Payoffs, Intensionality and Abstraction in Games

Chris Hankin[1] and Pasquale Malacaria[2]

[1] Institute for Security Science and Technology, Imperial College London
[2] School of Electrical Engineering and Computer Science,
Queen Mary University of London

Abstract. We discuss some fundamental concepts in Game Theory: the concept of payoffs and the relation between rational solutions to games like Nash equilibrium and real world behaviour. We sketch some connections between Game Theory and Game Semantics by exploring some possible uses of Game Semantics strategies enriched with payoffs. Finally we discuss potential contributions of Abstract Interpretation to Game Theory in addressing the state explosion problem of game models of real world systems.

1 Introduction

1.1 An Historical Note

Samson Abramsky joined the Department of Computing at Imperial College London in 1983 and Hankin joined him there in 1984. They had previously collaborated on the launch of an informal inter-collegiate PhD course on Theoretical Computer Science. Their scientific collaboration with Geoffrey Burn led to work on higher-order strictness analysis of functional programs [7] and ultimately to an edited volume on abstract interpretation of declarative languages [1]. Whilst Samson's main focus was on domain logics at this time, he also made important contributions to the theory of program analysis through his invention of the notion of *polymorphic invariance* [2] and a deep study of the role of logical relations in establishing the correctness of program analyses [3]. Even his work on domain logics found an application in program analysis through Jensen's development of strictness logics [14].

Malacaria came to Imperial College London in 1993. He worked with Samson and Radha Jagadeesan on Game Semantics, solving the long standing open problem of providing a fully abstract semantics for PCF [5].

The authors of this paper subsequently worked together on using Game Semantics as a basis for program analyses that were correct by construction. This work culminated in [15] which uses Game Semantics as a basis for an Information Flow analysis. More recently, we have been studying the use of Game Theory in decision support for cyber security.

1.2 This Paper

Textbook presentations of Game Theory are often extensional: game solutions are found on normal form games. We are interested in a more intensional

B. Coecke, L. Ong, and P. Panangaden (Eds.): Abramsky Festschrift, LNCS 7860, pp. 69–82, 2013.

approach and look to Game Semantics to provide a framework for approaching this problem. In the next section we address some of the criticisms that have recently been addressed at Game Theory. We then present a common framework based on [8]. We next discuss some of the potential uses of payoffs in Game Semantics. We conclude by considering the role of abstraction in making the problem of finding game solutions more tractable.

2 The Problem with Payoffs

Game Theory is sometimes criticised for providing solutions that are unrealistic. Classical examples enlightening this criticism are provided by the centipede game or the game of ultimatum bargaining, both extensively studied by economists and social scientists. More recently what we believe to be a similar criticism has also arisen in a cyber-security context. In this section we will discuss these criticisms.

2.1 The Centipede Example

The centipede game [18] is a well known example in the Game Theory literature illustrating a variety of features and issues about Nash equilibrium solutions in games. The centipede is a multi-stage alternating game where at each stage the player whose turn is to move can either decide to end the game with some payoffs or continue. Crucially the payoffs are arranged so that if one player were to decide to continue to the next stage but at the next stage the other player were to decide to stop, the first player would get an inferior payoff than if he had decided to stop at the previous round. Figure 1 illustrates a simple centipede game. If the red player were to stop at the first stage he would get 3.20, if the blue player were to stop at stage 2, the red player would however get an inferior payoff of 1.60. However if both players were to continue playing until the end they would end up with much higher payoffs than if they were to stop at any earlier stage.

One interesting aspect of this game is that the equilibrium solution says that players should stop at the very beginning, thus ending up with payoff 3.20 for

