

# Featured Games

Uli Fahrenberg  
École polytechnique, France  
uli@lix.polytechnique.fr

Axel Legay  
Université Catholique de Louvain, Belgium  
axel.legay@uclouvain.be

## ABSTRACT

Feature-based SPL analysis and family-based model checking have seen rapid development. Many model checking problems can be reduced to two-player games on finite graphs. A prominent example is mu-calculus model checking, which is generally done by translating to parity games, but also many quantitative model-checking problems can be reduced to (quantitative) games.

In their FASE'20 paper, ter Beek et al. introduce parity games with variability in order to develop family-based mu-calculus model checking of featured transition systems. We generalize their model to general featured games and show how these may be analysed in a family-based manner.

We introduce featured reachability games, featured minimum reachability games, featured discounted games, featured energy games, and featured parity games. We show how to compute winners and values of such games in a family-based manner. We also show that all these featured games admit optimal featured strategies, which project to optimal strategies for any product. Further, we develop family-based algorithms, using late splitting, to compute winners, values, and optimal strategies for all the featured games we have introduced.

## ACM Reference Format:

Uli Fahrenberg and Axel Legay. 2020. Featured Games. In *Proceedings of SPLC*. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

Managing variability between products is a key challenge in software product line (SPL) engineering. In feature-based SPL analysis, products are abstracted into features, so that any product is a combination of a set of given features, specifying characteristics that are present or absent in the particular product.

*Featured transition systems* (FTS), introduced by Classen et al. [14], are high-level representations of SPL which allow for model checking of qualitative and quantitative properties of SPL. Model checking is an established technique for verifying the behavior of complex systems, and SPL model checking is an active research subject [9–11, 14, 15, 29, 34, 36, 37].

The number of products in an SPL grows exponentially with the number of features, hence model checking each individual product

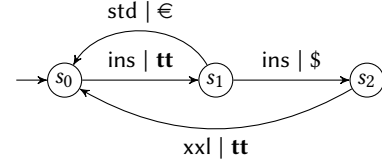


Figure 1: FTS model  $S$  of a simple coffee machine SPL.

$$\varphi = \nu X. \mu Y. \underbrace{\left( \langle \langle \text{ins} \rangle Y \vee \langle \text{xxl} \rangle Y \rangle \vee \langle \text{std} \rangle X \right)}_{\varphi_3} \quad \varphi_2$$

Figure 2:  $\mu$ -calculus specification for  $S$ .

is prohibitive. Thus, *family-based model checking* has been introduced in [14], allowing for the simultaneous verification of all products. The family-based approach has seen rapid development [1, 10–13, 34, 38] and has been extended to conformance model checking [16], abstraction-based model checking [18, 20, 21], real-time formalisms [4, 19], probabilistic systems [9, 18, 33, 35], and quantitative model checking [23, 30]; see [17] for a recent survey.

Many model checking problems can be reduced to two-player games on finite graphs. A prominent example is  $\mu$ -calculus model checking, which is generally done by translating to parity games [5], but also many quantitative model-checking problems can be reduced to (quantitative) games, see [22, 25].

In their recent paper [38], ter Beek et al. introduce a procedure for family-based  $\mu$ -calculus model checking of FTS. They define a translation to *parity games with variability* and then develop an algorithm for family-based analysis of such games. We give an example inspired by [38]. Figure 1 shows a toy model  $S$  of a coffee machine with feature set  $\{\epsilon, \$\}$  and three products  $\{\epsilon\}$ ,  $\{\$\}$ , and  $\{\epsilon, \$\}$ . The machine can accept coins at the ins transitions, deliver regular coffee at the std transition, and hand out extra large coffee at the xxl transition; but the std transition is only enabled if the  $\epsilon$  feature is present, and the second ins transition exists only if the  $\$$  feature is present.

In Fig. 2 we define a  $\mu$ -calculus formula  $\varphi$  which expresses the property that there exists an infinite run of the system along which infinitely many regular coffees are delivered. We quickly recall the translation introduced in [38], which is a feature-enriched version of the standard translation [5] from  $\mu$ -calculus model checking to parity games.

Let  $N$  be a set of features,  $\Sigma$  a set of actions, and  $F = (S, i, T, \gamma)$  an FTS, with states  $S$ , initial state  $i \in S$ , transitions  $T \subseteq S \times \Sigma \times S$ , and feature guards  $\gamma : T \rightarrow \mathbb{B}(N)$ , the set of boolean expressions over  $N$ . Let  $\varphi$  be a  $\mu$ -calculus formula and denote by  $\text{sub}(\varphi)$  the set of subformulas of  $\varphi$  (including  $\varphi$  itself). The featured parity game

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

SPLC, October 2020, Montreal, Canada

© 2020 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

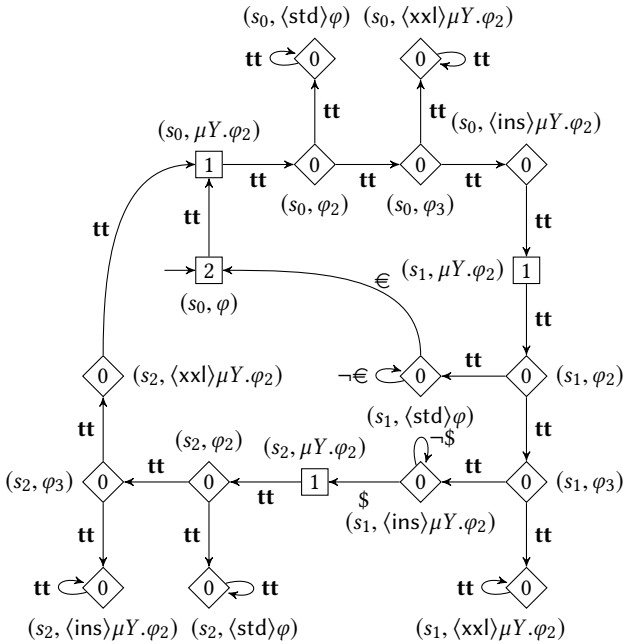
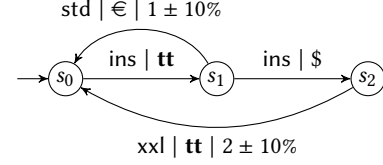
**Figure 3: Translation to featured parity games from [38].**

State	Owner	Successors	Priority
$(s, \mathbf{ff})$	1	$\emptyset$	0
$(s, \mathbf{tt})$	2	$\emptyset$	0
$(s, \psi_1 \vee \psi_2)$	1	$\{(s, \psi_1)/\mathbf{tt}, (s, \psi_2)/\mathbf{tt}\}$	0
$(s, \psi_1 \wedge \psi_2)$	2	$\{(s, \psi_1)/\mathbf{tt}, (s, \psi_2)/\mathbf{tt}\}$	0
$(s, \langle a \rangle \psi)$	1	$\{(s', \psi)/_Y \mid s \xrightarrow{a}_Y s'\}$	0
$(s, [a] \psi)$	2	$\{(s', \psi)/_Y \mid s \xrightarrow{a}_Y s'\}$	0
$(s, \nu X. \psi)$	2	$\{(s, \psi[X := \nu x. \psi])/\mathbf{tt}\}$	$2\lfloor \text{ad}_\psi(X)/2 \rfloor$
$(s, \mu X. \psi)$	2	$\{(s, \psi[X := \mu x. \psi])/\mathbf{tt}\}$	$2\lfloor \text{ad}_\psi(X)/2 \rfloor + 1$

associated with  $F$  and  $\varphi$  has states  $S \times \text{sub}(\varphi)$ , with initial state  $(i, \varphi)$ , and the owners, priorities and successors of states are given in Fig. 3, where  $\text{ad}_\psi(X)$  denotes the alternation depth of variable  $X$  in formula  $\psi$ .

