Exploring Options for Efficiently Evaluating the Playability of Computer Game Agents

Todd Wareham and Scott Watson
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL Canada A1B 3X5
Email: harold@mun.ca, saw104@mun.ca

***Abstract*—Automatic generation of game content is an important challenge in computer game design. Such generation requires methods that are both efficient and guaranteed to produce playable content. While existing methods are adequate for currently available types of games, it is possible that games based on more complex entities and structures may require new methods. A case in point is gameplay involving complex non-player agents capable of exchanging items and facts with each other and human players. In this paper, we use computational complexity analysis to explore under what restrictions efficient playability evaluation of groups of such agents as well as the efficient design of such groups is and is not possible,**

I. Introduction

Given the time and cost involved with the human design of computer games, the ability to automatically generate game content is an important problem in computer game design. Research on this problem currently goes under the name of Procedural Content Generation (PCG) [1], [2]. It is critical that such automatic methods generate playable content because "[g]iven the way most commercial games are designed, any risk of the player being presented with unplayable content is unacceptable" [1, p. 183]. They should also operate quickly, particularly if content is being generated in real time to accommodate unanticipated player actions or choices (a situation in which human-based methods such as testing or manually adjusting game parameters to ensure playability are not applicable). Though existing automatic methods appear to be adequate for currently available types of games, *e.g.*, [3], this adequacy has not been rigorously proven (particularly for large-scale real-time operation). Whether or not this turns out to be the case, it would be useful to know if there are other options for efficient playability-guaranteed content generation, particularly in the case of games incorporating more complex structures and entities.

A case in point is the automatic generation of advanced Non-Player Character (NPC) agents for computer games. Though simple finite-state models of game agents [4]–[6] suffice for modeling short-term action-based interactions with human players, they are not satisfactory for modeling more socially realistic agent-player interactions that take place over longer (possibly disjoint) periods of time. As a minimum, this requires that agents maintain collections of items and facts which can be both exchanged with and used in defining behavior with respect to other agents and human players. Initial work on generating groups of such agents [7] has demonstrated that genetic algorithms and backtracking-based agent group playability assessment suffice for the off-line and real-time generation of moderate- ($\sim$ 50) and small- ($\sim$ 5) size groups of playable NPCs, respectively. However, in the interests of improved scalability and real-time generation, it would be most useful to know if provably efficient playability-guaranteed methods for generating such groups are available and, if so, in what circumstances.

In this paper, we present initial results addressing both of these questions. First, using techniques from computational complexity theory [8], we show that evaluating the playability of a given group of computer game agents (in particular, determining if a human player can interact with a group of agents to obtain a specified goal-set of items and facts) is NP-hard and thus intractable in general. This holds even in the case where there is only one given agent and no time limit on achieving the goal, as well as whether or not the agents operate autonomously or under the control of a game narrative manager. Second, using techniques from parameterized complexity theory [9], we establish that surprisingly few restrictions on agents and human-agent interactions render playability evaluation tractable. Though these results are derived for the model of game agents and playability given in [7], we show that these results apply not only to evaluating the playability of a much broader class of models but also to the full playable agent-group generation process itself.

The remainder of this paper is organized as follows. In Section II, we present an augmented finite-state machine model of game agents that can exchange items and facts with other agents and human players and formalize playability evaluation for such agents. Section III demonstrates the intractability of this problem. Section IV describes a methodology for identifying conditions for tractability, which is then applied in Section V to identify such conditions for agent playability evaluation. In order to focus in the main text on the implications of our results for computer game design, all proofs of results are given in Appendix B. Finally, our conclusions and directions for future work are given in Section VI.

A. Related Work

Determining whether given game levels can be completed and are thus playable is known to be NP-hard (and not

efficiently solvable in general) for many types of games [10]–[14]. However this work has not been extended to address the problem of designing playable levels, let alone assessing the playability or designing playable groups of agents.

There is existing work on the computational complexity of verifying if given multi-agent systems can perform a specified task (and hence are in a sense "playable") as well as designing multi-agent systems to correctly perform specified tasks [15]–[18]. The formalizations of agent control and interaction mechanisms and the environments analyzed in this work are very general and powerful (*e.g.*, arbitrary Turing machines or Boolean propositional formulae), rendering the intractability of these problems unsurprising. Moreover, as these formalizations obscure almost all details of the agent mechanisms and environment, the derived results are also unenlightening with respect to possible restrictions that could yield tractability. Similar reasoning applies with respect to existing complexity analyses of verification problems relative to single robots and swarms of robots (see [19, Section 4.2.1] and references).

II. Formalizing Agent Playability Evaluation

At a minimum, an agent capable of exchanging items and facts with another agent (which could be a human player) should be able to do the following:

- Maintain an internal state as well as collections of personal items and facts;
- Perform actions (and possibly change internal state) in response to another agent's actions and offered items and facts; and
- As part of a performed action, give in return some of its own personal items and facts to that other agent.

Following [7], there can be at most one copy of an item in a game at any time (*i.e.*, an item can be possessed by at most one agent or human player) but there can be any number of copies of a fact (*i.e.*, any number of agents or human players can possess the same fact).

There are many ways of modeling agents with the requisite abilities described above. In this paper, we will augment the finite-state machine model typically used in implementing game agents [4]. Recall that for any set S, 2^S denotes the set of all possible subsets of S (including the empty set $\emptyset$). Define an **augmented finite machine (AFSM)** (see Figure 1) relative to game-overall action-, item-, and fact-sets A^G, I^G, and F^G as a 2-tuple $\langle Q, \delta \rangle$ where Q is a set of states and $\delta \subseteq Q \times A^G \times 2^{I^G} \times 2^{F^G} \times A^G \times 2^{I^G} \times 2^{F^G} \times Q$ is a state-transition relation. A transition $(q, a, I, F, a', I', F', q') \in \delta$ of an AFSM M can be interpreted as an interaction between M and another agent in which that other agent performs action a with item- and fact-sets I and F offered to M and M responds in turn via action a' with (1) a change from state q to state q' and (2) item- and fact-sets I' and F' being given to the other agent. Any unspecified proposed action and offered item- and fact-sets relative to a state q whose result is not explicitly stated in δ is assumed to loop back on q with no effect, *e.g.*, M ignores the offered amulet and mumbles under its breath. There are many possible ways