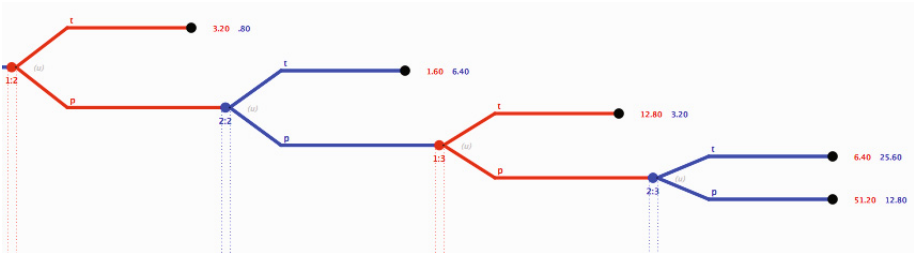

Fig. 1. The Centipede game

red and .80 for blue. This solution is explained as follows: at the very last move, which is a blue move, the blue player has interest in choosing the move giving him payoff 25.6 which would be bad for the red player who would get 6.40. Hence the red player should have stopped at the previous round as he would have gotten the higher payoff 12.8; but in that case the blue player would have gotten 3.20, which is inferior to 6.40 the reward he would have gotten had he stopped at the previous round; and so on...

The problem is that when this game is played "in reality" the outcome is different. Experimental studies show that people tend not to stop at the early stages, a notable exception being chess grand masters who when playing the game tend to stop at the beginning and so behave consistently with the equilibrium solution.

So what is wrong with Nash equilibrium? Are ordinary people irrational? Is Game Theory unrealistic?

These are fundamental questions that arise over and over in one form or another in any context where Game Theory is applied.

A classic game theoretical answer to these questions is that there is not much wrong with Nash equilibrium itself: the point is that the game with the payoffs described above is not the game ordinary people play when they are asked to play it.... In real life, e.g. when the payoffs are money, people will reason that the other player may have interest to go ahead too and not stop at the next round. Hence as long as the red player thinks that his expected future payoff, given the probability that the blue player stops at the next round, exceeds the payoff he would get by stopping at the current round then he will carry on. Therefore to match reality the payoffs of the game should be adjusted to match the real payoffs. This matching is tricky and depends on very specific circumstances, e.g. the amount involved compared with the wealth and greediness of the individual, the suspicion about the other player motives and trustworthiness, etc. Somehow the payoffs need to be tailor-made for each player.

Without taking into account all these factors in the payoffs, comparing the formal game with the game played in the real world is just comparing apples and pears.

2.2 The Prime Factorization Game

Halpern and Pass [17] consider the following game. A player is given a random odd n-bit number x and he is supposed to decide whether x is prime or composite. Guessing correctly will give him \$2, however an incorrect guess will give him −\$1000, i.e. he will have to pay a penalty of \$1000. The player can however choose the "playing safe" strategy by giving up, in which case he receives \$1.

The game theoretical solution is to play, i.e. not to give up: game theoretical players have mathematical unbounded power so they never make wrong guesses and so they will always get \$2. However clearly in reality people wouldn't choose the equilibrium solution: we don't need experiments to see that.

So what is wrong with Nash equilibrium? Are ordinary people irrational? Is Game Theory unrealistic?

We have already seen these questions above and the game theoretical answer is the same as we saw above: people are not playing the game those rules describe. In order to model reality we need to take into account the cost of computation and so the crucial point in matching the above payoffs to the real game is that guessing correctly will give the player payoff $\$2 - F(x)$, where $F(x)$ is a function representing the cost of determining whether x is prime. Again this $F(x)$ ought to be tailor-made for specific players, e.g. players having available thousands of powerful machines will tend to play for larger x as long as other factors like the electricity required by the factorisation will make it profitable still. Even if general guidelines from computability and complexity may be helpful they still miss factors that may be important to the model, e.g. the cost of electricity: for example in real world, crypto contexts like Bitcoin mining, electricity and the cost of GPUs are the key criteria in deciding whether to play the game. Once the right $F(x)$ is used the game theorist would claim that the solution matches reality.

This game is interesting because $F(x)$ depends heavily on intensional aspects like the computational resources available and limits of computational devices. These factors are external to classical Game Theory: an attempt to develop a Game Theory where computational limitations are taken into account is developed in [17]. We will suggest in section 4.1 that the intensional aspect of Game Semantics can be used for similar purposes.

3 A Common Framework

To get further in the discussion and relate Game Theory and Game Semantics it is necessary to have a formalism common to both. This has been developed by Chatterjee, Jagadeesan and Pitcher in [8]. We present some definitions based on that paper that will help in the following discussion.