We show the result of the translation applied to our example in Fig. 4, depicting only the reachable part of the featured parity game. Here, diamond-shaped states are owned by player 1 and box-shaped states by player 2, and the priorities are indicated inside states. Player 1 is said to *win* the game if she can enforce an infinite path through the game graph for which the highest priority occurring infinitely often is *even*. By the properties of the translation [38], player 1 wins the game for a product  $p$  iff the projection  $\text{proj}_p(S)$  satisfies  $\varphi$ ; in our case iff  $\epsilon \in p$ . [38] gives a family-based algorithm for solving featured parity games.

For another example of the use of games, we turn to the *quantitative* setting. Figure 5 displays our toy model of the coffee machine together with approximate annotations for energy consumption: brewing a standard coffee consumes 1 energy unit, plus/minus 10%;

**Figure 4: Featured parity game for checking whether  $S \models \varphi$ .****Figure 5: Coffee machine model  $S$  with energy annotations.**

brewing an extra large coffee consumes  $2 \pm 10\%$  energy units. (Quite naturally, inserting coins does not consume energy.)

We may now inquire about the *robustness* of this SPL: given that the energy annotations are approximate, what are the *long-run* deviations in energy consumption that we should expect, depending on the particular product? As a simple example, one machine might always consume 1.1 energy units at a *std* transition and another always 0.9, so that in an infinite run (*ins*, *std*, *ins*, *std*, ...) the two machines would accumulate a difference in energy consumption of 0.2 every second step.

Taking the standard point of view that *the future is discounted*, we fix a discounting factor  $\lambda < 1$  and multiply differences by  $\lambda$  at each step. For the two runs above, the long-run energy difference would thus evaluate to  $0 + \lambda \cdot 0.2 + \lambda^2 \cdot 0 + \lambda^3 \cdot 0.2 + \dots = 0.2 \frac{\lambda}{1 - \lambda^2}$ , which becomes 9.95 for a standard discounting factor of  $\lambda = 0.99$ .

Following [22], robustness of a model for a product  $p$  may be computed using the  $\lambda$ -discounted bisimulation distance: let  $S_1$  and  $S_2$  be the versions of the projection  $\text{proj}_p(S)$  with the minimal, resp. maximal, energy consumption on every transition and write  $S_i = \{s_0^i, s_1^i, s_2^i\}$  for  $i \in \{1, 2\}$ , then the discounted bisimulation distance between  $S_1$  and  $S_2$  is  $d(s_0^1, s_0^2)$ , where  $d : S_1 \times S_2 \rightarrow \mathbb{R}$  is the unique solution to the equation system given by

$$d(s^1, s^2) = \max \begin{cases} \max_{s^1 \xrightarrow{a} t^1} \min_{s^2 \xrightarrow{a} t^2} |x - y| + \lambda d(t^1, t^2) \\ \max_{s^2 \xrightarrow{a} t^2} \min_{s^1 \xrightarrow{a} t^1} |x - y| + \lambda d(t^1, t^2) \end{cases}$$

for all  $s^1 \in S_1, s^2 \in S_2$ . (Here  $s \xrightarrow{a} t$  indicates a transition from  $s$  to  $t$  with label  $a$  and energy consumption  $x$ .)

In [25] it is shown that  $\lambda$ -discounted bisimulation distances may be computed by translating to  $\sqrt{\lambda}$ -discounted games [40]. We recall the translation and extend it to FTS. Let  $F_1 = (S_1, i_1, T_1, \gamma_1)$ ,  $F_2 = (S_2, i_2, T_2, \gamma_2)$  be weighted FTS, with transitions  $T_j \subseteq S_j \times \Sigma \times \mathbb{Q} \times S_j$ . The states of the game for computing the  $\lambda$ -discounted bisimulation distance between  $F_1$  and  $F_2$  are  $V_1 = S_1 \times S_2$  (owned by player 1) and  $V_2 = S_1 \times S_2 \times \Sigma \times \mathbb{Q} \times \{1, 2\}$ , with initial state  $i = (i_1, i_2) \in V_1$ . The transitions of the game are of four types:

$$\begin{aligned} &\{(s_1, s_2) \xrightarrow{\varphi}^0 (s'_1, s'_2, a, x, 1) \mid (s_1, a, x, s'_1)/\varphi \in T_1\} \\ &\{(s_1, s_2) \xrightarrow{\varphi}^0 (s_1, s'_2, b, x, 2) \mid (s_2, b, x, s'_2)/\varphi \in T_2\} \\ &\{(s'_1, s_2, a, x, 1) \xrightarrow{\varphi}^{\lambda^{-1/2}|a-b|} (s'_1, s'_2) \mid (s_2, b, x, s'_2)/\varphi \in T_2\} \\ &\{(s_1, s'_2, b, x, 2) \xrightarrow{\varphi}^{\lambda^{-1/2}|a-b|} (s'_1, s'_2) \mid (s_1, a, x, s'_1)/\varphi \in T_1\} \end{aligned}$$

We show the result of the translation applied to our example in Fig. 6, where we have omitted some states and transitions due to symmetry. For  $\lambda = 0.99$  and  $p = \{\epsilon, \$\}$ , the distance evaluates to 13.2.

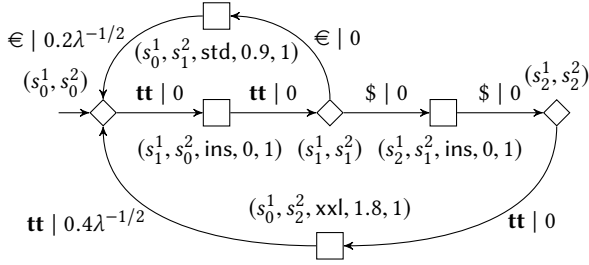


Figure 6: Game for computing discounted distance.

Games are also important in *controller synthesis*: the problem of generating controllers for discrete event systems [28, 32]. In this setting, the model is a game in which player 1 is the *controller* and player 2 the *environment*, and then the task is to find a *strategy* for the controller which ensures a given property one wishes to enforce.

We give a simple example in Fig. 7, inspired by [2]. This is a toy model of a mars robot which collects rocks, with an operations cycle consisting of charging its batteries, searching for rocks, collecting a rock, and depositing the rock in a container. Charging the battery adds 3 energy units to its battery; unless the ext feature is present, in which case the charge transition may add 5 energy units. Searching and depositing both cost 1 energy unit, as does collecting a small rock. If the big feature is present, then also big rocks may be collected, with an energy consumption of 3. The size of a collected rock is controlled by the environment.

The property we wish to enforce is that the system have an infinite run in which the battery charge never drops below 0. That is, player 1 should have a strategy of choosing her transitions so that no matter the behavior of player 2, battery charge never goes negative. A simple analysis shows that this is the case precisely for all products which satisfy the formula  $\neg \text{big} \vee \text{ext}$ : if feature big is not present, then the search-collect-deposit cycle always consumes 3 energy units which can be recharged also without the ext feature; and if both big and ext are present, then charging 5 energy units ensures that also big rocks can be deposited.

In this paper we concern ourselves with several types of games which have been used in model checking and controller synthesis. We lift these games to featured versions useful in an SPL context, and we show how to compute their values and optimal strategies

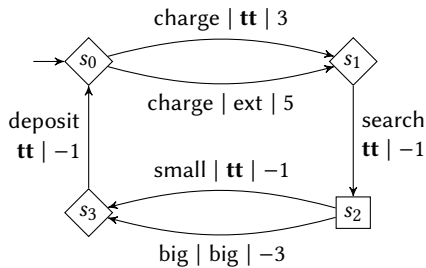


Figure 7: A simple energy game

in a family-based manner using late splitting [1]. We treat the following types of games:

- reachability games;
- minimum reachability games;
- discounted games;
- energy games;
- parity games.

