

Modelling MAS with Finite Analytic Stochastic Processes

Luke Dickens, Kryisia Broda and Alessandra Russo¹

Abstract. The Multi-Agent paradigm is becoming increasingly popular as a way of capturing complex control processes with stochastic properties. Many existing modelling tools are not flexible enough for these purposes, possibly because many of the modelling frameworks available inherit their structure from single agent frameworks. This paper proposes a new family of modelling frameworks called FASP, which is based on *state encapsulation* and powerful enough to capture multi-agent domains. It identifies how the FASP is more flexible, and describes systems more naturally than other approaches, demonstrating this with a number of robot football (soccer) formulations. This is important because more natural descriptions give more control when designing the tasks, against which a group of agents' collective behaviour is evaluated and regulated.

1 Introduction

Modelling stochastic processes is a time consuming and complicated task that involves many issues of concern. Among these, some that may appear high on most modellers' list are highlighted below. What type of language to use for building the model is often among the first decisions to make. Models are often finite state but they represent real-life continuous domains, so approximations are needed. A common problem is deciding what approximations to make: it is often difficult to determine how drastically a model will suffer in terms of results from making choice A rather than choice B; moreover if the language is restrictive or arcane it can hamper further. Furthermore, the modeller may wish to relate small problems to more complex tasks, especially if they want to partially solve or learn solutions on the simpler systems, and then import them to larger and more complicated ones². A related issue is one of tuning — a system may have some parameters which are hard to anticipate; instead experimentation and incremental changes might be needed, so a language which supports natural representations with tunable features is desirable. Additionally, it is preferable to have as many analytic and learning tools available as is realistically possible. This means that the modelling framework should either support these directly or allow models to be transformed into frameworks which support them. Finally, single agent control paradigms may not be sufficient, nor may single real valued measures for evaluation; the modern experimenter may want more flexibility; potentially using multiple non-cooperative, isolated, agents each with a number of orthogonal constraints.

This paper highlights some of the issues above, and aims to address them by proposing a family of modelling frameworks known as Finite Analytic Stochastic Processes (FASP). We will demonstrate: how the multi-agent scenarios alluded to above can be described

within this family; the naturally descriptive nature of the representations thus produced; and which real world applications might benefit. We highlight some of the features of this family by examining an often visited problem domain in the literature, that of robot soccer³. To do this, we introduce a relatively simple problem domain first developed by Littman [16]. We will show how this can be rewritten in the FASP framework in a number of ways, with each one giving more flexibility to re-tune in order to examine certain implicit choices of Littman. Then we will examine some other more recent robot soccer models from the related literature, which extend and add more interest to Littman's original formulation. We will show that these can also be represented by the FASP family, and will show how our more flexible framework can be exploited here. Ultimately, we generate a tunable turn-based non-cooperative solution, which can be scaled up to finer granularity of pitch size and a greater number of players with simultaneous team and independent objectives.

We finish with a discussion of the added value provided by the FASP languages, who might benefit from them, and some of the new challenges this presents to the reinforcement learning community (amongst others). The guiding principle here is that richer domains and evaluation — provided by better modelling tools, allows us to regulate agent and group behaviour in more subtle ways. It is expected that advanced learning techniques will be required to develop strategies concordant with these more sophisticated demands. This paper is a stepping stone on that longer journey.

2 Preliminaries

This section introduces the concept of Stochastic Map and Function, and uses them to construct a modelling framework for stochastic processes. Initially, we formalise a way of probabilistically mapping from one finite set to another.

Definition 1 (Stochastic Map) *A stochastic map m , from finite independent set X to finite dependent set Y , maps all elements in X probabilistically to elements in Y , i.e. $m : X \rightarrow \text{PD}(Y)$, where $\text{PD}(Y)$ is the set of all probability distributions over Y . The set of all such maps is $\Sigma(X \rightarrow Y)$; notationally the undecided probabilistic outcome of m given x is $m(x)$ and the following shorthand is defined $\text{Pr}(m(x) = y) = m(y|x)$. Two such maps, $m_1, m_2 \in \Sigma(X \rightarrow Y)$, identical for each such conditional probability, i.e. $(\forall x, y) (m_1(y|x) = m_2(y|x))$, are said to be equal, i.e. $m_1 = m_2$.*

The stochastic map is a generalisation of the functional map, see [8]. Another method for generating outcomes probabilistically is the stochastic function, and relies on the concept of probability density function (PDF). For readers unfamiliar with the PDF, a good definition can be found in [19].

¹ Imperial College London, South Kensington Campus, UK, email: luke.dickens@imperial.ac.uk

² This step-wise training is sometimes called shaping, and is an active area of research, see [7, 14, 22]

³ Or as we like to call it outside the US and Australia, robot football

Definition 2 (Stochastic Function) A stochastic function f from finite independent set X to the set of real numbers \mathbb{R} , maps all elements in X probabilistically to the real numbers, i.e. $f : X \rightarrow \text{PDF}(\mathbb{R})$. The set of all such functions is $\Phi(X \rightarrow \mathbb{R})$; notationally $f(x) = F_x$, where $F_x(y)$ is the $\text{PDF}(\mathbb{R})$ associated with x and belonging to f ; $f(x)$ is also used to represent the undecided probabilistic outcome of F_x — it should be clear from the context which meaning is intended; the probability that this outcome lies between two bounds $\alpha_1, \alpha_2 \in \mathbb{R}$, $\alpha_1 < \alpha_2$, is denoted by $[f(x)]_{\alpha_1}^{\alpha_2}$. Two functions, $f, g \in \Phi(X \rightarrow \mathbb{R})$, identical for every PDF mapping, i.e. $(\forall x, \alpha_1, \alpha_2) ([f(x)]_{\alpha_1}^{\alpha_2} = [g(x)]_{\alpha_1}^{\alpha_2})$, are said to be equal, i.e. $f = g$; if the two functions are inversely equal for each PDF mapping, i.e. $(\forall x, \alpha_1, \alpha_2) ([f(x)]_{\alpha_1}^{\alpha_2} = [g(x)]_{-\alpha_2}^{-\alpha_1})$, then the stochastic functions are said to be inversely equal, i.e. $f = -g$.