3.1 Turn-Based Probabilistic Games

We consider two person games with alternating moves. At each state, exactly one player can make a move, following which the system may evolve into a new state with some probability. At this point, the other player can make a move. Hence, a typical evolution has the form:

> The system with players A and B is in state s. In this state A can move with action a resulting in the system moving to state t from which it evolves to state t_i with probability p_i. In this state B can move with action b and so on.

In [8] this is formalized by a structure $((S, E), (S_1, S_2, S_{1\bullet}, S_{2\bullet}), \delta)$ where (S, E) is a graph with nodes S and labelled edges E, $(S_1, S_2, S_{1\bullet}, S_{2\bullet})$ is a partition of S s.t. $E \subseteq \cup_i (S_i \times L \times S_{i\bullet})$ (with L the label set) and $\delta : S_{i\bullet} \to \mathcal{D}(S_{(i+1)\%2})$ associates a distribution over the states to each target state of a player move, from which the other player can move.

We use player A as a synonym for player 1 and player B as a synonym for player 2. We will consider finite games, i.e. all plays have a finite length.

As argued in [8] this is a general framework for stochastic games, e.g. non strictly alternating games can be interpreted using *dummy* moves. To recover Game Semantics of sequential languages we can use point mass distributions, in effect eliminating probabilities.

Edges (in E) have rewards associated to them, these are encoded by two maps r_1, r_2 with $r_i : E_i \to \mathbb{R}$ where E_i is the set of transition whose source is a state where player i can move.

As is usual in the literature, a strategy for a player is a method to extend a play. Given the history of the play where player A can move, a strategy for A chooses a successor state and an action to extend the play. A pure strategy is one where this choice is given by a function. A mixed strategy is a choice of several pure strategies according to some probabilities. Notice however that in a play, once a strategy is chosen by a player, that player will stick to it along that particular play, i.e. the same function is used to decide what move to make at each stage.

Since we can encode histories into states it will be enough to consider history-free or Markovian strategies, i.e. maps from states to transitions having that state as origin. Hence by pure strategies in this paper we refer to maps $\sigma, \tau : s \mapsto e$ where $e = s \to^l t$ for some l, t, with σ associated to player 1 and τ to player 2, and ρ a generic strategy. We will often write transitions as triples (s, l, t).

Given a path in the game tree, i.e. a sequence of transitions $P = e_1, ..., e_n$ we consider the mean value of this sequence to each player, so

$$\mathtt{val}_i(P) = \frac{\sum_j r_i(e_j)}{m}$$

where the transitions e_j have source states where player i can move and m is the number of such transitions in the path P. As usual the probability of the path P is the product of the probabilities of the edges in P.

A *path* in a strategy ρ for player i is a path in the game tree where whenever it is the turn of player i to move, it will use ρ to choose the move. The set of all paths for ρ is denoted by Π^ρ.

Given a pair of strategies (σ, τ) the payoff v_ρ for player i is the expected values of the mean values of the sequences possible according to (σ, τ), i.e. the payoff (v_σ, v_τ) is defined as

$$v_\sigma = E\{\mathtt{val}_\sigma(P) \mid P \in \Pi^\sigma \cap \Pi^\tau\}, v_\tau = E\{\mathtt{val}_\tau(P) \mid P \in \Pi^\sigma \cap \Pi^\tau\}$$

A Nash equilibrium (N.E.) is a strategy pair (σ^*, τ^*) from which no player has advantage in deviating unilaterally, i.e.:

$$\forall \sigma, \tau. \;\; (v_\sigma, v^*_\tau) \leq (v^*_\sigma, v^*_\tau) \wedge (v^*_\sigma, v_\tau) \leq (v^*_\sigma, v^*_\tau),$$

3.2 Game Algebras

An interesting contribution of [8] is to consider how to build up composite games and in particular how equilibrium solutions are preserved by such constructions.

The authors present a rich algebra of games which includes operators for synchronous product, restriction, sequencing, iteration, player choice, probabilistic choice and tensor. Here, we just consider binary player choice – player i is able to choose between games with player i start states.

