Leveraging FOND Planning Technology to Solve Multi-Agent Planning Problems

Christian Muise, Paolo Felli, Tim Miller, Adrian R. Pearce, Liz Sonenberg
Department of Computing and Information Systems, University of Melbourne
{christian.muise, paolo.felli, tmiller, adrianrp, l.sonenberg}@unimelb.edu.au

Abstract

Single-agent planning in a multi-agent environment is challenging because the actions of other agents can affect our ability to achieve a goal. From a given agent's perspective, actions of others can be viewed as non-deterministic outcomes of that agent's actions. While simple conceptually, this interpretation of planning in a multi-agent environment *as* non-deterministic planning is challenging due to the non-determinism resulting from others' actions, and because it is not clear how to compactly model the possible actions of others in the environment. In this paper, we cast the problem of planning in a multi-agent environment as one of Fully-Observable Non-Deterministic (FOND) planning. We extend a non-deterministic planner to plan in a multi-agent setting, given the goals and possible actions of other agents. We use the other agents' goals to reduce their set of possible actions to a set of *plausible* actions, allowing non-deterministic planning technology to solve a new class of planning problems in first-person multi-agent environments. We demonstrate our approach on new and existing multi-agent benchmarks, demonstrating that modelling the other agents' goals reduces complexity.

1 Introduction

Synthesising a plan for an agent operating in a multi-agent domain is a challenging and increasingly important problem (Bolander and Herzig 2014). When an agent acts in an environment with other agents, it must be aware of the possible actions of others and how they might affect the continued feasibility of achieving a goal. In this work, we focus on a version of the multi-agent planning problem where the goals and possible actions of all agents are known, and the state of the world is fully observable. These restrictions capture a wide class of multi-agent problems, including many collaborative and competitive games. In Fully Observable Non-Deterministic (FOND) planning, an agent must synthesize a policy to achieve a goal with some guarantee. The actions in a FOND problem are *non-deterministic*; that is, any one of a number of outcomes may occur when the agent executes the action. The world is fully observable, so the agent knows the outcome of the action after execution.

In this paper, we cast the first-person view of a multi-agent planning (MAP) problem as a FOND planning problem, taking advantage of recent advances in that field. The key to our approach is that the choice of action by other agents can be viewed as a non-deterministic effect of our own action. Any valid sequence of actions from other agents can potentially result in a different state of the world, which we encode as the non-deterministic outcomes of our own action. As an example, consider the partially played game of Tic-Tac-Toe in Figure 1 where we are playing **X** and it is our turn. Playing the right column / middle row can subsequently lead to four possible states where it is our turn again: these are the non-deterministic outcomes of our move.

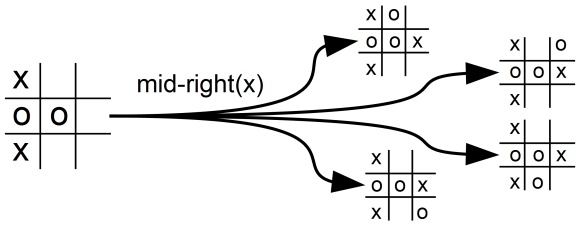


Figure 1: Tic-Tac-Toe example from **X**'s perspective

A number of recent advances in FOND planning have improved the scalability of the solvers significantly (Fu et al. 2011; Alford et al. 2014; Muise, McIlraith, and Beck 2012; Ramirez and Sardina 2014). We wish to take advantage of these improvements for solving MAP problems, but achieving this is not simply a matter of encoding the problem in a form that FOND planners can solve. If we naively encode the problem as input to a FOND planner, then the encoding must enable the planner to take account of the context in which other agents execute their actions: in particular to respect that for any particular state of the world, there may be only a small subset of actions that are applicable. As a result, to capture the behaviour of other agents as non-deterministic action effects must involve either: (1) enumerating the actions in some way to find those that are applicable; or (2) fully specifying the actions and effects to account for every situation that the other agents may encounter. Both options result in a combinatorial explosion on the size of the encoded FOND problem. Our approach takes an alternative direction and modifies an existing solver to support consideration of applicable actions, allowing us to keep the encoding of the MAP problem compact and naturally expressed.

Considering all of the applicable actions for other agents can result in a prohibitively large amount of non-

determinism. To mitigate this, we consider restricting the possible actions of another agent to a set of *plausible* actions, which are those that lead to states with the highest heuristic value for the agent given their goal. We do not consider how the goals of others are known, but in many scenarios the goal is known apriori or can be inferred; e.g., using goal recognition (Ramírez and Geffner 2010). This gives us a general means to focus on the worst case scenario, if we are considering agents with conflicting or opposing goals, or to focus on the expected scenarios if we are working towards the same goal as other agents (e.g., on the same team). Note that we do not presume authority over our teammates actions – rather, we wish to plan for what they would plausibly do given that they share a goal similar or the same as our own.