Armed with these two stochastic generators, we can formalise the Finite Analytic Stochastic Process (FASP), an agent oriented modelling framework.

Definition 3 (FASP) A FASP is defined by the tuple $(S, A, O, t, \omega, F, i, \Pi)$, where the parameters are as follows; S is the finite state space; A is the finite action space; O is the finite observation space; $t \in \Sigma(S \times A \rightarrow S)$ is the transition function that defines the actions' effects on the system; $\omega \in \Sigma(S \rightarrow O)$ is the observation function that generates observations; $F = \{f^1, f^2, \dots, f^N\}$ is the set of measure functions, where for each i , $f^i \in \Phi(S \rightarrow \mathbb{R})$ generates a real valued measure signal, f_n^i , at each time-step, n ; $i \in \Sigma(\emptyset \rightarrow S)$ is the initialisation function that defines the initial system state; and Π is the set of all control policies available.

Broadly similar to MDP style constructions, it can be used to build models representing agent interactions with some environmental system, in discrete time-steps. The FASP allows an observation function, which probabilistically generates an observation in each state, and multiple measure signals — analogous to the MDP's reward signal, which generate a measure signal at each state. One important feature of the FASP is that it generates observations and measures from state information alone. There is no in-built interpretation of the measure functions, their meaning and any preferred constraints on their output are left to be imposed by the modeller.

Care needs to be taken when interpreting the measure signals, and setting the desired output. These can be considered separate reward functions for separate agents - see section Section 3, or conflicting constraints on the same agent. A FASP does not have a natively implied solution (or even set of solutions): there can be multiple policies that satisfy an experimenters constraints; or there may be none — this is a by-product of the FASP's flexibility.

Fortunately, small problems (those that are analytically soluble) can be solved in two steps, first by finding the policy dependent state occupancy probabilities, and then solving the expected output from the measure functions. This means that measure function interpretation can be imposed late and/or different possibilities can be explored without having to resolve the state probabilities again, [8].

If we confine ourselves to a purely reactive policy space, i.e. where action choices are based solely on the most recent observation, hence $\Pi = \Sigma(O \rightarrow A)$, and a policy, $\pi \in \Pi$ is fixed, then the dynamics of this system resolves into a set of state to state transition probabilities, called the Full State Transition Function.

Definition 4 (FASP Full State Transition Function) The FASP $M = (S, A, O, t, \omega, F, i, \Pi)$, with $\Pi = \Sigma(O \rightarrow A)$, has full state

transition (FST) function $\tau_M : \Pi \rightarrow \Sigma(S \rightarrow S)$, where for $\pi \in \Pi$, $\tau_M(\pi) = \tau_M^\pi$ (written τ^π when M is clear), and, $\forall s, s' \in S$,

$$\tau_M^\pi(s'|s) = \sum_{o \in O} \sum_{a \in A} \omega(o|s) \pi(a|o) t(s'|s, a)$$

It is the FST function that concisely defines the policy dependent stochastic dynamics of the system, each fixed policy identifying a Markov Chain, and together giving a policy labelled family of Markov Chains. A more detailed examination of FASPs (including the Multi-Agent derivatives appearing later in this paper) can be found in [8].

3 Multi-Agent Settings

This section shows how the FASP framework can be naturally extended to multi-agent settings. We distinguish two domains; models with simultaneous joint observations and subsequent actions by synchronised agents, and asynchronous agents forced to take turns, observing and acting on the environment. To model the first situation, we define the action space as being a tuple of action spaces — each part specific to one agent, similarly the observation space is a tuple of agent observation spaces. At every time-step, the system generates a joint observation, delivers the relevant parts to each agent, then combines their subsequent action choices into a joint action which then acts on the system. Any process described as a FASP constrained in such a way is referred to as a Synchronous Multi-Agent (SMA)FASP.

Definition 5 (Synchronous Multi-Agent FASP) A Synchronous multi-agent FASP (SMAFASP), is a FASP, with the set of enumerated agents, G , and the added constraints that; the action space A , is a Cartesian product of action subspaces, A^g , for each $g \in G$, i.e. $A = \times_{g \in G} A^g$; the observation space O , is a Cartesian product of observation subspaces, O^g , for each $g \in G$, i.e. $O = \times_{g \in G} O^g$; and the policy space, Π , can be rewritten as a Cartesian product of sub-policy spaces, Π^g , one for each agent g , i.e. $\Pi = \times_{g \in G} \Pi^g$, where Π^g generates agent specific actions from A^g , using previous such actions from A^g and observations from O^g .

To see how a full policy space might be partitioned into agent specific sub-policy spaces, consider a FASP with purely reactive policies; any constrained policy $\pi \in \Pi (= \Sigma(O \rightarrow A))$, can be rewritten as a vector of sub-policies $\pi = (\pi^1, \pi^2, \dots, \pi^{|G|})$, where for each $g \in G$, $\Pi^g = \Sigma(O^g \rightarrow A^g)$. Given some vector observation $o_i \in O$, $\vec{o}_i = (o_i^1, o_i^2, \dots, o_i^{|G|})$, and vector action $\vec{a}_j \in A$, $a_j = (a_j^1, a_j^2, \dots, a_j^{|G|})$, the following is true,

$$\pi(\vec{a}_j | \vec{o}_i) = \prod_{g \in G} \pi^g(a_j^g | o_i^g)$$

In all other respects \vec{o}_i and \vec{a}_j behave as an observation and an action in a FASP. The FST function depends on the joint policy, otherwise it is exactly as for the FASP. Note that the above example is simply for illustrative purposes, definition 5 does not restrict itself to purely reactive policies. Many POMDP style multi-agent examples in the literature could be formulated as Synchronous Multi-Agent FASPs (and hence as FASPs), such as those found in [6, 12, 13, 20], although the formulation is more flexible, especially compared to frameworks that only allow a single reward shared amongst agents, as used in [20].