We consider rooted game graphs:

$$G^A = ((S^A, E^A), (S_1^A, S_2^A, S_{1\bullet}^A, S_{2\bullet}^A), \delta^A)$$

and

$$G^B = ((S^B, E^B), (S_1^B, S_2^B, S_{1\bullet}^B, S_{2\bullet}^B), \delta^B)$$

with start states $s^A \in S_i^A$ and $s^B \in S_i^B$. Then the player choice between these two games, written $G^A \oplus_i G^B$, is a game graph $((S, E), (S_1, S_2, S_{1\bullet}, S_{2\bullet}), \delta)$ such that:

$$\begin{aligned}
S_i &= S_i^A \uplus S_i^B \uplus \{\langle s^A, s^B \rangle\} \\
S_{(i+1)\%2} &= S_{(i+1)\%2}^A \uplus S_{(i+1)\%2}^B \\
E &= E^A \uplus E^B \uplus \{(\langle s^A, s^B \rangle, l, t) \mid (s^A, l, t) \in E^A \vee (s^B, l, t) \in E^B\} \\
\delta &= \delta^A \uplus \delta^B
\end{aligned}$$

The reward functions are also modified in the following way (for $j = 1, 2$):

$$r_j(e) = \begin{cases} r_j^A(e), \text{if } e \in E^A \\ r_j^A((s^A, l, t)), \text{if } e = (\langle s^A, s^B \rangle, l, t) \ \wedge \ (s^A, l, t) \in E^A \\ r_j^B(e), \text{if } e \in E^B \\ r_j^B((s^B, l, t)), \text{if } e = (\langle s^A, s^B \rangle, l, t) \ \wedge \ (s^B, l, t) \in E^B \end{cases}$$

Following [8] we then have that any equilibrium payoff of a component game is an equilibrium payoff of the choice game if and only if there is no other equilibrium payoff in the other component with a higher value for player i. The cited paper discusses similar results for the other games constructions.

4 Payoffs in General Game Semantics

4.1 Some General Remarks

The games algebra from [8] is closely related to game constructions in Game Semantics, but while that work shows some interactions between equilibria and constructions on Game Semantics, it doesn't demonstrate their relevance for game theoretical problems.

We now consider more general relationships between these theories.

Game Semantics were first introduced in [5,13]. Games are "types" i.e. for example the game $N \to N$ represents the space of programs taking a natural

number as input and returning a natural number. The basic types are simple one stage games where the player named Opponent starts by asking a question (he has only one move available, i.e. a single question) and the player named Proponent answers the question with a value of that basic type, for example the boolean game B can be described by the game graph on the left of Figure 2: Hence each basic value corresponds to a strategy for Proponent. The game graph

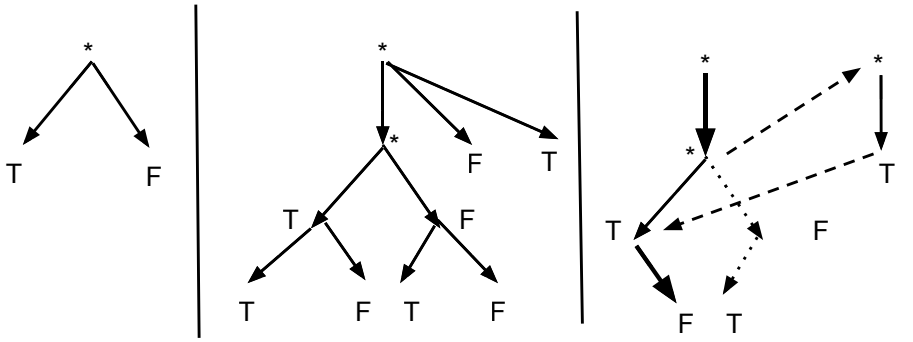

Fig. 2. Boolean Game

for $B \multimap B$ is in the centre of Figure 2. The affine implication $B \multimap B$ is a weaker form of $B \to B$ and represents the type of all algorithms from B to B using their argument at most once. Notice that the role of Proponent and Opponent change at each level in the game graph: * is a move for Opponent at the first level of the game graph but becomes a Proponent move at the second level: this is the program asking for its input. At the third level Opponent plays data values: this corresponds to the environment providing values for the program. Next the program plays a data value which, in the $B \multimap B$ example, is the final value returned by the program.

Programs are interpreted as strategies for Proponent; for example in the game $B \multimap B$ the program that performs the negation of a boolean input is interpreted by the strategy whose paths in the game graph are $**TF, **FT$. The constant programs that do not inspect their argument are: $\lambda x.T$ with path $*T$ and $\lambda x.F$ with path $*F$.

