will need to assign subsets V_j to each free X_j in A. Then to define $\pi(A)$, the clauses for the first two of the conditions above are obvious. Of course $\pi(X_j)$ is taken to be V_j. Since $\pi(A)$ depends on the V_j, let us write it as $\pi(A,V_1,\ldots,V_k)$. Then for the μX condition we take,

$\pi(\mu X_k.A,V_1,\ldots,V_{k-1})$=The least V_k such that $\pi(A,V_1,\ldots,V_k)=V_k$.

It is known that the μ-calculus is decidable and properly includes PDL, see [K], [KP2].

Appendix II

A Translation of Game Logic into the μ-calculus: We said earlier that Game logic can be translated into the μ-calculus, but there is a slight difficulty with this statement as it stands. In a model of the μ-calculus, at least as considered by Kozen and Pratt, programs are binary relations on the state space W. However, in Game Logic, the basic objects are not binary relations, but games. This disparity needs to be resolved before we can speak of a translation.

Of course, a binary relation gives rise to a game. Namely, if R is the relation, the we can define the game R' to be $\{(s,X)\mid(\exists t\in X)((s,t)\in R)\}$. However, the game defined this way has the property that $(s,X\cup Y)\in R'$ iff $(s,X)\in R'$ or $(s,Y)\in R'$. In other

words, R' is disjunctive, and of course not every game has this property.

The solution is to go one type upwards. Given an arbitrary game α on W, we can think of it also as a relation between W and P(W), the power set of W. Thus it can be thought of as a two move game on the space $W' = W \cup P(W)$. Specifically, in his first move from a state $s \in W$, player I chooses an X such that $(s,X) \in R(\alpha)$. Note that since $X \in P(W)$, $X \in W'$. Now player II chooses an element $t \in X$. Thus α is represented on W' as $\langle \alpha' \rangle [e]$ where both α' and e are binary relations and, as a matter of fact, e does not depend on α at all. Incidentally, the letter e stands for the word "element" since II is choosing an element of the set X chosen by I.

It is clear now that if a Game Logic formula A has a model of size k, then there will be a corresponding formula A^+ of the μ-calculus with a model of size $k+2^k$. However, the translation must preserve satisfiability in both directions, and that means that if the translation A^+ of a Game Logic formula A has a model M', then we should be able to decode from it a model M of the formula A. This means that we can separate W and P(W) inside W' etc. These possibilities need to be built into the structure of A^+. Thus A^+ will be a formula $A' \wedge G$ where A' translates A in a μ-calculus model of a special kind and G' which does not depend on A, says that the given μ-calculus model *is* of that special kind. The language of G and A' will include the atomic predicates of A plus a new atomic predicate symbol E (for "is an element"). There is an atomic program a_i for each game g_i and a new atomic program e (choosing an element). First we define G to be the (μ-calculus translation of) the PDL formula

$$E \wedge [((\cup a_i) \cup e)^*] \Big(\bigwedge_i (\langle a_i \rangle \mathrm{true} \Rightarrow E) \wedge \bigwedge_i ([a_i] \neg E) \wedge \bigwedge_i (P_i \Rightarrow E) \wedge ([e]E) \wedge (\langle e \rangle \mathrm{true} \Leftrightarrow \langle e \rangle E) \Big)$$

Intuitively, G says that the universe W' is divided into

two parts $\pi(E)$ and $W'-\pi(E)$, and that all programs a_i go from $\pi(E)$ to $W'-\pi(E)$ whereas the program e goes in the opposite direction.

Now A' is defined inductively as follows:

If A is:	Then A' is
an atomic predicate,	A
$(B\vee C)$	$B'\vee C'$
$\neg B$	$E\wedge\neg B'$
$(g_i)B$	$\langle a_i\rangle[e]B'$
$(\alpha;\beta)B$	$((\alpha)(\beta)B)'$
$\langle(\alpha)^*\rangle B$	$\mu X.(B'\vee((\alpha)X)')$
$(\alpha^d)B$	$(\neg(\alpha)\neg B)'$
$(\alpha\vee\beta)B$	$(\alpha)B'\vee(\beta)B'$

Finally we take A^+ to be $A'\wedge G$. Then as we indicated earlier, any Game Logic *model* M of A can be converted into a *model* of A^+. Conversely, any model of A^+ is a model of G and hence all games of the form $\langle a_i\rangle[e]$ begin and end in the extension $\pi(E)$ of E. Thus a model of A^+, which is, therefore a model of both A' and G, can be converted into a Game Logic model of A with $\pi(E)$ as W. It follows that A is Game Logic satisfiable iff A^+ is μ-calculus satisfiable.

References

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

[E] Ehrenfeucht, A., " An Application of Games to to the *Completeness Problem for Formalised Theories*". *Fundamenta* Mathematica, 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 ACM-STOC Symposium*, (1982) 60-65.

[HR] J. Halpern and. M. Rabin, "A Logic to Reason About Likelihood", *Proceedings of the 15th ACM-STOC Symposium* (1983) 310-319.

[K] Kozen, D., "Results on the Propositional μ-calculus", *Proc 9th ICALP* (1982) Springer LNCS #140 348-359

[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", *Proceedings of the CMU Conference on the Logic of Programs*, Springer LNCS #164, 313-25.

[LR] D. Luce and H. Raiffa, *Games and Decisions*, Wiley 1957.

[MP] A. Meyer and R. Parikh, "Definability in Dynamic Logic", *Jour. Comp. and System Science* 23 (1981) 279-298.

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

[Pa1] R. Parikh, "The Completeness of Propositional Dynamic Logic", *Proc. 7th Symp. on Math. Found. Comp. Sci*, Springer LNCS #64 403-415.

[Pa2] R. Parikh, "Propositional Dynamic Logics of Programs: A Survey", *Logics of Programs* Ed. E. Engeler, Springer LNCS #125 102-144.

[Pa3] R. Parikh, "Propositional Logics of Programs: New Directions", *FCT 83* Springer LNCS #158, 347-359.

[Pa4] R. Parikh, "Propositional Game Logic", *IEEE-FOCS 83* 195-200.

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

[R] A. Rapaport, *N-Person Game Theory*, Michigan Press 1970.

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

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

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

[VW] M. Vardi and P. Wolper, "Automata Theoretic Techniques in the Logic of Programs", Research Report (1984)