Fuzzy Pushdown Termination Games

Haiyu Pan, Fu Song, Yongzhi Cao, *Senior Member, IEEE*, and Junyan Qian

***Abstract*—The computational study on finite/infinite-state systems, probabilistic systems, and finite-state fuzzy systems, has received much attention recently. In contrast, there are very few results for algorithmic analysis of infinite-state fuzzy systems. In this paper, we introduce fuzzy pushdown termination games (FPDTGs), which are an extension of fuzzy pushdown automata with a game feature and can serve as a formal model of infinite-state fuzzy systems. We investigate some computational issues of the games under termination objectives for two players: the goal of player-1 is to maximize the truth value of eventually terminating at some given configurations with the empty stack, while player-2 aims at the opposite. Some interesting results are obtained. For example, we show that both players have optimal memoryless strategies and the same value. The problem of computing the value can be solved in exponential time when the triangular norm is chosen as the minimum one. Furthermore, we present efficient algorithms for computing the values of two special subclasses of FPDTGs. The potential for practical use of our model is demonstrated by a case study on a manufacturing system.**

***Index Terms*—Fuzzy automata, pushdown automata (PDAs), pushdown game, reachability problem, triangular norm.**

I. Introduction

Finite-state transition systems and more general models, such as *game graphs* [4], play a crucial role in the modeling and analysing of *finite-state* systems in various areas. In this paper, many different types of transition systems and game graphs have been adopted for *infinite-state* systems. For instance, *pushdown automata* (PDAs) [24] (i.e., *pushdown systems*) have been proposed as an adequate formalism to *sequential programs with (recursive) procedural calls*, and have been extensively studied in software verification community [8], [19], [22], [37]. Games over PDA, called *pushdown games*, are an appropriate model for the verification and synthesis of infinite-state open systems. The reachability and other properties of pushdown games have been studied in [10], [35], and [38].

Recently, transition systems and game graphs have been extended to model and analyze fuzzy systems. For finite-state fuzzy transition systems (FTSs), algorithms for various fuzzy temporal logics, such as fuzzy versions of LTL, CTL, and μ-calculus, have been presented in [27], [28], [30], [31], and [36], and algorithms for fuzzy (bi)simulations have been proposed in [13], [15], and [40]. More recently, Pan *et al.* [32] generalized (two-player) games on *finite* transition systems with *reachability objectives* to the fuzzy setting, called FRGs. They showed that FRGs are determined in the sense that for every state, the two players have the same value, there exist optimal memoryless strategies for both players, and there exists an effective algorithm for computing the values of FRGs.

On the other hand, a number of works have been done on the modeling and analyzing of *infinite-state* fuzzy systems. Cao *et al.* [11] considered bisimulation for infinite-state fuzzy systems by modeled them as FTSs. Wu and Deng [39] provided a logical characterization of fuzzy (bi)simulation for (infinite-state) FTSs. In [18], Du *et al.* proposed fuzzy hybrid machines to model fuzzy hybrid systems. *Ding et al.* [16] used differential Petri nets as a unified model to represent switched fuzzy systems, which are also infinite-state fuzzy systems. Fuzzy PDAs (FPDAs) are obtained by associating each transition rule with a value from some domain of weights, such as the unit interval [0, 1] [43], complete residuated lattices [42], orthomodular lattices [34], and type-00 lattices [41]. The relation between FPDAs and fuzzy context-free grammars [5], [6] has been established in [41] and [42]. Although the underlying semantics of these models are defined in *infinite-state* FTSs, the computational problems on these models are largely restricted to the *finite-state* cases.

In this paper, we investigate a game version of FPDAs under termination objectives, which is a continuation of [32]. There are a large number of results on termination problems for PDAs, pushdown games, and their probabilistic extensions, but the fuzzy extension has so far not been considered. Recall that a PDA consists of a finite set of *control states*, a finite set of *stack alphabet*, a finite set of *input alphabet*, and a finite set of *transition rules* of the form $pX \xrightarrow{a} q\alpha$, where p, q are control states, X is a stack symbol, a is an input symbol, and α is a finite sequence of stack symbols. In general, a PDA gives rise to an *infinite-state transition system* having *configurations* of the form

Manuscript received December 20, 2017; revised July 2, 2018; accepted August 31, 2018. Date of publication September 10, 2018; date of current version March 29, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61672023, Grant 61751210, Grant 61772035, Grant 61532019, Grant 61761136011, Grant 61673352, and Grant 61562015, in part by the Qing Lan Project, in part by Guangxi Natural Science Foundation of China under Grant 2015GXNSFDA139038 and Grant 2018GXNSFDA138003, in part by the High Level of Innovation Team of Colleges and Universities in Guangxi and Outstanding Scholars Program Funding, and in part by Guangxi Key Laboratory of Trusted Software under Grant kx201609. *(Corresponding author: Yongzhi Cao.)*

H. Pan and J. Qian are with the Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China (e-mail: phyu76@126.com; qjy2000@gmail.com).

F. Song is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: songfu@shanghaitech.edu.cn).

Y. Cao is with the Key Laboratory of High Confidence Software Technologies, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China, and also with the School of Mathematics and Statistics, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China (e-mail: caoyz@pku.edu.cn).

Digital Object Identifier 10.1109/TFUZZ.2018.2869127

(*control state, stack content*). In the verification community, one is not interested in the *languages* recognized by PDAs, but in the infinite-state transition systems they generate.

To define two-player games with termination objectives on *infinite-state* FTSs generated by FPDAs, we partition the configurations into player-1 set and player-2 set such that a player gets to choose the move when the current configuration belongs to its own partition. The obtained games are called *fuzzy pushdown termination games* (FPDTGs). Each FPDTG induces an associated *infinite-state* FRG, where the states are configurations of FPDTGs and the transition rules are determined in the natural way. A subclass of FPDTGs, called *fuzzy basic process algebra termination games* (FBPATGs), is obtained by considering singleton control state set.

We are interested in FPDTGs due to two reasons. First, it is impossible to computationally treat arbitrary infinite-state models, because in general one cannot handle infinite-state models as input. This is possible only if the infinite-state model has a finite representation, that is, each finite instance represents an underlying infinite-state system. Second, FPDTGs are of interest in themselves as a basic model that combines three very common modeling primitives: *fuzziness*, *recursion*, and *game*.

This paper focuses on studying the computational complexity of FPDTGs and their special cases mentioned above. Termination objectives are central to all of the more general analysis one might wish to perform on such models, such as games on FPDAs with winning conditions expressed in fuzzy temporal logics. More specifically, we consider FPDTGs and FBPATGs with a target set R consisting of some selected configurations with the *empty* stack where the goal of player 1 is to *maximize* the truth value of eventually terminating at R, while the goal of player 2 is to *minimize* this truth value. An FPDTG is called *maximizing* FPDTG (Max-FPDTG) if no configurations are assigned to *player 2*, while it is called *minimizing* FPDTG (Min-FPDTG) if no configurations are assigned to *player 1*. We also consider Max-FPDTGs and Min-FPDTGs with the target set R, where the objective of player 1 (player 2) is to maximize (minimize) the truth value of eventually terminating at R. In this paper, we try to generalize all the results on *finite-state* FRGs to the setting of FPDTGs and their special cases. Surprisingly, it turns out to be non-trivial to extend the results from *finite-state* FRGs to FPDTGs which are a kind of *infinite-state* FRGs.

We start by resolving the determinacy issue. We prove that FPDTGs are determined and have optimal memoryless strategies for two players. We show that the values of FPDTGs can be computed in exponential time when *triangular norm* [33] (*t*-norm) is chosen as the minimum one. It remains open whether there exists an algorithm for computing the values of FPDTGs for any other *t*-norms. As a compromise, we show that FPDTGs and Min-FPDTGs are linear-time reducible to each other. Then, for FBPATGs, we first give a *fixed-parameter polynomial time* (FPT) (w.r.t. the computational time of *t*-norms) algorithm for computing the values of FBPATGs. We prove that both players have optimal *stackless* and *memoryless* strategies for FBPATGs (see Section III for the detailed definition), which does not hold in general for FPDTGs. Finally, we provide an efficient algorithm for computing the values of Max-FPDTGs.

Related work: We give a summary of the existing results about algorithmic analysis of reachability/termination problems for some variants of PDAs.

Bouajjani *et al.* [8] and Finkel *et al.* [22] independently introduced a *polynomial-time* algorithm for reachability problem for PDAs. In [38], Walukiewicz showed that the reachability problem for pushdown games is EXPTIME-complete. By the way, Alur *et al.* in [2] presented recursive state machines (RSMs) to model sequential programs with recursive procedure calls. They showed that RSMs and PDAs are essentially equivalent, and can be translated in linear time to each other. Moreover, they investigated the reachability problem for RSMs in a somewhat different way.

Probabilistic extension of PDAs, probabilistic PDAs (PPDAs), are obtained by associating probabilities to transition rules of PDAs so that the total probabilities of all rules applicable to a given configuration is one. Esparza *et al.* [20] proved that the quantitative reachability problem for PPDAs can be solved in PSPACE. Etessami and Yannakakis in [21] examined the PPDA and probabilistic BPA (PBPA) games with termination objectives. For PBPA termination games, two players have optimal stackless and memoryless strategies. The quantitative termination problem for PBPA games can be decidable in PSPACE. However, for PPDA games, there do not necessarily exist optimal strategies for two players, and the quantitative termination problem is undecidable, even for maximizing or minimizing PPDA games.

For the convenience of the reader, we summarize the results for PDA and BPA games with termination objectives and their variants in Table I, where "–" denotes no, "?" denotes unknown, and "SM" denotes "stackless and memoryless."

Organization: After reviewing some basic facts on fuzzy sets and FRGs in Section II, we introduce the concept of FPDTGs in Section III and study the termination problem for FPDTGs in Section IV. Sections V and VI are devoted to efficient algorithms for computing the values of FBPATGs and Max-FPDTGs, respectively. We further discuss the potential applications of our results in Section VII and conclude the paper in Section VIII.

II. Fuzzy Reachability Games

In this section, we recap some basic notions of fuzzy sets and FRGs.

We denote by $\mathbb{N}$ the set of nonnegative integers and by $|S|$ the cardinality of a set S. Let Σ be a finite alphabet. The symbols Σ^* and Σ^ω denote the sets of all finite words and infinite words over Σ, respectively. Given a finite word $\sigma = a_0, \ldots, a_n \in \Sigma^*$, let $|\sigma|$ denote the length $n+1$ of σ and $\sigma(i)$ denote a_i for all $i : 0 \leq i \leq n$. The empty word is denoted by ϵ. Σ^+ stands for $\Sigma^* \backslash \{\epsilon\}$.

Let S be a universal set. A *fuzzy set* A in S is a function from S to $[0, 1]$ [33]. The fuzzy set A is *crisp* if $A(s) \in \{0, 1\}$ for all $s \in S$. The *support* $\text{supp}(A)$ of A is the set $\{s \in S : A(s) > 0\}$. We denote $\mathcal{F}(S)$ as the set of all fuzzy sets in S, $\mathcal{F}_f(S)$ as the set of all fuzzy sets with finite-support, and $\mathcal{P}(S)$ as the power set of S. A fuzzy set A with finite-support

TABLE I
SUMMARY OF QUANTITATIVE TERMINATION PROBLEMS IN PDA AND BPA GAMES AND THEIR VARIANTS

Model	Complexity	Optimal strategies
PDA termination games	EXPTIME-complete [38]	Memoryless [38]
BPA termination games	PTIME [11]	SM [11]
PPDA termination games	Undecidable [22]	-
PBPA termination games	PSPACE [22]	SM [22]
FPDTGs	?	Memoryless (Proposition 3)
FBPATGs	FPT w.r.t. the computational time of t-norms (Theorem 3)	SM (Theorem 4)

$\text{supp}(A) = \{s_1, \ldots, s_n\}$ can be written in Zadeh's notation as follows: $A = A(s_1)/s_1 + \cdots + A(s_n)/s_n$.

Given a set I of indices, the union $\bigcup_{i\in I} A_i$ of an arbitrary family $\{A_i : i \in I\}$ of fuzzy sets in S is a mapping from S into $[0, 1]$ defined by $(\bigcup_{i\in I} A_i)(s) = \sup_{i\in I} A_i(s)$, for all $s \in S$. For any $A, B \in \mathcal{F}(S)$, A is *contained* in B, denoted by $A \subseteq B$, if $A(s) \leq B(s)$ for all $s \in S$. Moreover, $A = B$ if both $A \subseteq B$ and $B \subseteq A$. We use $\emptyset$ to denote the *empty* fuzzy set with $\emptyset(s) = 0$ for all $s \in S$. For any $s \in S$, we write $\widehat{s}$ for the fuzzy set satisfying $\widehat{s}(s') = 1$ if $s' = s$, and 0 otherwise.

A binary operation $\otimes$ on $[0, 1]$ is called a *triangular norm* (*t*-norm) [25] if $\otimes$ is commutative, associative, monotone, and has 1 as its identity. A *t*-norm $\otimes$ is *continuous* if for all convergent sequences $(x_n)_{n\in\mathbb{N}}$, $(y_n)_{n\in\mathbb{N}}$, we have

$$\lim_{n\to\infty} x_n \otimes \lim_{n\to\infty} y_n = \lim_{n\to\infty}(x_n \otimes y_n). \tag{1}$$

A *t*-norm $\otimes$ is *left-continuous* if for each $y \in [0, 1]$ and for all nondecreasing sequences $(x_n)_{n\in\mathbb{N}}$, we have

$$\lim_{n\to\infty} x_n \otimes y = \lim_{n\to\infty}(x_n \otimes y).$$

Example 1: The four commonly used *t*-norms are: Minimum *t*-norm (also known as Gödel *t*-norm): $x \otimes y = \min\{x, y\}$; Łukasiewicz *t*-norm: $x \otimes y = \max\{x + y - 1, 0\}$; product *t*-norm: $x \otimes y = xy$; and nilpotent minimum *t*-norm [23]: $x \otimes y = \min\{x, y\}$ if $x + y > 1$ and 0 otherwise. The first three are all continuous, while the fourth is left-continuous.

Let $\otimes$ be a *t*-norm. For any $x \in [0, 1]$, we can inductively define the power operation of x as follows: $x^0 = 1$; $x^{n+1} = x^n \otimes x$. Then, for any subset D of $[0, 1]$, the *subalgebra* of $([0, 1], \otimes)$ generated by D is defined as follows:

$$\langle D\rangle = \{x_1^{n_1} \otimes \cdots \otimes x_k^{n_k} : x_1, \ldots, x_k \in D,\, n_1, \ldots, n_k \in \mathbb{N}\}.$$

We say that a *t*-norm $\otimes$ satisfies the *finite generated condition*, if for any finite subset D of $[0, 1]$, $\langle D\rangle$ is finite. For example, minimum, Łukasiewicz, and nilpotent minimum *t*-norms satisfy the finite generated condition, while product *t*-norm does not satisfy the condition [26].