In real world scenarios with multiple rational decision makers, the likelihood that each decision maker chooses actions in step with every other, or even as often as every other, is small. Therefore it is natural to consider an extension to the FASP framework to allow each agent to act independently, yielding an Asynchronous Multi-Agent Finite Stochastic Process (AMAFASP). Here agents' actions affect the system as in the FASP, but any pair of action choices by two different agents are strictly non-simultaneous, each action is either before or after every other. This process description relies on the FASPs state encapsulated nature, i.e. all state information is accessible to all agents, via the state description.

Definition 6 (AMAFASP) An AMAFASP is the tuple $(S, G, A, O, t, \omega, F, i, u, \Pi)$, where the parameters are as follows; S is the state space; $\{G\}g_1, g_2, \dots, g_{|G|}$ is the set of agents; $A = \bigcup_{g \in G} A^g$ is the action set, a disjoint union of agent specific action spaces; $O = \bigcup_{g \in G} O^g$ is the observation set, a disjoint union of agent specific observation spaces; $t \in \Sigma(S \times A \rightarrow S)$ is the transition function and is the union function $t = \bigcup_{g \in G} t^g$, where for each agent g , the agent specific transition function $t^g \in \Sigma((S \times A^g) \rightarrow S)$ defines the effect of each agent's actions on the system state; $\omega \in \Sigma(S \rightarrow O)$ is the observation function and is the union function $\omega = \bigcup_{g \in G} \omega^g$, where $\omega^g \in \Sigma(S \rightarrow O^g)$ is used to generate an observation for agent g ; $\{F\}f^1, f^2, \dots, f^N$ is the set of measure functions; $i \in \Sigma(\emptyset \rightarrow S)$ is the initialisation function as in the FASP; $u \in \Sigma(S \rightarrow G)$ is the turn taking function; and $\Pi = \times_{g \in G} \Pi^g$ is the combined policy space as in the SMAFASP.

A formal definition of the union map and union function used here, can be found in [8], for simplicity these objects can be thought of as a collection of independent stochastic maps or functions.

As with the FASP and SMAFASP, if we consider only reactive policies, i.e. $\Pi^g = \Sigma(O^g \rightarrow A^g)$ for each g , it is possible to determine the probability of any state-to-state transition as a function of the joint policy, and hence write full state transition function.

Definition 7 (AMAFASP Full State Transition) An AMAFASP, M , with $\Pi^g = \Sigma(O^g \rightarrow A^g)$ for each g , has associated with it a full state transition function τ_M , given by,

$$\tau_M^\pi(s' | s) = \sum_{g \in G} \sum_{o^g \in O^g} \sum_{a^g \in A^g} u(g | s) \omega^g(o^g | s) \pi^g(a^g | o^g) t^g(s' | s, a^g)$$

To our knowledge, there are no similar frameworks which model asynchronous agent actions within stochastic state dependent environments. This may be because without state encapsulation it would not be at all as straightforward. A discussion of the potential benefits of turn-taking appears in Appendix A. From this point on, the umbrella term FASP covers FASPs, SMAFASPs and AMAFASPs.

The multi-agent FASPs defined above allow for the full range of cooperation or competition between agents, dependent on our interpretation of the measure signals. This includes general-sum games, as well as allowing hard and soft requirements to be combined separately for each agent, or applied to groups. Our paper focuses on problems where each agent, g , is associated with a single measure signal, f_n^g at each time-step n . Without further loss of generality, these measure signals are treated as rewards, $r_n^g = f_n^g$, and each agent g is assumed to prefer higher values for r_n^g over lower ones⁴.

⁴ It would be simple to consider cost signals rather than rewards by inverting the signs

For readability we present the general-sum case first, and incrementally simplify.

The general-sum scenario considers agents that are following unrelated agendas, and hence rewards are independently generated.

Definition 8 (General-Sum Scenario) A FASP is general-sum, if for each agent g there is a measure function $f^g \in \Phi(S \rightarrow \mathbb{R})$, and at each time-step n with the system in state s_n , g 's reward signal $r_n^g = f_n^g$, where each f_n^g is an outcome of $f^g(s_n)$, for all g . An agent's measure function is sometimes also called its reward function, in this scenario.

Another popular scenario, especially when modelling competitive games, is the zero-sum case, where the net reward across all agents at each time-step is zero. Here, we generate a reward for all agents and then subtract the average.

Definition 9 (Zero-Sum Scenario) A FASP is zero-sum, if for each agent g there is a measure function $f^g \in \Phi(S \rightarrow \mathbb{R})$, and at each time-step n with the system in state s_n , g 's reward signal $r_n^g = f_n^g - \bar{f}_n$, where each f_n^g is an outcome of $f^g(s_n)$ and $\bar{f}_n = \sum_h f_n^h / |G|$.

With deterministic rewards, the zero-sum case can be achieved with one less measure than there are agents, the final agent's measure is determined by the constraint that rewards sum to 1 (see [4]). For probabilistic measures with more than two agents, there are subtle effects on the distribution of rewards, so to avoid this we generate rewards independently.

If agents are grouped together, and within these groups always rewarded identically, then the groups are referred to as teams, and the scenario is called a team scenario.