To realize our approach, we modified the state-of-the-art FOND planner PRP (Muise, McIlraith, and Beck 2012). We evaluate the approach on three multi-agent domains adapted from existing problems. The evaluation addresses our strategies for restricting non-determinism, and demonstrates the capability to solve MAP problems with existing FOND planning technology. By using a FOND planner at the core of our approach, which subsequently uses a classical planner at its core, we take advantage of every new advancement in either FOND or classical planning. We also discuss the issues surrounding the interpretation of MAP as FOND.

Next we describe the background concepts and notation required for our approach. Following this, we describe our approach for solving MAP problems as FOND, and how to heuristically reduce the amount of non-determinism in the domain. Next, we provide details on an evaluation of our approach, and we conclude with a discussion of related work and future directions.

2 Background

2.1 (FOND) Planning Notation

We describe briefly here the requisite planning background and notation (cf. Ghallab et al. (2004) for a full treatment). A *Fully Observable Non-Deterministic* (FOND) planning problem P consists of a tuple $\langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$; $\mathcal{F}$ is a set of fluents, and we use $\mathcal{S}$ as the set of all possible states; $\mathcal{I} \subseteq \mathcal{F}$ is the initial state; $\mathcal{G} \subseteq \mathcal{F}$ characterizes the goal to be achieved; and $\mathcal{A}$ is the set of actions. An action $a \in \mathcal{A}$ is a tuple $\langle \text{Pre}_a, \text{Eff}_a \rangle$ where $\text{Pre}_a \subseteq \mathcal{F}$ is the precondition (i.e., the fluents that must hold for a to be executable) and Eff_a is a set of one or more outcomes. An action with only one outcome is *deterministic*; otherwise it is *non-deterministic*. Each $e \in \text{Eff}_a$ contains a set of positive and negative effects that update the state of the world, and we use $Prog(s, a, e)$ to denote the state reached when action a is executed with outcome e in state s. After executing an action, *exactly one* outcome is used to update the state of the world. The planning agent does not know which outcome in advance, and so must plan for every contingency.

Following Cimatti et al. (2003), we consider three types of solution to a FOND planning problem: *weak*, *strong*, and *strong cyclic*. The representation for all three is in the form of a policy P that maps the state of the world to an action: $P : \mathcal{S} \rightarrow \mathcal{A}$. We say that a state s' is *reachable* from s by the plan P if it is equal to s or if there exists some other state s'' reachable from s such that s' is a possible successor to the execution of $P(s'')$ in state s''.

P is a *weak plan* if there is some state that is reachable from $\mathcal{I}$ where $\mathcal{G}$ holds. P is a *strong cyclic plan* if for every state s that is reachable from $\mathcal{I}$, there is another state reachable from s where $\mathcal{G}$ holds. Finally, P is a *strong plan* if it is a strong cyclic plan and no state s reachable from the initial state is reachable from the possible successors of s (i.e., a reachable cycle is impossible). The distinction between strong and strong cyclic plans is, as the terms imply, the presence of cycles in the reachable state space of the plan P. Finally, the *all-outcomes determinization* of a FOND problem $\langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$ is the classical planning problem $\langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A}' \rangle$ where $\mathcal{A}'$ is defined as: $\mathcal{A}' = \{\langle \text{Pre}_a, [e] \rangle \mid a \in \mathcal{A} \text{ and } e \in \text{Eff}_a\}$

Solving the all-outcomes determinization is a technique used to compute weak plans, as any classical plan for the all-outcomes determinization represents a weak plan for the original FOND problem.

2.2 A First-person View of Multi-agent Planning

There are a variety of multi-agent planning (MAP) formalisms (e.g., see (Brafman and Domshlak 2008; Brenner 2003; Kovacs 2012)), however, we consider a first-person view of planning in the simplified setting where the world is fully known and is observable to all agents. Actions may be non-deterministic, but outcomes are known immediately by all agents in the domain. This setting is most related to the NAPL syntax for multi-agent planning introduced in (Jensen and Veloso 2000).