Proposition 1 collects some properties of *t*-norms, which will be used in this paper.

Proposition 1: Let $x, y, z \in [0, 1]$, I be an index set, and $\{x_i\}_{i\in I}$, $\{y_i\}_{i\in I}$ be two sets of numbers in $[0, 1]$. The following properties hold for any *t*-norm:

1) If $\otimes$ is left-continuous, then $x \otimes \sup_{i\in I} y_i = \sup_{i\in I}(x \otimes y_i)$;
2) $x \otimes \min\{y, z\} = \min\{x \otimes y, x \otimes z\}$;
3) $\sup_{i\in I}(x_i \otimes y_i) \leq \sup_{i\in I} x_i \otimes \sup_{i\in I} y_i$;
4) $\inf_{i\in I} x_i \otimes \inf_{i\in I} y_i \leq \inf_{i\in I}(x_i \otimes y_i)$.

We now review the concept of FRGs [32] and related notions.

Let $i = 1, 2$. A (two-player) FRG is a tuple $G = (S, S_1, S_2, \Sigma, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$, where:

S is a countable nonempty set of states partitioned into *player-1* states S_1 and *player-2* states S_2;
Σ is a finite set of moves;
$\Delta_i : S_i \to \mathcal{P}(\Sigma)$ is a move assignment, which associates with each state $s \in S_i$ the nonempty set $\Delta_i(s) \subseteq \Sigma$ of moves available to player i at state s;
$\delta_i : S_i \times \Sigma \to \mathcal{F}(S)$ is a partial *transition* function, where for any $s \in S_i$, $\delta_i(s, a)(t)$ is the truth value of the transition from s to t when player i chooses the move a from $\Delta_i(s)$; and
$R \subseteq S$ is a set of target states.

We assume that for any $s \in S_i$ and $a \in \Delta_i(s)$, $\text{supp}(\delta_i(s, a)) \neq \emptyset$. An FRG is *finitely branching* if for any state $s \in S_i$ and $a \in \Delta_i(s)$, $\text{supp}(\delta_i(s, a))$ is finite. An FRG is *finite* if S is finite. Clearly, a finite FRG is finitely branching.

An FRG G is played by two players, 1 and 2, who select the moves in the states of S_1 and S_2, respectively. A *strategy* for player i in G is a function $\pi : S^* \cdot S_i \to \Sigma$, such that $\pi(\sigma s) \in \Delta_i(s)$ for any $\sigma \in S^*$ and $s \in S_i$. The strategy π for player i is *memoryless* if $\pi(\sigma s) = \pi(s)$ for any $\sigma \in S^*$ and $s \in S_i$. As a result, a memoryless strategy π for player i is equivalent to a function $S_i \to \Sigma$, and thus, we usually express a memoryless strategy as such a function. We denote by Π_i the set of all strategies for player i and by Π_i' the set of all memoryless strategies for player i.

Given a starting state s, two strategies π_1 and π_2 for the players of the game G yield a *play*, denoted by $G_s^{\pi_1,\pi_2}$, which is also an FRG such that there is exactly one available move for each player at any state.

A *path* in $G_s^{\pi_1,\pi_2}$ is an infinite sequence $\sigma = s_0 \xrightarrow{a_0|x_0} s_1 \xrightarrow{a_1|x_1} \cdots$ where $s_0 = s$, and for $i = 1, 2$ and $k \in \mathbb{N}$, if $s_k \in S_i$, then $\pi_i(s_0 s_1 \cdots s_k) = a_k$ and $\delta_i(s_k, a_k)(s_{k+1}) = x_k$. We denote by $\text{Paths}(G_s^{\pi_1,\pi_2})$ the set of all infinite paths in $G_s^{\pi_1,\pi_2}$. The *value* of a path σ (reaching some target states in R), denoted by $G_s^{\pi_1,\pi_2}(\sigma)$, is defined as follows:

$$G_s^{\pi_1,\pi_2}(\sigma) = \begin{cases} 1, & \text{if } s_0 \in R \\ \sup\{x_0 \otimes \cdots \otimes x_{n-1} : s_n \in R\}, & \text{otherwise.} \end{cases}$$

The *value* of the play $G_s^{\pi_1,\pi_2}$, denoted by $T(G_s^{\pi_1,\pi_2})$, is the supremum of the values of all the paths in $G_s^{\pi_1,\pi_2}$, namely,

$\sup\{G_s^{\pi_1,\pi_2}(\sigma) : \sigma \in \text{Paths}(G_s^{\pi_1,\pi_2})\}$. The goal of player 1 is to *maximize* the value of plays, while the goal of player 2 is to *minimize* the value of plays.

Value functions V_1^G, $V_2^G : S \to [0,1]$ for players 1 and 2, respectively, are defined as follows: for every state $s \in S$

$$V_1^G(s) = \sup_{\pi_1\in\Pi_1} \inf_{\pi_2\in\Pi_2} T(G_s^{\pi_1,\pi_2})$$

$$V_2^G(s) = \inf_{\pi_2\in\Pi_2} \sup_{\pi_1\in\Pi_1} T(G_s^{\pi_1,\pi_2}).$$

The FRG G is called *determined* if $V_1^G(s) = V_2^G(s)$ for every state $s \in S$. In this case, the value V_1^G (also V_2^G) is referred to as the *value* of G, denoted by V^G. In the following, if the FRG in question is understood, G is dropped from the functions V_1^G, V_2^G, and V^G. We say that the strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$ are *optimal* for the respective player, if for all $s \in S$

$$V_1(s) = \inf_{\pi_2'\in\Pi_2} T(G_s^{\pi_1,\pi_2'}),\ V_2(s) = \sup_{\pi_1'\in\Pi_1} T(G_s^{\pi_1',\pi_2}).$$

Given a state $s \in S$ and a number $n \in \mathbb{N}$, let $T_n(G_s^{\pi_1,\pi_2})$ denote the value $T(G_s^{\pi_1,\pi_2})$ in $G_s^{\pi_1,\pi_2}$ where only paths $\sigma = s_0 \xrightarrow{a_0|x_0} s_1 \xrightarrow{a_1|x_1} \cdots s_k \xrightarrow{a_k|x_k} s_{k+1}$ with $k < n$ are considered. We write $V_1^n(s)$ for $\sup_{\pi_1\in\Pi_1}\inf_{\pi_2\in\Pi_2} T_n(G_s^{\pi_1,\pi_2})$. Hence, $T(G_s^{\pi_1,\pi_2}) = \sup_{n\in\mathbb{N}} T_n(G_s^{\pi_1,\pi_2})$ and $V_1(s) = \sup_{n\in\mathbb{N}} V_1^n(s)$. $V_2^n(s)$ and $V^n(s)$ are defined in a similar way.

The following gives the main result on finite FRGs.

Proposition 2 ([32]): Let G be a finite FRG. Then, G is determined and there exist optimal memoryless strategies for two players.

III. Determinacy of FPDTGss

In this section, we first introduce FPDTGs, then generalize Proposition 2 to the setting of FPDTGs. From now on, the *t*-norms $\otimes$ considered are required to satisfy the following two conditions: *continuous* and *finite generated conditions*. We start by recalling *fuzzy pushdown automata* (FPDAs) without ϵ-transition, and the details are referred to [34], [41]–[43].

An FPDA without ϵ-transition (that accepts a language by the *empty stack*) is a tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, where:

Q is a finite nonempty set of control states;
Σ and Γ are two finite sets called the input and stack alphabets, respectively;
$q_0 \in Q$ is the initial state;
$Z_0 \in \Gamma$ is a special stack symbol called the start symbol; and
$\delta : Q \times \Sigma \times \Gamma \to \mathcal{F}_f(Q \times \Gamma^*)$ is a set of transition rules.

In the rest of this paper, we shall use the following:
1) X, Y, Z or X_i, Y_i, Z_i for elements of Γ;
2) α or α_i for a word in Γ^*;
3) $\Gamma^{\le n}$ for $\{\alpha : \alpha \in \Gamma^*, |\alpha| \le n\}$ and Γ^n for $\{\alpha : \alpha \in \Gamma^*, |\alpha| = n\}$;
4) Γ_ϵ for $\Gamma \cup \{\epsilon\}$; (5) $p\alpha$ for the tuple (p, α) of $Q \times \Gamma^*$.

We sometimes write p instead of $p\epsilon$ when the context is clear.

A *configuration* of P is an element of $Q \times \Gamma^*$. The head of a configuration $pX\alpha$ is pX. For every $\alpha' \in \Gamma^*$, the *value* of a transition from the configuration $pZ\alpha'$ to the configuration $q\alpha\alpha'$ is $\delta(p, a, Z)(q, \alpha)$. The degree to which P accepts an input string is the truth of the proposition "P can consume the input string and at the same time empty its stack."

Now we give the definition of FPDTGs, which is a fuzzy correspondence of probabilistic pushdown termination games (PDTGs) [9]. Intuitively, the configuration graph of an FPDA is an *infinite-state* fuzzy transition system (FTS), by assigning each configuration of the graph to one of two players, we obtain a finitely branching FRG.

Definition 1: An FPDTG is a tuple $P = (Q, \Sigma, \Gamma, H_1, H_2, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$, where Q, Σ, Γ are the same as in FPDA defined above, and the other components are defined as follows:
1) H_1 and H_2 are a partition of $Q \times \Gamma$;
2) for $i = 1, 2$, $\Delta_i : H_i \to \mathcal{P}(\Sigma)$ is a move assignment;
3) for $i = 1, 2$, $\delta_i : H_i \times \Sigma \to \mathcal{F}_f(Q \times \Gamma^*)$ is a partial transition function;
4) $R \subseteq Q$ is a set of target control states, called *termination objective*.

For $i = 1, 2$, we assume that for all $pX \in H_i$, $\Delta_i(pX) \ne \emptyset$, and if $a \in \Delta_i(pX)$, then $\text{supp}(\delta_i(pX, a)) \ne \emptyset$. An FPDTG is called PDTG if for every $pX \in H_i$, $a \in \Delta_i(pX)$, $\delta_i(pX, a)$ is crisp. An FPDTG is *simple* if for every $pX \in H_i$, $a \in \Delta_i(pX)$, $|\text{supp}(\delta_i(pX, a))| = 1$. We say that an FPDTG is a *simple PDTG* if it is both a PDTG and simple. A FBPATG is an FPDTG with $|Q| = 1$. In this case, we sometimes omit the control state from the representation of a configuration (for example, we write just α instead of $p\alpha$). An FPDTG where $H_2 = \emptyset$ (resp. $H_1 = \emptyset$) is called a *maximizing* (resp. *minimizing*) FPDTG. We denote by Max-FPDTGs (resp. Min-FPDTGs) the maximizing (resp. minimizing) FPDTGs. We also denote by Max-FBPATGs the maximizing FPDTGs, which are also FBPATGs. We often omit (Q), $(H_1, H_2, \Delta_2, \delta_2)$ and $(H_1, H_2, \Delta_1, \delta_1)$ in the descriptions of FBPATGs, Max-FPDTGs, and Min-FPDTGs, respectively. For the sake of simplicity, we also write Δ, δ instead of Δ_1, δ_1 (resp. Δ_2, δ_2) in the Max-FPDTGs (resp. Min-FPDTGs).

Given an FPDTG $P = (Q, \Sigma, \Gamma, H_1, H_2, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$, the finitely-branching FRG yielded by P is $G_P = (Q\Gamma^*, H_1\Gamma^* \cup Q\{\epsilon\}, H_2\Gamma^*, \Sigma, \Delta_1^P, \Delta_2^P, \delta_1^P, \delta_2^P, R\{\epsilon\})$, where for $i = 1, 2$, and $\alpha \in \Gamma^*$,
1) $\Delta_i^P(pX\alpha) = \Delta_i(pX)$ if $pX \in H_i$; and $\Delta_1^P(p\epsilon) = \{a\}$ if $p \in Q$ and $a \in \Sigma$;
2) $\delta_i^P(pX\alpha, a)(q\beta\alpha) = \delta_i(pX, a)(q\beta)$ if $pX \in H_i$, $a \in \Delta_i(pX)$, and $q\beta \in Q\Gamma^*$;
3) $\delta_1^P(p\epsilon, a) = \widehat{p\epsilon}$.

Let us illustrate the above-mentioned notions by the following example.

Example 2: Consider the FPDTG $P = (Q, \Sigma, \Gamma, H_1, H_2, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$, where $Q = \{p, q\}$, $\Sigma = \{a, b\}$, $\Gamma = \{Z\}$, $H_1 = \{pZ\}$, $H_2 = \{qZ\}$, $\Delta_1(pZ) = \Delta_2(qZ) = \{a, b\}$, $R = \{q\}$, and δ_1 and δ_2 are defined as follows:

$$\delta_1(pZ, a) = 0.8/pZZ + 0.6/pZ, \quad \delta_1(pZ, b) = 0.9/qZ$$

$$\delta_2(qZ, a) = 0.8/pZZ + 0.8/q, \quad \delta_2(qZ, b) = 0.9/q.$$

The fragment of the finitely-branching FRG G_P generated by P is shown in Fig. 1, where the vertices shaped as circles are player-1 states and the vertices shaped as boxes are player-2 states. For any $s \in H_1\Gamma^* \cup Q \times \{\epsilon\}$ (resp. $s \in H_2\Gamma^*$), $a \in$

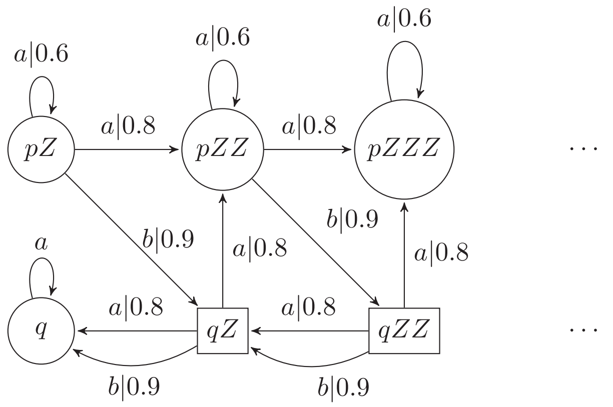


Fig. 1. Fragment of G_P.

$\Gamma_i(s)$, and $t \in \text{supp}(\delta_i^P(s,a))$, we draw an edge from s to t labeled by $a|\delta_i^P(s,a)(t)$, where $i=1,2$. When $\delta_i^P(s,a)(t)=1$, we simply label the edge by a instead of $a|1$.