Definition 10 (Team Scenario) A FASP is in a team scenario, if the set of agents G is partitioned into some set of sets $\{G_j\}$, so $G = \bigcup_j G_j$, and for each j , there is a team measure function f^j . At some time-step n in state s_n , each j 's team reward $r_n^j = f_n^j$, where f_n^j is an outcome of $f^j(s_n)$, and for all $g \in G_j$, $r_n^g = r_n^j$.

The team scenario above is general-sum, but can be adapted to a zero-sum team scenario in the obvious way. Other scenario's are possible, but we restrict ourselves to these. It might be worth noting that the team scenario with one team (modelling fully cooperative agents) and the two-team zero-sum scenario, are those most often examined in the associated multi-agent literature, most likely because they can be achieved with a single measure function and are thus relatively similar to the single agent POMDP, see [1, 9, 10, 11, 15, 16, 18, 23, 24, 25].

4 Examples

This section introduces the soccer example originally proposed in [16], and revisited in [2, 3, 5, 20, 24]. We illustrate both the transparency of the modelling mechanisms and ultimately the descriptive power this gives us in the context of problems which attempt to recreate some properties of a real system. The example is first formulated below as it appears in Littman's paper [16], and then recreated in two different ways.

Formulation 1 (Littman's adversarial MDP soccer) An early model of MDP style multi-agent learning and referred to as soccer, the game is played on a 4×5 board of squares, with two agents (one of whom is holding the ball) and is zero-sum, see Fig. 1. Each

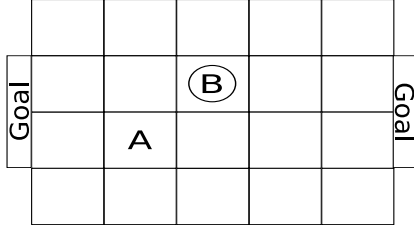


Figure 1. The MDP adversarial soccer example from [16].

agent is located at some grid reference, and chooses to move in one of the four cardinal points of the compass (N, S, E and W) or the H(ol)d action at every time-step. Two agents cannot occupy the same square. The state space, S_{L1} , is of size $20 \times 19 = 380$, and the joint action space, A_{L1} is a cartesian product of the two agents action spaces, $A_{L1}^A \times A_{L1}^B$, and is of size $5 \times 5 = 25$. A game starts with agents in a random position in their own halves. The outcome for some joint action is generated by determining the outcome of each agent's action separately, in a random order, and is deterministic otherwise. An agent moves when unobstructed and does not when obstructed. If the agent with the ball tries to move into a square occupied by the other agent, then the ball changes hands. If the agent with the ball moves into the goal, then the game is restarted and the scoring agent gains a reward of +1 (the opposing agent getting -1).

Therefore, other than the random turn ordering the game mechanics are deterministic.

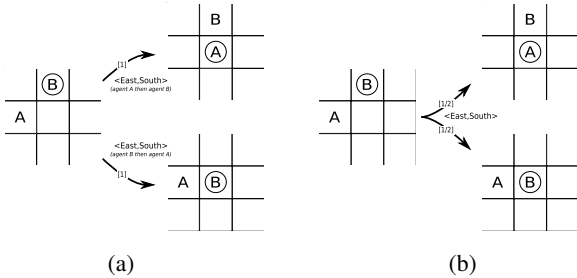


Figure 2. If agent A chooses to go East and agent B chooses to go South when diagonally adjacent as shown, the outcome will be non-deterministic depending on which agent's move is calculated first. In the original Littman version this was achieved by resolving agent specific actions in a random order, fig. (a). In the SMAFASP version this is translated into a flat transition function with probabilities on arcs (square brackets), fig. (b).

The Littman soccer game can be recreated as a SMAFASP, with very little work. We add a couple more states for the reward function, add a trivial observation function, and flatten the turn-taking into the transition function.

Formulation 2 (The SMAFASP soccer formulation) The SMAFASP formulation of the soccer game, is formed as follows: the state space $S_{L2} = S_{L1} + s_A^* + s_B^*$, where s_g^* is the state immediately after g scores a goal; the action space $A_{L2} = A_{L1}$; the observation space $O_{L2} = S_{L2}$; the observation function is the identity mapping; and the measure/reward function for agent A, $f^A \in \Phi(S_{L2} \rightarrow \mathbb{R})$ is as follows,

$$f^A(s_A^*) = 1, \quad f^A(s_B^*) = -1, \quad \text{and } f^A(s) = 0 \text{ for all other } s \in S.$$

Agent B's reward function is simply the inverse, i.e. $f^B(s) = -f^A(s)$, for all s .

To define the transition function we need first to imagine that Littman's set of transitions were written as agent specific transition functions, $t^g L1 \in \Sigma(S_{L1} \times A_{L1} \rightarrow S_{L1})$ for each agent g — although this is not explicitly possible within the framework he uses, his description suggests he did indeed do this. The new transition function, $t_{L2} \in \Sigma(S_{L2} \times A_{L2} \rightarrow S_{L2})$, would then be defined, for all $s_n, s_{n+1} \in S_{L1}$, $a_n^A \in A_{L2}^A$, $a_n^B \in A_{L2}^B$, in the following way,

$$t_{L2}(s_{n+1}|s_n, (a_n^A, a_n^B)) = \frac{1}{2} \cdot \sum_{s \in S_{L1}} \left(\begin{array}{c} t_{L1}^A(s|s_n, a_n^A) \cdot t_{L1}^B(s_{n+1}|s, a_n^B) \\ + t_{L1}^B(s|s_n, a_n^B) \cdot t_{L1}^A(s_{n+1}|s, a_n^A) \end{array} \right).$$

The transition probabilities involving the two new states, s_A^* and s_B^* , would be handled in the expected way.