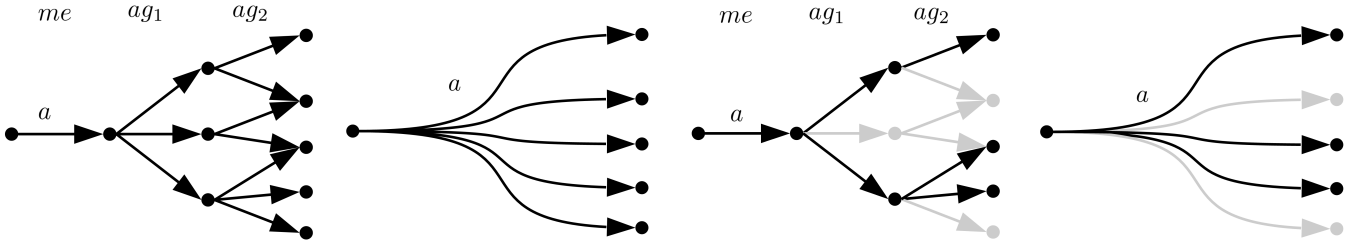
As input to the planner, we have a collection of agents, each of which has its own set of actions, $\mathcal{A}_i$, and possibly its own goal $\mathcal{G}_i \subseteq \mathcal{F}$. The planning agent's task in our MAP setting is to synthesize a policy that maps states to actions given the uncertainty of what other agents will do, and the analogies to FOND solutions are direct. Ideally the goal state of the planning agent should be guaranteed, but if this is not possible the agent should strive to achieve robust policies that achieve the goal with a high probability of success. While we do not compute optimal solutions, we do evaluate the policies based on this notion of solution quality.

3 Approach

First, we present our general approach for solving fully observable MAP problems using FOND planning, and discuss the issues that would arise if we instead used FOND as a black box. We then describe how to heuristically restrict the problem's non-determinism in cases where the goals of other agents in the domain are given.

3.1 MAP as FOND

Our method of solving MAP problems as FOND is straightforward on the surface, but powerful in practice: *we view the set of possible states that result from other agents conducting actions as non-deterministic outcomes to the planning agent's own choice of action*. Figure 2a shows how, in a setting with 3 agents, $\{me, ag_1, ag_2\}$, *me* choosing action a can



(a) Full MAP execution (b) Full FOND interpretation (c) Plausible MAP execution (d) Plausible FOND outcomes

Figure 2: Example execution for agents ag_1 and ag_2 after *me* executes action a. Subfigures (c) and (d) show plausible outcomes.

lead to 5 different states depending on the actions that ag_1 and ag_2 decide to execute on their turn. Figure 2b shows the conceptual interpretation of our decision to perform action a. Intuitively, we care about only the states that we may arrive in after a is executed, so we can therefore view the collective decisions of other agents as non-determinism in the world if we choose action a.

For simplicity, we assume agents execute in round-robin fashion according to the sequence $\overrightarrow{ag}$, although this is not strictly necessary to apply our approach.[1] The planning agent, labelled *me* and corresponding to agent ag_0 in $\overrightarrow{ag}$, acts first followed by every agent from $\overrightarrow{ag}$; and then the process repeats. Thus, for every action $a \in \mathcal{A}_i$ in agent i's action set, the precondition of a is augmented with a fluent stating that agent i is the current acting agent, and the effect is augmented with a proposition stating that agent i+1 mod $|\overrightarrow{ag}|$ is the next acting agent. Initially, *me* is the current actor. We use $App(s, ag)$ for $s \in \mathcal{S}$ and $ag \in \overrightarrow{ag}$ to signify the actions that are applicable by agent ag in state s. $App(s, ag)$ is empty if ag is not the current acting agent for state s. Figure 2a shows how one action a executed by *me* can be followed by three possible actions of ag_1 and then potentially seven different actions by ag_2 before agent *me* has a turn again.

Not all FOND planners are built using the same solving technique, but many approaches (e.g., NDP (Alford et al. 2014), FIP (Fu et al. 2011), and PRP (Muise, McIlraith, and Beck 2012)) build a policy by exploring the reachable state space and augmenting the policy with new plan fragments when encountering a previously unseen state. Adapted from the description of PRP (Muise, McIlraith, and Beck 2012), Algorithm 1 gives a high level view of computing a policy.

The GenPlanPairs($\langle \mathcal{F}, s, s_*, \mathcal{A} \rangle, P$) algorithm attempts to find a weak plan from s to the goal *assuming that we are omniponent* (i.e., we control the actions of others). It does this by simply merging actions from all agents into a single set and planning as normal using this set; that is $\mathcal{A} = \bigcup_{i \in \overrightarrow{ag}} \mathcal{A}_i$,. This is the result of using the all-outcomes determinization, which assumes that the world is deterministic and we can choose any outcome of a non-deterministic action.