Based on FPDTG P, we can construct a PDTG $P_1 = (Q_1, \Sigma_1, \Gamma_1, H_{1,1}, H_{1,2}, \Delta_{1,1}, \Delta_{1,2}, \delta_{1,1}, \delta_{1,2}, R_1)$ where $Q_1 = Q$, $\Sigma_1 = \Sigma$, $\Gamma_1 = \Gamma$, $H_{1,1} = H_1$, $H_{1,2} = H_2$, $\Delta_{1,1} = \Delta_1$, $\Delta_{1,2} = \Delta_2$, $R_1 = R$, and $\delta_{1,1}$ and $\delta_{1,2}$ are defined as follows:

$$\delta_{1,1}(pZ, a) = 1/pZZ + 1/pZ, \quad \delta_{1,1}(pZ, b) = 1/qZ$$
$$\delta_{1,2}(qZ, a) = 1/pZZ + 1/q, \quad \delta_{1,2}(qZ, b) = 1/q.$$

Consider FPDTG $P_2 = (Q_2, \Sigma_2, \Gamma_2, H_{2,1}, H_{2,2}, \Delta_{2,1}, \Delta_{2,2}, \delta_{2,1}, \delta_{2,2}, R_2)$, where $Q_2 = Q$, $\Sigma_2 = \Sigma$, $\Gamma_2 = \Gamma$, $H_{2,1} = H_1$, $H_{2,2} = H_2$, $\Delta_{2,1} = \Delta_1$, $\Delta_{2,2} = \Delta_2$, $R_2 = R$, and $\delta_{2,1}$ and $\delta_{2,2}$ are defined as follows:

$$\delta_{2,1}(pZ, a) = 0.8/pZZ, \quad \delta_{2,1}(pZ, b) = 0.9/qZ$$
$$\delta_{2,2}(qZ, a) = 0.8/pZZ, \quad \delta_{2,2}(qZ, b) = 0.9/q.$$

By definition, FPDTG P_2 is simple.

Consider FPDTG $P_3 = (Q_3, \Sigma_3, \Gamma_3, H_{3,1}, H_{3,2}, \Delta_{3,1}, \Delta_{3,2}, \delta_{3,1}, \delta_{3,2}, R_3)$, where $Q_3 = \{p\}$, $\Sigma_3 = \Sigma$, $\Gamma_3 = \{Y, Z\}$, $H_{3,1} = \{pY\}$, $H_{3,2} = \{pZ\}$, $\Delta_{3,1}(pY) = \Delta_{3,2}(pZ) = \Gamma_3$, $R_3 = \{p\}$, and $\delta_{3,1}$ and $\delta_{3,2}$ are defined as follows:

$$\delta_{3,1}(pY, a) = 0.8/pZZ, \quad \delta_{3,1}(pY, b) = 0.9/pZ$$
$$\delta_{3,2}(pZ, a) = 0.8/pZZ, \quad \delta_{3,2}(pZ, b) = 0.9/p.$$

By definition, FPDTG P_3 is an FBPATG.

Consider FPDTG $P_4 = (Q_4, \Sigma_4, \Gamma_4, H_{4,1}, H_{4,2}, \Delta_{4,1}, \Delta_{4,2}, \delta_{4,1}, \delta_{4,2}, R_4)$, where $Q_4 = Q$, $\Sigma_4 = \Sigma$, $\Gamma_4 = \Gamma$, $H_{4,1} = H_1 \cup H_2$, $H_{4,2} = \emptyset$, $\Delta_{4,1} = \Delta_1$, $\Delta_{4,2} = \emptyset$, $\delta_{4,2} = \emptyset$, $R_4 = R$, and $\delta_{4,1}$ is defined as follows:

$$\delta_{4,1}(pZ, a) = 0.8/pZZ + 0.6/pZ, \quad \delta_{4,1}(pZ, b) = 0.9/qZ$$
$$\delta_{4,1}(qZ, a) = 0.8/pZZ + 0.8/q, \quad \delta_{4,1}(qZ, b) = 0.9/q.$$

Clearly, P_4 is a Max-FPDTG. Similarly, we can also define a min-FPDTG $P_5 = (Q_5, \Sigma_5, \Gamma_5, H_{5,1}, H_{5,2}, \Delta_{5,1}, \Delta_{5,2}, \delta_{5,1}, \delta_{5,2}, R_5)$, where $Q_5 = Q$, $\Sigma_5 = \Sigma$, $\Gamma_5 = \Gamma$, $H_{5,1} = \emptyset$, $H_{5,2} = H_1 \cup H_2$, $\Delta_{5,1} = \emptyset$, $\Delta_{5,2} = \Delta_1$, $\delta_{5,5} = \emptyset$, $R_5 = R$, and $\delta_{5,1}$ is defined as follows:

$$\delta_{5,1}(pZ, a) = 0.8/pZZ + 0.6/pZ, \quad \delta_{5,1}(pZ, b) = 0.9/qZ$$
$$\delta_{5,1}(qZ, a) = 0.8/pZZ + 0.8/q, \quad \delta_{5,1}(qZ, b) = 0.9/q.$$

Realize that all of the previously introduced game-theoretic notions (strategies, values, etc.) apply to G_P, not directly to P. For finitely-branching FRGs generated from FPDTGs, we can define a special type of memoryless strategies: A strategy π is a *stackless* and *memoryless* strategy (SM-strategy) of player i if π is a memoryless strategy of player i such that $\pi(pX\alpha) = \pi(pX)$ for every $pX \in H_i$ and $\alpha \in \Gamma^*$. It follows from the following theorem that FPDTGs are determined and there exist memoryless strategies for two players. In the following, we denote Im(P) by $\{\delta_i(pX, a)(q\alpha) : pX \in H_i, a \in \Delta_i(pX), q\alpha \in Q\Gamma^*, i = 1, 2.\}$.

Theorem 1: Let $G = (S, S_1, S_2, \Sigma, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$ be a finitely-branching FRG. Define a mapping $F : \mathcal{F}(S) \longrightarrow \mathcal{F}(S)$ by

$$F(A)(s) = \begin{cases} 1, & \text{if } s \in R \\ \sup\limits_{a\in\Delta_1(s)} \sup\limits_{t\in S}(\delta_1(s,a)(t) \otimes A(t)), & \text{if } s \in S_1 \backslash R \\ \inf\limits_{b\in\Delta_2(s)} \sup\limits_{t\in S}(\delta_2(s,b)(t) \otimes A(t)), & \text{otherwise} \end{cases}$$

for any $A \in \mathcal{F}(S)$. Then, the following properties hold.

1) G is determined, i.e., for all states $s \in S$, $V_1(s) = V_2(s)$.
2) There exists an optimal memoryless strategy for player 2 in every state.
3) An optimal strategy for player 1 does not necessarily exist.
4) If G is generated by some FPDTG P, then there exists an optimal memoryless strategy for player 1 in every state.
5) $V = F(V)$.

Proof: See the Appendix. ■

Remark 1:

1) Recall that we have assumed that the *t*-norm in Theorem 1 satisfies the following two conditions: continuous and finite generated conditions. In fact, we can obtain some more general results from the proof of the above-mentioned theorem: The assertions (1)–(3) hold for any *t*-norm, (4) holds for any *t*-norm satisfying the finite generated condition, and (5) holds for any left-continuous *t*-norm.
2) Given an FRG G and $r \in [0, 1]$, we say that player 1 in state s has a r-winning strategy if there exists $\pi_1 \in \Pi_1$ such that for all $\pi_2 \in \Pi_2$, $T(G_s^{\pi_1,\pi_2}) \geq r$; and player 2 in state s has a r-winning strategy if there exists $\pi_2 \in \Pi_2$ such that for all $\pi_1 \in \Pi_1$, $T(G_s^{\pi_1,\pi_2}) < r$. It follows from Theorem 1 that for FPDTG P, either player 1 has a r-winning strategy, or player 2 has a r-winning strategy.

With a slight abuse of notation, we write V^P instead of V^{G_P}. In the following, we often just write V, if the underlying system P is clear. Given an FPDTG, FBPATG, or Max/Min-FPDTG P, a configuration $p\alpha \in Q\Gamma^+$, in this paper, we consider the following questions.

1) The *quantitative* termination problem: Given $r \in [0, 1]$, is $V(p\alpha) \geq r$?
2) The termination problem: How to compute $V(p\alpha)$?

Without loss of generality, we assume, henceforth, that for every $\delta_i(pX, a)(q\alpha) = x$ and $x > 0$, $|\alpha| \leq 2$. Note that this condition does not restrict the expressiveness of the model, as every FPDTG can be transformed into an equivalent one in linear time such that the above-mentioned condition holds. The similar transformation method can be found in [20].

IV. Termination Values in FPDTGs

In this section, we show that there exists an effective algorithm to compute the value of a given FPDTG over the minimum *t*-norm.

For complexity analysis, we assume that operations on $[0, 1]$ (comparison, addition, and multiplication) can be performed in constant time; moreover, $T_\otimes$ denotes the worst computational cost of $\otimes$. We make the assumption that the computation time of $\otimes$ is bounded and that $T_\otimes$ corresponds to an upper bound. Based on the assumption, the computation costs of the minimum, nilpotent minimum, and Łukasiewicz *t*-norms can be both performed in constant time.

In the rest of this section, let us fix an FPDTG $P = (Q, \Sigma, \Gamma, H_1, H_2, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$. We define the *size* of FPDTG P as $|P| = \sum_{i=1}^{2} \sum_{pX \in H_i} \sum_{a \in \Delta_i(pX)} |\text{supp}(\delta_i(pX, a))|$. For the complexity of results in this paper, we assume that all elements in $\text{Im}(P)$ are all *rational*.

We first study the (quantitative) termination problem for FPDTGs and variants thereof. We start with the quantitative termination problem for simple PDTGs, which has a close relation with the reachability problem for pushdown game systems studied by Walukiewicz [38].

Lemma 1: The quantitative termination problem for simple PDTGs is EXPTIME-complete.

Proof: Recall that in [38], a pushdown game is defined as a tuple $(Q, Q_1, Q_2, \Gamma, \delta)$, where $Q = Q_1 \uplus Q_2$ is a finite set of control states, Q_i indicates the game position of player i ($i = 1, 2$), Γ is a finite stack alphabet, and $\delta \subseteq Q \times \Gamma \times Q \times \Gamma^{\leq 2}$ is a finite set of (unlabelled) transition rules. The reachability problem for pushdown games is EXPTIME-complete. Hence, the quantitative termination problem for pushdown games is EXPTIME-complete, since the termination objective is a kind of reachability objective.

Clearly, pushdown games with termination objectives are a special kind of simple PDTGs. Hence, the quantitative termination problem for simple PDTGs is EXPTIME-hard. By applying the saturation procedure of [10], the termination problem of simple PDTGs can be solved in exponential time. The result immediately follows. ■

The following lemma shows that FPDTGs can be transformed into simple FPDTGs, which preserve the expected property:

Lemma 2: There is a linear-time reduction from an FPDTG P to a simple FPDTG P' such that for all $p\alpha \in Q\Gamma^*$, $V^P(p\alpha) = V^{P'}(p\alpha)$.

Proof: Suppose $i = 1, 2$. For simplicity, we assume that $|\Sigma| \geq \sum_{a \in \Delta_i(pX)} |\text{supp}(\delta_i(pX, a))|$ for all $pX \in H_i$. We define an FPDTG $P' = (Q', \Sigma, \Gamma, H_1', H_2', \Delta_1', \Delta_2', \delta_1', \delta_2', R)$, where:

$Q' = Q \cup \{q_{[pXa]} : pX \in H_i, a \in \Delta_i(pX), i = 1, 2\}$;

$H_1' = H_1 \cup (Q' \backslash Q)\Gamma$;

$H_2' = H_2$;

$\Delta_i'(pX) = \Delta_i(pX)$ for every $pX \in H_i$. If $a \in \Delta_i(pX)$ and $\text{supp}(\delta_i(pX, a)) = \{q_1\alpha_1, \ldots, q_n\alpha_n\}$, then $\delta_i'(pX, a) = \widehat{q_{[pXa]}X}$, $\Delta_1'(q_{[pXa]}X) = \{a_j : a_j \in \Sigma, 1 \leq j \leq n\}$, and $\delta_1'(q_{[pXa]}X, a_j) = \delta_i(pX, a)(q_j\alpha_j)/q_j\alpha_j$, $1 \leq j \leq n$.

Clearly, P' is a simple FPDTG and it takes linear time to construct P' from P. By Theorem 1, we conclude that for any $p\alpha \in Q\Gamma^*$, $V^P(p\alpha) = V^{P'}(p\alpha)$. Hence, this reduction works. ■

A consequence of the above-mentioned lemmas is the following.

Corollary 1: The quantitative termination problem for PDTGs is EXPTIME-complete.

Example 3: Consider the FPDTG P in Example 2. Using the construction method in the proof of Lemma 2, we obtain a simple FPDTG $P' = (Q', \Sigma, \Gamma, H_1', H_2', \Delta_1', \Delta_2', \delta_1', \delta_2', R)$, where δ_1' and δ_2' are defined as follows:

$$\delta_1'(pZ, a) = \widehat{q_{[pZa]}Z}, \qquad \delta_1'(q_{[pZa]}Z, a) = 0.8/pZZ$$
$$\delta_1'(q_{[pZa]}Z, b) = 0.6/pZ, \qquad \delta_1'(pZ, b) = \widehat{q_{[pZb]}Z}$$
$$\delta_1'(q_{[pZb]}Z, a) = 0.9/qZ, \qquad \delta_2'(qZ, a) = \widehat{q_{[qZa]}Z}$$
$$\delta_1'(q_{[qZa]}Z, a) = 0.8/pZZ, \qquad \delta_1'(q_{[pZa]}Z, b) = 0.8/q$$
$$\delta_2'(qZ, b) = \widehat{q_{[qZb]}Z}, \qquad \delta_1'(q_{[qZb]}Z, b) = 0.9/q.$$

We now address the termination problem for FPDTGs where the *t*-norm is chosen as the minimum *t*-norm.

Theorem 2: The termination problem for FPDTGs with *t*-norm chosen as the minimum *t*-norm can be solved in exponential time.

Proof: See the Appendix. ■

Following Corollary 1 and Theorem 2, we have:

Corollary 2: The quantitative termination problem for FPDTGs with *t*-norm chosen as the minimum *t*-norm is EXPTIME-complete.

Remark 2: Notice that Corollary 2 only works for FPDTGs with *t*-norm chosen as the minimum *t*-norm. It is open whether the quantitative termination problem for FPDTGs with other *t*-norms is decidable or not. So far, we can show that FPDTGs can be reduced to Min-FPDTGs. Moreover, all the considered problems of FBPATGs and Max-FPDTGs can be solved (cf. Sections V and VI).