The turn-taking is absorbed so that the random order of actions within a turn is implicit within the probabilities of the transition function, see Fig. 2(b), rather than as before being a product of the implicit ordering of agent actions, as in Fig. 2(a).

It is possible to reconstruct Littman's game in a more flexible way. To see how, it is instructive to first examine what Littman's motives may have been in constructing this problem, which may require some supposition on our part. Littman's example is of particular interest to the multi-agent community, in that there is no independently optimal policy for either agent; instead each policy's value is dependent on the opponent's policy — therefore each agent is seeking a policy referred to as the *best response* to the other agent's policy. Each agent is further limited by Littman's random order mechanism, see Fig. 2(a), which means that while one agent is each turn choosing an action based on current state information, in effect the second agent to act is basing its action choice on state information that is one time-step off current; and because this ordering is random, neither agent can really rely on the current state information. Littman doesn't have much control over this turn-taking, and as can be seen from the SMAFASP formulation, the properties of this turn-taking choice can be incorporated into the transition function probabilities (see Fig. 2 (b)). Different properties would lead to different probabilities, and would constitute a slightly different system with possibly different solutions.

However, consider for example, that an agent is close to their own goal defending an attack. Its behaviour depends to some degree on where it expects to see its attacker next: the defender may wish to wait at one of these positions to *ambush* the other agent. The range of these positions is dependent on how many turns the attacker might take between these observations, which is in turn dependent on the turn-taking built into the system. For ease of reading, we introduce an intermediate AMAFASP formulation. The individual agent action spaces are as in the SMAFASP, as is the observation space, but the new state information is enriched with the positional states of the previous time-step, which in turn can be used to generate the observations for agents.

Formulation 3 (The first AMAFASP soccer formulation) This AMAFASP formulation of the soccer game, M_{L3} , is formed as follows: the new state space $S_{L3} = S_{L2} \times S_{L2}$, so a new state at some time-step n , is given by the tuple (s_n, s_{n-1}) , where $s_{n-1}, s_n \in S_{L2}$ and records the current and most recent positional states; there are two action space, one for each agent, $A_{L3}^A = A_{L2}^A$ and $A_{L3}^B = A_{L2}^B$; and two identical agent specific observation

spaces, $O_{L3}^A = O_{L3}^B = O_{L2}$; the new agent specific transition functions, $t_{L3}^g \in \Sigma(S_{L3} \times A_{L3}^g \rightarrow S_{L3})$, are defined, for all $s_{n-1}, s_n, s'_n, s_{n+1} \in S_{L2}$, $a_n^g \in A_{L3}^g$, in the following way:

$$t_{L3}^g((s_{n+1}, s'_n)|(s_n, s_{n-1}), a_n^g) = \begin{cases} t_{L1}^g(s_{n+1}|s_n, a_n^g) & \text{iff } s'_n = s_n, \\ 0 & \text{otherwise.} \end{cases}$$

where t_{L1}^g represents agent g 's deterministic action effects in Littman's example, as in Formulation 2. The goal states, s_A^* and s_B^* , are dealt with as expected.

Recalling that $O_{L3} = S_{L2}$, the observation function, $\omega_{L3}^g \in \Sigma(S_{L3} \rightarrow O_{L3})$, is generated, for all $(s_{n-1}, s_n) \in S_{L3}$, $o_n \in O_{L3}^g$, and $g \in \{A, B\}$, in the following way,

$$\omega_{L3}^g(o_n|(s_n, s_{n-1})) = \begin{cases} 1 & \text{iff } s_{n-1} = o_n, \\ 0 & \text{otherwise.} \end{cases}$$

The reward function is straightforward and left to the reader.

Finally, we construct the turn-taking function $u_{L3} \in \Sigma(S_{L3} \rightarrow \{A, B\})$, which simply generates either agent in an unbiased way at each time-step. The turn taking function is defined, for all $(s_n, s_{n-1}) \in S_{L3}$, as

$$u_{L3}(A|(s_n, s_{n-1})) = u_{L3}(B|(s_n, s_{n-1})) = 1/2.$$

This does not fully replicate the Littman example, but satisfies the formulation in spirit in that agents are acting on potentially stale positional information, as well as dealing with an unpredictable opponent. In one sense, it better models hardware robots playing football, since *all* agents observe slightly out of date positional information, rather than a mix of some and not others. Both this and the Littman example do, however, share the distinction between turn ordering and game dynamics typified by Fig. 2 (a), what is more, this is now explicitly modelled by the turn-taking function.

To fully recreate the mix of stale and fresh observations seen in Littman's example along with the constrained turn-taking, we need for the state to include turn relevant information. This can be done with a tri-bit of information included with the other state information, to differentiate between; the start of a Littman time-step, when either agent could act next; when agent A has just acted in this time-step – and it must be B next; and vice versa when A must act next; we shall label these situations with l_0 , l_B and l_A respectively. This has the knock on effect that in l_0 labelled states the observation function is as Formulation 2; in l_A and l_B labelled states the stale observation is used – as in Formulation 3. Otherwise Formulation 4 is very much like formulation Formulation 3.