An essential aspect of Game Semantics is composition; composing two strategies corresponds to synchronizing them and hiding their interaction. For example to compose the strategy for the boolean negation with the constant T-function results in the following interaction: start by playing the strategy negation, so to the initial Opponent question play the following question asking for the argument, now see this question as the initial question for the strategy constant, hence the strategy constant will answer T, this will be seen as the Opponent answer to the Proponent question and so Proponent will now play the negation i.e. F. The crucial point here is that these interactions where Proponent moves are transferred to the other game as the Opponent moves is then hidden, so the composition results in the simple strategy in $B \multimap B$ that to the initial Opponent

question answers T. This is illustrated in the right hand side of Figure 2 where only the first and last move are left once the interaction (dotted arrows) are factored out.

The important features of Game Semantics include:

1. compositionality
2. trace level description of computational processes
3. abstractions of games available to allow for static analysis of program behaviours [15]

The formalism introduced in [5] is consistent with the common framework introduced above and treated in more detail in [8]. The reward functions have not previously been used in the work on programming language semantics[1] – we could assume that the reward functions are constant zero functions.

A first simple question is: can payoffs discriminate different strategies?

The answer is yes: we can distinguish Game Semantics strategies by appropriate rewards. For basic types we could just distinguish them by associating as reward the $n+1$ where n is the answer move, e.g. in the boolean game F would have Proponent reward of 1 and T reward 2. To recover Game Semantics composition we can think of assigning zero rewards to moves in the hidden part. In this interpretation hence two programs representing the same function become indistinguishable in a game theoretical sense.

This simple rewards structure can be elaborated to make finer distinctions, for example we can make the rewards in the hidden interaction count. e.g. by "accumulating the rewards in the hidden part" we can get a cost for the length of computation, so two strategies answering the same number but doing so with different computational cost will differ. Also complexity distinctions would appear at this stage, e.g. polynomial vs non-polynomial strategies. A step in this direction is already taken in [11]. This line of investigation may provide a way to find the "right payoffs" in a computational setting in the spirit of [17].

5 Game Theory in Game Semantics

We now explore a novel way to think of Game Semantics where Nash equilibrium becomes a tool for deciding which strategy the system and the environment should choose, given a particular objective quantified by payoffs. Notice how we now move the focus, from the classical single strategy game semantics view, to a space of candidate strategies in a game type.

In the following two examples we play classical Game Theory games in the context of Game Semantics. This is specially interesting in that these games enlighten deep issues about social interaction, competition and cooperation.

We interpret these issues in computer science terms of trustworthiness, usability and security.

[1] There is however some recent interesting work by Clairambault and Winskel [9] on payoffs for concurrent games.

5.1 The Centipede PCF Game

We revisit the centipede game within Game Semantics as follows. Consider the game

$$(((B \multimap B) \multimap B) \multimap B)$$

In this game there is a strategy for Proponent answering immediately the initial Opponent question and one following the initial Opponent question with another question about the second rightmost B and so on. By assigning the appropriate rewards to the answer moves we recover the centipede game. The rewards are as follows: questions have 0 rewards for both players, the answer T at level 0 has rewards 3.20 for Proponent and 0.8 for Opponent, the answer F has reward 0 for both players (at any level), the answer T at level $i+1$ has Proponent reward $2n$ where n is the Opponent reward for the Opponent T answer at level i and symmetrically the answer T at level $i+1$ has Opponent reward $2m$ where m is the Proponent reward for the Proponent T answer at level i.

We now consider Opponent and Proponent strategies whose first answer is T and all the following answers are F. This models the centipede game. Hence the equilibrium would tell us that in this game the rational thing to do is to play the constant program $\lambda x.T$. What is the meaning of the equilibrium in this context? It expresses a mistrust by the system that the environment has any reliability, e.g. the system fears that the environment will not perform correctly, if at all, the required computation.

It is easy to see, consistently with the previous discussion about the original centipede game, how degrees of trust of the system with respect to the environment could be reflected in the payoffs and so how to make the choices about what the best thing the system should do given a specific degree of trust.