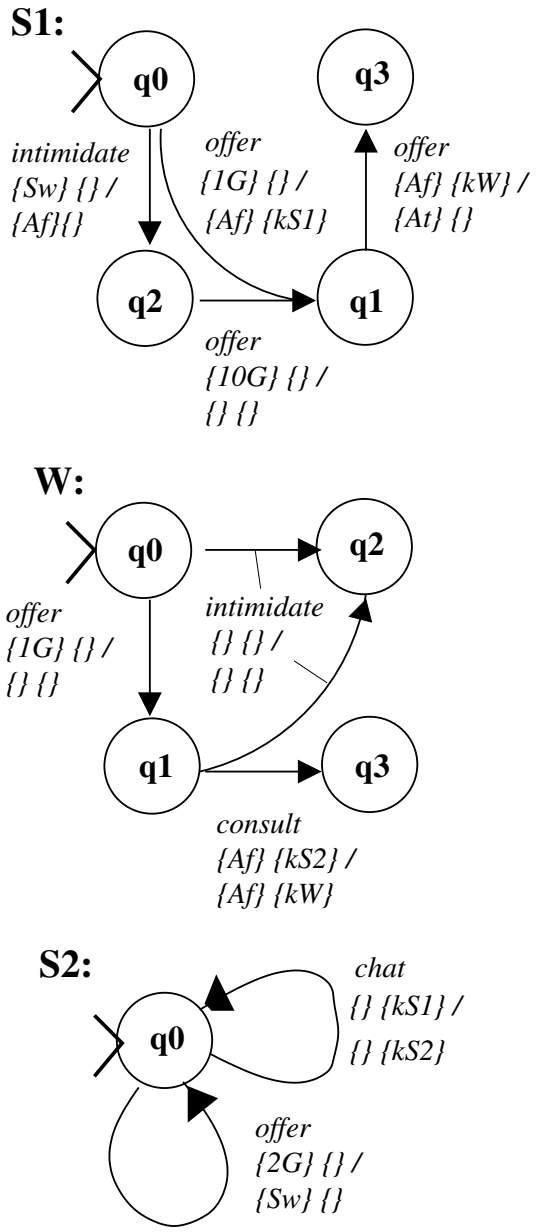

Fig. 1. Three Example Augmented Finite-State Machine (AFSM) agents

of specifying determinism and non-determinism relative to AFSM; in this paper, we will focus on AFSM that are **offered (o-) deterministic**, *i.e.*, for any given q, a, I, and F, there is at most one $(q, a, I, F, a', I', F', q') \in \delta$.

Examples of three possible AFSM representing two shopkeepers $S1$ and $S2$ and a wizard W are shown in Figure 1. These AFSM are defined relative to the action-, item-, and fact-sets {*chat, consult, intimidate, offer*}, {*false amulet (Af), true amulet (At), gold piece (G), sword (Sw)*}, and {*know shopkeeper #1 (kS1), know shopkeeper #2 (kS2), know wizard (kW)*}, respectively. Each transition $(q, a, I, F, a', I', F', q')$ is written as an arrow between q and q' with the label "$a\{I\}\{F\}/\{I'\}\{F'\}$", *i.e.*, a' is ignored. For example, $S1$ has

Interaction-sequence #1

Interaction	P	$S1$	$S2$	W
–	$\{2G\},\{\}$	$q0:\{Af,At\},\{kS1\}$	$q0:\{Sw\},\{kS2\}$	$q0:\{\},\{kW\}$
$S1$: *offer* $\{1G\},\{\}$	$\{1G,Af\},\{kS1\}$	$q1:\{1G,At\},\{kS1\}$	”	”
$S2$: *chat* $\{\},\{kS1\}$	$\{1G,Af\},\{kS1,kS2\}$	”	”	”
W: *offer* $\{1G\},\{\}$	$\{Af\},\{kS1,kS2\}$	”	”	$q1:\{1G\},\{kW\}$
W: *cnslt* $\{Af\},\{kS2\}$	$\{Af\},\{kS1,kS2,kW\}$	”	”	$q3:\{1G\},\{kW\}$
$S1$: *offer* $\{Af\},\{kW\}$	$\{At\},\{kS1,kS2,kW\}$	$q3:\{1G,Af\},\{kS1\}$	”	”

Interaction-sequence #2

Interaction	P	$S1$	$S2$	W
–	$\{20G\},\{\}$	$q0:\{Af,At\},\{kS1\}$	$q0:\{Sw\},\{kS2\}$	$q0:\{\},\{kW\}$
$S2$: *offer* $\{2G\},\{\}$	$\{18G,Sw\},\{\}$	”	$q0:\{2G\},\{kS2\}$	”
$S1$: *intim* $\{Sw\},\{\}$	$\{18G,Sw,Af\},\{\}$	$q2:\{At\},\{kS1\}$	”	”
W: *offer* $\{1G\},\{\}$	$\{17G,Sw,Af\},\{\}$	”	”	$q1:\{1G\},\{kW\}$
W: *cnslt* $\{Af\},\{\}$	”	”	”	”
$S1$: *offer* $\{10G\},\{\}$	$\{7G,Sw,Af\},\{kS1\}$	$q1:\{10G,At\},\{kS1\}$	”	”
$S2$: *chat* $\{\},\{kS1\}$	$\{7G,Sw,Af\},$ $\{kS1,kS2\}$	”	”	”
W: *cnslt* $\{Af\},\{kS2\}$	$\{7G,Sw,Af\},$ $\{kS1,kS2,kW\}$	”	”	$q3:\{1G\},\{kW\}$
$S1$: *offer* $\{Af\},\{kW\}$	$\{7G,Sw,At\},$ $\{kS1,kS2,kW\}$	$q3:\{10G,Af\},\{kS1\}$	”	”

Interaction-sequence #3

Interaction	P	$S1$	$S2$	W
–	$\{2G\},\{\}$	$q0:\{Af,At\},\{kS1\}$	$q0:\{Sw\},\{kS2\}$	$q0:\{\},\{kW\}$
$S2$: *offer* $\{2G\},\{\}$	$\{Sw\},\{\}$	”	$q0:\{2G\},\{kS2\}$	”
$S1$: *intim* $\{Sw\},\{\}$	$\{Sw,Af\},\{\}$	$q2:\{At\},\{kS1\}$	”	”
W: *intim* $\{\},\{\}$	”	”	”	$q2:\{\},\{kW\}$
$S2$: *intim* $\{\},\{\}$	”	”	”	”

Fig. 2. Three Example AFSM Agent – Human Player Interaction-sequences

a transition between $q0$ and $q2$ such that $S1$ hands over the fake amulet when intimidated by another agent with a sword.