The key distinction between Algorithm 1 and PRP's approach is that we generalize line 11. In this context, the original PRP algorithm defines GenerateSuccessors(s, a) to be: $\{Prog(s, a, e) \mid e \in \text{Eff}_a\}$.

[1] If joint actions are required, we need only to be able to enumerate the different possible action combinations given the state of the world and a single action choice for our agent.

Algorithm 1: Generate Strong Cyclic Plan

```
   Input: FOND planning task Π = ⟨F, I, G, A⟩
   Output: Partial policy P
1  Initialize policy P
2  while P changes do
3      Open = {I}; Seen = {};
4      while Open ≠ ∅ do
5          s = Open.pop();
6          if Satisfies(s, G) ∧ s ∉ Seen then
7              Seen.add(s);
8              if P(s) is undefined then
9                  GenPlanPairs(⟨F, s, G, A⟩, P);
10             if P(s) is defined then
11                 for s' ∈ GenerateSuccessors(s, P(s)) do
12                     Open.add(s');
13     ProcessDeadends();
14 return P;
```

Given the agent sequence $\overrightarrow{ag}$, GenerateSuccessors(s, a) enumerates the *applicable* options for each agent in turn, including the non-deterministic effects of their possible actions. Algorithm 2 outlines this procedure in detail.

Line 6 is critical because it determines the options to consider for another agent, and later in the paper we present alternatives to $App(s', ag)$ to improve the problem efficiency.

In addition to modifying GenerateSuccessors(s, a), we also account for the multi-agent setting. For example, during the computation of partial states for the policy and deadend detection, the variable representing the current acting agent is maintained in all cases. While not strictly necessary for soundness, it helps the planner to avoid focusing on unreachable parts of the state space.

Another modification was made to handle deadends: when PRP detects a deadend, it creates a *forbidden state-action pair* that avoids actions that have at least one outcome leading to the deadend. We modified this to cycle through the list of agents in reverse to prevent the planning agent from executing actions that could lead to deadends. For example, consider Figure 1. One omnipotent strategy may be to play in the top middle spot, as it brings us close to win-

Algorithm 2: GENERATESUCCESSORS(s, a)

Input: State s, action a
Output: Set of successor states S

```
1 S = {Prog(s, a, e) | e ∈ Eff_a} ;
2 for i = 1 ... |ag⃗| do
3     ag = ag⃗[i];
4     S' = ∅;
5     for s' ∈ S do
6         for a' ∈ App(s', ag) do
7             S' = S' ∪ {Prog(s', a', e) | e ∈ Eff_a'};
8     S = S';
9 return S;
```

ning with the top row. However, doing so means that O has a winning move at the middle right spot, which is a deadend for us. The modified version for the forbidden state-action procedure will create rules that forbid every move other than playing the one shown that blocks O from winning.

We now turn our attention to various considerations when solving MAP problems with FOND technology.

The value of doing nothing When planning in a multi-agent environment, it is important to consider whether or not to use *noop* actions. These actions simply change the current acting agent into the next agent in sequence without changing the state of the world. The advantage of allowing noop actions for the other agents is that the computed policies can be far more compact: if other agents cannot interfere with our plan, assigning noop actions to them will produce a compact policy. Another benefit of using noop actions is that for the acting agent ag in state s, the set $App(a, ag)$ will always be non-empty.

On the other hand, including a noop action can increase the applicable action space, causing the planner to work harder to compute a solution. This is evident in the Tic-Tac-Toe domain with the goal of drawing the game — solving the problem to completion without noop is roughly an order of magnitude faster than with. Ultimately, the decision to include noop actions should be made on a per-domain basis.

(Un)Fair non-determinism Typically, FOND planning assumes fairness (Cimatti et al. 2003): if an action is executed infinitely many times, every non-deterministic outcome will occur infinitely often. Agents are, of course, not always fair. By using a FOND planner, we are synthesizing policies under the assumption of fairness, although we evaluate the policies without this assumption, as discussed in Section 4.

While this assumption is not ideal, it does serve as a natural relaxation of the full multi-agent setting. Ultimately, it would be safer to produce strong plans rather than strong cyclic plans. There are encoding techniques that we do not discuss here that force all solutions to be strong plans, but it is crucial to observe that *many domains are inherently acyclic*. This means that a FOND planner, even with the assumption of fairness, will produce a strong plan. Examples include any game or setting where an agent can never return to the same state (e.g., Tic-Tac-Toe, board games, etc). In these settings, the fairness assumption does not play a role: an action will never occur infinitely often.