Proposition 3: There is a linear-time reduction from an FPDTG P to a Min-FPDTG P' such that for all $p\alpha \in Q\Gamma^*$, $V^P(p\alpha) = V^{P'}(p\alpha)$.

Proof: Consider Min-FPDTG $P' = (Q, \Sigma, \Gamma, \Delta', \delta', R)$, where Δ' and δ' are defined by:

1) If $pX \in H_1$, then $\Delta'(pX) = \{a\}(a \in \Sigma)$, and $\delta'(pX, a) = \bigcup_{b \in \Delta_1(pX)} \delta_1(pX, b)$;
2) If $pX \in H_2$ and $a \in \Delta_2(pX)$, then $a \in \Delta'(pX)$, and $\delta'(pX, a) = \delta_2(pX, a)$.

By Theorem 1, we derive that for any $p\alpha \in Q\Gamma^*$, $V^P(p\alpha) = V^{P'}(p\alpha)$. ■

Example 4: Let us illustrate the reduction by considering the FPDTG P in Example 2. We obtain the corresponding Min-FPDTG $P' = (Q, \Sigma, \Gamma, \Delta', \delta', R)$, where δ' is defined as follows:

$$\delta'(pZ, a) = 0.8/pZZ + 0.6/pZ + 0.9/qZ$$
$$\delta'(qZ, a) = 0.8/pZZ + 0.8/q, \ \delta'(qZ, b) = 0.9/q.$$

Remark 3: It should be pointed out that Theorem 2 and Corollary 1 also hold for the nilpotent minimum *t*-norm, and the other results obtained in this section are applicable to any left-continuous *t*-norm.

V. Termination Values in FBPATGs

In this section, we investigate the termination problem for FBPATGs, an important subclass of FPDTGs. We present an algorithm for solving the termination problem, and then show that there exist optimal SM-strategies for two players.

Let us fix an FBPATG $P_b = (\Sigma, \Gamma, H_1, H_2, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$. Recall that there is only one control state in FBPATGs, we, therefore, omit the control state for simplicity. Then, $R = \{\epsilon\}$. We will write $X \xrightarrow{a|x} \alpha$ to indicate $\delta_i(X, a)(\alpha) = x$ in the rest of this section when it is clear from the context.

First, we give the following lemma, which will be used later.

Lemma 3: Let P_b be an FBPATG and $X \in \Gamma$, $\alpha \in \Gamma^+$. Then,

1) For every $n \in \mathbb{N}$, $V^{n+1}(X\alpha) \leq V^n(X) \otimes V^n(\alpha) \leq V^{2n}(X\alpha)$.
2) $V(\alpha) = V(\alpha(0)) \otimes \cdots \otimes V(\alpha(|\alpha| - 1))$ if $|\alpha| \geq 2$.

Proof: See the Appendix. ∎

We present a reduction of computing the above-mentioned $V(\alpha(i))$ to compute the *least fixed point* of a monotone system E_{P_b} of equations. Let $V|_{\Gamma_\epsilon}$ denote the *restriction* of V to Γ_ϵ. Let us use a variable $\mu(X)$ for each unknown $V(X)$, where $\mu \in \mathcal{F}(\Gamma_\epsilon)$. The system of equations E_{P_b} has one equation of the form $\mu(X) = F_X(\mu)$ for every $X \in \Gamma_\epsilon$. For simplicity, we denote the system of equations E_{P_b} by $\mu = F(\mu)$. We now identify a particular solution to $\mu = F(\mu)$, called the least fixed point solution, which gives precisely $V|_{\Gamma_\epsilon}$ of an FBPATG.

Lemma 4: Define $F^0(\mu) = \mu$, and $F^n(\mu) = F(F^{n-1}(\mu))$, for $n \geq 1$. Suppose P_b is an FBPATG and $\mu = F(\mu)$ is the system of equations E_{P_b} associated with P_b, which is defined as follows:

$$\mu(X) = \begin{cases} 1, & \text{if } X = \epsilon \\ \sup\limits_{a \in \Delta_1(X)} \max \left\{ \begin{array}{l} \sup\limits_{X \xrightarrow{a|x} \epsilon} x, \\ \sup\limits_{X \xrightarrow{a|x} Y} (x \otimes \mu(Y)), \\ \sup\limits_{X \xrightarrow{a|x} YZ} (x \otimes \mu(Y) \otimes \mu(Z)) \end{array} \right\}, & \text{if } X \in H_1 \\ \inf\limits_{a \in \Delta_2(X)} \max \left\{ \begin{array}{l} \sup\limits_{X \xrightarrow{a|x} \epsilon} x, \\ \sup\limits_{X \xrightarrow{a|x} Y} (x \otimes \mu(Y)), \\ \sup\limits_{X \xrightarrow{a|x} YZ} (x \otimes \mu(Y) \otimes \mu(Z)) \end{array} \right\}, & \text{otherwise.} \end{cases}$$

Algorithm 1 Computing $V^{P'}(\alpha)$.

```
   Input  : An FBPATG $P'$ and a configuration $\alpha \in \Gamma^+$
   Output: $V^{P'}(\alpha)$
1  foreach $X \in \Gamma'$ do
2      $v[X] \leftarrow 0$;
3      if $X \in H_2'$ then  $c[X] \leftarrow 0$;
4  $v[\epsilon] \leftarrow 1$;
5  $\mathcal{Q} \leftarrow \emptyset$;
6  foreach $X \in \Gamma_\epsilon'$ do
7      INSERT($\mathcal{Q}$, $X$)          //insert $X$ into queue $\mathcal{Q}$
8  while $\mathcal{Q} \neq \emptyset$ do
9      $Y \leftarrow$ EXTRACT-MAX($\mathcal{Q}$);
10     foreach $X \in \mathcal{Q}$ such that $Y \in Succ(X)$ do
11         if $X \in H_1'$ then
12             foreach $a \in \Delta_1'(X)$ and $Z \in \Gamma_\epsilon'$ such that
                       $X \xrightarrow{a|x} YZ$ or $X \xrightarrow{a|x} ZY$ do
13                 INCREASE($\mathcal{Q}$, $X$, $x \otimes v[Y] \otimes v[Z]$);
14         if $X \in H_2'$ then
15             $c[X] \leftarrow c[X] + |\{a \in \Delta_2'(X) : \delta_2'(X, a) = \widehat{Y}\}|$;
16             if $c[X] = |\Delta_2'(X)|$ then
17                 INCREASE($\mathcal{Q}$, $X$, $v[Y]$);
18 if $|\alpha| = 1$ then return ($v[\alpha]$);
19 else return ($v[\alpha_0] \otimes \cdots \otimes v[\alpha_{|\alpha|-1}]$);
```

Then, $V|_{\Gamma_\epsilon}$ is the least fixed point of the system of equations E_{P_b}. Moreover, there exists some $m \in \mathbb{N}$ such that $V|_{\Gamma_\epsilon} = F^m(\emptyset)$.

Proof: See the Appendix. ∎

Lemma 4 suggests that we can compute $V|_{\Gamma_\epsilon}$ by value iterations, which can be done in $O(|\Gamma| \cdot |\langle \mathrm{Im}(P_b) \rangle| \cdot |P_b| \cdot T_\otimes)$ time. Hence, by Lemma 3, we can compute the value $V(\alpha)$ of the FBPATG P_b in $O((|\Gamma| \cdot |\langle \mathrm{Im}(P_b) \rangle| \cdot |P_b| + |\alpha| - 1)T_\otimes)$ time.

Based on Lemmas 3 and 4, we provide a more efficient algorithm to compute V rather than using value iterations. Our algorithm consists of two steps. Taking an FBPATG P_b and $\alpha \in \Gamma^+$ as inputs, the first step transforms P_b into another FBPATG P' such that $V^{P_b}(\alpha) = V^{P'}(\alpha)$. In the second step, we use Algorithm 1 to compute $V^{P'}(\alpha)$.

Step 1: For the given FBPATG P_b, we construct an FBPATG $P' = (\Sigma, \Gamma', H_1', H_2', \Delta_1', \Delta_2', \delta_1', \delta_2', R)$ such that the following:

1) $\Gamma' = H_1' \cup H_2'$, $H_1' = H_1 \cup H_\Sigma$, and $H_2' = H_2$, where $H_\Sigma = \{X_a : X \in H_2, a \in \Delta_2(X)\}$;
2) $\Delta_1'(X) = \Delta_1(X)$ for all $X \in H_1$, $\Delta_1'(X_a) = \{a\}$ for all $X_a \in H_\Sigma$; and $\Delta_2' = \Delta_2$;
3) for all $X \in H_1$, $a \in \Delta_1(X)$, $\mathrm{supp}(\delta_1'(X, a)) = \mathrm{supp}(\delta_1(X, a))$, and $\delta_1'(X, a)(\alpha) = \delta_1(X, a)(\alpha)$ for any $\alpha \in \mathrm{supp}(\delta_1'(X, a))$; for all $X_a \in H_\Sigma$, $\delta_1'(X_a, a)(\alpha) = \delta_2(X, a)(\alpha)$ for any $\alpha \in \mathrm{supp}(\delta_1'(X_a, a))$ with $\mathrm{supp}(\delta_1'(X_a, a)) = \mathrm{supp}(\delta_2(X, a))$;
4) for all $X \in H_2'$ and $a \in \Delta_2'(X)$, $\delta_2'(X, a) = \widehat{X_a}$.

Step 2: Inspired by [1], [30], and [32], we use a max-priority queue data structure [14] to compute $V^{P'}(\alpha)$ by Algorithm 1.

Suppose $i = 1, 2$ and $X \in H'_i$. Let $\text{Succ}(X) = \{Y \in \Gamma'_\epsilon :$ there exist $a \in \Delta'_i(X)$ and $Z \in \Gamma'_\epsilon$ such that $ZY \in \text{supp}(\delta'_i(X, a))$ or $YZ \in \text{supp}(\delta'_i(X, a))\}$. Recall that $Y\epsilon = \epsilon Y = Y$ for any $Y \in \Gamma'_\epsilon$. We say that Y is a *successor* of X if $Y \in \text{Succ}(X)$.

Algorithm 1 uses three data structures that are: an array of counters $c[X]$ for each $X \in H'_2$, an array of values $v[X]$ for $X \in \Gamma'_\epsilon$, and a max-priority queue $\mathcal{Q}$ ordered by the value of $v[X]$. The counter $c[X]$ records the number processed moves of X ($X \in H'_2$) during the algorithm. $v[X]$ stores the value of X. The max-priority queue $\mathcal{Q}$ stores the stack elements whose values still need to be computed. Lines 1–7 initialize these three data structures.

Line 1 removes an element Y with the maximal priority in queue $\mathcal{Q}$. For each $X \in \mathcal{Q}$ such that Y is a successor of X, if $X \in H'_1$, the value $v[X]$ is increased to $x \otimes v[Y] \otimes V[Z]$ at line 1 if $v[X] < x \otimes v[Y] \otimes V[Z]$. Otherwise, if $X \in H'_2$, the number of moves $a : X \xrightarrow{a} Y$ is added onto the counter $c[X]$ at line 1. If $c[X]$ equals to $|\Delta'_2(X)|$ (i.e., all the successors of X have been dequeued from $\mathcal{Q}$), then assigns the value $v[Y]$ to $v[X]$ at line 1 if $v[Y] > v[X]$.

Lemma 5: Algorithm 1 computes $V(\alpha)$.

Proof: See the Appendix. ■

The time complexity of the above-mentioned algorithm for computing the value of a given FBPATG is as follows.

Theorem 3: Given an FBPATG P_b and a configuration $\alpha \in \Gamma^+$, the value $V(\alpha)$ can be computed in time $O\big((|P_b| + |\Gamma| \cdot |\Sigma| + |\alpha| - 1)T_\otimes + (|P_b| + |\Gamma| \cdot |\Sigma|)\log(|\Gamma| \cdot |\Sigma|)\big)$ and space $O(|\Gamma| \cdot |\Sigma|)$.

Proof: See the Appendix. ■

Since Step 1 only handles H_2, Δ_2, and δ_2, then for Max-FBPATGs, we get the following result:

Corollary 3: Given a Max-FBPATG P_b and a configuration $\alpha \in \Gamma^+$, the value $V(\alpha)$ can be computed in time $O((|P_b| + |\alpha| - 1)T_\otimes + |P_b| \log |\Gamma|)$ and space $O(|\Gamma|)$.

We identify a very restricted kind of strategy that suffices as an optimal strategy for FBPATGs.

Theorem 4: For every FBPATG, both players 1 and 2 have optimal SM-strategies.

Proof: See the Appendix. ■

Note that Theorem 4 cannot be extended to FPDTGs. We illustrate this by an example.

Example 5: Consider FPDTG $P = (Q, \Sigma, \Gamma, H_1, H_2, \Delta_1, \Delta_2, \delta_1, \delta_2, R)$, where $Q = \{p_1, p_2, p_3, q_1, q_2, q_3\}$; $\Sigma = \{a, b\}$; $\Gamma = \{X, Y\}$; $H_1 = \{p_1, p_2, p_3\}\Gamma$; $H_2 = \{q_1, q_2, q_3\}\Gamma$; $R = \{p_3, q_3\}$; Δ_1 and Δ_2 are defined as $\Delta_1(p_1X) = \Delta_2(q_1X) = \{a, b\}$, $\Delta_1(p_jX) = \Delta_2(q_jX) = \Delta_1(p_jY) = \Delta_2(q_jY) = \{a\}$, for $j = 2, 3$; and δ_1 and δ_2 are defined as

$$\delta_1(p_1X, a) = \delta_1(p_2X, a) = \widehat{p_3}, \qquad \delta_1(p_1X, b) = \widehat{p_2Y}$$

$$\delta_1(p_2Y, a) = \widehat{p_2},\ \delta_1(p_3X, a) = \widehat{p_3X}, \qquad \delta_1(p_3Y, a) = \widehat{p_3Y}$$

$$\delta_2(q_1X, a) = \delta_2(q_2X, a) = \widehat{q_3}, \qquad \delta_2(q_1X, b) = \widehat{q_2Y}$$

$$\delta_2(q_2Y, a) = \widehat{q_2},\ \delta_2(q_3X, a) = \widehat{q_3X}, \qquad \delta_2(q_3Y, a) = \widehat{q_3Y}.$$

Consider a memoryless strategy $\pi \in \Pi_1$ satisfying

$$\pi(p_1X) = a,\ \pi(p_1XX) = b.$$

Hence, $V(p_1X) = 1$ and $V(p_1XX) = 1$. Note that there exist exactly two SM-strategies for player 1, say π_1, π'_1. Let

$$\pi_1(p_1X) = \pi_1(p_1XX) = a,\ \pi'_1(p_1X) = \pi'_1(p_1XX) = b.$$

It is easy to compute that for any strategy $\pi_2 \in \Pi_2$

$$T(P^{\pi_1,\pi_2}_{p_1X}) = 1,\ T(P^{\pi_1,\pi_2}_{p_1XX}) = 0$$

$$T(P^{\pi'_1,\pi_2}_{p_1X}) = 0,\ T(P^{\pi'_1,\pi_2}_{p_1XX}) = 1.$$

Hence, there does not exist any optimal SM-strategy for player 1. Similarly, it is easy to show that there does not exist any optimal SM-strategy for player 2.