We define the execution of interactions of an AFSM $M = \langle Q, \delta \rangle$ with another agent X as follows. A transition $(q, a, I, F, a', I', F', q')$ is **enabled** relative to $M = \langle Q, \delta \rangle$ and X, where M and X currently possess the items and facts in sets I_M, F_M, I_X, and F_X, respectively, if:

1) $(q, a, F, I, a', F', I', q') \in \delta$;
2) M is currently in state q;
3) $I \subseteq I_X$ and $F \subseteq F_X$; and
4) $I' \subseteq I_M \cup I$ and $F' \subseteq F_M$.

The **execution** of a transition $(q, a, I, F, a', I', F', q')$ that is enabled relative M and X has the following effects:

1) The state of M is set to q';
2) I_X is set to $(I_X - I) \cup I'$;
3) F_X is set to $F_X \cup F'$; and
4) I_M is set to $(I_M \cup I') - I'$.

In this paper, for simplicity, we only consider the case in which the other agent X is a human player, *i.e.*, agent actions can only be triggered by human players. Three possible sequences of interactions of the AFSM in Figure 1 with a human player are shown in Figure 2. Note that in each such interaction-sequence, the players and agents start with specified item- and fact-sets, each agent starts in a designated state $q0$, and if a player temporarily stops interacting an agent M left in state q with current item- and fact-sets I and F, the next interaction of the player with M resumes with M in state q with current item- and fact-sets I and F.

Playability of a group of AFSMs relative to a human player can be formalized in terms of hard (inviolable) and soft (violable) constraints [7]. Example hard and soft constraints are, respectively, that a specified goal must be achieved and that the interactions in any goal-achieving interaction-sequence should incorporate as many of the actions allowable to agents as possible. Assessments of playability are based on the degree to which these constraints can be satisfied by a human player interacting with the given agents. For simplicity, let us focus on minimum playability, *i.e.*, whether or not a human player can interact with a given set of agents to obtain specified

goal-sets of facts and items within a given time limit. In our running example, with respect to the goal consisting of having the true amulet and knowing the wizard, the first and second interaction-sequences in Figure 2 achieve the goal within 5 and 8 interactions, respectively, while the third interaction-sequence does not achieve the goal and moreover cannot be extended by any sequence of interactions to achieve the goal.

The above yields the following formalization:

AFSM AGENT PLAYABILITY EVALUATION (APE)
Input: A set $A = \{a_1, \ldots, a_n\}$ of AFSM with associated initial item- and fact-sets $\{I^0_{a_1}, \ldots, I^0_{a_n}\}$ and $\{F^0_{a_1}, \ldots F^0_{a_n}\}$, initial player item- and fact-sets I^0_P and F^0_P, goal item- and fact-sets I_G and F_G, and a positive integer t.
Question: Can the player obtain I_G and F_G by engaging in at most t interactions with the agents in A?

Note that this formalization applies regardless of whether the agents in A operate autonomously or under the direction of a game narrative manager; hence, results derived relative to this formalization will apply in both of these cases.

III. AGENT PLAYABILITY EVALUATION IS INTRACTABLE

In this section, we address whether or not agent playability evaluation can be done efficiently relative to the model described in Section II. Following general practice in Computer Science [8], we define efficient solvability as being solvable in the worst case in time polynomially bounded in the input size, and show that a problem is not polynomial-time solvable, *i.e.*, not in the class P of polynomial-time solvable problems, by proving it to be at least as difficult as the hardest problems in problem-class NP (see [8] and Appendix A for details).

Result A: APE is NP-hard.

Modulo the conjecture $P \neq NP$ which is widely believed to be true [20], the above shows that APE is not polynomial-time solvable. Note that this result holds even in the very restricted case in which the player only interacts with a single agent, *i.e.*, $|A| = 1$, and an unlimited number of interactions between the player and that agent is allowed, *i.e.*, $t = \infty$.

IV. A METHOD FOR IDENTIFYING TRACTABILITY CONDITIONS

A computational problem that is intractable for unrestricted inputs may yet be tractable for non-trivial restrictions on the input. This insight is based on the observation that some NP-hard problems can be solved by algorithms whose running time is polynomial in the overall input size and non-polynomial only in some aspects of the input called *parameters*. In other words, the main part of the input contributes to the overall complexity in a "good" way, whereas only the parameters contribute to the overall complexity in a "bad" way. In such cases, the problem Π is said to be **fixed-parameter tractable** for that respective set of parameters. The following definition states this idea more formally.

Definition 1: Let Π be a problem with parameters $k_1, k_2, \ldots$. Then Π is said to be **fixed-parameter (fp-) tractable** for parameter-set $K = \{k_1, k_2, ...\}$ if there exists at least one algorithm that solves Π for any input of size n in time $f(k_1, k_2, ...)n^c$, where $f(\cdot)$ is an arbitrary function and c is a constant. If no such algorithm exists then Π is said to be **fixed-parameter (fp-) intractable** for parameter-set K.

In other words, a problem Π is fp-tractable for a parameter-set K if all superpolynomial-time complexity inherent in solving Π can be confined to the parameters in K. In this sense the "unbounded" nature of the parameters in K can be seen as a reason for the intractability of the unconstrained version of Π.

There are many techniques for designing fp-tractable algorithms [9], [21], and fp-intractability is established in a manner analogous to classical polynomial-time intractability by proving a parameterized problem is at least as difficult as the hardest problems in one of the problem-classes in the W-hierarchy $\{W[1], W[2], ...\}$ (see [9] and Appendix A for details). Additional results are typically implied by any given result courtesy of the following lemmas:

Lemma 1: [22, Lemma 2.1.30] If problem Π is fp-tractable relative to parameter-set K then Π is fp-tractable for any parameter-set K' such that $K \subset K'$.

Lemma 2: [22, Lemma 2.1.31] If problem Π is fp-intractable relative to parameter-set K then Π is fp-intractable for any parameter-set K' such that $K' \subset K$.

Observe that it follows from the definition of fp-tractability that if an intractable problem Π is fp-tractable for parameter-set K, then Π can be efficiently solved even for large inputs, provided only that the values of all parameters in K are relatively small. This strategy has been successfully applied to a wide variety of intractable problems (see [9], [23] and references). In the next section we investigate how the same strategy may be used to render the problem APE tractable.

V. WHAT MAKES AGENT PLAYABILITY EVALUATION TRACTABLE?

The AFSM agent playability evaluation problem has a number of parameters whose restriction could render agent playability evaluation tractable. An overview of the parameters that we considered in our fp-tractability analysis is given in Table I. These parameters can be divided into three groups:

1) Restrictions on the game agents;
2) Restrictions on the human player; and
3) Restrictions on the game itself.

In the remainder of this section, we will assess the fp-tractability of APE relative to all parameters in Table I (Section V-A), show how these results apply in more general settings (Section V-B) as well as to playable AFSM agent generation (Section V-C), and discuss the implications of these results for computer game design (Section V-D).