Our treatment is based on the computation of *attractors*, which in general is the most efficient technique for solving games and typically gives rise to (pseudo)polynomial algorithms. Our first main contribution is showing how to lift attractor computations to the featured setting, in Sections 2 through 6. (Compared to [38], we use a different algorithm for parity games which is known to be more efficient [39].)

Our second main contribution, in Section 7, is the family-based computation of optimal strategies. We show that in all featured games considered here, optimal featured strategies may be found during the attractor computation, and these project to optimal strategies for *any* product.

Finally, Section 8 exhibits our third main contribution: family-based algorithms, using late splitting, to compute attractors for all the featured games we have introduced. Due to space restrictions, most proofs had to be omitted from this paper; these can be found in the long version [24]

## 2 FEATURED REACHABILITY GAMES

### 2.1 Reachability Games

A *game structure*  $G = (S_1, S_2, i, F, T)$  consists of two disjoint sets  $S = S_1 \sqcup S_2$  of *states*, *initial* and *accepting states*  $i \in S$ ,  $F \subseteq S$ , and *transitions*  $T \subseteq S \times S$ . For simplicity we assume  $G$  to be *non-blocking*, so that for all  $s \in S$  there exists  $s' \in S$  for which  $(s, s') \in T$ .

As customary, we write  $s \rightarrow s'$  to indicate that  $(s, s') \in T$ . Intuitively, a game on a game structure  $G$  as above is played by two players, player 1 and player 2, taking turns to move a token along the directed graph with vertices  $S$  and edges  $T$ . We to make this intuition precise.

A *finite path* in  $G$  is a finite sequence  $\pi = (s_1, \dots, s_k)$  in  $S$  such that  $s_i \rightarrow s_{i+1}$  for all  $i = 1, \dots, k-1$ . The set of finite paths in  $G$  is denoted  $\text{fPaths}(G)$ . The *end state* of a path  $\pi = (s_1, \dots, s_k)$  is  $\text{end}(\pi) = s_k$ . An *infinite path* in  $G$  is an infinite sequence  $(s_1, s_2, \dots)$  in  $S$  such that  $s_i \rightarrow s_{i+1}$  for all  $i \geq 1$ , and the set of these is denoted  $\text{iPaths}(G)$ .

The *configurations* for player  $i$ , for  $i \in \{1, 2\}$ , are  $\text{Conf}_i = \{\pi \in \text{fPaths}(G) \mid \text{end}(\pi) \in S_i\}$ . A *strategy* for player  $i$  is a function  $\theta : \text{Conf}_i \rightarrow S$  such that for all  $\pi \in \text{Conf}_i$ ,  $\text{end}(\pi) \rightarrow \theta(\pi)$ . The set of strategies for player  $i$  is denoted  $\Theta_i$ .

Any pair of strategies  $\theta_1 \in \Theta_1$ ,  $\theta_2 \in \Theta_2$  induces a unique infinite path  $\text{out}(\theta_1, \theta_2) = (s_1, s_2, \dots) \in \text{iPaths}(G)$ , called the *outcome* of the pair  $(\theta_1, \theta_2)$  and defined inductively as follows:

$$s_1 = i \quad s_{2k} = \theta_1(s_1, \dots, s_{2k-1}) \quad s_{2k+1} = \theta_2(s_1, \dots, s_{2k})$$

Note that the outcome is indeed infinite due to our non-blocking assumption.

The *reachability game* on a game structure  $G = (S_1, S_2, i, F, T)$  is to decide whether there exists a strategy  $\theta_1 \in \Theta_1$  such that for all  $\theta_2 \in \Theta_2$ , writing  $\text{out}(\theta_1, \theta_2) = (s_1, s_2, \dots)$ , there is an index  $k \geq 1$

for which  $s_k \in F$ . In the affirmative case, player 1 is said to *win* the reachability game.

In order to solve reachability games, one introduces a notion of *game attractor*  $\text{attr} : (S \rightarrow \mathbb{B}) \rightarrow (S \rightarrow \mathbb{B})$ , where  $\mathbb{B} = \{\mathbf{ff}, \mathbf{tt}\}$  is the boolean lattice, defined for any  $U : S \rightarrow \mathbb{B}$  by

$$\text{attr}(U)(s) = \begin{cases} \bigvee_{s \rightarrow s'} U(s') & \text{if } s \in S_1, \\ \bigwedge_{s \rightarrow s'} U(s') & \text{if } s \in S_2. \end{cases}$$

Hence  $\text{attr}(U)(s)$  is true precisely if *there exists* a player-1 transition to a state  $s'$  for which  $U(s') = \mathbf{tt}$ , or if it holds *for all* player-2 transitions  $s \rightarrow s'$  that  $U(s') = \mathbf{tt}$ .

Let  $\text{attr}^* = \text{id} \vee \text{attr} \vee \text{attr}^2 \vee \dots$ , where  $\vee$  is the supremum operator on the complete lattice  $S \rightarrow \mathbb{B}$ . The following is then easy to see.

LEMMA 2.1. *Let  $G = (S_1, S_2, i, F, T)$  be a game structure and define  $I : S \rightarrow \mathbb{B}$  by  $I(s) = \mathbf{tt}$  iff  $s \in F$ . Player 1 wins the reachability game in  $G$  iff  $\text{attr}^*(I)(i) = \mathbf{tt}$ .*

The operator  $\text{attr}$  is monotone on the complete lattice  $S \rightarrow \mathbb{B}$ , thus  $\text{attr}^*$  can be computed using a fixed-point algorithm, in time quadratic in the size of  $S$ . Hence reachability games can be decided in polynomial time.

## 2.2 Featured Reachability Games

Let  $N$  be a finite set of *features* and  $px \subseteq 2^N$  a set of *products* over  $N$ . A *feature guard* is a Boolean expression over  $N$ , and we denote the set of these by  $\mathbb{B}(N)$ . We write  $p \models \gamma$  if  $p \in px$  satisfies  $\gamma \in \mathbb{B}(N)$ . For each  $p \in px$  let  $\gamma_p \in \mathbb{B}(N)$  be its *characteristic formula* satisfying that  $p' \models \gamma_p$  iff  $p' = p$ .

A *featured game structure*  $G = (S_1, S_2, i, F, T, \gamma)$  consists of a game structure  $(S_1, S_2, i, F, T)$  together with a mapping  $\gamma : T \rightarrow \mathbb{B}(N)$ . We also assume our featured game structures to be non-blocking, in the sense that for all  $s \in S$  and all  $p \in px$ , there exists  $(s, s') \in T$  with  $p \models \gamma(s, s')$ .

The *projection* of a featured game structure  $G$  as above to a product  $p \in px$  is the game structure  $\text{proj}_p(G) = (S_1, S_2, i, F, T')$  with  $T' = \{t \in T \mid p \models \gamma(t)\}$ . All projections of non-blocking featured game structures are again non-blocking.

We are interested in solving the reachability game for each product  $p \in px$ , but in a family-based manner. We will thus compute a function  $\mathbb{B}(N) \rightarrow \mathbb{B}$  which for each feature expression indicates whether player 1 wins the reachability game on  $G$ .

To this end, define the *featured attractor*  $\text{fattr} : (S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B})) \rightarrow (S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B}))$  by

$$\text{fattr}(U)(s)(\varphi) = \begin{cases} \bigvee_{s \rightarrow s'} U(s')(\gamma((s, s')) \wedge \varphi) & \text{if } s \in S_1, \\ \bigwedge_{s \rightarrow s'} U(s')(\gamma((s, s')) \wedge \varphi) & \text{if } s \in S_2. \end{cases}$$

and let  $\text{fattr}^* = \text{id} \vee \text{fattr} \vee \text{fattr}^2 \vee \dots$ , the supremum in the complete lattice  $S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B})$ .