VI. Termination Values in Max-FPDTGs

In this section, we show how to solve the termination problem for Max-FPDTGs. Let us fix the Max-FPDTG $P = (Q, \Sigma, \Gamma, \Delta, \delta, R)$ for the rest of this section.

We first show how to compute $V(p\alpha)$, $p\alpha \in Q\Gamma$. Inspired by the transformation from an FPDA into an equivalent fuzzy context-free grammar [41], we transform every Max-FPDTG P into a Max-FBPATG P_b, so that we can obtain $V(p\alpha)$ by running Algorithm 1. Let us denote $V(p\alpha, q)$ the value of player 1 reaching the target set $\{q\}$ from the configuration $p\alpha$ in the underlying FRG G_P.

We define the Max-FBPATG $P_b = (\Sigma, \Gamma', \Delta', \delta', \{\epsilon\})$, where $\Gamma' = \{[pXq] : p, q \in Q,\ X \in \Gamma\}$, Δ' and δ' are defined as follows: for any $[pXq] \in \Gamma'$

1) $a \in \Delta'([pXq])$ and $[pXq] \xrightarrow{a|x} \epsilon$, if $\delta(pX, a)(q\epsilon) = x$;
2) $a \in \Delta'([pXq])$ and $[pXq] \xrightarrow{a|x} [q'Yq]$, if $\delta(pX, a)(q'Y) = x$;
3) $a \in \Delta'([pXq])$ and $[pXq] \xrightarrow{a|x} [q'Yq''][q''Zq]$ for every $q'' \in Q$, if $\delta(pX, a)(q'YZ) = x$.

Lemma 6: For every $p \in Q$ and $X \in \Gamma$, it holds that $V^P(pX) = \sup_{q \in R} V^{P_b}([pXq])$ for any *t*-norm.

Proof: See the Appendix. ■

We will show how to compute $V(p\alpha)$ ($|\alpha| \geq 2$) in P by reducing to the single-destination longest-path problem of weighted digraphs.

A *weighted digraph* $\mathcal{G}$ over $[0, 1]$ is a tuple (U, E, w), where U is the set of vertices of $\mathcal{G}$, E is the set of edges, and $w : E \to [0, 1]$ is the weight function mapping edges to elements of $[0, 1]$. The *length* $w(\sigma)$ of a path $\sigma = u_0u_1 \cdots u_k$ is the $\otimes$-product of the weights of its constituent edges:

$$w(\sigma) = w(u_0, u_1) \otimes \cdots \otimes w(u_{k-1}, u_k).$$

Given a set of paths, the path whose length is the largest one is called the *longest-path*. Given two vertices u and t, let $d(u, t)$ denote the length of the longest-path from u to t.

Let $t \in U$ be a fixed vertex of $\mathcal{G}$ called the *destination*. The *single-destination longest-path problem* (about *t*-norm $\otimes$) is to compute $d(u, t)$ for a given vertex u. The problem can be solved in $O(|U| \log |U| + |E|T_\otimes)$ time by using a modified Dijkstra's algorithm in [29] and [30]. In the classical single-destination longest-path problem, for a given weighted digraph, the weight of each edge is an element of real numbers, and the weight of a path is the sum of the weights of its constituent edges. The classical single-destination longest-path problem is NP-hard, meaning that it cannot be solved in polynomial time for arbitrary graphs unless $P = \mathrm{NP}$. However, it has a linear time solution for directed acyclic graphs (DAGs).

As mentioned previously, we will compute $V(p\alpha)$ by a reduction to the single-destination longest-path problem. We will see that the resulting weighted digraph is a DAG, for which we will first give a more efficient algorithm to solve the single-destination longest-path problem of weighted DAGs.

Suppose that $\mathcal{G}$ is a weighted DAG. Then, the following algorithm solves the single-destination longest-path problem of $\mathcal{G}$.

1) Initially, for every $u \in U\backslash\{t\}$, let $d(u, t) = w(u, t)$ if $(u, t) \in E$, and 0 otherwise.
2) Create a reverse topological order of all vertices.
3) Repeat following rule for every vertex u in reverse topological order until no update: if $(u, v) \in E$ and $d(u, t) < d(v, t) \otimes w(u, v)$, then $d(u, t) = d(v, t) \otimes w(u, v)$.

Indeed, the above-mentioned algorithm is an adaption of the algorithm for classical single-source shortest paths in weighted DAGs [14].

Proposition 4: Given a weighted DAG $\mathcal{G} = (U, E, w)$, the above-mentioned algorithm correctly solves its single-destination longest-path problem in $O(|U| + |E|T_\otimes)$ time.

The proof is trivial (cf. [14, Th. 24.5]), hence, omitted here.

Now we show how the algorithm for the single-destination longest-path problem in weighted DAGs can be used to compute $V(p\alpha)$ $(|\alpha| \geq 2)$ of the Max-FPDTG. For the given Max-FPDTG P and $p\alpha$, we define the weighted digraph $\mathcal{G} = (U, E, w)$, where

$$U = \{s, t\} \cup \left\{s^0_{[p\alpha(0)q]} : q \in Q\right\} \cup \left\{s^{|\alpha|-1}_{[q\alpha(|\alpha|-1)]} : q \in Q\right\} \cup \left\{s^i_{[q\alpha(i)q']} : q, q' \in Q, 1 \leq i < |\alpha| - 1\right\}$$

$$E = \begin{cases} \left\{\left(s, s^0_{[p\alpha(0)q]}\right) : q \in Q\right\} \cup \left\{(s^1_{[q\alpha(1)]}, t) : q \in Q\right\} \cup \\ \left\{\left(s^0_{[p\alpha(0)q]}, s^1_{[q\alpha(1)]}\right) : q \in Q\right\}, \text{if } |\alpha| = 2 \\ \left\{\left(s, s^0_{[p\alpha(0)q]}\right) : q \in Q\right\} \cup \left\{\left(s^{|\alpha|-1}_{[q\alpha(|\alpha|-1)]}, t\right) : q \in Q\right\} \cup \\ \left\{\left(s^i_{[q\alpha(i)q']}, s^{i+1}_{[q'\alpha(i+1)q'']}\right) \right. \\ \left. 1 \leq i < |\alpha| - 2, q, q', q'' \in Q\right\} \cup \\ \left\{\left(s^{|\alpha|-2}_{[q\alpha(|\alpha|-2)q']}, s^{|\alpha|-1}_{[q'\alpha(|\alpha|-1)]}\right) : q, q' \in Q\right\} \cup \\ \left\{\left(s^0_{[p\alpha(0)q]}, s^1_{[q\alpha(1)q']}\right) : q, q' \in Q\right\}, \text{otherwise} \end{cases}$$

and w is defined as

$$w\left(s, s^0_{[p\alpha(0)q]}\right) = 1$$

$$w\left(s^{|\alpha|-1}_{[q\alpha(|\alpha|-1)]}, t\right) = V\left(q\alpha(|\alpha| - 1)\right), \text{for all } q \in Q$$

if $|\alpha| = 2$, then

$$w\left(s^0_{[p\alpha(0)q]}, s^1_{[q\alpha(1)]}\right) = V(p\alpha(0), q), \text{ for all } q \in Q$$

if $|\alpha| > 2$, then

$$w\left(s^0_{[p\alpha(0)q]}, s^1_{[q\alpha(1)q']}\right) = V(p\alpha(0), q)$$

$$w\left(s^{|\alpha|-2}_{[q\alpha(|\alpha|-2)q']}, s^{|\alpha|-1}_{[q'\alpha(|\alpha|-1)]}\right) = V(q\alpha(|\alpha| - 2), q')$$

and for all $1 \leq i < |\alpha| - 2$, $q, q', q'' \in Q$

$$w\left(s^i_{[q\alpha(i)q']}, s^{i+1}_{[q'\alpha(i+1)q'']}\right) = V(q\alpha(i), q').$$

Clearly, the weighted digraph $\mathcal{G}$ is acyclic.

Lemma 7: $V(p\alpha) = d(s, t)$.

Proof: We first make a simple observation

$$V(p\alpha) = \sup\{V(p\alpha(0), q_0) \otimes V(q_0\alpha(1), q_1) \otimes \cdots \otimes V(q_{|\alpha|-2}\alpha(|\alpha| - 1)) : q_0, \ldots, q_{|\alpha|-2} \in Q\}.$$

It is easy to check the above-mentioned equation holds by induction on the length of α and the proof is omitted here. We, thus, see that $V(p\alpha) = d(s, t)$. ■

The following theorem gives the time complexity of our algorithm for computing $V(p\alpha)$.

Theorem 5: Let P be a Max-FPDTG and $p\alpha \in Q\Gamma^+$. Then

1) If $|\alpha| = 1, 2$, then the value $V(p\alpha)$ can be computed in $O(|Q|^2|P|T_\otimes + |Q|^2|P|\log(|Q|^2|\Gamma|))$ time and $O(|Q|^2|P|)$ space.
2) If $|\alpha| > 2$, then the value $V(p\alpha)$ can be computed in $O(|Q|^2|P|T_\otimes + |Q|^2|P|\log(|Q|^2|\Gamma|) + |Q|^2(|\alpha| - 2)T_\otimes)$ time and $O(|Q|^2|P| + |Q|^2(|\alpha| - 2))$ space.

Proof: According to the construction of P_b, $|\Gamma'| = |Q|^2|\Gamma|$ and $|P_b| = O(|Q|^2|P|)$. Hence, by Corollary 3, the value $V|_{Q\Gamma}$ of Max-FPDTG P can be computed in $O(|Q|^2|P|T_\otimes + |Q|^2|P|\log(|Q|^2|\Gamma|))$ time and $O(|Q|^2|P|)$ space.

To compute $V(p\alpha)$, $|\alpha| \geq 2$, we need to construct the weighted digraph $\mathcal{G}$: $|U| = 2(|Q| + 1) + |Q|^2(|\alpha| - 2)$ and $|E| = 3|Q| + |Q|^2(|\alpha| - 2)$. We see by Proposition 4 that the value $d(s, t)$ can be computed in $O((|Q| + |Q|^2(|\alpha| - 2)) \cdot T_\otimes)$ time and $O(|Q| + |Q|^2(|\alpha| - 2))$ space. Hence, the theorem holds. ■

Let us illustrate the above-mentioned algorithm by an example.

Example 6: Consider Max-FPDTG $P = (Q, \Sigma, \Gamma, \Delta, \delta, R)$, where $Q = \{p, q\}$, $\Sigma = \{a, b\}$, $\Gamma = \{Z\}$, $\Delta(pZ) = \Delta(qZ) = \{a, b\}$, $R = \{p\}$, and the transition rules are as follows:

$$\delta(pZ, a) = 0.8/qZZ, \qquad \delta(pZ, b) = 0.4/p + 0.5/q$$
$$\delta(qZ, a) = 0.9/pZZ, \qquad \delta(qZ, b) = 0.8/p + 0.4/q.$$

Then, the stack alphabet of the associating Max-FBPATG P_b is $\{[pZp], [pZq], [qZp], [qZq]\}$, the move assignment of each

$$\begin{tikzcd}
 & s^0_{[pZp]} \arrow[r, "0.5"] & s^1_{[pZ]} \arrow[dr, "0.5"] & \\
s \arrow[ur, "1"] \arrow[dr, "1"'] & & & t \\
 & s^0_{[pZq]} \arrow[r, "0.5"] & s^1_{[qZ]} \arrow[ur, "0.8"'] &
\end{tikzcd}$$

Fig. 2. Weighted DAG $\mathcal{G}$.

stack element is all $\{a, b\}$, the transitions are the following:

$$\delta'([pZp], a) = 0.8/[qZp][pZp] + 0.8/[qZq][qZp]$$
$$\delta'([pZq], a) = 0.8/[qZp][pZq] + 0.8/[qZq][qZq]$$
$$\delta'([qZp], a) = 0.9/[pZp][pZp] + 0.9/[pZq][qZp]$$
$$\delta'([qZq], a) = 0.9/[pZp][pZq] + 0.9/[pZq][qZq]$$
$$\delta'([pZp], b) = 0.4/\epsilon,\ \delta'([pZq], b) = 0.5/\epsilon$$
$$\delta'([qZp], b) = 0.8/\epsilon,\ \delta'([qZq], b) = 0.4/\epsilon.$$

We want to know the value of $V(pZZ)$. Using Algorithm 1, we get that

$$V([pZp]) = V([pZq]) = V([qZq]) = 0.5,\ V([qZp]) = 0.8$$

when we choose the minimum *t*-norm as the *t*-norm. Hence

$$V(pZ) = 0.5,\ V(qZ) = 0.8.$$

Now we construct a weighted DAG $\mathcal{G}$ shown in Fig. 2 . We get by Lemma 7 that $V(pZZ) = 0.5$.

VII. Illustrating Example

In this section, we apply the previous results to an example arising from manufacturing systems [17].

A manufacturing system consists of three components: the customer component for order placing and product acceptance, the manufacturing factory component for product production, and the transportation component for delivering products.