TABLE I
FIXED-PARAMETER INTRACTABILITY RESULTS FOR THE AGENT PLAYABILITY EVALUATION PROBLEM

Parameter		Result				
Description	Name	B	C	D	E	F
AGENTS:						
# agents	$\|A\|$	–	P	P	P	1
max # items / agent	i_A	0	1	1	1	–
max # facts / agent	f_A	3	–	–	–	–
max # items / interaction	i_I	0	1	1	2	1
max # facts / interaction	f_I	P	–	2	2	2
max # states / agent	$\|Q\|$	2	–	–	–	P
max # interactions / state	$\|I\|$	1	–	2	–	–
PLAYER:						
max # items / player	i_P	0	–	–	–	–
max # facts / player	f_P	P	–	–	P	P
GAME:						
max # interactions in game	t	P	P	–	P	P
max # items in goal	i_G	0	0	0	0	0
max # facts in goal	f_G	1	1	1	1	1

A. Results

Our parameterized intractability results are summarized in Table I. Each column describes an intractability result that holds relative to the set of all parameters whose entries in that column are not dashes ("–"); if the result holds when a non-dashed parameter has constant value c, this indicated by an entry for that parameter with the value c. Result B is notable because it, when combined with results implied by Lemma 2, establishes the intractability of APE relative to all subsets of the considered parameters that do not include $|A|$; the intractability of many (but not all) of those remaining subsets including $|A|$ is then established by Results C–F.

At present, we have a lone tractability result:

Result G: APE is fp-tractable for $\{|A|, |I|, t\}$.

Results B, D, F, and G, combined with those implied by Lemmas 1 and 2, establish the intractability of APE relative to all subsets of $\{|A|, i_I, f_I, |I|, t, i_G, f_G\}$. This in turn establishes that the parameter-set in Result G is minimal in the sense that no parameter in that set can be deleted to yield fp-tractability.

B. Generality of Agent Playability Evaluation Results

Our intractability results, though defined relative to an admittedly simple model of game agent and human-agent interaction, have a remarkable generality. Observe that this model is a special case of many more realistic models, *e.g.*,

- deterministic AFSM are special cases of both nondeterministic and probabilistic AFSM (AFSM without nondeterminism or in which all all actions have probability of execution 1.0 if their triggering conditions are satisfied are deterministic);
- player-activated AFSM are special cases of autonomous AFSM (restrict non-player-triggered interaction); and
- basic AFSM ares special cases of AFSM with extra abilities (restrict use these extra abilities).

Intractability results for these more realistic models then follow from the well-known observation in computational complexity theory that intractability results for a problem Π also hold for any problem Π' that has Π as a special case (suppose Π is intractable; if Π' is tractable, then any algorithm for Π' can be used to solve Π efficiently, which contradicts the intractability of Π – hence, Π' must also be intractable).

Our fp-tractability result is more fragile, as innocuous changes to agent or game models may in fact violate assumptions critical to the operation of the algorithm underlying this result. For now, we can say that as our fp-tractability results depend on the combinatorics of possible player-agent interaction and require only that such an interaction can be checked for validity and performed in time polynomial in the sizes of the entities involved in that interaction, these results apply relative to playability evaluation for *all* choices of agent and game models whose player-agent interactions are polynomial-type verifiable relative to these models.

C. Applicability to Playable Agent Generation

The results given so far for APE are useful, especially in suggesting improvements to the playability-assessment module in the system described in Watson et al [7]. However, our ultimate goal is still the efficient generation of playable agents, regardless of whether or not an explicit playability-assessment module is used. In this section, we will sketch how our results for APE apply to this larger problem.

Though a full formalization of the AFSM agent-group generation problem is beyond the scope of this paper, we can informally sketch what such a problem might look like. It is trivial to construct an agent-group A that will allow a player to obtain specified goal item- and fact-sets within t steps (let A consist of a single AFSM whose lone transition gives the player all required items and facts in response to an arbitrary action on the part of the player). Hence, a specification of the characteristics of the desired agent-group must be given; let us call such a specification S_A. As a minimum, S_A should specify two types of characteristics:

1) Overall characteristics of agent-group and individual-agent structure; and
2) Required internal structures of individual agents.

The first type of characteristics correspond to the AGENTS parameters in Table I while the second could consist of specifications of required states and transitions along the lines of the system described in Watson et al [7].

The above yields the following:

AFSM PLAYABLE AGENT GENERATION (PAG)
Input: Item- and fact-sets I and F, an AFSM-group specification S_A, initial player item- and fact-sets I_P^0 and F_P^0, goal item- and fact-sets I_G and F_G, and a positive integer t.
Output: An AFSM-group A consistent with S_A such that the player can obtain I_G and F_G by engaging in at most t interactions with the agents in A, if such an A exists, and special symbol $\perp$ otherwise.

This informal version can be fully formalized relative to a particular format in which specifications are written. Consider the set of specification-formats in which one can create in time polynomial in $|A|$ a specification that can only be satisfied by a given AFSM-set A; let us call this set $\mathcal{S}$.

To see how the intractability results given in Sections III, V-A, and V-B apply to PAG, note the following – namely, any algorithm a for a version of PAG formalized relative to any member of $\mathcal{S}$ can be used to answer any instance of APE (given an instance I of APE with agent-set A, construct an instance I' of PAG such that S_A generates A; return "No" for I if a run on I' returns $\perp$ and "Yes" otherwise). Hence, *any* intractability result (including *all* intractability results in Sections III, V-A, and V-B) that forbids the existence of a certain type of algorithm for APE also then forbids that type of algorithm for any version of PAG formalized relative to any member of $\mathcal{S}$. Our lone tractability result for APE does not appear to apply in such a general manner to PAG; however it may hold relative to specific members of $\mathcal{S}$.

D. Discussion

We have found that evaluating agent playability is NP-hard (Result A). This NP-hardness holds for a basic agent model and a minimal playability condition that a human player can attain a specified goal by interacting with the given group of agents, even when that group consists of a single agent; moreover, as pointed out in Section V-C, this also applies to plausible schemes for generating playable agents.

Our results immediately imply that it is unlikely that deterministic polynomial-time methods exist for these problems. The scope of these results is actually broader still. It is widely believed that $P = BPP$ [24, Section 5.2] where BPP is considered the most inclusive class of problems that can efficiently solved using probabilistic methods (in particular, methods whose probability of correctness can be efficiently boosted to be arbitrarily close to probability one). Hence, our results also imply that unless $P = NP$, there are no probabilistic polynomial-time methods which correctly evaluate or generate playable agent-groups with high probability for all inputs. This then constitutes the first proof that *no* currently-used method (including the automated search and simulated-play-based processes described in [1], [2] or evolutionary algorithms such as that employed in [7]) can guarantee both efficient and correct operation for all inputs for these problems.