Winning -vs- not losing In settings such as Tic-Tac-Toe, where draws are possible, there is no distinction between losing by a wide margin and drawing a game: if we cannot win, then the state is a deadend. This falls under a generalization of goal achievement where we would like to satisfy a hierarchy of objectives: e.g., I would like to win, and if I cannot win I would like to draw.

Although the goal could be "any final state that is not a loss", as we did for one of the Tic-Tac-Toe problems,this falls short of the goal to create a policy that wins whenever possible. It is unclear how to evaluate the quality of a policy when multiple objectives are at play. For example, is a policy that wins more often but loses some of the time better than a policy that wins less often but never loses? We leave this question as part of future work.

Issues with FOND as a black box We considered a variety of approaches for encoding MAP problems directly to use FOND planners in an off-the-shelf manner. However, this leads to complications, making the approach unmanageable. Whether we use a monolithic action that captures all of the possibilities for an agent, or many individual actions that are all considered before the agent's turn is up (with only one being accepted), we would need heavy use of conditional effects. The only state-of-the-art FOND planner capable of solving problems with conditional effects (Muise, McIlraith, and Belle 2014) would be hampered by the range of conditions on the encoded actions — all savings due to leveraging state relevance would be lost, and as a result the size of the policies would grow exponentially.

In general, encoding the options for how other agents can act as part of the FOND problem itself comes with a variety of issues that range from prohibitively many actions to overly-restricted conditional effects. We sidestep these obstacles by modifying the FOND planner directly, as outlined in 3.3.

3.2 Reducing Non-determinism

Though we avoided the difficulties discussed above by modifying a FOND planner directly, non-determinism can still be unwieldy. If each agent has an average of k actions applicable in an arbitrary state of the world, the number of non-deterministic successors is on the order of $O(k^{|\vec{ag}|})$. Here, we consider restricting the reasoning to only a subset of the applicable actions, thus making k a low constant.

At Line 6 of Algorithm 2, rather than considering all actions in $App(s, ag)$, we use a *plausibility function* Γ, where $\Gamma(s, ag) \subseteq App(s, ag)$. Some examples include:

- $\Gamma_{full}(s, ag) = App(s, ag)$: The original set of all applicable actions for agent ag.
- $\Gamma_{rand_k}(s, ag) = \text{RANDOM}(\Gamma_{full}(s, ag), k)$: A random subset of (maximum) size k which is drawn from the set of applicable actions for agent ag.

Figures 2c and 2d show an example of using plausible action for agents ag_1 and ag_2, after agent *me* executes action a. Notice that from the non-deterministic perspective, the three plausible outcomes are entirely disassociated from the agents that lead us there. When using a random subset of the actions as the plausibility function, we will indeed reduce the amount of non-determinism in the encoded domain. However, this will clearly ignore some important actions.

Goal-based plausibility In many multi-agent settings, the goals of the agents are known or can be inferred. Examples include any competitive or collaborative game where the goal is to win, and teamwork scenarios where the goal is common among all agents. If not known, we can try to infer the goals (e.g., see (Ramírez and Geffner 2010)). For this work, we assume that we have every agents' goal.

We consider a heuristic for calculating the plausible actions based on these goals. Given the goals of the other agents, we limit the set of plausible actions to those that work towards the goal using any state heuristic function. We extend the plausibility function to include the goal function $\mathcal{G}$, which maps an agent to its goal:

$$\Gamma(s, ag, \mathcal{G}) \subseteq App(s, ag)$$

Considering an agent's goal enables a wide variety of plausibility functions, such as actions at the start of a plan for the other agent's goal, and nesting the reasoning of other agents so that the next agent considers the goal of the following agent. In this paper, we consider only one possibility: we select the top k actions based on successor state quality using a given state heuristic function.

Definition 1. Best Successor Plausibility Function

Let h be a state heuristic function that maps a state and goal to a value. Given the state s, current agent ag, and goal function $\mathcal{G}$, we define $\Gamma_{h_k}(s, ag, \mathcal{G})$ to be the function that returns the top k actions from $App(s, ag)$ ranked by the following scoring function:

$$Score(s, a, ag, \mathcal{G}) = \max_{e \in \text{Eff}_a} h(Prog(s, a, e), \mathcal{G}(ag))$$

We can use any existing state heuristic in place of h. For our current implementation, we use a variant of the FF heuristic where successors are sorted first based on whether or not the action was a "helpful operator" (Hoffmann and Nebel 2001), and then subsequently based on the estimated distance to the goal. We prioritize helpful operators to encourage the planner to consider the actions that seem helpful for the other agent. As a plausibility function, this may work well in some domains and poorly in others. Nonetheless, it represents a reasonable first step for evaluating the potential to reduce the non-determinism in the problem.