The manufacturing factory may have several branch factories to fulfill the orders. Every branch factory may have many suppliers to provide the required raw material, and may have many production lines to select to produce. In this paper, we assume that there are two branch factories, denoted by B_1 and B_2, can be selected. We also assume that there are two suppliers, denoted by $S_{i,1}$ and $S_{i,2}$, and two production lines, denoted by $L_{i,1}$ and $L_{i,2}$, can be selected for the branch factories B_i, $i = 1, 2$. When a branch factory is selected to fulfill the order, it should select one appropriate supplier and one production line. In practice, many factors can affect the selections of suppliers and production lines. To simplify the problem, we only consider the following two factors to analyze the selection of suppliers: 1) the quality of the raw material, and 2) the price of the material; and the following two factors to analyze the selection of production lines: 1) production quality, and 2) the equipment maintenance and replacement cost. Assume that the quality of the raw material provided by $S_{i,1}$ is better than that provided by $S_{i,2}$, while the price of the raw material provided by $S_{i,1}$ is higher than that provided by $S_{i,2}$; and the quality of the products produced by $L_{i,1}$ is better than that produced by $L_{i,2}$, while the equipment maintenance and replacement cost of $L_{i,1}$ is more expressive than that of $L_{i,2}$, $i = 1, 2$.

$$\begin{tikzcd}
X_0 \arrow[d, "\textit{receiving orders}"'] \\
X_1 \arrow[d, bend right, "\textit{selecting } B_1"'] \arrow[d, bend left, "\textit{selecting } B_2"] \\
X_2 \arrow[d, "\textit{delivering products}"'] \\
X_3 \arrow[d, "\textit{return}"'] \\
\bullet
\end{tikzcd}$$

Flow graph (X)

$$\begin{tikzcd}
Y_{i,0} \arrow[d, bend right, "\textit{selecting } S_{i,1}"'] \arrow[d, bend left, "\textit{selecting } S_{i,2}"] \\
Y_{i,1} \arrow[d, bend right, "\textit{selecting } L_{i,1}"'] \arrow[d, bend left, "\textit{selecting } L_{i,2}"] \\
Y_{i,2} \arrow[d, "\textit{return}"'] \\
\bullet
\end{tikzcd}$$

Flow graph (Y_i)

$$\begin{tikzcd}
Z_{i,j,0} \arrow[d, "\textit{return} | x_{i,j}"] \\
\bullet
\end{tikzcd}$$

Flow graph $(Z_{i,j})$

$$\begin{tikzcd}
Z'_{i,j,0} \arrow[d, "\textit{return} | y_{i,j}"] \\
\bullet
\end{tikzcd}$$

Flow graph $(Z'_{i,j})$

Fig. 3. Flow graphs.

The objective of the manufacturing system is to select a branch factory such that the possibility that the branch factory can fulfill the order is as "large" as possible. To solve this problem, we use the FBPATGs to model the manufacturing system and its branch factories, suppliers, and production lines. To construct the associated FBPATGs, we proceed in two steps.

In the first step, we represent the manufacturing system by a set of *flow graphs* shown in Fig. 3, where the flow graphs X, Y_i, $Z_{i,j}$, and $Z'_{i,j}$ correspond to the manufacturing system, the branch factory B_i, the supplier $S_{i,j}$, and the production line $L_{i,j}$, $i, j = 1, 2$, respectively. Each flow graph has a distinguished start node and an end node with a *return* action. $x_{i,j}$ in the flow graph $Z_{i,j}$ in Fig. 3 is the truth value that the supplier $S_{i,j}$ provides the raw material satisfying the requirements of the order if $S_{i,j}$ is chosen as a supplier by the branch factory B_i, while $y_{i,j}$ in the flow graph $Z'_{i,j}$ is the truth value that the production line $L_{i,j}$ completes the orders on time and with high quality if the production line $L_{i,j}$ is chosen to produce by the branch factory B_i, $i,\ j = 1,\ 2$. Their values are as follows:

$$x_{11} = 0.9, \quad x_{12} = 0.7, \quad x_{21} = 0.9, \quad x_{22} = 0.8$$
$$y_{11} = 0.8, \quad y_{12} = 0.7, \quad y_{21} = 0.9, \quad y_{22} = 0.8.$$

The branch factories and suppliers form the environment of the manufacturing system, which cannot be controlled by the manufacturing system. Hence, we model the control points in flow graph X in Fig. 3 as the stack elements of player 1, while the control points in other flow graphs are modeled as the stack elements of player 2.

In the second step, we apply a straightforward transformation to obtain its FBPATG model P_b. The transition rules generated by the flow graph X are as follows:

$$\delta_1(X_0,\ \text{receiving orders}) = \widehat{X_1}$$
$$\delta_1(X_1,\ \text{selecting}\ B_1) = \widehat{Y_{1,0}X_2}$$
$$\delta_1(X_2,\ \text{delivering products}) = \widehat{X_3}$$
$$\delta_1(X_1,\ \text{selecting}\ B_2) = \widehat{Y_{2,0}X_2}$$
$$\delta_1(X_3,\ \text{return}) = \widehat{\epsilon}.$$

Intuitively, the transition rule $\delta_1(X_0,\ \text{receiving orders}) = \widehat{X_1}$ represents that control passes from X_0 to X_1 when the action receiving orders is activated in control point X_0; $\delta_1(X_1,\ \text{selecting}\ B_1) = \widehat{Y_{1,0}X_2}$ (resp. $\delta_1(X_1,\ \text{selecting}\ B_2) = \widehat{Y_{2,0}X_2}$) represents an edge between control points X_1 and X_2 containing a call to flow graph Y_1 (resp. Y_2) by action selecting B_1 (resp. selecting B_2), X_2 can be seen as the return address of that call; and $\delta_1(X_3,\ \text{return}) = \widehat{\epsilon}$ represents an edge leaving X_3 containing a return statement.

The transition rules generated by flow graph Y_i, $i = 1,\ 2$, are as follows:

$$\delta_2(Y_{i,0},\ \text{selecting}\ S_{i,1}) = \widehat{Z_{i,1,0}Y_{i,1}}$$
$$\delta_2(Y_{i,0},\ \text{selecting}\ S_{i,2}) = \widehat{Z_{i,2,0}Y_{i,1}}$$
$$\delta_2(Y_{i,1},\ \text{selecting}\ L_{i,1}) = \widehat{Z'_{i,1,0}Y_{i,2}}$$
$$\delta_2(Y_{i,1},\ \text{selecting}\ L_{i,2}) = \widehat{Z'_{i,2,0}Y_{i,2}}$$
$$\delta_2(Y_{i,2},\ \text{return}) = \widehat{\epsilon}.$$

The transition rules generated by flow graphs $Z_{i,j}$ and $Z'_{i,j}$, $i,\ j = 1, 2$, are as follows:

$$\delta_2(Z_{i,j,0},\ \text{return}) = x_{i,j}/\epsilon,\ \delta_2(Z'_{i,j,0},\ \text{return}) = y_{i,j}/\epsilon.$$

We see that the objective of the manufacturing system is equivalent to looking for the optimal strategies for player 1. Using our previous theoretical results, we can get that $V(X_0) = 0.8$ when we choose the minimum *t*-norm as the *t*-norm. The value 0.8 expresses the maximal possibility of completing the order of the manufacturing system. The manufacturing system can choose branch factory B_2 to produce the production, which ensures that the truth value of completing the order is 0.8.

VIII. Conclusion

In this paper, we introduced FPDTGs that are a natural model for studying the termination problem for infinite-state systems involving fuzzy, recursion, and game features. We showed that the computational complexity of the termination problems for FPDTGs and subclasses thereof: FBPATGs and Max-FPDTGs. We proved that the termination problems are exponential time for FPDTGs over the minimum *t*-norm and FPT with respect to the computational time of *t*-norms for FBPATGs and Max-FPDTGs. We also studied the optimal strategies for termination problems, and showed that there exist optimal memoryless strategies for two players in FPDTGs, and there exist SM-strategies for two players in FBPATGs. A simple example illustrated that optimal SM-strategies do not exist for FPDTGs, even for Max-FPDTGs and Min-FPDTGs.

There are two problems that are worth further study in our framework. First, we plan to consider model checking fuzzy temporal logics against the transition systems generated by FPDTGs. In addition, we are going to extend FPDTGs to the concurrent game structure setting [3] and investigate their termination problem.

Appendix

Proof of Theorem 1:

1) Observe that $V_1^n(s)$ and $V_2^n(s)$ can be seen as the values of players 1 and 2 of a finite FRG in state s, respectively, since G is finitely branching. It follows from Proposition 2 that $V_1^n(s) = V_2^n(s)$. We, therefore, see by definition that $V_1(s) = V_2(s)$. Hence, G is determined.

2) We define a sequence $A_0, A_1, \ldots$ of fuzzy sets of S as follows: For any state $s \in S$

$$A_0(s) = \begin{cases} 1, & \text{if } s \in R \\ 0, & \text{otherwise} \end{cases}$$
$$A_{n+1}(s) = F(A_n)(s).$$

By the proof of [32, Th. 2], we get that $V^n(s) = A_n(s)$ for any state $s \in S$ and $n \in \mathbb{N}$. Let $s \in S_2 \backslash R$. By finitely branching property of G, there must be some $b \in \Delta_2(s)$ such that for infinitely many $n \in \mathbb{N}$

$$A_{n+1}(s) = \sup_{t \in S}(\delta_2(s,\ b)(t) \otimes A_n(t)).$$

We define the memoryless strategy $\pi \in \Pi_2$ by setting $\pi(s) = b$.

We shall prove that for any $s \in S$, $\pi_1 \in \Pi_1$, and $n \in \mathbb{N}$, there exists some m such that

$$T_n(G_s^{\pi_1,\pi}) \leq A_{n+m}(s). \tag{2}$$

This implies that assertion (2) holds.

If $s \in R$, (2) obviously holds. We prove (2) holds for $s \in S \backslash R$ by induction on n. For $n = 0$, (2) holds trivially. Assume that for $n = n_1$, (2) holds. In other words, if $n = n_1$, then for every $s \in S$ and $\pi_1 \in \Pi_1$, there exists $m' \in \mathbb{N}$ such that $T_{n_1}(G_s^{\pi_1,\pi}) \leq$

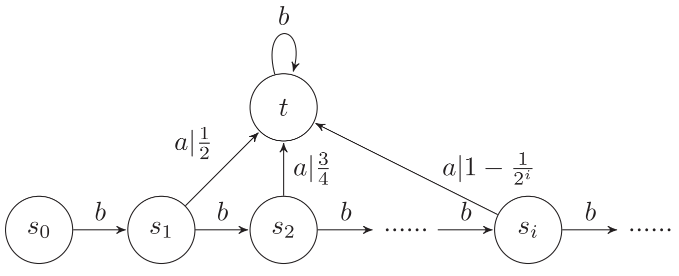


Fig. 4. Finitely-branching FRG G with the target set $R = \{t\}$.

$A_{n_1+m'}(s)$. For $n = n_1 + 1$, there are two cases needed to be considered

Case 1: $s \in S_1 \backslash R$. By definition, we have that

$$\begin{aligned} T_n(G_s^{\pi_1,\pi}) &= T_{n_1+1}(G_s^{\pi_1,\pi}) \\ &= \sup_{t\in S}\big(\delta_1(s,\,\pi_1(s))(t) \otimes T_{n_1}(G_t^{\pi_{1,s},\pi})\big) \\ &\le \sup_{t\in S}\big(\delta_1(s,\,\pi_1(s))(t) \otimes A_{n_1+m'}(t)\big) \\ &\le \sup_{a\in\Delta_1(s)}\sup_{t\in S}(\delta_1(s,a)(t) \otimes A_{n_1+m'}(t)) \\ &= A_{n_1+m'+1}(s) = A_{n+m'}(s) \end{aligned}$$

where $\pi_{1,s}$ is the strategy satisfying $\pi_{1,s}(t\sigma) = \pi_1(st\sigma)$ for any $\sigma \in S^*$.

Case 2: $s \in S_2 \backslash R$. By definition, we have that

$$\begin{aligned} T_n(G_s^{\pi_1,\pi}) &= T_{n_1+1}(G_s^{\pi_1,\pi}) \\ &= \sup_{t\in S}\big(\delta_2(s,\,\pi(s))(t) \otimes T_{n_1}(G_t^{\pi_{1,s},\pi})\big) \\ &\le \sup_{t\in S}\big(\delta_2(s,\,\pi(s))(t) \otimes A_{n_1+m'}(t)\big) \end{aligned}$$

where $\pi_{1,s}$ is the strategy satisfying $\pi_{1,s}(t\sigma) = \pi_1(st\sigma)$ for any $\sigma \in S^*$.

By definition, there must be some m, $m > m'$, such that

$$A_{n_1+m}(s) = \sup_{t\in S}\big(\delta_2(s,\,\pi(s))(t) \otimes A_{n_1+m-1}(t)\big).$$

By the monotonicity of A_n, we have that

$$\begin{aligned} T_{n_1+1}(G_s^{\pi_1,\pi}) &\le \sup_{t\in S}\big(\delta_2(s,\,\pi(s))(t) \otimes A_{n_1+m-1}(t)\big) \\ &= A_{n_1+m}(s). \end{aligned}$$

Based on the above-mentioned analysis, we conclude that if $n = n_1 + 1$, (2) holds.

3) We give an example to show that the assertion holds. Consider a finitely-branching FRG G depicted as in Fig. 4. It is easy to see that $V(s_0) = 1$, but for any strategy π, $T(G_{s_0}^{\pi}) < 1$.

4) Suppose that $p\alpha \in Q\Gamma^+$. Recall that $V(p\alpha) = \sup_{n\in\mathbb{N}} V^n(p\alpha)$. Hence, for any $\epsilon > 0$, there exists some $n \in \mathbb{N}$ such that $V^n(p\alpha) > V(p\alpha) - \epsilon$. We then can see by Proposition 2 that for every $\epsilon > 0$, player 1 has a memoryless strategy $\pi_1 \in \Pi_1'$ such that for any strategy $\pi_2 \in \Pi_2$, $T(G_{p\alpha}^{\pi_1,\pi_2}) > V(p\alpha) - \epsilon$. Consequently, the value of $p\alpha$ can be written as $V(p\alpha) = \sup_{\pi_1\in\Pi_1'}\inf_{\pi_2\in\Pi_2} T(G_{p\alpha}^{\pi_1,\pi_2})$. Note that $\mathrm{Im}(P)$ is a finite set and the *t*-norm $\otimes$ considered in this paper satisfies the finite generated condition. Hence, $V(p\alpha)$ can be further written as $V(p\alpha) = \max_{\pi_1\in\Pi_1'}\min_{\pi_2\in\Pi_2} T(G_{p\alpha}^{\pi_1,\pi_2})$, which implies that there exists an optimal memoryless strategy for player 1.