As described in Section IV, efficient correctness-guaranteed methods may yet exist relative to plausible restrictions on the input and output. To our knowledge, no such restrictions have been proposed in the literature for either agent playability evaluation or playable agent generation. It seems reasonable to conjecture that some restrictions relative to the parameters listed in Table I should render these problems tractable. However, no single one or indeed many possible combinations of these restrictions can yield tractability, even when the parameters involved are restricted to very small constants (Results B–F and Section V-C).

The one exception that we have found to date (and only for agent playability evaluation) is that of simultaneously restricting $|A|$, $|I|$, and t (Result G). Though this may initially seem of limited interest in that it overly restricts the form of games whose playability can be checked efficiently, it actually suggests several reasonable ways in which games can be decomposed into sub-games whose playability can be checked efficiently. For example, a long game could be decomposed into several shorter ones (restrict t). Alternatively, the game could be structured such that only a very small number of agents or player-agent interactions are necessary and/or relevant to achieving the goal (restrict $|A|$ and/or $|I|$); this could be done while preserving a larger game environment by embedding the goal-relevant set of agents and interactions within a goal-irrelevant set of agents and interactions, *e.g.*, only a few shopkeepers, wizards, or travellers are worth talking to and only about specific matters.

A valid objection to this lone tractability result is that the running time of the underlying algorithm is impractical. This is often true of the initial algorithms derived relative to a parameter-set. However, our result is important nonetheless because it establishes fixed-parameter tractability relative to a set of parameters which (by reasoning like that above) can be of small value in practice. Once this has been done, surprisingly effective parameterized algorithms can frequently be developed with both greatly diminished non-polynomial terms and polynomial terms that are quadratic and even linear in the input size (see [9], [21] and references).

A final very important proviso is in order – namely, as illuminating as the results given here are in demonstrating basic forms of (in)tractability for the agent playability evaluation and playable agent generation problems, these results do not necessarily imply that methods currently being applied to evaluate or generate agents are impractical. Differing agent models, the particular situations in which these methods are being applied, and accepted standards by which method practicality is assessed may render the results given here irrelevant. For example, current methods may already be implicitly exploiting restrictions on the input and output such that both efficient and correct operation (or operation that is correct with probability very close to one) are guaranteed. That being said, not knowing the precise conditions under which such practicality holds could have very damaging consequences, *e.g.*, drastically slowed gameplay and/or unplayable game content, for systems (in particular, real-time-adaptable systems) using such methods that stray outside these conditions. Given that (as noted earlier in Section I) playability and its efficient evaluation and enforcement are very important properties of game systems, the acquisition of such knowledge via a combination of rigorous empirical and theoretical analyses should be a priority. With respect to theoretical analyses, it is our hope that the techniques and results in this paper comprise a useful first step.

VI. Conclusions

We have presented a formal characterization of the problem of game agent playability evaluation relative to an augmented finite-state machine model of game agents. Our complexity analyses reveal that, while this problem is computationally intractable in general, there are conditions that render it tractable. Knowledge of this and other such conditions can be exploited in computer game design to create efficient playability-guaranteed content generation methods with respect to more complex and interesting gameplay involving player interactions with more socially realistic game agents.

In future research, we plan to explore the computational consequences of additional types of restrictions on agent playability evaluation and playable agent design relative to both the agent-model described in this paper and more complex agent-models (*e.g.*, agents that are truly autonomous rather than player-activated) as well as minimal and broader conceptions of playability. We will also build on previous work establishing the NP-hardness of assessing the playability of and generating playable game levels by applying parameterized analysis to establish under which restrictions these problems can and cannot be solved efficiently. Finally, given work positing connections between human cognition and fixed-parameter tractability [19], [25], we will investigate the extent to which results such as those we have derived here can help in creating games whose level of difficulty not only is more appropriate to human players but can also be efficiently customized to the abilities of those players [2].

Acknowledgments

The authors would like to thank Rod Byrne, Andrew Vardy, and Wolfgang Banzhaf for encouraging them to embark on this research. TW was supported by NSERC Discovery Grant RGPIN 228104-2010 and SW was supported by NSERC Discovery Grant RGPIN 283304-2012 to Wolfgang Banzhaf and a doctoral award from the Dean of the School of Graduate Studies at MUN.

References

[1] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based Procedural Content Generation: A Taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[2] G. N. Yannakakis and J. Togelius, "Experience-driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[3] D. D. Williams-King, J. Denzinger, J. Aycock, and B. Stephenson, "The Gold Standard: Automatically Generating Puzzle Game Levels," in *Proceedings of AIIDE 2012*. Palo Alto, CA: AAAI Press, 2012, pp. 191–196.

[4] M. Dawe, S. Gargolinski, L. Dicken, T. Humphries, and D. Mark, "Behavior Selection Algorithms: An Overview," in *Game AI Pro: Collected Wisdom of Game AI Professionals*, S. Rabin, Ed. Boca Raton, FL: CRC Press, 2014, pp. 47–60.

[5] A. Nareyek, "Game AI is Dead. Long Live Game AI!" *IEEE Intelligent Systems*, vol. 22, no. 1, pp. 9–11, 2007.

[6] G. N. Yannakakis, "Game AI Revisited," in *Proceedings of the 9th Conference on Computing Frontiers*. ACM, 2012, pp. 285–292.

[7] S. Watson, W. Banzhaf, and A. Vardy, "Automated Design for Playability in Computer Game Agents," in *Proceedings of the 2014 IEEE Conference on Computational Intelligence in Games*. IEEE Press, 2014.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W.H. Freeman, 1979.

[9] R. Downey and M. Fellows, *Fundamentals of Parameterized Complexity*. Berlin: Springer, 2013.

[10] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta, "Classic Nintendo Games are (Computationally) Hard," in *7th International Conference on Fun with Algorithms*, ser. Lecture Notes in Computer Science, A. Ferro, F. Luccio, and P. Widamayer, Eds., no. 8496. Berlin: Springer, 2014, pp. 40–51.

[11] M. Forišek, "Computational Complexity of Two-dimensional Platform Games," in *5th International Conference on Fun with Algorithms*, ser. Lecture Notes in Computer Science, P. Boldi, Ed., no. 6099. Berlin: Springer, 2010, pp. 214–227.