THEOREM 2.2. *Let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured game structure and define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B})$  by  $I(s)(\varphi) = \mathbf{tt}$  if  $s \in F$ ;  $\mathbf{ff}$  if  $s \notin F$ . Let  $p \in px$ , then Player 1 wins the reachability game in  $\text{proj}_p(G)$  iff  $\text{fattr}^*(I)(i)(\gamma_p) = \mathbf{tt}$ .*

The operator  $\text{fattr}$  is monotone on the complete lattice  $S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B})$ , thus  $\text{fattr}^*$  can be computed using a fixed-point algorithm. In Section 8 we will give an algorithm which uses *guard partitions* [23] and *late splitting* [1] to compute the fixed point.

## 3 FEATURED MINIMUM REACHABILITY

We now enrich the above problem to compute featured *minimum* reachability in *weighted* game structures.

### 3.1 Minimum Reachability Games

A *weighted game structure*  $G = (S_1, S_2, i, F, T)$  consists of two disjoint sets  $S = S_1 \sqcup S_2$  of states, initial and accepting states  $i \in S$ ,  $F \subseteq S$ , and transitions  $T \subseteq S \times \mathbb{N} \times S$ . Note that all weights are non-negative. We also assume our weighted game structures to be non-blocking, and we write  $s \rightarrow_x s'$  to indicate that  $(s, x, s') \in T$ .

Games on such structures are played as before, only now the goal of player 1 is not only to reach a state in  $F$ , but to do so as cheaply as possible. Let us make this precise. A path in  $G$  is now a sequence  $\pi = (s_1, x_1, s_2, x_2, \dots)$  such that  $s_i \rightarrow_{x_i} s_{i+1}$  for all  $i = 1$ . The notion of configuration is unchanged, and a strategy for player  $i$  is now a function  $\theta : \text{Conf}_i \rightarrow \mathbb{N} \times S$  such that for all  $\pi \in \text{Conf}_i$ ,  $\text{end}(\pi) \rightarrow_{\theta(\pi)_1} \theta(\pi)_2$ , where  $\theta(\pi) = (\theta(\pi)_1, \theta(\pi)_2)$ . The outcome of a strategy pair is an infinite path  $(s_1, x_1, s_2, x_2, \dots) \in \text{iPaths}(G)$  defined as expected.

The *reachability value* of an infinite path  $\pi = (s_1, x_1, s_2, x_2, \dots)$  is defined to be  $\text{val}_R(\pi) = \min\{x_1 + \dots + x_{k-1} \mid s_k \in F\}$ , where  $\min \emptyset = \infty$  by convention, and the *minimum reachability value* of  $G$  is  $\text{val}_R(G) = \inf_{\theta_1 \in \Theta_1} \sup_{\theta_2 \in \Theta_2} \text{val}_R(\text{out}(\theta_1, \theta_2))$ . That is,  $\text{val}_R(\text{out}(\theta_1, \theta_2))$  is the minimum sum of weights along any accepting finite path, and the goal of player 1 is to minimize this value.

In order to compute minimum reachability in a weighted game structure  $G$ , define the *weighted attractor*  $\text{wattr} : (S \rightarrow \mathbb{N} \cup \{\infty\}) \rightarrow (S \rightarrow \mathbb{N} \cup \{\infty\})$  by

$$\text{wattr}(U)(s) = \begin{cases} \min_{s \rightarrow_x s'} x + U(s') & \text{if } s \in S_1, \\ \max_{s \rightarrow_x s'} x + U(s') & \text{if } s \in S_2 \end{cases}$$

and let  $\text{wattr}^* = \min(\text{id}, \text{wattr}, \text{wattr}^2, \dots)$ . The following seems to be folklore; note that it only holds under our assumption that all weights are non-negative. (See [6] for an extension to negative weights.)

LEMMA 3.1. *The minimum reachability value of a weighted game structure  $G = (S_1, S_2, i, F, T)$  is  $\text{val}_R(G) = \text{wattr}^*(I)(i)$ , where  $I : S \rightarrow \mathbb{N} \cup \{\infty\}$  is defined by  $I(s) = 0$  if  $s \in F$ ;  $\infty$  if  $s \notin F$ .*

The operator  $\text{wattr}$  is monotone on the complete lattice of functions  $S \rightarrow \mathbb{N} \cup \{\infty\}$ , thus  $\text{wattr}^*$  can be computed using a fixed-point algorithm, in time quadratic in the size of  $S$  and linear in the maximum of the weights on the transitions of  $G$ . That is, minimum reachability values can be computed in pseudo-polynomial time.

### 3.2 Featured Minimum Reachability Games

A *featured weighted game structure*  $G = (S_1, S_2, i, F, T, \gamma)$  consists of a weighted game structure  $(S_1, S_2, i, F, T)$  together with a mapping  $\gamma : T \rightarrow \mathbb{B}(N)$ . We again assume our featured weighted game structures to be non-blocking. Projections of such structures to products  $p \in px$  are defined as before.

Define the *featured weighted attractor* operator  $\text{fwattr} : (S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\})) \rightarrow (S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\}))$  by

$$\text{fwattr}(U)(s)(\varphi) = \begin{cases} \min_{s \rightarrow x s'} x + U(s')(\gamma((s, x, s')) \wedge \varphi) & \text{if } s \in S_1, \\ \max_{s \rightarrow x s'} x + U(s')(\gamma((s, x, s')) \wedge \varphi) & \text{if } s \in S_2 \end{cases}$$

and let  $\text{fwattr}^* = \min(\text{id}, \text{fwattr}, \text{fwattr}^2, \dots)$ .

**THEOREM 3.2.** *Let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured weighted game structure and define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\})$  by  $I(s)(\varphi) = 0$  if  $s \in F$ ;  $\infty$  if  $s \notin F$ . Let  $p \in px$ , then the minimum reachability value of  $\text{proj}_p(G)$  is  $\text{val}_R(\text{proj}_p(G)) = \text{fwattr}^*(I)(i)(\gamma_p)$ .*

## 4 FEATURED DISCOUNTED GAMES

### 4.1 Discounted Games

We now turn our attention towards *discounted* games. These are also played on weighted game structures, but now the accepting states are ignored, and the restriction on non-negativity of weights can be lifted. That is, we are now working with weighted game structures  $G = (S_1, S_2, i, T)$  with  $T \subseteq S \times \mathbb{Z} \times S$ .

The notions of configurations, strategies, and outcome remain unchanged from the previous section. Let  $0 < \lambda < 1$  be a real number, called the *discounting factor* of the game. The *discounted value* of an infinite path  $\pi = (s_1, x_1, s_2, x_2, \dots)$  is  $\text{val}_\lambda(\pi) = x_1 + \lambda x_2 + \lambda^2 x_3 + \dots$ , and the discounted value of a game  $G$  is  $\text{val}_\lambda(G) = \sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} \text{val}_\lambda(\text{out}(\theta_1, \theta_2))$ . That is, the value of a path is the sum of its weights, progressively discounted along its run, and the goal of player 1 is to maximize this value.