Formulation 4 (The second AMAFASP soccer formulation) This AMAFASP formulation of the soccer game, M_{L4} , is formed as follows: there is a set of turn labels, $L = \{l_0, l_A, l_B\}$; the state space is a three way Cartesian product, $S_{L4} = S_{L2} \times S_{L2} \times L$, where the parts can be thought of as current-positional-state, previous-positional-state and turn-label respectively; the action spaces and observation spaces are as before, i.e. $A_{L4}^g = A_{L3}^g$, $O_{L4}^g = O_{L3}^g$, for each agent g ; the transition and reward functions are straightforward and are omitted for brevity; the observation and turn taking functions are defined, for all $(s_n, s_{n-1}, l_n) \in S_{L4}$, $o_n \in O_{L4}^g$ and all agents g , in the following way,

$$\omega_{L4}^g(o_n|(s_n, s_{n-1}, l_n)) = \begin{cases} 1 & \text{if } s_n = o_n \text{ and } l_n = l_0, \\ 1 & \text{if } s_{n-1} = o_n \text{ and } l_n = u_g, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$u_{L4}(g_n|(s_n, s_{n-1}, l_n)) = \begin{cases} \frac{1}{2} & \text{if } g_n = g \text{ and } l_n = l_0, \\ 1 & \text{if } g_n = g \text{ and } l_n = u_g, \\ 0 & \text{otherwise} \end{cases}$$

The above formulation recreates Littman's example precisely, and instead of the opaque turn-taking mechanism hidden in the textual description of the problem, it is transparently and explicitly modelled as part of the turn-taking function.

So the Littman example can be recreated as a SMAFASP or AMAFASP, but more interestingly both AMAFASP formulations, 3 and 4, can be tuned or extended to yield new, equally valid, formulations. What is more, the intuitive construction means that these choices can be interpreted more easily.

Consider Formulation 3; the turn-taking function u can be defined to give different turn-taking probabilities at different states. For instance, if an agent is next to its own goal, we could increase its probability of acting (over the other agent being chosen) to reflect a defender behaving more fiercely when a loss is anticipated. Alternatively, if an agent's position has not changed since the last round, but the other's has then the first agent could be more likely to act (possible as two steps of positional data are stored); giving an advantage to the H(old) action, but otherwise encouraging a loose alternating agent mechanism.

While Formulation 4 recreates the Littman example, it again can be adjusted to allow different choices to the turn taking mechanism; in particular it is now possible to enforce strictly alternating agents. This would be done by flipping from state label l_A to l_B or vice versa, at each step transition, and otherwise keeping things very much as before. It is important to note that many specific models built in this way, can be recreated by implicit encoding of probabilities within existing frameworks, but it is difficult to see how the experimenter would interpret the group of models as being members of a family of related systems.

4.1 Flexible Behaviour Regulation

If we increase the number of players in our game, we can consider increasing the number of measure functions for a finer degree of control over desired behaviour. With just 2 agents competing in the Littman problem, it is difficult to see how to interpret any extra signals, and adding agents will increase the state space and hence the policy size radically. So, before we address this aspect of the FASP formalisms, it is useful to examine a more recent derivative soccer game, namely Peshkin et al.'s partially observable identical payoff stochastic game (POIPSG) version [20], which is more amenable to scaling up.

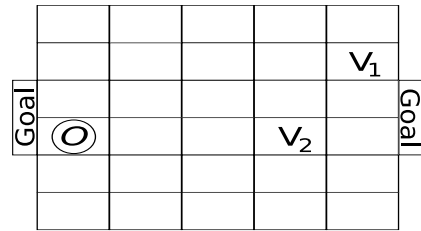


Figure 3. The POIPSG cooperative soccer example from [20].

Peshkin et al.’s example is illustrated in Fig. 3. There are two teammates, V_1 and V_2 , and an opponent O , each agent has partial observability and can only see if the 4 horizontally and vertically adjacent squares are occupied, or not. Also, players V_1 and V_2 have an extra *pass* action when in possession of the ball. Otherwise the game is very much like Littman’s with joint actions being resolved for individual agents in some random order at each time-step. Contrary to expectations, the opponent is not modelled as a learning agent and does not receive a reward; instead the two teammates share a reward and learn to optimise team behaviour versus a static opponent policy; for more details see [20].

As with its progenitor, the Peshkin example could be simply reworked as a SMAFASP in much the same way as in Formulation 2. Recreating it as an AMAFASP is also reasonably straightforward; as before, the trick of including the previous step’s positional state in an AMAFASP state representation, allows us to generate *stale* observations – which are now also partial. As this is relatively similar to the Littman adaptations, the details are omitted. The focus here instead, is to show that, in the context of Peshkin’s example, a zero- or general-sum adaption with more agents, could have some utility. Firstly, agent O could receive the opposite reward as the shared reward of V_1 and V_2 , this could be done without introducing another measure function, merely reinterpreting Peshkin’s reward function; now, the opponent could learn to optimise against the cooperative team. More interestingly, the players V_1 and V_2 could be encouraged to learn different roles by rewarding V_1 (say) more when the team scores a goal and penalising V_2 more when the team concedes one, all this requires is a measure function for each agent and a zero-sum correction. Further, we can add a second opponent (giving say O_1 and O_2), either rewarding them equally or encouraging different roles as with V_1 and V_2 . In this way we could explore the value of different reward structures by competing the teams. If more agents were added, even up to 11 players a side, and a much larger grid, the AMAFASP framework supports a much richer landscape of rewards and penalties, which can encourage individual roles within the team, while still differentiating between good and bad team collaborations.

5 Discussion

In this paper, we have examined a new family of frameworks — the FASP, geared towards modelling stochastic processes, for one or many agents. We focus, on the multi-agent aspects of this family, and show how a variety of different game scenario’s can be explored within this context. Further, we have highlighted the difference between synchronous and asynchronous actions within the multi-agent paradigm, and shown how these choices are explicit within the FASP frameworks. As a package, this delivers what we consider to be a much broader tool-set for regulating behaviour within cooperative, non-cooperative and competitive problems. This is in sharp contrast to the more traditional pre-programmed expert systems approaches, that attempt to prescribe agent intentions and interactions.