[12] L. Gualà, S. Leucci, and E. Natale, "Bejeweled, Candy Crush and other Match-Three Games are (NP-)Hard," in *Proceedings of the 2014 IEEE Conference on Computational Intelligence in Games*. IEEE Press, 2014.

[13] J. Lynch, "Collecting Things Under Time Pressure is Hard," *Tiny Transactions on Computer Science*, vol. 1, 2012.

[14] G. Viglietta, "Lemmings is PSPACE-complete," *arXiv preprint arXiv:1202.6581*, 2012.

[15] M. Wooldridge and P. E. Dunne, "The computational complexity of agent verification," in *Intelligent Agents VIII*. Springer, 2002, pp. 115–127.

[16] P. E. Dunne, M. Laurence, and M. Wooldridge, "Complexity results for agent design," *Annals of Mathematics, Computing & Teleinformatics*, vol. 1, no. 1, pp. 19–36, 2003.

[17] I. A. Stewart, "The complexity of achievement and maintenance problems in agent-based systems," *Artificial Intelligence*, vol. 2, no. 146, pp. 175–191, 2003.

[18] M. K. Valiev and M. I. Dekhtyar, "Complexity of verification of nondeterministic probabilistic multiagent systems," *Automatic Control and Computer Sciences*, vol. 45, no. 7, pp. 390–396, 2011.

[19] I. van Rooij and T. Wareham, "Parameterized Complexity in Cognitive Modeling: Foundations, Applications, and Opportunities," *Computer Journal*, vol. 51, no. 3, pp. 385–404, 2008.

[20] L. Fortnow, "The Status of the P Versus NP Problem," *Communications of the ACM*, vol. 52, no. 9, pp. 78–86, 2009.

[21] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
[22] T. Wareham, *Systematic Parameterized Complexity Analysis in Computational Phonology*. Ph.D. thesis, Department of Computer Science, University of Victoria, 1999.
[23] U. Stege, "The Impact of Parameterized Complexity to Interdisciplinary Problem Solving," in *The Multivariate Algorithmic Revolution and Beyond*, ser. Lecture Notes in Computer Science, H. L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, Eds. Berlin: Springer, 2012, no. 7370, pp. 56–68.
[24] A. Wigderson, "P, NP and mathematics — A computational complexity perspective," in *Proceedings of ICM 2006: Volume I.* Zurich: EMS Publishing House, 2007, pp. 665–712.
[25] I. van Rooij, "The Tractable Cognition Thesis," *Cognitive Science*, no. 32, pp. 939–984, 2008.
[26] L. Cai, J. Chen, R. Downey, and M. Fellows, "On the parameterized complexity of short computation and factorization," *Archives of Mathematics and Logic*, vol. 36, pp. 321–337, 1997.

APPENDIX A
PROVING INTRACTABILITY

Given some criterion of tractability like polynomial-time or fixed-parameter solvability, we can define the class T of all computational problems that are tractable relative to that criterion. For example, T could be the class P of decision problems (see below) solvable in polynomial-time, or FPT, the class of parameterized problems that are fp-tractable. We can show that a particular problem is not in T (and thus that this problem is intractable) by showing that this problem is at least as hard as the hardest problem in some class C that properly includes (or is strongly conjectured to properly include) T. For example, C could be NP, the class of decision problems whose candidate solutions can be verified in polynomial time, or a class of parameterized problems in the W-hierarchy $= \{W[1], W[2], \ldots, W[P], \ldots, XP\}$ (see [8] and [9], respectively, for details).

We will focus here on reducibilities between pairs of **decision problems**, *i.e.*, problems whose outputs are either "Yes" or "No". The two types of reductions used in this paper are as follows.

Definition 2: Given a pair Π, Π' of decision problems, Π **polynomial-time many-one reduces to** Π if there is a polynomial-time computable function f mapping instances I of Π to instances $f(I)$ of Π' such that the answer to I is "Yes" if and only if the answer to $f(I)$ is "Yes".

Definition 3: Given a pair Π, Π' of parameterized decision problems with parameters p and p, respectively, , Π **fp-reduces to** Π if there is a function f mapping instances $I = (x, p)$ of Π to instances $I' = (x', p')$ of Π' such that (i) f is computable in $g(p)|x|^{\alpha}$ time for some function $g()$ and constant α, (ii) $p' = h(p)$ for some function $h()$, and (iii) the answer to I is "Yes" if and only if the answer to $I' = f(I)$ is "Yes".

A reducibility is appropriate for a tractability class T if whenever Π reduces to Π' and $\Pi' \in T$ then $\Pi \in T$. We say that a problem Π is C**-hard** for a class C if every problem in C reduces to Π. A C-hard problem is essentially as hard as the hardest problem in C.

Reducibilities become particularly useful when the following three properties hold:

1) If Π reduces to Π' and Π is C-hard then Π' is C-hard.
2) If Π is C-hard and $T \subset C$ then $\Pi \notin T$, *i.e.*, Π is not tractable.
3) If Π is C-hard and $T \subseteq C$ then $\Pi \notin T$ unless $T = C$, *i.e.*, Π is not tractable unless $T = C$.

These properties are easily provable for many commonly-used reducibilities, including those given in Definitions 2 and 3 above. The first and third of these properties will be used to show intractability below relative to tractable T-classes P and FPT and enclosing but not provably properly enclosing C-classes NP, $W[1]$, and XP. Note that these intractability results hold relative to the conjectures $P \neq NP$ and $FPT \neq W[1]$ which, though not proved, have strong empirical support and are commonly accepted as true within the Computer Science community (see [8], [9], [20] for details).

APPENDIX B
PROOFS OF RESULTS

All of our intractability results will be derived using reductions from the following NP-hard decision problems:

NONDETERMINISTIC TURING MACHINE COMPUTATION (NTMC)
Input: A single-tape, single-head nondeterministic Turing machine $M = \langle \Sigma, Q, \Delta, s, F \rangle$ (where Σ is an alphabet, Q is a set of internal states, $\Delta \subseteq Q \times \Sigma \times Q$ is a set of transitions, $s \in Q$ is the start state, and $f \in Q$ is the final state), a word $x \in \Sigma^*$, and a positive integer k.
Question: Is there a computation of M on x starting in s that reaches some final state $f \in F$ in at most k steps?

DOMINATING SET [8, Problem GT2]
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Does G contain a dominating set of size $\leq k$, *i.e.*, is there a subset $V' \subseteq V$, $|V'| \geq k$, such that for all $v \in V'$, either $v \in V'$ or there is a $v' \in V'$ such that $(v, v') \in E$?

