A Logic for Games

Alexandru Baltag
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands[*]

The goal of this paper is to introduce a modal logic for sequential games with incomplete information. The logic is a mixture of dynamic and epistemic logics, designed to capture relevant game-theoretic notions, e.g. *perfect information*, *Nash equilibrium*, *subgame perfect equilibrium*, *knowledge of the best strategy* etc. I consider it as a first step towards a full modal formalization of the basic notions of game theory. Important notions e.g. *randomization, mixed strategies, beliefs* and *rationality* are missing from the present approach; a logical treatment of these notions would involve us in the complexities of *probabilistic modal logic.*

The main originality of the present approach consists in giving the semantics in terms of a Kripke structure formed of *pairs of states* (in a game) and *strategy profiles.* This is the basic feature of the logic. There are various natural choices for the language to be considered on this structure, and in further work I plan to explore some more of them. In this paper I chose to consider only the following: an *epistemic logic*, modelling player's knowledge of their current state and strategy, and a *dynamic logic* with programs constructed (in the usual manner) from basic *actions (moves) for each player i*, from *test* actions and from a deterministic *next-step action* (induced by the strategy profile at the given state).

The *notion of game* I consider here is an enrichment of the standard notion of game with imperfect information. Namely, as observed by Bonano and others, the standard notion does not provide a full semantics for epistemic logic, due to the fact that *each player's information sets consist only of states at which he/she has to move.* In order to talk about player's knowledge in other states, one needs to extend the information sets to obtain a full partition of the set of all the states. Bonano observed this extending can be done in many different ways, and he proposed some natural solutions to this problem. In the following, I take for granted that such a standard extension is given, and I identify a game with its extension. Consequently, for a given finite set N of *players* and a given set A of *actions*, a *game* is a tree $G = (S, P, \sim_i, u_i)_{i \in N}$, where $S \subseteq A^* \cup A^\omega$ is set of *states*, i.e. (finite or infinite) sequences of actions, P is the*player function* from the nonterminal states to the set of players, $\sim_i$ are equivalence relations on S and u_i are functions from the terminal states to some linearly ordered set (usually taken to be R). We denote by $s \to^a s'$ the fact that $s' = sa$, and by $s \to s'$ the fact that $s \to^a s'$ for some action a; similarly, we put $s \to^a$ if we have $s \to^a s'$ for some state s', and $s \to$ if we have $s \to^a$ for some action a. In the above definition, a *non-terminal state* is simply a state s such that $s \to$. The initial state is $s_0 = \emptyset$ (the empty sequence). A game is assumed to satisfy suitable conditions: first, S has indeed to be *a tree*; secondly, if $s \sim_i s'$ then $s \to^a$ iff $s' \to^a$. In the following, we shall also assume all our games to be *finite.* As a consequence, the set of all utility

[*]Email: `abaltag@cwi.nl`

scores $\{u_i(s) : s \in S\}$ is finite. Moroever, we fix a finite set U of utility scores, and we consider only games having scores in U. A *strategy for player* i is a function $\sigma_i : \{s \in S : P(s) = i\} \to A$, satisfying the following conditions: $(s, \sigma_i(s)) \in S$; and if $s \sim_i s'$ then $\sigma_i(s) = \sigma_i(s')$. A *strategy profile* is a tuple $\sigma = (\sigma_i)_{i \in N}$ of strategies for each player. A strategy profile determines a unique function (also denoted by σ) from the set of non-terminal states into A, defined by: $\sigma(s) =: \sigma_i(s)$ if $P(s) = i$. I will denote by $\Sigma = \Sigma_G$ the set of all the strategy profiles of the game G. Our logical language consists of a set $Prog$ of *programs* and a set For of *formulas*. The programs are formed from *tests* $?\varphi$ (as usually), *basic action labels* - one for each action $a \in A$, and a new action label $next$, called the *next-step action*. We allow the usual dynamic-logic constructions for programs: composition ;, union (non-deterministic choice) $\cup$ and Kleene star $*$. We have *atomic sentences* p_i, one for each possible utility score $p \in U$ and player $i \in N$. The meaning of p_i will be "the current state is a terminal state at which player i's score is greater or equal to p". We also have an atomic sentece i for each player i, to denote the fact that it is player i's turn to move. Besides these, we allow propositional variables $P, Q, \ldots$, whose meaning will depend on the model. Denote by $AtProp$ the set of all the atomic propositions, both variables and constants. Formulae are built from atomic sentences, using propositional connectives, dynamic modalities $[\pi]\varphi$ ("after program π is executed, φ is true") for each program π, and epistemic modalities $K_i\varphi$ ("player i knows that φ"). For expressing important game properties involving *common knowledge*, we might have to add later the common-knowledge operator $K_I\varphi$ ("φ is common knowledge among all the members of the group I"). The semantics is given in terms of pairs $(state, strategy-profile)$, as announced. Namely, we associate to the game G a Kripke frame $K_G = (S \times \Sigma, R_\pi, R_{next}, R_i)_{\pi \in Prog, i \in N}$, where the frame relations are defined by:

$$(s,\sigma)R_a(s',\sigma') \text{ iff } s \to^a s', \sigma = \sigma';$$
$$(s,\sigma)R_{next}(s',\sigma') \text{ iff } s \to^{\sigma(s)} s', \sigma = \sigma';$$
$$(s,\sigma)R_i(s',\sigma') \text{ iff } s \sim_i s', \sigma_i = \sigma_i';$$

while $R_{\pi \cup \pi'}, R_{\pi;\pi'}$ and R_{π^*} are defined as usual in dynamic logic semantics. A *model* $M_G = (K_G, V)$ on this frame is given by a valuation $V : AtProp \to \mathcal{P}(S \times \Sigma)$, such that: $V(p_i) = \{(s,\sigma) : u_i(s) = p\}$, $V(i) = \{(s,\sigma) : P(s) = i\}$. Truth and validity are defined as usually. We use the following abbrebiations for special programs: $\epsilon_i =: (?i; \bigcup_{a \in A} a) \cup (?\neg i; next)$, and $\epsilon =: \bigcup_{i \in N} \epsilon_i$. With this definition, we obtain the following characterizations:

1. σ is a Nash equilibrium profile in the game G iff the pair (s_0, σ) satisfies the following sentence
$$\theta^G_{Nash} \quad =: \quad \bigwedge_{i \in N} \bigwedge_{p \in U} (< \epsilon_i^* > p_i \to < next^* > p_i);$$

2. σ is a subgame perfect equilibrium iff the pair (s_0, σ) satisfies the sentence $[\epsilon^*]\theta^G_{Nash}$;

3. G is a perfect information game iff the schema $\bigwedge_{a \in A, i \in N}([a]P \to [a]K_iP)$ is valid on the frame K_G.