The following is a reformulation of a result from [40] in terms of attractors. Define the *discounted attractor*  $\text{dattr} : (S \rightarrow \mathbb{R}) \rightarrow (S \rightarrow \mathbb{R})$  by

$$\text{dattr}(U)(s) = \begin{cases} \max_{s \rightarrow x s'} x + \lambda U(s') & \text{if } s \in S_1, \\ \min_{s \rightarrow x s'} x + \lambda U(s') & \text{if } s \in S_2. \end{cases}$$

**LEMMA 4.1.** *Let  $G = (S_1, S_2, i, T)$  be a weighted game structure. The equation system  $V = \text{dattr}(V)$  has a unique solution  $\text{dattr}^*$ , and the discounted value of  $G$  is  $\text{val}_\lambda(G) = \text{dattr}^*(i)$ .*

### 4.2 Featured Discounted Games

Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure. Define the *featured discounted attractor*  $\text{fdattr} : (S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{R})) \rightarrow (S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{R}))$  by

$$\text{fdattr}(U)(s)(\varphi) = \begin{cases} \max_{s \rightarrow x s'} x + \lambda U(s')(\gamma((s, x, s')) \wedge \varphi) & \text{if } s \in S_1, \\ \min_{s \rightarrow x s'} x + \lambda U(s')(\gamma((s, x, s')) \wedge \varphi) & \text{if } s \in S_2. \end{cases}$$

**THEOREM 4.2.** *Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure. The equation system  $V = \text{fdattr}(V)$  has a unique solution  $\text{fdattr}^*$ , and for any  $p \in px$ , the discounted value of  $\text{proj}_p(G)$  is  $\text{val}_\lambda(\text{proj}_p(G)) = \text{fdattr}^*(i)(\gamma_p)$ .*

*Example.* We show the computation of  $\text{fdattr}^*$  for the example from Fig. 6; recall that this is a  $\sqrt{\lambda}$ -discounted game. For any  $\varphi \in \mathbb{B}(N)$ , and writing  $i = ((s_0^1, s_0^2))$ , we have

$$\begin{aligned} \text{fdattr}^*(i)(\varphi) &= \sqrt{\lambda} \text{fdattr}^*((s_0^1, s_0^2, \text{ins}))(\varphi) \\ &= \sqrt{\lambda}^2 \text{fdattr}^*((s_1^1, s_1^2))(\varphi) \end{aligned}$$

and, skipping computations for player-2 states from now,

$$\begin{aligned} &= \max \begin{cases} \sqrt{\lambda}^3 \cdot 0.2 \frac{1}{\sqrt{\lambda}} + \sqrt{\lambda}^4 \text{fdattr}^*(i)(\varphi \wedge \epsilon) \\ \sqrt{\lambda}^4 \text{fdattr}^*((s_2^1, s_2^2))(\varphi \wedge \$) \end{cases} \\ &= \max \begin{cases} \sqrt{\lambda}^3 \cdot 0.2 \frac{1}{\sqrt{\lambda}} + \sqrt{\lambda}^4 \text{fdattr}^*(i)(\varphi \wedge \epsilon) \\ \sqrt{\lambda}^5 \cdot 0.4 \frac{1}{\sqrt{\lambda}} + \sqrt{\lambda}^6 \text{fdattr}^*(i)(\varphi \wedge \$) \end{cases} \\ &= \max \begin{cases} 0.2\lambda + \lambda^2 \text{fdattr}^*(i)(\varphi \wedge \epsilon) \\ 0.4\lambda^2 + \lambda^3 \text{fdattr}^*(i)(\varphi \wedge \$) \end{cases} \end{aligned}$$

For  $p = \{\epsilon\}$ , we have  $\text{fdattr}^*(i)(\gamma_{\{\epsilon\}}) = 0.2\lambda + \lambda^2 \text{fdattr}^*(i)(\gamma_{\{\epsilon\}})$ , hence  $\text{val}_\lambda(\text{proj}_{\{\epsilon\}}(G)) = 0.2 \frac{\lambda}{1-\lambda^2}$ . Given that  $(\text{ins}, \text{std})^\omega$  is the only infinite run in the projection  $\text{proj}_{\{\epsilon\}}(S)$  of the original model, this is as expected.

For  $p = \{\$, \}$ , the equation simplifies to  $\text{fdattr}^*(i)(\gamma_{\{\$, \}}) = 0.4\lambda^2 + \lambda^3 \text{fdattr}^*(i)(\gamma_{\{\$, \}})$ , hence  $\text{val}_\lambda(\text{proj}_{\{\$, \}}(G)) = 0.4 \frac{\lambda^2}{1-\lambda^3}$ . For  $p = \{\epsilon, \$\}$ , no simplifications are possible; for  $\lambda = 0.99$  a standard fixed-point iteration yields  $\text{val}_\lambda(\text{proj}_{\{\epsilon, \$\}}(G)) = 13.2$ .

## 5 FEATURED ENERGY GAMES

### 5.1 Energy Games

Energy games are played on the same type of weighted game structures as the discounted games of the previous section, and also the notions of configurations, strategies, and outcome remain unchanged.

Let  $v_0 \in \mathbb{N}$ . An infinite path  $(s_1, x_1, s_2, x_2, \dots) \in \text{iPaths}(G)$  in a weighted game structure  $G$  is *energy positive with initial credit*  $v_0$  if all finite sums  $v_0 + x_1, v_0 + x_1 + x_2, \dots$  are non-negative; that is, if  $v_0 + \sum_{i=1}^k x_i \geq 0$  for all  $k \geq 1$ . The *energy game* on  $G$  with initial credit  $v_0$  is to decide whether there exists a strategy  $\theta_1 \in \Theta_1$  such that for all  $\theta_2 \in \Theta_2$ ,  $\text{out}((\theta_1, \theta_2))$  is energy positive with initial credit  $v_0$ .

The following procedure, first discovered in [7], can be used to solve energy games. Let  $G = (S_1, S_2, i, T)$  be a weighted game structure and define  $M = \sum_{s \in S} \max(\{0\} \cup \{-x \mid (s, x, s') \in T\})$ . Let  $W = \{0, \dots, M, \top\}$ , where  $\top$  is the greatest element, and define an operation  $\ominus : W \times \mathbb{Z} \rightarrow W$  by  $x \ominus y = \max(0, x - y)$  if  $x \neq \top$  and  $x - y \leq M$ ;  $\top$  otherwise.

Now define the *energy attractor*  $\text{eattr} : (S \rightarrow W) \rightarrow (S \rightarrow W)$  by

$$\text{eattr}(U)(s) = \begin{cases} \min_{s \rightarrow x s'} U(s') \ominus x & \text{if } s \in S_1, \\ \max_{s \rightarrow x s'} U(s') \ominus x & \text{if } s \in S_2 \end{cases}$$

and let  $\text{eattr}^* = \max(\text{id}, \text{eattr}, \text{eattr}^2, \dots)$ . The following is proven in [7] which also shows that energy games can be decided in pseudo-polynomial time.

**LEMMA 5.1.** *Let  $G = (S_1, S_2, i, T)$  be a weighted game structure and  $v_0 \in \mathbb{N}$ . Player 1 wins the energy game on  $G$  with initial credit  $v_0$  iff  $v_0 \geq \text{eattr}^*(I)(i)$ , where  $I : S \rightarrow W$  is defined by  $I(s) = 0$  for all  $s \in S$ .*

### 5.2 Featured Energy Games

Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure. Define the *featured energy attractor*  $\text{feattr} : (S \rightarrow (\mathbb{B}(N) \rightarrow W)) \rightarrow$

$(S \rightarrow (\mathbb{B}(N) \rightarrow W))$  by

$$\text{feattr}(U)(s)(\varphi) = \begin{cases} \min_{s \rightarrow s'} U(s')(\gamma((s, x, s')) \wedge \varphi) \ominus x & \text{if } s \in S_1, \\ \max_{s \rightarrow s'} U(s')(\gamma((s, x, s')) \wedge \varphi) \ominus x & \text{if } s \in S_2 \end{cases}$$

and let  $\text{feattr}^* = \max(\text{id}, \text{feattr}, \text{feattr}^2, \dots)$ .