CLIQUE [8, Problem GT19]
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Does G contain a clique of size $\geq k$, *i.e.*, is there a subset $V' \subseteq V$, $|V'| \geq k$, such that for all $v, v' \in V'$, $(v, v') \in E$?

Lemma 3: $\{k\}$-NTMC fp-reduces to $\{i_A, f_A, i_I, f_I, |Q|, |I|, i_G, f_G, t, i_P, f_P\}$-APE such that in the constructed instance of APE, $i_A = 0$, $f_A = 3$, $i_I = 0$, $|Q| = 2$, $|I| = 1$, $i_P = 0$, $i_G = 0$, and $f_G = 1$, and the values of f_I, f_P, and t are all functions of k in the given instance of NTMC.

Proof: Given an instance $\langle M = \langle \Sigma, Q, \Delta, s, F \rangle, x, k \rangle$ of NTMC, construct an instance of APE in which the state of the NTM at time t, $0 \leq t \leq k$, is encoded by a time-fact t, a time/head-position fact t/i, $1 \leq i \leq k$, a time/state fact t/q, $q \in Q$, and k time/tape square position/tape square contents (TTT) facts $t/i/s$, $1 \leq i \leq k$ and $s \in \Sigma$. Each write transition (q, x, q') in M is encoded by $k \times k \times \Sigma$ agents each consisting of states q_0 and q_1 with a single transition that is enabled by the time-fact t, time/head position fact t/i, time/state fact t/q,

and TTT fact $t/i/s$ and returns the corresponding facts $(t+1)$, $(t+1)/i$, $(t+1)/q'$, and $(t+1)/i/s$ for $0 \leq t < k$, $1 \leq i \leq k$, and $s' \in \Sigma$. Analogous sets of agents are constructed for all left-move and right-move transitions in M. The following four sets of two-state single-transition agents are also required:

1) A set of agents that individually enable on time-fact t and TTT fact $(t-1)/i/s$ and return the TTT fact $t/i/s$ for $1 \leq t, i \leq k$ and $s \in \Sigma$ (*i.e.*, bring forward in time all tape-square contents not updated by a write-transition at time $(t-1)$);
2) A set of agents that individually enable on time/state fact t/f and return time/state fact $(t+1)/f$ for $1 \leq t < k$ and $f \in F$ (*i.e.*, bring forward in time any final state reached at or before time $(t-1)$);
3) A set of agents that individually enable on time-fact k and TTT fact $k/i/s$ and return TTT fact $k/i/s''$ for some $s'' \notin \Sigma$ (*i.e.*, erase the contents of the tape at time k); and
4) A set of agents that enable on time-fact k, time/state fact t/f, and TTT facts $k/1/s'', k/2/s'', \ldots k/k/s''$ and return completion-fact c for $f \in F$.

Each agent starts with no items and facts it returns and the player starts with no items and the facts corresponding to an initial state q_0, head position 1, and x on the first $|x|$ squares of the tape and s'' in the remaining $k - |x|$ squares. Finally, set the goal to c and $t = (k+1)k+1$. Note that the instance of APE described above can be constructed in time that is fp-tractable with respect to k and the size of then given instance of DOMINATING SET (this is necessary because k is stored in binary in this given instance and the value of k is exponential in $\log_2 k$).

If there is a transition-sequence of length at most k for M computing on x from s that reaches a final state, there is a sequence of exactly t agent-player interactions that will achieve the goal (as all tape-squares must be updated to time k and be available for subsequent erasure in order to obtain goal-fact c). Conversely, if there is an interaction-sequence of length t that achieves the goal, there must be embedded in this sequence a subsequence of interactions of length $\leq k$ that allowed time/state fact k/f to be derived from time/state fact $0/q_0$, time/head position fact $0/1$, and the TTT facts encoding of x on the tape, which corresponds to a sequence k transitions that allow M computing on x from s to reach a final state.

To complete the proof, note that in the constructed instance of APE, $i_A = 0$, $f_A = 3$, $i_I = 0$, $|Q| = 2$, $|I| = 1$, $i_P = 0$, $i_G = 0$, $f_G = 1$, $f_I = k+2$, $f_P = k(k+3)+k+1$, and $t = (k+1)k+1$. ∎

Lemma 4: DOMINATING SET polynomial-time many-one reduces to APE such that in the constructed instance of APE, $i_A = i_I = 1$, $i_G = 0$, $f_G = 1$, and $|A|$ and t are both a function of k in the given instance of DOMINATING SET.

Proof: Given an instance $\langle G = (V, E), k \rangle$ of DOMINATING SET, the constructed instance of APE consists of k identical agents plus an additional final agent. Each of the identical agents consists of an initial state q_0 and a transition from q_0 to each of the $|V|$ states q_i, $1 \leq i \leq |V|$, in which the offered item v_i is exchanged for the set of facts corresponding to all vertices in the neighbourhood of of v_i (including v_i itself) in G. The final agent consists of two states q_0 and q_1 and a transition from q_0 to q_1 that exchanges the complete set of vertex-facts for G for a completion-fact. Each identical agent starts with no items and the complete set of vertex-facts for G, the final agent starts with no items and the completion-fact, and the player starts with the complete set of vertex-items for G and no facts. Finally, the goal is the completion-fact and $t = k+1$. Note that the instance of APE described above can be constructed in time polynomial in the size of the given instance of DOMINATING SET.

If there is a dominating set of size at most k in the given instance of DOMINATING SET, the player can exchange the vertices in that dominating set with at most k of the identical agents to obtain the complete set of vertex-facts for G and hence achieve the goal. Conversely, as the player can interact with each of the identical agents at most once to trade a vertex-item for its associated neighbourhood-set of vertex-facts in G, any set of at most $k+1$ interactions between the player and the agents that achieves the goal in the constructed instance of APE must correspond to a set of at most k vertices that form a dominating set in G.

To complete the proof, note that in the constructed instance of APE, $i_A = i_I = 1$, $i_G = 0$, $f_G = 1$, and $|A| = t = k+1$. ∎

Lemma 5: DOMINATING SET polynomial-time many-one reduces to APE such that in the constructed instance of APE, $i_A = i_I = 1$, $f_I = |I| = 2$, $i_G = 0$, and $f_G = 1$, and $|A|$ is a function of k in the given instance of DOMINATING SET.