The overarching motivation for our approach is to provide the modeller with *transparent mechanisms*, roughly this means that all the pertinent mechanisms are defined explicitly within the model. There are two such mechanisms that are visited repeatedly in this paper, these are multiple measures and turn-taking, but they are not the only such mechanisms. In fact, the FASP borrows from the MDP, and POMDP, frameworks a few (arguably) transparent mechanisms. The transition function is a good example, and the observation and reward functions as they are defined in the POMDP have a degree of transparency; we argue that strict *state encapsulation* improves upon

this though. The general agent-environment relationship is the same as the POMDP, meaning existing analytic and learning tools can still be applied where appropriate⁵. Moreover, techniques for shaping and incremental learning developed for related single agent frameworks [14, 22] and multi-agent frameworks [7] can also be applied without radical changes.

Other ideas for good transparent mechanisms exist in the literature, and the FASP family would benefit by incorporating the best of them. For instance, modelling extraneous actions/events can reduce the required size of a model, by allowing us to approximate portions of an overall system, without trying to explicitly recreate the entire system. Imagine a FASP with a external event function $X \in \Sigma(S \rightarrow S)$, which specifies how an open system might change from time-step to time-step, a similar extraneous event function can be found in [21]. This transition might occur between the observation and action outcomes, incorporating *staleness* into the observations as with our examples in Section 4. More interestingly, this function could be part of an AMAFASP, and be treated analogously to the action of a *null agent*; this would allow the turn-taking function to manage how rarely/often an extraneous events occurred. Another candidate for transparent mechanism, can be found in [4], where they simulate a broken actuator by separating intended action from actual action with what amounts to a stochastic map between the two. We would encourage modellers to incorporate such mechanisms as needed.

There are other clear directions for future work, the tools outlined in this paper enable a variety of multi-agent problems, with choice of measures for evaluation, but it leaves out how these measures might be used to solve or learn desirable solutions. The terms general- and zero-sum are not used accidentally, the similarities with traditional game theory are obvious, but the differences are more subtle. The extensive form game in the game theoretic sense, as described in [17], enforces that a game has a beginning and an end, only rewards at the end of a game run, players (agents) do not forget any steps in their game; and seeks to address the behaviour of intelligent players who have full access to the game’s properties. While this can be defended as appropriate for human players, our paradigm allows for an ongoing game of synthetic agents with potentially highly restricted memory capabilities and zero prior knowledge of the game to be played; and rewards are calculated at every step. In fact, if we were to combine the AMAFASP with extraneous actions, it would constitute a generalisation of the extensive form game, as described in [17], but we omit the proof here.

It is sufficient to say that the FASP family demands a different solution approach than the extensive form game, and certainly a different approach than the easily understood reward-maximisation/cost-minimisation required by single-agent and cooperative MDP style problems. Some preliminary research has been done with respect to such systems, [4, 11, 26], we envisage this as being an active area of research in the next few years, and hope that the FASP tool-set facilitates that study.

REFERENCES

- [1] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman, ‘The complexity of decentralized control of markov decision processes’, in *Proceedings of the 16th Annual Conference on Uncertainty in Artificial*

⁵ Tools for evaluating the expected long term reward in a POMDPs, can be used without change within a FASP except that multiple measure signals could be evaluated simultaneously. Maximisation procedures are still possible too, but care needs to be taken with what is maximised and where multiple agents are maximising different measures independently [4, 26].

- Intelligence (UAI-00)*, pp. 32–37, San Francisco, CA, (2000). Morgan Kaufmann.
- [2] Reinaldo A. C. Bianchi, Carlos H. C. Ribeiro, and Anna H. Reali Costa, ‘Heuristic selection of actions in multiagent reinforcement learning’, in *IJCAI*, ed., Manuela M. Veloso, pp. 690–695, (2007).
- [3] Michael Bowling, Rune Jensen, and Manuela Veloso, ‘A formalization of equilibria for multiagent planning’, in *Proceedings of the AAAI-2002 Workshop on Multiagent Planning*, (August 2002).
- [4] Michael Bowling and Manuela Veloso, ‘Existence of Multiagent Equilibria with Limited Agents’, *Journal of Artificial Intelligence Research* 22, (2004). Submitted in October.
- [5] Michael H. Bowling, Rune M. Jensen, and Manuela M. Veloso, ‘Multiagent planning in the presence of multiple goals’, in *Planning in Intelligent Systems: Aspects, Motivations and Methods*, John Wiley and Sons, Inc., (2005).
- [6] Michael H. Bowling and Manuela M. Veloso, ‘Simultaneous adversarial multi-robot learning’, in *IJCAI*, eds., Georg Gottlob and Toby Walsh, pp. 699–704. Morgan Kaufmann, (2003).
- [7] Olivier Buffet, Alain Dutech, and François Charpillat, ‘Shaping multi-agent systems with gradient reinforcement learning’, *Autonomous Agents and Multi-Agent Systems*, 15(2), 197–220, (2007).
- [8] Luke Dickens, Krycia Broda, and Alessandra Russo, ‘Transparent Modelling of Finite Stochastic Processes for Multiple Agents’, Technical Report 2008/2, Imperial College London, (January 2008).
- [9] Alain Dutech, Olivier Buffet, and François Charpillat, ‘Multi-agent systems by incremental gradient reinforcement learning’, in *IJCAI*, pp. 833–838, (2001).
- [10] Jerzy Filar and Koos Vrieze, *Competitive Markov decision processes*, Springer-Verlag New York, Inc., New York, NY, USA, 1996.
- [11] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings, 2004.
- [12] Amy Greenwald and Keith Hall, ‘Correlated q-learning’, in *AAAI Spring Symposium Workshop on Collaborative Learning Agents*, (2002).
- [13] Junling Hu and Michael P. Wellman, ‘Multiagent reinforcement learning: theoretical framework and an algorithm’, in *Proc. 15th International Conf. on Machine Learning*, pp. 242–250. Morgan Kaufmann, San Francisco, CA, (1998).
- [14] Adam Daniel Laud, *Theory and Application of Reward Shaping in Reinforcement Learning*, Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2004. Advisor: Gerald DeJong.
- [15] Martin Lauer and Martin Riedmiller, ‘An algorithm for distributed reinforcement learning in cooperative multi-agent systems’, in *Proc. 17th International Conf. on Machine Learning*, pp. 535–542. Morgan Kaufmann, San Francisco, CA, (2000).
- [16] Michael L. Littman, ‘Markov games as a framework for multi-agent reinforcement learning’, in *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pp. 157–163, New Brunswick, NJ, (1994). Morgan Kaufmann.
- [17] Roger B. Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, September 1997.
- [18] Fans Oliehoek and Arnoud Visser, ‘A Hierarchical Model for Decentralized Fighting of Large Scale Urban Fires’, in *Proceedings of the Fifth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, eds., P. Stone and G. Weiss, Hakodate, Japan, (May 2006).
- [19] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw Hill, 3rd edn., 1991.
- [20] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie P. Kaelbling, ‘Learning to cooperate via policy search’, in *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 307–314, San Francisco, CA, (2000). Morgan Kaufmann.
- [21] Bharaneedharan Rathnasabapathy and Piotr Gmytrasiewicz. Formalizing multi-agent pomdp’s in the context of network routing.
- [22] Mark B. Ring, ‘Child: A first step towards continual learning’, *Machine Learning*, 28, 77–104, (May 1997).
- [23] Yoav Shoham, Rob Powers, and Trond Grenager, ‘If multi-agent learning is the answer, what is the question?’, *Artif. Intell.*, 171(7), 365–377, (2007).
- [24] William T. B. Uther and Manuela M. Veloso, ‘Adversarial reinforcement learning.’, Technical report, Computer Science Department, Carnegie Mellon University, (April 1997).
- [25] Erfu Yang and Dongbing Gu, ‘Multiagent reinforcement learning for multi-robot systems: A survey’, Technical report, University of Essex,