5.2 The System Administrator Dilemma

In a similar vein we can consider the famous prisoner dilemma game and we look for a simple interpretation in Game Semantics. In the prisoner dilemma both players have two options: to cooperate or to not cooperate (defect). Each player receives his maximum payoff if he defects and the other cooperates, his minimum if he cooperates and the other player defects, mutual cooperation provides second best payoff and mutual defection second worst payoff; so for each player payoffs have the property $c > a > d > b$ where c is defect-cooperate, a mutual cooperation, d mutual defection and b is cooperate-defect.

Again we can see this as a system and environment game where the system may want to guarantee security and the environment usability. So the system may have the highest security by denying all access requests; in fact this is the only secure strategy in an authentication system. The environment on the other hand may have interest, from a usability point of view, in bypassing any authentication by providing access to any request.

We can model this in the space

$$((N \multimap B) \multimap B)$$

Here Proponent can choose the answer false to the initial opponent question (defect) or may instead decide to cooperate by choosing the strategy that engages with the environment by asking the initial question in the authentication module of type $(N \multimap B)$. Here Opponent can choose to answer true and so provide access to any user or he can act with due diligence and ask for the user credentials at type N which the system provides, following which Opponent checks the user credentials and provides an authentication response (true/false, accept/reject) and based on this answer Proponent can now decide whether to allow user access.

The game theoretical solution is that in the one shot prisoner dilemma the only Nash equilibrium is for both players to defect: so the system and opponent are better off not engaging with each other: the system is secure and usable, because both operate in a vacuum. A major point of debate in the Game Theory community is that the payoff the players get from both defecting may be much below the payoff they would get by both cooperating: it would make much more sense for the system to have its resource used instead of being paralysed under the fear of unauthorized intrusion.

An interesting twist is in considering the iterated prisoner dilemma. Here we are hence thinking of something like an operating system where authentication and resource access are available an unbounded number of times. In this case the solution favours cooperation and a richer panorama of equilibria with higher payoff than defection is obtained, e.g. when each player plays cooperation until the other player defects at which point the player will defect too (this is the Grim trigger strategy). This strategy well reflects the idea of a good compromise between usability and security: play nice until the other player plays dirty.

6 Games and Abstraction

Many classic examples in Game Theory like the prisoner dilemma, the centipede game and prey-predator games can be seen as illustrating the richness and conceptual complexity of rational interaction. However when applying Game Theory to the real world a state explosion problem often occurs. A large set of actions or states usually makes equilibrium non-computable; also the conceptual aspect is obfuscated by such complexity. Some form of ad-hoc abstraction is then introduced for these computational and conceptual reasons.

Our longer term objective is to replace this ad-hoc approach to abstraction by a more rigorous framework akin to Abstract Interpretation. In [15] we developed an Abstract Interpretation for Game Semantics; the most salient features of this abstraction were

1. replacing all data values by a single abstract data value
2. replacing the potentially infinite depth tree game by cyclic graphs

One application of this abstraction was in security, by tracking information flow along paths in the graphs and, for example, disallowing paths where a "high" move was followed by an observable "low" move. Similar ideas have been developed, in the context of access control, by Abramsky and Jagadeesan in [4]. We

should mention also related work on abstract game semantics which has been developed by Ghica [12] and Ong [16].

Following our earlier discussion a natural development of this line of work would be to add payoffs. This would pose two important questions:

1. what are the appropriate payoffs to use in such abstract games?
2. how should we interpret Nash equilibria in the abstract game – what do they tell us about Nash equilibria in the concrete game?

Most recently, the authors of this paper have been studying the use of Game Theory in cyber security. In the last decade Game Theory has been increasingly applied in this area. Examples of applications are in the field of intrusion detection systems, anonymity and privacy, economics of network security and cryptography: for a survey of these applications we refer to [6].

The basic idea is that many cyber security situations can be modelled in terms of an Attacker attempting to breach security of the system and/or damaging its services and a Defender aiming to enhance the system security both in terms of design and response. The Attacker and the Defender clearly have conflicting goals that can be quantified in terms of economic gain/loss or disruption time; even more serious criminal threats like cyber terrorism can, with some effort, be quantified in a reasonable way. The Attacker and the Defender will, in general, interact with some knowledge of each others possible actions: to be effective both players need to be clever or, in Game Theoretical terms, rational. Because of this the notion of Nash equilibrium, discussed above, is important; an equilibrium describes a possible outcome of decision makers trying to optimize their gain while being aware of each others possible actions. Often this will result in mixing possible actions according to some probability.