Proof (sketch): Modify the instance of APE constructed in Lemma 4 as follows: (1) Replace all $|V|$ transitions in each identical agent with a transition-tree rooted at q_0 consisting of a $|V|$-;length "spine" of transitions, each of which is enabled by a move-fact, with $|V|$ branches off this spine, where each branch is a $|V|$-length chain of transitions which are initially enabled by item v_i and deliver (one at a time) the vertex-facts corresponding to the neighbourhood vertex-facts for v_i before terminating at q_i; (2) Replace the single transition in the final agent with a $|V|$-length chain of transitions that are enabled by the individual vertex-facts in G before terminating in q_1 and the final exchange of the completion-fact; (3) make the move fact the initial fact-set for the player; and (4) set $t = (k \times 2|V|) + |V| = (2k+1)|V|$. The proof of correctness is a modification of that given in Lemma 4. Note that in the instance of APE described above, $i_A = i_I = 1$, $f_I = |I| = 2$, $i_G = 0$, $f_G = 1$, and $|A| = k+1$. ∎

Lemma 6: CLIQUE polynomial-time many-one reduces to APE such that in the constructed instance of APE, $i_A = i_I = 1$, $i_I = f_I = 2$, $u_G = 1$, and $f_G = 0$, and $|A|$, f_P, and t are all functions of k in the given instance of CLIQUE.

Proof: Given an instance $\langle G = (V, E), k \rangle$ of CLIQUE, construct an instance of APE consisting of two groups of k and $k(k-1)/2$ agents, respectively. The agents in the first

group are the vertex-selection agents from Lemma 4 modified so that agent i exchanges vertex-item v for vertex/position-fact v/i. Each of the agents in the second group corresponds to a distinct pair i, j, $1 \leq i < j \leq k$ which checks if the vertices selected in positions i and j have an edge between them in G. For the lth pair, $1 \leq l \leq k(k-1)/2$, this is done using two states q_0 and q_1 and $2|E|$ transition between q_0 and q_1 which, for each edge $(u, v) \in E$, trade items $u/i, v/j$ $(v/i, u/j)$ and fact $echk_{(l-1)}$ for items $u/i, v/j$ $(v/i, u/j)$ and fact $echk_l$, respectively. Each vertex-selection agent i starts with no items and the entire vertex/position i-fact-set, each edge-check agent l starts with no items and edge-check fact $echk_l$, and the player starts with the entire vertex-item-set for G and edge-check fact $echk_0$. Finally, the goal is $c_{k(k-1)/2}$ and $t = k + k(k-1)/2$. Note that the instance of APE described above can be constructed in time polynomial in the size of the given instance of CLIQUE.

If there is a clique of size at k in the given instance of CLIQUE, the player can exchange the vertices in that clique with the vertex/position agents in any order to obtain a "sequence" of vertex/position facts that will satisfy the edge-check agents and hence achieve the goal. Conversely, as the player can interact with each of the vertex-selection agents at most once to trade a vertex-item for its associated vertex/position fact, any set of at most $k + k(k-1)/2$ interactions between the player and the agents that achieves the goal in the constructed instance of APE must correspond to a set of k vertices that form a clique in G.

To complete the proof, note that in the constructed instance of APE, $i_A = 1$, $i_I = f_I = 2$, $i_G = 0$, $f_G = 1$, and $|A| = k + k(k-1)/2$, $f_P = k(k-1)/2 + 1$, and $t = k + k(k-1)/2$. ■

Lemma 7: CLIQUE polynomial-time many-one reduces to APE such that in the constructed instance of APE, $|A| = 1$, $i_I = 1$, $f_I = 2$, $i_0 = 1$, $f_G = 1$, and $|Q|$, f_P, and t are all functions of k in the given instance of CLIQUE.

Proof (sketch): Note that all of the agents in the reduction in Lemma 6 can be chained together in a single agent consisting of a chain of $1 + k + k(k-1)/2$ states, and that all edge-check facts except the last can be eliminated as they are no longer necessary, *e.g.*, the state q_1 for what was originally the $k(k-1)/2$st edge-check agent can only be reached if all other edge-checks are satisfied. The goal and value of t are unchanged. The proof of correctness of this reduction is a modification of that given in Lemma 6. Note that in the instance of APE described above, $|A| = 1$, $i_I = 1$, $f_I = 2$, $i_G = 0$, $f_G = 1$ $|Q| = 1 + k + k(k-1)/2$, $f_P = k$, and $t = k + k(k-1)/2$. ■

Observe that setting t to any specified value is actually unnecessary for the reductions in Lemmas 3 – 7 to work.

Result A *APE is NP-hard when $|A| = 1$.*

Proof: Follows from the NP-hardness of CLIQUE and the reduction in Lemma 7. ■

Result B *APE is fp-intractable for the parameter-set $\{i_A, f_A, i_I, f_I, |A|, |A|, i_P, f_P, t, i_G, f_G\}$.*

Proof: Follows from the $W[1]$-hardness of NTMC for parameter-set $\{k\}$ [26] and the reductions from NTMC to APE given in Lemma 3. ■

Result C *APE is fp-intractable for the parameter-set $\{|A|, i_A, i_I, t, i_G, f_G\}$.*

Proof: Follows from the $W[1]$-hardness of DOMINATING SET for parameter-set $\{k\}$ [9] and the reductions from DOMINATING SET to APE given in Lemma 4. ■

Result D *APE is fp-intractable for the parameter-set $\{|A|, i_A, i_I, f_I, |I|, i_G, f_G\}$.*

Proof: Follows from the $W[1]$-hardness of DOMINATING SET for parameter-set $\{k\}$ [9] and the reductions from DOMINATING SET to APE given in Lemma 5. ■

Result E *APE is fp-intractable for the parameter-set $\{|A|, i_A, i_I, f_I, f_P, t, i_G.f_G\}$.*

Proof: Follows from the $W[1]$-hardness of CLIQUE for parameter-set $\{k\}$ [9] and the reductions from CLIQUE to APE given in Lemma 6. ■

Result F *APE is fp-intractable for the parameter-set $\{|A|, i_I, f_I, |Q|, f_P, t, i_G, f_G\}$.*

Proof: Follows from the $W[1]$-hardness of CLIQUE for parameter-set $\{k\}$ [9] and the reductions from CLIQUE to APE given in Lemma 7. ■

Result G *APE is fp-tractable for the parameter-set $\|A|, |I|, t\}$.*

Proof: Consider the gamespace search tree whose nodes encode the current item- and fact-sets of each agent and the player as well as the current state of each agent. Observe that there are at most $|A| \times |I|$ possibilities for interactions relative to each node (as each agent's current state has at most $|I|$ enabled transitions outwards from that state). As we require that the goal be reachable within t agent-player interactions, the tree has at most $(|A||I|)^t$ nodes that must be considered. As each node can be generated and evaluated in time polynomial in the size of the given instance of APE, the above is an algorithm for APE whose runtime is fp-tractable for parameter-set $\{|A|, |I|, t\}$. ■