**THEOREM 5.2.** *Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure,  $v_0 : \mathbb{B}(N) \rightarrow \mathbb{N}$ , and define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow W)$  by  $I(s)(\varphi) = 0$  for all  $s \in S, \varphi \in \mathbb{B}(N)$ . Let  $p \in px$ , then player 1 wins the energy game on  $\text{proj}_p(G)$  with initial credit  $v_0(\gamma_p)$  iff  $v_0(\gamma_p) \geq \text{feattr}^*(I)(i)(\gamma_p)$ .*

*Example.* We show the computation of  $\text{fdattr}^*$  for the example from Fig. 7. Note that the example includes labels on transitions; for energy computations, these are ignored. We have  $M = 3$  and thus  $W = \{0, 1, 2, 3, \top\}$ . Denote  $\text{feattr}^*(I) = f^*$ , then for any  $\varphi \in \mathbb{B}(N)$ ,

$$\begin{aligned} f^*(i)(\varphi) &= \min \begin{cases} f^*(s_1)(\varphi) \ominus 3 \\ f^*(s_1)(\varphi \wedge \text{ext}) \ominus 5 \end{cases} \\ &= \min \begin{cases} (f^*(s_2)(\varphi) \ominus 1) \ominus 3 \\ (f^*(s_2)(\varphi \wedge \text{ext}) \ominus 1) \ominus 5 \end{cases} \\ &= \min \begin{cases} \max \begin{cases} ((f^*(s_3)(\varphi) \ominus 1) \ominus 1) \ominus 3 \\ ((f^*(s_3)(\varphi \wedge \text{big}) \ominus 3) \ominus 1) \ominus 3 \end{cases} \\ \max \begin{cases} ((f^*(s_3)(\varphi \wedge \text{ext}) \ominus 1) \ominus 1) \ominus 5 \\ ((f^*(s_3)(\varphi \wedge \text{ext} \wedge \text{big}) \ominus 3) \ominus 1) \ominus 5 \end{cases} \end{cases} \\ &= \min \begin{cases} \max \begin{cases} (((f^*(i)(\varphi) \ominus 1) \ominus 1) \ominus 1) \ominus 3 \\ (((f^*(i)(\varphi \wedge \text{big}) \ominus 1) \ominus 3) \ominus 1) \ominus 3 \end{cases} \\ \max \begin{cases} (((f^*(i)(\varphi \wedge \text{ext}) \ominus 1) \ominus 1) \ominus 1) \ominus 5 \\ (((f^*(i)(\varphi \wedge \text{ext} \wedge \text{big}) \ominus 1) \ominus 3) \ominus 1) \ominus 5 \end{cases} \end{cases} \end{aligned}$$

For  $p = \emptyset$ , only the first of these four lines contributes to the fixed point, which thus becomes  $f^*(i)(\gamma_\emptyset) = \text{feattr}^*(I)(i)(\gamma_\emptyset) = 0$ . Hence the minimum necessary initial credit in the energy game without any extra features is 0, as expected. For the other three products, standard fixed-point iterations yield  $f^*(i)(\gamma_{\{\text{big}\}}) = \top$  (player 1 cannot win this game) and  $f^*(i)(\gamma_{\{\text{ext}\}}) = f^*(i)(\gamma_{\{\text{ext}, \text{big}\}}) = 0$ .

## 6 FEATURED PARITY GAMES

### 6.1 Parity Games

A *priority game structure*  $G = (S_1, S_2, i, T, p)$  is a game structure (without weights) together with a *priority* mapping  $p : S \rightarrow \mathbb{N}$ ; we again assume these to be non-blocking. The notions of configurations, strategies and outcomes remain unchanged.

For an infinite path  $\pi = (s_1, s_2, \dots) \in \text{iPaths}(G)$  let  $\text{prio}(\pi) = \liminf_{n \rightarrow \infty} p(n)$  be the lowest priority which occurs infinitely often in  $\pi$ . The *parity game* on  $G$  is to decide whether there exists a strategy  $\theta_1 \in \Theta_1$  such that for all  $\theta_2 \in \Theta_2$ ,  $\text{prio}(\text{out}(\theta_1, \theta_2))$  is an even number.

Note that this is, thus, a *minimum* parity game, whereas the game we exposed in the introduction was a *maximum* parity game. This unfortunate dissonance between model checking and game

theory, which we choose to embrace rather than fix here, can easily be overcome by inverting all priorities and then adding their former maximum.

The following procedure for solving minimum parity games was discovered in [27]. Let  $G = (S_1, S_2, i, T, p)$  be a priority game structure and  $d = \max\{p(s) \mid s \in S\}$ . For every  $i \in \{0, \dots, d\}$  let  $p_i = |\{s \in S \mid p(s) = i\}|$  be the number of states with priority  $i$  and define  $M' \subseteq \mathbb{N}^d$  to be the following (finite) set: if  $d$  is odd, then  $M' = \{0\} \times \{0, \dots, p_1\} \times \{0\} \times \{0, \dots, p_3\} \times \dots \times \{0, \dots, p_d\}$ ; if  $d$  is even, then  $M' = \{0\} \times \{0, \dots, p_1\} \times \{0\} \times \{0, \dots, p_3\} \times \dots \times \{0\}$ .

We need some notation for lexicographic orders on  $\mathbb{N}^d$ . For  $x = (x_1, \dots, x_d), y = (y_1, \dots, y_d) \in \mathbb{N}^d$  and  $k \in \{1, \dots, d\}$ , say that  $x \leq_k y$  if  $x_i \leq y_i$  for all components  $i \in \{1, \dots, k\}$ . Relations  $=_k, <_k, \geq_k$  and  $>_k$  are defined similarly.

Let  $M = M' \cup \{\top\}$ , where  $\top$  is the greatest element in all the orders  $\leq_k$ , and define the relations  $\leq_k$  on  $M$  by  $x \leq_k y$  iff  $x \leq_k y$  if  $k$  is odd;  $x <_k y$  or  $x = y = \top$  if  $k$  is even. Define a function  $\text{prog} : (S \rightarrow M) \times S \times S \rightarrow M$  by  $\text{prog}(U, s, s') = \min\{m \in M \mid m \geq_{p(s)+1} U(s')\}$ .

Now define the *parity attractor*  $\text{pattr} : (S \rightarrow M) \rightarrow (S \rightarrow M)$  by

$$\text{pattr}(U)(s) = \begin{cases} \min_{s \rightarrow s'} \text{prog}(U, s, s') & \text{if } s \in S_1, \\ \max_{s \rightarrow s'} \text{prog}(U, s, s') & \text{if } s \in S_2 \end{cases}$$

and let  $\text{pattr}^* = \max(\text{id}, \text{pattr}, \text{pattr}^2, \dots)$ . The following is shown in [27], together with the fact that parity games are decidable in pseudo-polynomial time.

**LEMMA 6.1.** *Let  $G = (S_1, S_2, i, T, p)$  be a priority game structure and define  $I : S \rightarrow M$  by  $I(s) = (0, \dots, 0)$  for all  $s \in S$ . Player 1 wins the parity game on  $G$  iff  $\text{pattr}^*(I)(i) \neq \top$ .*

### 6.2 Featured Parity Games

A *featured priority game structure*  $G = (S_1, S_2, i, T, p, \gamma)$  consists of a priority game structure  $G = (S_1, S_2, i, T, p)$  together with a mapping  $\gamma : T \rightarrow \mathbb{B}(N)$ . We again assume these to be non-blocking. Let  $d = \max\{p(s) \mid s \in S\}$  and  $M$  be defined as above.