(2003).

- [26] Martin Zinkevich, Amy Greenwald, and Michael Littman, ‘Cyclic Equilibria in Markov Games’, in *Advances in Neural Information Processing Systems 18*, 1641–1648, MIT Press, Cambridge, MA, (2005).

A The Benefits of Turn-Taking

Modelling with turn-taking is not only more natural in some cases, in certain cases it allows for a more concise representation of a system. Consider a system with 2 agents, A and B , and 9 positional states (in a 3×3 grid), where agents cannot occupy the same location (so there are $9 \times 8 = 72$ states), and movement actions are in the 4 cardinal directions for each agent.

Let us first consider how this might be formulated without explicit turn taking. Imagine, this is our prior-state,

$$s_1 = \begin{array}{|c|c|c|} \hline & B & \\ \hline A & & \\ \hline & & \\ \hline \end{array}$$

Imagine also that agent A chooses action E(ast), agent B chooses action S(outh) — making joint action $\langle E, S \rangle$, and that these actions will be resolved in a random order. This results in one of three post-action states, with the following probabilities;

$$\Pr \left(s' = \begin{array}{|c|c|c|} \hline & B & \\ \hline & A & \\ \hline & & \\ \hline \end{array} \mid s = \begin{array}{|c|c|c|} \hline & B & \\ \hline A & & \\ \hline & & \\ \hline \end{array}, a = \langle E, S \rangle \right) = \alpha_1,$$

$$\Pr \left(s' = \begin{array}{|c|c|c|} \hline & & \\ \hline A & B & \\ \hline & & \\ \hline \end{array} \mid s = \begin{array}{|c|c|c|} \hline & B & \\ \hline A & & \\ \hline & & \\ \hline \end{array}, a = \langle E, S \rangle \right) = \beta_1,$$

and

$$\Pr \left(s' = \begin{array}{|c|c|c|} \hline & B & \\ \hline A & & \\ \hline & & \\ \hline \end{array} \mid s = \begin{array}{|c|c|c|} \hline & B & \\ \hline A & & \\ \hline & & \\ \hline \end{array}, a = \langle E, S \rangle \right) = \gamma_1,$$

where $\alpha_1 + \beta_1 + \gamma_1 = 1$.

Let’s assume that they are transparently modelled by agents A and B , whose actions fail with the following probabilities; $\Pr(A\text{’s action fails}) = p$ and $\Pr(B\text{’s action fails}) = q$, acting in some order, where $\Pr(A \text{ is first}) = r$, and such that agents cannot move into an already occupied square. We can use these values to determine the *flattened* probabilities as $\alpha_1 = (1-p)(r + q(1-r))$, $\beta_1 = pq$, and $\gamma_1 = (1-q)(pr + (1-r))$. α_1 , β_1 and γ_1 are not independent, they represent only 2 independent variables, but are fed into by p , q and r , which are independent. It may seem that α_1 , β_1 and γ_1 model the situation more concisely, since any one combination of α_1 , β_1 and γ_1 could correspond to many different choices for p , q and r . However, this isn’t the whole story. Consider instead the prior-state s_2 , where

$$s_2 = \begin{array}{|c|c|c|} \hline & B & \\ \hline A & & \\ \hline & & \\ \hline \end{array},$$

and the combined action $\langle S, E \rangle$. Treated naively the possible outcomes might all be specified with separate probabilities (say α_2 , β_2 and γ_2) as before. However, the modeller can, with transparent mechanisms, choose to exploit underlying symmetries of the system. If, as is quite natural, the turn ordering probabilities are independent of position, then there need be no extra parameters specified for this situation, the original p , q and r are already sufficient to define the transition probabilities here too.