3.3 Modified FOND planner

We made a number of modifications to the planner PRP to improve the efficiency of non-deterministic planning. In Section 3.1 we discussed the key modifications that we made to account for the multi-agent setting. We also changed other aspects of the planner including: (1) parsing the various goals of the agents and planning for the appropriate goal as discussed in Section 3.2; (2) always keeping the fluent that represents the active agent in the partial states for the policy; (3) disabling the feature that attempts to repair a policy locally before planning for the goal; and (4) changing the processing of unhandled states from a depth-first to a breadth-first manner (i.e., the *Open* data structure in Algorithm 1 was made into a queue as opposed to a stack). The latter three changes are not strictly required, but did improve the efficiency of the problems that we tested by a fair margin. Additionally, we were forced to disable PRP's "strong cyclic detection" feature, because conceptually it would need significant changes in order to work within the multi-agent setting.

4 Evaluation

As our approach opens FOND planning to a new class of problems, there are no publicly available benchmarks to evaluate on as far as we are aware. Instead, we provide new benchmark problems for three domains: Blocksworld, Tic-Tac-Toe and Sokoban. We use these to evaluate our proposed strategies for mitigating non-determinism, and the general ability of the planner to solve fully-observable MAP problems.

Experiment design We ran the generated policies against 1000 simulation trials, in which the moves of other agents were selected by taking the best applicable action measured using monte carlo rollouts, with the stopping condition of achieving the agent's goal. Policies were generated by running the planner for a maximum of 30 minutes and with a limit of 2Gb memory. If the planner did not complete in the time limit, the best policy found so far was returned. For each problem, we report on the following:

1. Success rate: The percentage of the 1000 trials that ended in success for the planning agent.
2. Policy size: The number of partial state-action pairs in the best found policy. Note that because of relevance analysis, the policy size typically is far smaller than the number of actual states that it can handle.
3. Planning time: The number of seconds required to compute the policy. **30m** indicates that the best policy found by the 30-minute mark was used. ✘ indicates that the planner ran out of memory, and no plan was returned.

Results

Table 1 show the results for each of the problems tested, and we discuss the results for each domain in turn.

Blocksworld The Blocksworld domain includes the first 10 problems from the FOND benchmark set of the 2008 International Planning Competition (IPC), with one additional agent added. We set the goal of the second agent to be the same as the acting agent, so this is a collaborative task. Note that in this domain, the actions for either agent may be non-deterministic (e.g., blocks may slip out of the hand).

		Blocksworld										Tic-Tac-Toe				Sokoban				
	N	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p1	p2	p3	p4	p1	p2	p3	p4	p5
Success	1	38	63	31	20	0	4	0	19	12	11	0	1	45	48	100	99	97	67	28
rate (%)	2	62	66	61	73	17	22	47	81	18	39	0	2	47	100	100	98	96	98	100
	3	92	97	90	96	49	62	81	90	84	76	0	4	47	100	100	98	96	97	100
	4	100	100	100	100	95	70	89	100	98	86	16	3	47	100	100	✘	✘	99	100
	∞	100	100	100	100	100	68	85	100	100	100	100	100	79	100	100	✘	✘	✘	✘
	Rnd	100	100	100	100	100	68	88	100	100	100	39	81	52	100	100	71	48	37	✘
Policy	1	32	27	31	111	49	25	68	63	43	30	42	47	14	7	11	27	27	138	906
size	2	48	77	78	316	141	99	175	217	73	83	88	107	26	15	11	26	31	5990	7891
	3	125	114	157	576	298	246	445	459	126	164	121	260	19	15	11	26	31	10952	10223
	4	337	236	593	932	757	973	892	830	593	526	871	250	22	15	11	✘	✘	10312	9270
	∞	550	286	631	1113	1149	785	987	699	867	818	1358	651	69	15	11	✘	✘	✘	✘
	Rnd	586	444	671	1045	1076	662	1006	763	745	818	827	606	27	12	11	11	11	11	✘
Planning	1	0.01	0.01	0.01	0.02	0.02	0.01	0.02	0.02	0.02	0.01	5	1	0.01	0.01	0.06	0.08	0.08	0.46	195
time (s)	2	0.02	0.04	0.04	0.14	0.06	0.04	0.08	0.12	0.02	0.04	155	11	0.26	0.01	0.06	0.08	0.10	30m	30m
	3	0.06	0.06	0.06	0.32	0.24	0.16	0.56	0.36	0.04	0.10	658	40	0.50	0.01	0.06	0. 10	0.12	30m	30m
	4	0.24	0.14	0.62	1.02	1.10	1.08	0.98	0.90	0.68	0.54	978	39	7.34	0.01	0.06	✘	✘	30m	30m
	∞	0.14	0.06	0.26	0.44	0.46	0.30	0.40	0.22	0.32	0.48	1765	114	39.32	0.01	0.06	✘	✘	✘	✘
	Rnd	0.20	0.12	0.28	0.44	0.44	0.28	0.44	0.26	0.28	0.36	30m	130	0.01	0.01	0.08	0.06	0.08	0.08	✘