Let  $\text{fprog} : (S \rightarrow (\mathbb{B}(N) \rightarrow M)) \times S \times S \rightarrow (\mathbb{B}(N) \rightarrow M)$  be the function given by  $\text{fprog}(U, s, s')(\varphi) = \min\{m \in M \mid m \geq_{p(s)} U(s')(\varphi)\}$ . Define the *featured parity attractor*  $\text{fpattr} : (S \rightarrow (\mathbb{B}(N) \rightarrow M)) \rightarrow (S \rightarrow (\mathbb{B}(N) \rightarrow M))$  by

$$\text{fpattr}(U)(s)(\varphi) = \begin{cases} \min_{s \rightarrow s'} \text{fprog}(U, s, s')(\gamma((s, s')) \wedge \varphi) & \text{if } s \in S_1, \\ \max_{s \rightarrow s'} \text{fprog}(U, s, s')(\gamma((s, s')) \wedge \varphi) & \text{if } s \in S_2 \end{cases}$$

and let  $\text{fpattr}^* = \max(\text{id}, \text{fpattr}, \text{fpattr}^2, \dots)$ .

**THEOREM 6.2.** *Let  $G = (S_1, S_2, i, T, p, \gamma)$  be a featured priority game structure and define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow M)$  by  $I(s)(\varphi) = (0, \dots, 0)$  for all  $s \in S, \varphi \in \mathbb{B}(N)$ . Let  $p \in px$ , then player 1 wins the parity game on  $\text{proj}_p(G)$  iff  $\text{fpattr}^*(I)(i)(\gamma_p) \neq \top$ .*

## 7 OPTIMAL FEATURED STRATEGIES

A player-1 strategy in a game is said to be *optimal* if it realizes the value of the game against any player-2 strategy. That is, in a game with boolean objective such as reachability, energy, or parity games, an optimal strategy for player 1 ensures that she wins the

game against any player-2 strategy if it is at all possible for her to win the game.

In a game with quantitative objective, such as minimum reachability games or discounted games, an optimal player-1 strategy  $\tilde{\theta}_1$  is one which realizes the value of the game against any player-2 strategy, that is, such that the value  $\sup_{\theta_2 \in \Theta_2} \text{val}_R(\text{out}(\tilde{\theta}_1, \theta_2)) = \inf_{\theta_1 \in \Theta_1} \sup_{\theta_2 \in \Theta_2} \text{val}_R(\text{out}(\theta_1, \theta_2))$  for reachability games;  $\inf_{\theta_2 \in \Theta_2} \text{val}_\lambda(\text{out}(\tilde{\theta}_1, \theta_2)) = \sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} \text{val}_\lambda(\text{out}(\theta_1, \theta_2))$  for discounted games.

We show how to compute optimal player-1 strategies for all featured games introduced in the previous sections.

## 7.1 Featured Reachability Games

Let  $G = (S_1, S_2, i, F, T)$  be a game structure. A player-1 strategy  $\theta_1 \in \Theta_1$  is *memoryless* if it depends only on last states of configurations, that is, if  $\text{end}(\pi) = \text{end}(\pi')$  implies  $\theta_1(\pi) = \theta_1(\pi')$  for all  $\pi, \pi' \in \text{Conf}_1$ . Hence memoryless player-1 strategies are mappings  $S_1 \rightarrow S$ . It is well-known that it suffices to consider memoryless strategies for reachability games.

Define again  $I : S \rightarrow \mathbb{B}$  by  $I(s) = \mathbf{tt}$  iff  $s \in F$ . A memoryless player-1 strategy  $\theta_1 : S_1 \rightarrow S$  is *locally optimal* if, for all  $s \in S_1$ ,  $\text{attr}^*(I)(s) = \text{attr}^*(I)(\theta_1(s))$ ; that is, among all options  $s \rightarrow s'$ , it  $\theta_1(s)$  is such that  $\text{attr}^*(I)(s) = \bigvee_{s \rightarrow s'} \text{attr}^*(I)(s')$  is maximized.

It is well-known that locally optimal strategies are optimal, hence if player 1 wins the reachability game on  $G$ , then she can do so using a locally optimal strategy. Further, such a strategy can be trivially extracted after the computation of  $\text{attr}^*$ , hence optimal player-1 strategies in reachability games can be computed in polynomial time.

Now let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured game structure. We extend the domain of  $\gamma : T \rightarrow \mathbb{B}(N)$  to finite paths in  $G$  by defining  $\gamma((s_1, \dots, s_k)) = \gamma((s_1, s_2)) \wedge \dots \wedge \gamma((s_{k-1}, s_k))$ .

A *featured strategy* for player  $i$ , for  $i \in \{1, 2\}$ , is a function  $\xi_i : \text{Conf}_i \rightarrow (\mathbb{B}(N) \rightarrow S)$  such that for all  $\pi \in \text{Conf}_i$  and  $\varphi \in \mathbb{B}(N)$ ,  $\text{end}(\pi) \rightarrow \xi_i(\pi)(\varphi)$ . The set of featured strategies for player  $i$  is denoted  $\Xi_i$ . We define mappings  $\Xi_i \times \mathbb{B}(N) \rightarrow \Theta_i$ , denoted  $(\xi_i, \varphi) \mapsto \xi_i(\varphi)$  and defined by  $\xi_i(\varphi)(\pi) = \xi_i(\pi)(\varphi)$  for all  $\pi \in \text{Conf}_i$ .

A pair of featured strategies  $\xi_1 \in \Xi_1, \xi_2 \in \Xi_2$  defines a mapping  $\text{out}(\xi_1, \xi_2) : \mathbb{B}(N) \rightarrow \text{iPaths}(G)$  from feature guards to infinite paths in  $G$ , where  $\text{out}(\xi_1, \xi_2)(\varphi) = (s_1, s_2, \dots)$  is given by

$$s_1 = i, \quad s_{2k} = \xi_1(s_1, \dots, s_{2k-1})(\varphi), \quad s_{2k+1} = \xi_2(s_1, \dots, s_{2k})(\varphi).$$

Let  $\varphi \in \mathbb{B}(N)$ . Player 1 *wins the  $\varphi$ -reachability game* if there exists a strategy  $\xi_1 \in \Xi_1$  such that for all  $\xi_2 \in \Xi_2$ , with  $\text{out}(\xi_1, \xi_2)(\varphi) = (s_1, s_2, \dots)$ , there is an index  $k \geq 1$  for which  $s_k \in F$  and  $\varphi \wedge \gamma((s_1, \dots, s_k)) \neq \mathbf{ff}$ .

**LEMMA 7.1.** *Let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured game structure and  $p \in \text{px}$ . Player 1 wins the reachability game in  $\text{proj}_p(G)$  iff she wins the  $\gamma_p$ -reachability game in  $G$ .*

A featured player-1 strategy  $\xi_1 \in \Xi_1$  is *memoryless* if  $\text{end}(\pi) = \text{end}(\pi')$  implies  $\xi_1(\pi) = \xi_1(\pi')$  for all  $\pi, \pi' \in \text{Conf}_1$ . Hence memoryless featured strategies are mappings  $S_1 \rightarrow (\mathbb{B}(N) \rightarrow S)$ .

Define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B})$  by  $I(s)(\varphi) = \mathbf{tt}$  if  $s \in F$ ;  $\mathbf{ff}$  if  $s \notin F$ . A memoryless featured player-1 strategy  $\xi_1 : S_1 \rightarrow (\mathbb{B}(N) \rightarrow S)$

is *locally optimal* if, for all  $s \in S_1$  and  $\varphi \in \mathbb{B}(N)$ ,  $\text{fattr}^*(I)(s)(\varphi) = \text{fattr}^*(I)(\xi_1(s)(\varphi))(\gamma((s, \xi_1(s)(\varphi))) \wedge \varphi)$ .