5) It is obvious that $V \subseteq F(V)$, since $V(s) = \sup_{n\in\mathbb{N}} A_n(s)$ and $A_n(s) \le F(V)(s)$ for any $s \in S$, $n \in \mathbb{N}$. We next show that $F(V)(s) \le V(s)$. We only consider the case for $s \in S_2\backslash R$, the others are similar. If $s \in S_2\backslash R$, then for every $b \in \Delta_2(s)$, there exists $t_b \in S$ such that $F(V)(s) \le \delta_2(s,\,b,\,t_b) \otimes V(t_b)$. Hence, by Proposition 1(1), we have that $F(V)(s) \le \sup_{n\in\mathbb{N}}(\delta_2(s,\,b,\,t_b) \otimes V^n(t_b))$. This implies that for any $\varepsilon > 0$, there exists $n_{t_b} \in \mathbb{N}$ such that $F(V)(s) - \varepsilon < \delta_2(s,\,b,\,t_b) \otimes V^{n_{t_b}}(t_b)$. Notice that $\delta_2(s,\,b,\,t_b) \otimes V^{n_{t_b}}(t_b) \le \delta_2(s,\,b,\,t_b) \otimes V(t_b)$. As a consequence, $F(V)(s) < V(s) + \varepsilon$. Now we can see that $F(V)(s) \le V(s)$. ■

Proof of Theorem 2: Due to Lemma 2, it is sufficient to show that the theorem holds for any simple FPDTG P with the property that if $pX \in H_2$, $a \in \Delta_2(pX)$, and $\delta_2(pX,\,a)(q\alpha) > 0$, then $\delta_2(pX,\,a)(q\alpha) = 1$. For every $x \in [0,\,1]$, we define a PDTG $P^x = (Q,\,\Sigma,\,\Gamma,\,H_1,\,H_2,\,\Delta_1,\,\Delta_2,\,\delta_1^x,\,\delta_2^x,\,R)$, where δ_1^x and δ_2^x are defined as: For $i = 1,\,2$, $pX \in H_i$, $a \in \Delta_i(pX)$, $q\alpha \in Q\Gamma^{\le 2}$, if $\delta_i(pX,\,a)(q\alpha) \ge x$, then $\delta_i^x(pX,\,a)(q\alpha) = 1$, otherwise $\delta_i^x(pX,\,a)(q\alpha) = 0$.

Similarly to the proof of [26, Th. 2.1], it is easy to check that if the *t*-norm is chosen as the minimum *t*-norm, then for $p\alpha \in Q\Gamma^+$, $V(p\alpha) \ge x$ if and only if player 1 in P^x has a 1-winning strategy from $p\alpha$. Note that $|\langle\mathrm{Im}(P)\rangle| = |\mathrm{Im}(P)|$ for the minimum *t*-norm. Hence

$$V(p\alpha) = \max\{x:\ \text{player 1 in } P^x \text{ has a 1-winning strategy from } p\alpha,\ x \in \mathrm{Im}(P)\}$$

which implies that the termination problem for the simple FPDTG P can be reducible to the termination problem for simple PDTGs. It directly follows from Lemma 1 that the termination problem for simple FPDTGs with *t*-norm chosen as the minimum *t*-norm can be solved in exponential time. ■

Proof of Lemma 3:

1) We prove assertion (1) in two parts. First, we show that $V^{n+1}(X\alpha) \le V^n(X) \otimes V^n(\alpha)$ holds. By Proposition 2, there exist two strategies $\pi_2',\,\pi_2'' \in \Pi_2$ such that

$$V^n(X) = \max_{\pi_1\in\Pi_1} T_n^{\pi_1,\pi_2'}(X),\ V^n(\alpha) = \max_{\pi_1\in\Pi_1} T_n^{\pi_1,\pi_2''}(\alpha).$$

Hence, by Proposition 1(3), we conclude that

$$\begin{aligned} V^n(X) \otimes V^n(\alpha) &= \max_{\pi_1 \in \Pi_1} T_n^{\pi_1,\pi_2'}(X) \otimes \max_{\pi_1 \in \Pi_1} T_n^{\pi_1,\pi_2''}(\alpha) \\ &\geq \max_{\pi_1 \in \Pi_1} (T_n^{\pi_1,\pi_2'}(X) \otimes T_n^{\pi_1,\pi_2''}(\alpha)) \\ &\geq \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} (T_n^{\pi_1,\pi_2}(X) \otimes T_n^{\pi_1,\pi_2}(\alpha)) \\ &\geq \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} T_{n+1}^{\pi_1,\pi_2}(X\alpha) = V^{n+1}(X\alpha). \end{aligned}$$

Next, we show that $V^n(X) \otimes V^n(\alpha) \leq V^{2n}(X\alpha)$. By Proposition 2, there exist two strategies $\pi_1', \pi_1'' \in \Pi_1$ such that

$$V^n(X) = \min_{\pi_2 \in \Pi_2} T_n^{\pi_1',\pi_2}(X),\ V^n(\alpha) = \min_{\pi_2 \in \Pi_2} T_n^{\pi_1'',\pi_2}(\alpha).$$

By Proposition 1(4), we see that

$$\begin{aligned} V^n(X) \otimes V^n(\alpha) &= \min_{\pi_2 \in \Pi_2} T_n^{\pi_1',\pi_2}(X) \otimes \min_{\pi_2 \in \Pi_2} T_n^{\pi_1'',\pi_2}(\alpha) \\ &\leq \min_{\pi_2 \in \Pi_2} (T_n^{\pi_1',\pi_2}(X) \otimes T_n^{\pi_1'',\pi_2}(\alpha)) \\ &\leq \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} (T_n^{\pi_1,\pi_2}(X) \otimes T_n^{\pi_1,\pi_2}(\alpha)) \\ &\leq \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} T_{2n}^{\pi_1,\pi_2}(X\alpha) = V^{2n}(X\alpha). \end{aligned}$$

2) Notice that $\sup_{n\in\mathbb{N}} V^n(\alpha) = \lim_{n\to\infty} V^n(\alpha)$, since for any $n \in \mathbb{N}$, $V^n(\alpha) \in [0, 1]$, and $V^n(\alpha) \leq V^{n+1}(\alpha)$. Hence, by assertion 1) and (1), we see that

$$\begin{aligned} &V(\alpha(0)) \otimes \cdots \otimes V(\alpha(|\alpha|-1)) \\ &\quad = \sup\nolimits_{n\in\mathbb{N}} V^n(\alpha(0)) \otimes \cdots \otimes \sup_{n\in\mathbb{N}} V^n(\alpha(|\alpha|-1)) \\ &\quad = \lim_{n\to\infty} V^n(\alpha(0)) \otimes \cdots \otimes \lim_{n\to\infty} V^n(\alpha(|\alpha|-1)) \\ &\quad = \lim_{n\to\infty} \Big(V^n(\alpha(0)) \otimes \cdots \otimes V^n(\alpha(|\alpha|-1))\Big) \\ &\quad = \lim_{n\to\infty} V^n(\alpha) = \sup_{n\in\mathbb{N}} V^n(\alpha) = V(\alpha) \end{aligned}$$

which finishes the proof of the assertion. ■

Proof of Lemma 4: By the monotonicity properties of $\otimes$, sup, and inf operators, it is easy to check that for all $n \geq 0$, $F^n(\emptyset) \subseteq F^{n+1}(\emptyset)$. Note that the *t*-norms we consider satisfy finite generated condition and Γ is a finite set. Hence, there exists some $m \in \mathbb{N}$ such that for all $n \geq m$, $F^n(\emptyset) = F^m(\emptyset)$. By Kleene's theorem, $\bigcup_{n\in\mathbb{N}} F^n(\emptyset)$ is the least fixed point of E_{P_b}.

By Theorem 1(5) and Lemma 3(2), we see that $V|_{\Gamma_\epsilon}$ is the fixed point of the system of equations E_{P_b}. Thus, $\bigcup_{n\in\mathbb{N}} F^n(\emptyset) \subseteq V|_{\Gamma_\epsilon}$. To show that $V|_{\Gamma_\epsilon} \subseteq \bigcup_{n\in\mathbb{N}} F^n(\emptyset)$, it suffices to show that for any $X \in \Gamma_\epsilon$ and $n \in \mathbb{N}$

$$V^n(X) \leq F^{n+1}(\emptyset)(X). \tag{3}$$

We prove (3) holds by induction on n. For the base case, namely, $n = 0$, it is trivial. The induction hypothesis is that (3) holds for n. We now prove the same for $n+1$. For $n+1$, we only consider the case for $X \in H_1 \cap \Gamma$, all other cases can be easily proved along similar lines. Using Lemma 3(1) and induction hypothesis, we have the following:

$$\begin{aligned} V^{n+1}(X) &= \sup_{a\in\Delta_1(X)} \max \left\{ \begin{array}{l} \sup\limits_{X \xrightarrow{a|x} \epsilon} x,\ \sup\limits_{X \xrightarrow{a|x} Y} (x \otimes V^n(Y)), \\ \sup\limits_{X \xrightarrow{a|x} YZ} (x \otimes V^n(YZ)) \end{array} \right\} \\ &\leq \sup_{a\in\Delta_1(X)} \max \left\{ \begin{array}{l} \sup\limits_{X \xrightarrow{a|x} \epsilon} x,\ \sup\limits_{X \xrightarrow{a|x} Y} (x \otimes V^n(Y)), \\ \sup\limits_{X \xrightarrow{a|x} YZ} (x \otimes V^{n-1}(Y) \otimes V^{n-1}(Z)) \end{array} \right\} \\ &\leq \sup_{a\in\Delta_1(X)} \max \left\{ \begin{array}{l} \sup\limits_{X \xrightarrow{a|x} \epsilon} x,\ \sup\limits_{X \xrightarrow{a|x} Y} (x \otimes V^n(Y)), \\ \sup\limits_{X \xrightarrow{a|x} YZ} (x \otimes V^n(Y) \otimes V^n(Z)) \end{array} \right\} \\ &\leq \sup_{a\in\Delta_1(X)} \max \left\{ \begin{array}{l} \sup\limits_{X \xrightarrow{a|x} \epsilon} x,\ \sup\limits_{X \xrightarrow{a|x} Y} (x \otimes F^{n+1}(\emptyset)(Y)), \\ \sup\limits_{X \xrightarrow{a|x} YZ} (x \otimes F^{n+1}(\emptyset)(Y) \\ \qquad\qquad \otimes F^{n+1}(\emptyset)(Z)) \end{array} \right\} \\ &= F^{n+2}(\emptyset)(X) \end{aligned}$$

which proves the claim. ■

Proof of Lemma 5: By Theorem 1, $V^{P_b}(\alpha') = V^{P'}(\alpha')$ for any $\alpha' \in \Gamma^+$. By Lemma 3, it is enough to prove that upon termination of Algorithm 1, $v[X] = V(X)$ for any $X \in \Gamma'$. We give the sketch of its proof, the similar proof was given in that of [32, Th. 4]. First, we can see that during the execution of Algorithm 1, if the stack element X does not belong to $\mathcal{Q}$, then $v[X]$ is greater than or equal to the maximum priority in $\mathcal{Q}$. Next, we show that during the execution of Algorithm 1, a stack element X is never assigned a priority greater than $V(X)$. The property holds after initialization because then $v[X] = 0 \leq V(X)$. Assume that the property is true and that the function INCREASE is called. Then, there exists a stack element Y with $Y \in \mathrm{Succ}(X)$ and which was just removed from $\mathcal{Q}$. We only consider the case $X \in H_1'$, the other is the similar. By Lemma 4 and induction hypothesis, we get that

$$V(X) \geq x \otimes V(Y) \otimes V(Z) \geq x \otimes v[Y] \otimes v[Z] = v[X].$$

Finally, we show that when Algorithm 1 terminates, $V(X) \leq v[X]$ holds for any $X \in \Gamma'$. Let $F^n(\emptyset)$, $n \in \mathbb{N}$, be the fuzzy set of Γ_ϵ defined in the proof of Lemma 4. It follows from Lemma 4 that the assertion holds if we can show that $F^n(\emptyset)(X) \leq v[X]$ holds for any $X \in \Gamma'$. We prove it by induction on n. The detailed proof is omitted, and similar proof can be found in that of [32, Lemma 7]. ■

Proof of Theorem 3: We use aggregate analysis to discuss the time complexity of the algorithm above. The time spent in Step 1 is $O(|P_b|)$, which results in an FBPATG P' of size $O(|P_b| + |\Gamma| \cdot |\Sigma|)$, moreover, $|\Gamma'| = O(|\Gamma| \cdot |\Sigma|)$.

We continue to analyze the time spent in Step 2, i.e., Algorithm 1. In Algorithm 1, lines 1–7 take $O(|\Gamma'|)$ time. The loop at lines 8–17 takes $O(|P'|T_\otimes + |P'| \log |\Gamma'|)$ time, since INCREAS is executed at most $|P'|$ times and each time takes

$O(\log|\Gamma'|)$ time; moreover, the $\otimes$ operation is executed $O(|P'|)$ times. Lines 18 and 19 take $O((|\alpha|-1)T_{\otimes})$ time. In total, the time of executing Algorithm 1 is $O((|P'|+|\alpha|-1)T_{\otimes}+|P'|\log|\Gamma'|)$. Thus the time complexity of computing $V(\alpha)$ is dominated by Step 2. In other words, the problem of computing $V(\alpha)$ can be solved in time $O\big((|P_b|+|\Gamma|\cdot|\Sigma|+|\alpha|-1)T_{\otimes}+(|P_b|+|\Gamma|\cdot|\Sigma|)\log(|\Gamma|\cdot|\Sigma|)\big)$.

Memory is needed for δ_2' and the three data structures: an array of counters, an array of values, and a max-priority queue. Hence the algorithm takes $O(|\Gamma|\cdot|\Sigma|)$ space. ■

Proof of Theorem 4: Consider the SM-strategy π for player 2 such that for all $X\in H_2$, $\alpha\in\Gamma^*$, $\pi(X\alpha)=a$ if $a\in\Delta_2(X)$ and

$$V(X)=\max\left\{\begin{array}{l}\sup\limits_{X\xrightarrow{a|x}\epsilon} x,\ \sup\limits_{X\xrightarrow{a|x}Y}(x\otimes V(Y)),\\ \sup\limits_{X\xrightarrow{a|x}YZ}(x\otimes V(Y)\otimes V(Z))\end{array}\right\}$$

We show by induction on n that for all $\alpha\in\Gamma^+$, $\sup_{\pi_1\in\Pi_1}T_n(P_{\alpha}^{\pi_1,\pi})\le V(\alpha)$. This implies that π is an optimal strategy for player 2. For $n=0$, the result is immediate. For $n>0$, we only consider the case for $\alpha\in H_2\Gamma^*$, the proof of the other case is analogous to that of Theorem 1(2).

Assume that $n=k+1$. For the case $\alpha\in H_2\Gamma^*$, we need to consider two subcases of α: $\alpha=X\in H_2$ and $\alpha=X\alpha'\in H_2\Gamma^+$.