Table 1: Success rate over 1000 simulated trials, generated policy size, and the time to synthesize a plan. ✘ indicates a memory violation, 30m indicates the solving was capped at 30 minutes, and N indicates the level of restricted non-determinism (∞ meaning no restriction and Rnd meaning a random subset of 3 applicable actions).

We found that in general, the policies were easy to compute for any level of restricted non-determinism, and the quality of the policies that restrict possible actions to 3 or 4 achieved near perfect performance in all problems. The notable exceptions are problems p6 and p7. For these, the loss in quality is a direct result of the lack of fairness in this domain. For many simulations, both agents simply "did nothing" expecting that the other would make the first move. This reveals the need for some form of deadlock-breaking mechanism in modelling problems that involve multiple agents.

The good performance of the random outcomes is to be expected in such a collaborative setting. However, because the search is not focused on the other agents plausible actions, the subsequent size of the policies increase.

Sokoban For the Sokoban domain, each player must push a box to a goal location, while an opposing agent attempts to push it to a different location. Across the four problems, the second agent starts closer to both our starting location and the block (i.e., in problem p1 the other agent cannot interfere, while in problem p5 the other agent can interfere to a large extent). The domain is inherently competitive.

We found an interesting threshold for the construction of policies in this domain. If the exploration considers any behaviour of the opponent that attempts to thwart our approach to the goal, the planner becomes overwhelmed handling the deadends. Partially, this is due to the type of deadends that occur in Sokoban, and which the underlying planner does not detect easily. Table 1 shows the effect of this as either the maximum time limit (**30m**) or memory limit (✘).

There is an interesting distinction between plans that consider only a few outcomes (which is quite effective) and those that scrutinize all outcomes (which run out of memory in 4 of the 5 problems). When focusing on the actions that help the opponent to achieve its goal, the planner can find a highly successful solution. It struggles, however, when considering actions that serve no other purpose for the opponent than to prevent its own goal, as these include actions that push the block to a position that is no use for any agent.

Tic-Tac-Toe For Tic-Tac-Toe, problems p1 and p2 start with an empty board and our goal is to draw while the opponent's goal is to win. In problem p2, we do not allow for noop actions, and as discussed earlier the performance improvement is substantial. The low success rate of the reduced non-determinism is a result of how poorly our heuristic function approximates the simulated behaviour of the other agent. Randomly selecting 3 actions for the opponent (i.e., roughly half of the applicable actions), on the other hand, covers a wider set of states and thus improves the success rate. However, our solver was able to find the perfect strategy to ensure a draw, when given enough time.

In problems p3 and p4, both players' goal is to win. Problem p3 starts with an open board that has no guaranteed strategy to win, and problem p4 starts with one move each (beginning in the bottom corners) so that a perfect strategy exists. In the latter case, we find that very few of the outcomes are required to generate a highly successful policy, and the perfect strategy is computed by the planner in a fraction of a

second.

Problem p3 is the typical starting configuration for Tic-Tac-Toe, and it poses an interesting challenge for FOND technology. State relevance, deadend detection and avoidance, effective search heuristics, etc., all play an important role in producing a successful and compact policy. Further, because no state is repeatable in the game (aside from the potential of noops), the assumption of fairness is not a concern.

5 Summary and Related Work

In this work, we presented a novel application of FOND planning in multi-agent environments based on the intuition that actions taken by others in the world can be viewed as non-deterministic outcomes of our own choices. This is in contrast with treating the environment and the other agents as an explicit adversary, which is the idea behind game structures used in verification (Piterman, Pnueli, and Sa'ar 2006), which in turn are at the base of ATL interpretation structures (Alur, Henzinger, and Kupferman 2002), in which a successor state is selected depending on the action that each agent performs. The latter approach is the one captured, in a planning setting, by the notion of joint state-action pairs in Bowling et al. (2003). Although conceptually different, these two approaches allow us to model actions whose effects are not completely determined by the state of the world.