Attacker and the Defender are already an abstraction whose appropriateness is often questionable e.g. when multiple players and coalitions are more appropriate. More crucially we tend to abstract on possible behaviours, i.e. while in the real world a very large number of choices may be possible we abstract them to few, for example in a cyber-security scenario there could be thousands of different types of malware leading to botnets but we may find it convenient, or even indispensable, to reduce them to a small set of "Attacker's choices". The underlying argument is that if they are "similar enough" then the reasoning on the reduced set is applicable to the original set.

6.1 A Simple Example

If we restrict ourselves for simplicity to the case of games in normal form we can think of the following scenario: Attacker A can choose between strategies $A_1, A_2, \ldots A_n$ with A_1 being "no attack" and $A_2, \ldots A_n$ being similar potential malware attacks. Suppose defender D can choose between three strategies, D_1, D_2, D_3 e.g. D_1 could be do nothing, D_2 to alert the user, D_3 to stop the service. Suppose moreover that $A_2, \ldots A_n$ result in very close payoffs. In most cases then we would be inclined to translate this into a simpler game where

A has two strategies $A_1, A_{2,\ldots,n}$ with the $A_{2,\ldots,n}$ payoff being a function of the original payoffs e.g. the average, or the minimum or maximum of the original payoffs. A stable Nash equilibrium in the simpler game (2×3) matrix would then suggest the following strategy for D in the original $n \times 3$ matrix: Play D_k with $A_{i,\ i>1}$ with the same probability of playing D_k with $A_{2,\ldots,n}$ in the reduced game. Intuitively if the $A_2, \ldots A_n$ are almost indistinguishable play the same way with each of them individually.

A concrete example is given in Figure 3 where the Attacker has three choices (columns) the last two having very similar payoffs and Figure 4 where the two last choices are reduced to one by averaging. The meaning of the payoff values should be clear e.g. $(10, -22)$ indicates that blocking malware 2 is very good for the Defender (value 10) and very bad for the Attacker (value -22). Both games have a unique and very close Nash equilibrium: in the original Defender plays 1 (resp 3) with probability $\frac{3}{7}$ (resp $\frac{4}{7}$) and Attacker plays 1 (resp 3) with probability $\frac{15}{22}$ (resp $\frac{7}{22}$) . In the reduced game equilibrium Defender plays 1 (resp 3) with probability $\frac{16}{35}$ (resp $\frac{19}{35}$) and Attacker plays 1 (resp 2) with probability $\frac{15}{22}$ (resp $\frac{7}{22}$).

A formal way to build this abstraction is to consider the matrices α, γ defined in Table 1.

The abstraction is given by α and the concretisation by γ. The abstract game in Figure 4 is obtained by the original game from Figure 3 by multiplying it with the matrix α.

	1		2		3	
1	2	2	-5	20	-5	22
2	-2	-1	0	-5	1	-7
3	-5	-5	10	-22	10	-20

Fig. 3. The original malware game

	1		2	
1	2	2	-5	21
2	-2	-1	0.5	-6
3	-5	-5	10	-21

Fig. 4. The abstract malware game

Table 1. Matrices α and γ

$$\begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

The general form of α, γ for a general normal form game is given by γ being the normalized transpose of α and α is the matrix whose columns are buckets of columns of the original matrix and coefficients are: $\alpha_{i,j} = 0$ if i, j are not in the same bucket, $\alpha_{i,j} = \frac{1}{n}$ if i, j are in the same bucket and n is the size of the bucket.

Then γ is the Moore-Penrose pseudo-inverse of α i.e. it satisfies the following properties:

1. $\alpha\gamma\alpha = \alpha$
2. $\gamma\alpha\gamma = \gamma$
3. $(\alpha\gamma)^* = \alpha\gamma$
4. $(\gamma\alpha)^* = \gamma\alpha$

An important consequence of γ being the Moore-Penrose pseudo-inverse of α is that γ provides the least square approximation [10]. In other terms whilst $\gamma\alpha = 1$ which, consistently with the classical theory of Abstract Interpretation, means that the abstraction α is surjective, the other composition $\alpha\gamma$ provides a "best fit" (least square) approximation.