Subcase 1. $\alpha\in H_2$: By induction hypothesis and Proposition 1(1), we have that

$$\begin{aligned}&\sup_{\pi_1\in\Pi_1}T_{n+1}(P_{\alpha}^{\pi_1,\pi})\\ &=\max\left\{\begin{array}{l}\sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}\epsilon}x,\ \sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}Y}\sup\limits_{\pi_1\in\Pi_1}(x\otimes T_n(P_Y^{\pi_1,\pi}))\\ \sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}YZ}\sup\limits_{\pi_1\in\Pi_1}(x\otimes T_n(P_{YZ}^{\pi_1,\pi}))\end{array}\right\}\\ &=\max\left\{\begin{array}{l}\sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}\epsilon}x,\ \sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}Y}(x\otimes\sup\limits_{\pi_1\in\Pi_1}T_n(P_Y^{\pi_1,\pi}))\\ \sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}YZ}(x\otimes\sup\limits_{\pi_1\in\Pi_1}T_n(P_{YZ}^{\pi_1,\pi}))\end{array}\right\}\\ &\le\max\left\{\begin{array}{l}\sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}\epsilon}x,\ \sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}Y}(x\otimes V(Y))\\ \sup\limits_{\alpha\xrightarrow{\pi_2(\alpha)|x}YZ}(x\otimes V(YZ))\end{array}\right\}\\ &=V(\alpha).\end{aligned}$$

Subcase 2. $\alpha=X\alpha'\in H_2\Gamma^+$: Using Lemma 3(2) and the induction hypothesis, we have the following:

$$\begin{aligned}\sup_{\pi_1\in\Pi_1}T_{n+1}(P_{X\alpha'}^{\pi_1,\pi})&\le\sup_{\pi_1\in\Pi_1}T_n(P_X^{\pi_1,\pi})\otimes\sup_{\pi_1\in\Pi_1}T_n(P_{\alpha'}^{\pi_1,\pi})\\ &\le V(X)\otimes V(\alpha')=V(X\alpha').\end{aligned}$$

Observe that for an FBPATG P and an SM-strategy $\pi_2\in\Pi_2$, the result is a Max-FBPATG. To show that player 1 has an optimal SM-strategy for every FBPATG, it is sufficient to show that there is an optimal SM-strategy for player 1 in every Max-FBPATG. Given a Max-FBPATG, let π_1 be a strategy for player 1 defined as follows: For all $X\in H_1,\alpha\in\Gamma^*$, $\pi_1(X\alpha)$ takes the move of $X\overset{a|x}{\to}YZ$ or $X\overset{a|x}{\to}ZY$ at line 1 of Algorithm 1 such that $v[X]$ is the latest value. Then, by the idea of Algorithm 1, we get that π_1 is an optimal SM-strategy for player 1 in the Max-FBPATG. ■

Proof of Lemma 6: We first show that for any $pX\in Q\Gamma$ and $q\in Q$, $V^P(pX,q)\le V^{P_b}([pXq])$. To show this, it is sufficient to check the following property holds.

Claim 1: Let $pX\overset{\beta|x}{\longrightarrow}{}^* q$ denote that there exists a sequence of transitions $pX\overset{a_1|x_1}{\longrightarrow}q_1\alpha_1\cdots\overset{a_{|\beta|}|x_{|\beta|}}{\longrightarrow}q$ such that $\beta=a_1\cdots a_{|\beta|}$, $x=x_1\otimes\cdots\otimes x_{|\beta|}$, and $q_1\alpha_1\in Q\Gamma^{\le 2}$. If $\sigma_1=pX\overset{\beta|x}{\longrightarrow}{}^* q$ for some $\beta\in\Sigma^+$, $p,q\in Q$, and $X\in\Gamma$, is a finite path in P, then there exists a finite path $\sigma_2=[pXq]\overset{\beta|x}{\longrightarrow}{}^*\epsilon$ in P_b.

Claim 1 can be proved by induction on the length $|\beta|$. For the case of $|\beta|=1$, it is obvious. Assume that Claim 1 holds for any β with length $k\ge1$. Let β with length $k+1$. Then, we can write β as $a\beta'$, where $a\in\Sigma$ and $\beta'\in\Sigma^*$ with length k. Hence, by definition, σ_1 can be written as $pX\overset{a|x'}{\longrightarrow}p_1\alpha\overset{\beta'|x''}{\longrightarrow}{}^* q$ with $x=x'\otimes x''$ and $p_1\alpha\in Q\Gamma^{\le2}$. We only consider the case $\alpha=YZ$, the others being simpler. By definition, there exist $\beta_1',\beta_2'\in\Sigma^*$ and $q'\in Q$ such that $p_1YZ\overset{\beta_1'|x_1''}{\longrightarrow}{}^* q'Z\overset{\beta_2'|x_2''}{\longrightarrow}{}^* q$ with $x''=x_1''\otimes x_2''$ and $\beta'=\beta_1'\beta_2'$. Hence, we have by assumption that

$$[p_1Yq']\overset{\beta_1'|x_1''}{\longrightarrow}{}^*\epsilon \tag{4}$$

$$[q'Zq]\overset{\beta_2'|x_2''}{\longrightarrow}{}^*\epsilon. \tag{5}$$

Moreover, by construction of P_b, we have

$$[pXq]\overset{a|x'}{\longrightarrow}[p_1Yq'][q'Zq]. \tag{6}$$

Combining (4), (5), and (6), we obtain a finite path in P_b such that $\sigma_2=[pXq]\overset{\beta|x}{\longrightarrow}{}^*\epsilon$.

Likewise, we may show that for any $pX\in Q\Gamma$ and $q\in Q$, $V^P(pX,q)\ge V^{P_b}([pXq])$. Then, $V^P(pX,q)=V^{P_b}([pXq])$, and so $V^P(pX)=\sup_{q\in R}V^{P_b}([pXq])$. ■

Acknowledgment

The authors would like to thank the anonymous reviewers for their invaluable suggestions and comments.

References

[1] L. de Alfaro *et al.*, "Model checking discounted temporal properties," *Theor. Comput. Sci.*, vol. 345, pp. 139–170, 2005.
[2] R. Alur *et al.*, "Analysis of recursive state machines," *ACM Trans. Program. Lang. Syst.*, vol. 27, no. 4, pp. 786–818, 2005.
[3] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," *J. ACM*, vol. 49, no. 5, pp. 672–713, 2002.
[4] K. R. Apt and E. Grädel, *Lectures in Game Theory for Computer Scientists*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[5] P. R. J. Asveld, "Fuzzy context-free languages-part 1: Generalized fuzzy context-free grammars," *Theor. Comput. Sci.*, vol. 347, pp. 167–190, 2005.

[6] P. R. J. Asveld, "Fuzzy context-free languages-part 2: Recognition and parsing algorithms," *Theor. Comput. Sci.*, vol. 347, no. 1/2, pp. 191–213, 2005.

[7] C. Baier and J. P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[8] A. Bouajjani, J. Esparza, and O. Maler, "Reachability analysis of pushdown automata: Application to model-checking," in *Proc. Int. Conf. Concurrency Theory*, 1997, pp. 135–150.

[9] V. Brožek, "Basic model checking problems for stochastic games," Ph.D. thesis, Dept. Faculty Inf., Masaryk Univ., Brno, Czech Republic, 2009.

[10] T. Cachat, "Symbolic strategy synthesis for games on pushdown graphs," in *Proc. Int. Colloquium Automata, Lang. Program.*, 2002, pp. 704–715.

[11] Y. Cao, G. Chen, and E. E. Kerre, "Bisimulations for fuzzy transition systems," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 3, pp. 540–552, Jun. 2010.

[12] Y. Cao, S. X. Sun, H. Wang, and G. Chen, "A behaviorual distance for fuzzy-transition systems," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 4, pp. 735–747, Aug. 2013.

[13] M. Ćirić *et al.*, "Computation of the greatest simulations and bisimulations between fuzzy automata," *Fuzzy Sets Syst.*, vol. 208, pp. 22–42, 2012.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. New York, NY, USA: McGraw-Hill, 2001.

[15] W. L. Deng and D. W. Qiu, "Supervisory control of fuzzy discrete-event systems for simulation equivalence," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 1, pp. 178–192, Feb. 2015.

[16] Z. Ding, J. Ma, and A. Kandel, "Petri net representation of switched fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 16–29, Feb. 2013.

[17] Z. H. Ding, Y. Zhou, and M. C. Zhou, "Modeling self-adaptive software systems by fuzzy rules and Petri nets," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 3, pp. 967–984, Apr. 2018.

[18] X. Y. Du, H. Ying, and F. Lin, "On modeling of fuzzy hybrid systems," *J. Intell. Fuzzy Syst*, vol. 23, pp. 129–141, 2012

[19] J. Esparza, A. Kucera, and S. Schwoon, "Model checking LTL with regular valuations for pushdown systems," *Inf. Comput.*, vol. 186, pp. 355–376, 2003.

[20] J. Esparza, A. Kučera, and R. Mayr, "Model checking probabilistic pushdown automata," *Log. Methods Comput. Sci.*, vol. 2, pp. 1–31, 2006.

[21] K. Etessami and M. Yannakakis, "Recursive Markov decision processes and recursive stochastic games," *J. ACM.*, vol. 62, no. 2, 2015.

[22] A. Finkel, B. Willems, and P. Wolper, "A direct symbolic approach to model checking pushdown systems," *Electron. Notes Theor. Comput. Sci.*, vol. 9, pp. 27–37, 1997.

[23] J. C. Fodor, "Contrapositive symmetry of fuzzy implications," *Fuzzy Sets Syst.*, vol. 69, pp. 141–156, 1995.

[24] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Reading, MA, USA: Addison-Wesley, 1979.

[25] E. P. Klement, R. Mesiar, and E. Pap, *Triangular Norms*. Dordrecht, The Netherlands: Kluwer, 2000.

[26] Y. M. Li, "Approximation and robustness of fuzzy finite automata," *Int. J. Approx. Reason.*, vol. 47, pp. 247–257, 2008.

[27] Y. M. Li and L. J. Li, "Model checking of linear-time properties based on possibility measure," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 5, pp. 842–854, Oct. 2013.

[28] Y. M. Li and Z. Y. Ma, "Quantitative computation tree logic model checking based on generalized possibility measures," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 2034–2047, Dec. 2015.

[29] M. Mohri, "Semiring frameworks and algorithms for shortest-distance problems," *J. Autom. Lang. Combinatorics*, vol. 7, no. 3, pp. 321–350, 2002.

[30] H. Y. Pan, Y. M. Li, Y. Z. Cao, and Z. Y. Ma, "Model checking fuzzy computation tree logic," *Fuzzy Sets Syst.*, vol. 262, pp. 60–77, 2015.

[31] H. Y. Pan, Y. M. Li, Y. Z. Cao, and Z. Y. Ma, "Model checking computation tree logic over finite lattices," *Theor. Comput. Sci.*, vol. 612, pp. 45–62, 2016.

[32] H. Y. Pan, Y. M. Li, Y. Z. Cao, and Z. Y. Ma, "Reachability in fuzzy game graphs," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 4, pp. 972–984, Apr. 2017.

[33] W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*. Cambridge, MA, USA: MIT Press, 1998.

[34] D. W. Qiu, "Automata theory based on quantum logic: Reversibilities and pushdown automata," *Theor. Comput. Sci.*, vol. 386, pp. 38–56, 2007.

[35] O. Serre, "Note on winning positions on pushdown games with ω-regular conditions," *Inf. Process. Lett.* vol. 85, pp. 285–291, 2003.

[36] S. Shoham and O. Grumberg, "Multi-valued model checking games," *J. Comput. Syst. Sci.*, vol. 78, no. 2, pp. 354–369, 2005.

[37] F. Song and T. Touili, "Efficient CTL model-checking for pushdown systems," *Theor. Comput. Sci.*, vol. 549, pp. 127–145, 2014.

[38] I. Walukiewicz, "Pushdown processes: Games and model-checking," *Inf. Comput.*, vol. 164, pp. 234–263, 2001.

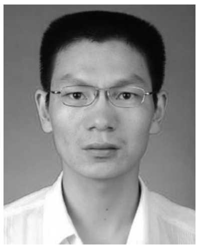
[39] H. Y. Wu and Y. X. Deng, "Logical characterizations of simulation and bisimulation for fuzzy transition systems," *Fuzzy Sets Syst.*, vol. 301, pp. 19–36, 2016.

[40] H. Y. Wu and Y. X. Deng, "Distribution-based behavioural distance for nondeterministic fuzzy transition systems," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 416–429, Apr. 2018.

[41] H. Y. Xing, "Fuzzy pushdown automata," *Fuzzy Sets Syst.*, vol. 158, pp. 1437–1449, 2007.

[42] H. Y. Xing, D. W. Qiu, and F. C. Liu, "Automata theory based on complete residuated lattice-valued logic: Pushdown automata," *Fuzzy Sets Syst.*, vol. 160, pp. 1125–1140, 2009.

[43] M. S. Ying, "A formal model of computing with words," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 5, pp. 640–652, Oct. 2002.



Haiyu Pan received the Ph.D degree in computer science from East China Normal University, Shanghai, China, in 2012.

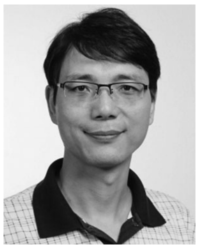
He is currently an Associate Professor of Computer Science with Guilin University of Electronic Technology, Guilin, China. He has authored or coauthored some papers in academic journals, such as the IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Theoretical Computer Science*, *International Journal of Approximate Reasoning*, *Fuzzy Sets and Systems*, and *Fundamenta Informaticae*. His research interests include formal methods and reasoning about uncertainty in artificial intelligence.

Fu Song received the B.S. degree in electronic information science and technology from Ningbo University, Ningbo, China, in 2006, the M.S. degree in software engineering from East China Normal University, Shanghai, China, in 2009, and the Ph.D. degree in computer science from the University Paris-Diderot (Paris 7), Paris, France, in 2013.

He is currently an Assistant Professor with ShanghaiTech University, Shanghai, China. His research interests include formal methods and computer security, especially about automata, logic, model checking, and program analysis.

Yongzhi Cao (SM'15) received the B.S. and M.S. degrees from Central China Normal University, Wuhan, China, in 1997 and 2000, respectively, and the Ph.D. degree from Beijing Normal University, Beijing, China, in 2003, all in mathematics.

He is currently a Professor of computer science with Peking University, Beijing, China. From 2003 to 2007, he was a Postdoctoral Researcher with Tsinghua University, Beijing, China, and from 2007 to 2105, he was an Associate Professor of computer science with Peking University. He has authored or coauthored some papers in academic journals, such as the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B: CYBERNETICS, *Theoretical Computer Science*, *Information and Computation*, and *Journal of Computer and System Sciences*. His current research interests include formal methods and reasoning about uncertainty in artificial intelligence.



Junyan Qian received the B.S. degree in electrical automation from the Anhui Polytechnic University, Wuhu, China, in 1996, the M.S. degree in computer science from the Guilin University of Electronic Technology, Guilin, China, in 2000, and the Ph.D. degree in computer science from the Southeast University, Nanjing, China, in 2008.

He is currently a Professor of computer science with Guilin University of Electronic Technology. His research interests include formal verification, optimization algorithm, and reconfigurable VLSI design.