The work of Bowling et al. (2003) considers a setting similar to ours where the goals of the other agents are known. The distinction, however, is that they use the model of agents' goals to devise a game-theoretic notion of equilibria for the agents, whereas we use the information to improve the efficiency of reasoning.

The main contribution of our work is the realization of the above intuition to leverage the recent advances in non-deterministic planning for solving problems in a multi-agent setting. A second key contribution is a means to reduce the non-determinism in the domain by restricting the set of *possible* actions for other agents to those that are *plausible* given their goal. We discussed some issues that arise when using our approach, and demonstrated its ability to solve multi-agent problems on a new suite of benchmarks that include both collaborative and competitive tasks.

The connection that we make between multi-agent planning and FOND planning presents an exciting initial step towards a range of more sophisticated techniques. The generality of a plausibility function opens the door to techniques ranging from nested simulations of agent behaviour to online learning methods that model other agents in the domain. It also provides an avenue to incorporate UCT and sample-based approaches (e.g., the PROST planner (Keller and Eyerich 2012)) with the more systematic search used by determinization-based planners such as PRP. As evidenced by the running example in this paper, our approach lends itself naturally to competitive games: we hope to apply this work to general game playing in the near future.

An important step forward is to bring this approach together with our recent work on planning over multi-agent epistemic states (Muise et al. 2015b). In that work, state is represented using a belief base with syntactic restrictions (Muise et al. 2015a), in which beliefs can be about the world or about other agents' belief, including their belief about us, etc.; so called *nested belief*. The formalism supports ontic actions: actions that modify the state of the world; and *deontic* actions: actions that modify the knowledge or belief of other agents. We encode these multi-agent epistemic planning problems as classical planning problems. However, the modelled actions can only be performed by the single planning agent. Bringing the multi-agent planning as FOND work from this paper together with multi-agent epistemic planning will enable us to solve a rich set of problems in which the planner considers both the actions others can take, the beliefs they have, and how these two interact.

Acknowledgements This research is partially funded by Australian Research Council Discovery Grant DP130102825, *Foundations of Human-Agent Collaboration: Situation-Relevant Information Sharing*

References

Alford, R.; Kuter, U.; Nau, D.; and Goldman, R. P. 2014. Plan aggregation for strong cyclic planning in nondeterministic domains. *Artificial Intelligence* 216:206–232.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM* 49(5):672–713.

Bolander, T., and Herzig, A. 2014. Group attitudes and multi-agent planning: overview and perspectives. Technical report.

Bowling, M. H.; Jensen, R. M.; and Veloso, M. M. 2003. A formalization of equilibria for multiagent planning. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 1460–1462.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, 28–35.

Brenner, M. 2003. A multiagent planning language. In *Proceedings of the Workshop on PDDL, ICAPS*, volume 3, 33–38.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1):35–84.

Fu, J.; Ng, V.; Bastani, F. B.; and Yen, I.-L. 2011. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *Proceedings of the 22nd International Joint Conference On Artificial Intelligence (IJCAI)*, 1949–1954.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated planning: theory & practice*. Elsevier.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 253–302.

Jensen, R. M., and Veloso, M. M. 2000. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research (JAIR)* 13:189–226.

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 119–127. AAAI Press.

Kovacs, D. L. 2012. A multi-agent extension of pddl3.1. 19.

Muise, C.; Miller, T.; Felli, P.; Pearce, A.; and Sonenberg, L. 2015a. Efficient reasoning with consistent proper epistemic knowledge bases. In Bordini; Elkind; Weiss; and Yolum., eds., *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Muise, C.; Belle, V.; Felli, P.; McIlraith, S.; Miller, T.; Pearce, A.; and Sonenberg, L. 2015b. Planning over multi-agent epistemic states: A classical planning approach. In *The 29th AAAI Conference on Artificial Intelligence*.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-deterministic Planning by Exploiting State Relevance. In *The 22nd International Conference on Automated Planning and Scheduling*, The 22nd International Conference on Automated Planning and Scheduling.

Muise, C.; McIlraith, S. A.; and Belle, V. 2014. Non-deterministic planning with conditional effects. In *The 24th International Conference on Automated Planning and Scheduling*.

Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2006. Synthesis of reactive(1) designs. In *VMCAI*, 364–380.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Ramirez, M., and Sardina, S. 2014. Directed fixed-point regression-based planning for non-deterministic domains. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.