The Abstract Interpretation ideas we have sketched here are applicable beyond normal form games, e.g. when considering stochastic games, and can be integrated with Abstract Interpretation of Game Semantics and approximate bisimulation.

7 Final Remarks

The paper should be seen as an initial roadmap rather than a completed piece of work. In particular, the role of abstraction in making the approximate solution of games a tractable problem needs considerably more study.

Some of the seeds of the ideas presented in this paper were sown twenty to thirty years ago through our joint and separate work with Samson. We wish Samson many years of continued scientific work and hope, one day, to be able to discuss our results with him.

Aknowledgments. We thank Dusko Pavlovic and Fabrizio Smeraldi for helpful discussions on this work. Both authors are partially supported by the "Games and Abstraction: the Science of Cyber Security" project (EPSRC projects EP/K005790/1 and EP/K005820/1).

References

1. Abramsky, S., Hankin, C. (eds.): Abstract Interpretation of Declarative Languages. Ellis Horwood (1987)
2. Abramsky, S.: Strictness analysis and polymorphic invariance. In: Ganzinger, H., Jones, N.D. (eds.) Programs as Data Objects. LNCS, vol. 217, pp. 1–23. Springer, Heidelberg (1986)

3. Abramsky, S.: Abstract Interpretation, Logical Relations and Kan Extensions. J. Log. Comput. 1(1), 5–40 (1990)
4. Abramsky, S., Jagadeesan, R.: Game Semantics for Access Control. Electr. Notes Theor. Comput. Sci. 249, 135–156 (2009)
5. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full Abstraction for PCF. Inf. and Comput. 163(2), 409–470 (2000)
6. Alpcan, T., Basar, T.: Network Security: A Decision and Game Theoretic Approach. Cambridge University Press (2011)
7. Burn, G.L., Hankin, C., Abramsky, S.: Strictness Analysis for Higher-Order Functions. Sci. Comput. Program. 7(3), 249–278 (1986)
8. Chatterjee, K., Jagadeesan, R., Pitcher, C.: Games for Controls. In: CSFW 2006, pp. 70–84 (2006)
9. Clairambault, P., Winskel, G.: On concurrent games with payoff, `http://www.cl.cam.ac.uk/~cdt25/ecsym/Publications/`
10. Di Pierro, A., Hankin, C., Wiklicky, H.: Measuring the confinement of probabilistic systems. Theor. Comput. Sci. 340(1), 3–56 (2005)
11. Ghica, D.R.: Slot games: a quantitative model of computation. In: Proc. POPL 2005, pp. 85–97 (2005)
12. Ghica, D.R.: Applications of Game Semantics: From Program Analysis to Hardware Synthesis. In: LICS 2009, pp. 17–26. IEEE Computer Society (2009)
13. Hyland, M., Ong, L.: On Full Abstraction for PCF: I, II, and III. Inf. and Comput. 163(2), 285–408 (2000)
14. Jensen, T.P.: Strictness Analysis in Logical Form. In: Hughes, J. (ed.) FPCA 1991. LNCS, vol. 523, pp. 352–366. Springer, Heidelberg (1991)
15. Malacaria, P., Hankin, C.: Non-Deterministic Games and Program Analysis: An Application to Security. In: LICS 1999, pp. 443–452. IEEE Computer Society (1999)
16. Ong, C.-H.L.: Some Results on a Game-Semantic Approach to Verifying Finitely-Presentable Infinite Structures (Extended Abstract). In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 31–40. Springer, Heidelberg (2006)
17. Pass, R., Halpern, J.: Game theory with costly computation: formulation and application to protocol security. In: Proceedings of the Behavioral and Quantitative Game Theory: Conference on Future Directions. ACM (2010)
18. Rosenthal, R.: Games of Perfect Information, Predatory Pricing, and the Chain Store. Journal of Economic Theory 25(1), 92–100 (1981), doi:10.1016/0022-0531(81)90018-1
19. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)