**THEOREM 7.2.** *Let  $G$  be a featured game structure, then there exists a locally optimal player-1 strategy. Further, if  $\xi_1 \in \Xi_1$  is locally optimal, then  $\xi_1(\gamma_p)$  is optimal in  $\text{proj}_p(G)$  for every  $p \in \text{px}$ .*

## 7.2 Featured Minimum Reachability

Let  $G = (S_1, S_2, i, F, T)$  be a weighted game structure. Memoryless player-1 strategies are now mappings  $\theta_1 : S_1 \rightarrow \mathbb{N} \times S$ . Such a strategy is locally optimal if  $\text{wattr}^*(I)(s) = \theta_1(s)_1 + \text{wattr}^*(I)(\theta_1(s)_2)$  for all  $s \in S_1$ , where  $I : S \rightarrow \mathbb{N}$  is defined by  $I(s) = 0$  if  $s \in F$ ;  $\infty$  if  $s \notin F$ , and  $\theta_1(s) = (\theta_1(s)_1, \theta_1(s)_2)$ .

It is again well-known that locally optimal strategies are optimal, hence optimal player-1 strategies in minimum reachability games can be computed in pseudo-polynomial time.

Now let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured weighted game structure. A featured player- $i$  strategy is a mapping  $\xi_i : \text{Conf}_i \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \times S)$ . The outcome of a pair  $\xi_1, \xi_2$  of featured strategies is again a mapping  $\text{out}(\xi_1, \xi_2) : \mathbb{B}(N) \rightarrow \text{iPaths}(G)$  defined as expected.

The *featured reachability value* of a mapping  $\pi : \mathbb{B}(N) \rightarrow \text{iPaths}(G)$  is the function  $\text{fval}_R(\pi) : \mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\}$  given by  $\text{fval}_R(\pi)(\varphi) = \text{val}_R(\pi(\varphi))$ , and the *featured minimum reachability value* of  $G$  is  $\text{fval}_R(G) = \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \text{fval}_R(\text{out}(\xi_1, \xi_2))$ , where the order in  $\mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\}$  is point-wise.

**LEMMA 7.3.** *For  $G = (S_1, S_2, i, F, T, \gamma)$  any featured weighted game structure and  $p \in \text{px}$ ,  $\text{val}_R(\text{proj}_p(G)) = \text{fval}_R(G)(\gamma_p)$ .*

Define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\})$  by  $I(s)(\varphi) = 0$  if  $s \in F$ ;  $\mathbf{ff}$  if  $s \notin F$ . Memoryless featured player-1 strategies are mappings  $\xi_1 : S_1 \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \times S)$ . Such a strategy is locally optimal if, for all  $s \in S_1$  and  $\varphi \in \mathbb{B}(N)$ ,  $\text{fwattr}^*(I)(s)(\varphi) = \xi_1(s)(\varphi)_1 + \text{fwattr}^*(I)(\xi_1(s)(\varphi)_2)(\gamma((s, \xi_1(s)(\varphi)_1, \xi_1(s)(\varphi)_2)) \wedge \varphi)$ .

**THEOREM 7.4.** *Let  $G$  be a featured weighted game structure, then there exists a locally optimal player-1 strategy. Further, if  $\xi_1 \in \Xi_1$  is locally optimal, then  $\xi_1(\gamma_p)$  is optimal in  $\text{proj}_p(G)$  for every  $p \in \text{px}$ .*

## 7.3 Featured Discounted Games

Let  $G = (S_1, S_2, i, T)$  be a weighted game structure,  $0 < \lambda < 1$ . A memoryless player-1 strategy  $\theta_1 : S_1 \rightarrow \mathbb{Z} \times S$  is locally optimal if  $\text{dattr}^*(s) = \theta_1(s)_1 + \lambda \text{dattr}^*(\theta_1(s)_2)$  for all  $s \in S_1$ . Locally optimal strategies always exist and are optimal [40].

Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure. The *featured discounted value* of a mapping  $\pi : \mathbb{B}(N) \rightarrow \text{iPaths}(G)$  is the function  $\text{fval}_\lambda(\pi) : \mathbb{B}(N) \rightarrow \mathbb{R}$  given by  $\text{fval}_\lambda(\pi)(\varphi) = \text{val}_\lambda(\pi(\varphi))$ . The featured discounted value of  $G$  is  $\text{fval}_\lambda(G) = \sup_{\xi_1 \in \Xi_1} \inf_{\xi_2 \in \Xi_2} \text{fval}_\lambda(\text{out}(\xi_1, \xi_2))$ .

**LEMMA 7.5.** *For  $G = (S_1, S_2, i, T, \gamma)$  any featured weighted game structure and  $p \in \text{px}$ ,  $\text{val}_\lambda(\text{proj}_p(G)) = \text{fval}_\lambda(G)(\gamma_p)$ .*

A memoryless featured player-1 strategy  $\xi_1 : S_1 \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{Z} \times S)$  is locally optimal if, for all  $s \in S_1$  and  $\varphi \in \mathbb{B}(N)$ ,  $\text{fdattr}^*(s)(\varphi) = \xi_1(s)(\varphi)_1 + \lambda \text{fdattr}^*(\xi_1(s)(\varphi)_2)(\gamma((s, \xi_1(s)(\varphi)_1, \xi_1(s)(\varphi)_2)) \wedge \varphi)$ .

**THEOREM 7.6.** *Let  $G$  be a featured weighted game structure, then there exists a locally optimal player-1 strategy. Further, if  $\xi_1 \in \Xi_1$  is locally optimal, then  $\xi_1(\gamma_p)$  is optimal in  $\text{proj}_p(G)$  for every  $p \in \text{px}$ .*

## 7.4 Featured Energy Games

Let  $G = (S_1, S_2, i, T)$  be a weighted game structure and define  $M = \sum_{s \in S} \max(\{0\} \cup \{-x \mid (s, x, s') \in T\})$ ,  $W = \{0, \dots, M, \top\}$ , and  $I : S \rightarrow W$  by  $I(s) = 0$  for all  $s \in S$  as before. A memoryless player-1 strategy  $\theta_1 : S_1 \rightarrow \mathbb{Z} \times S$  is locally optimal if  $\text{eattr}^*(I)(s) = \text{eattr}^*(I)(\theta_1(s)_2) \ominus \theta_1(s)_1$  for all  $s \in S_1$ . If player 1 wins the energy game on  $G$  with initial credit  $v_0 \in \mathbb{N}$ , then she can do so using a locally optimal strategy [7].

Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure,  $v_0 : \mathbb{B}(N) \rightarrow \mathbb{N}$ , and  $\varphi \in \mathbb{B}(N)$ . Player 1 wins the  $\varphi$ -energy game with initial credit  $v_0$  if there exists a featured strategy  $\xi_1 \in \Xi_1$  such that for all  $\xi_2 \in \Xi_2$ ,  $\text{out}(\xi_1, \xi_2)(\varphi)$  is energy positive with initial credit  $v_0(\varphi)$ .

**LEMMA 7.7.** *Let  $G = (S_1, S_2, i, T, \gamma)$  be a featured weighted game structure,  $v_0 : \mathbb{B}(N) \rightarrow \mathbb{N}$ , and  $p \in px$ . Player 1 wins the energy game with initial credit  $v_0(\gamma_p)$  in  $\text{proj}_p(G)$  iff player 1 wins the  $\gamma_p$ -energy game in  $G$  with initial credit  $v_0$ .*

Define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow W)$  by  $I(s)(\varphi) = 0$  for all  $s \in S$ ,  $\varphi \in \mathbb{B}(N)$ . A memoryless featured player-1 strategy  $\xi_1 : S_1 \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{Z} \times S)$  is locally optimal if, for all  $s \in S_1$  and  $\varphi \in \mathbb{B}(N)$ ,  $\text{feattr}^*(I)(s)(\varphi) = \text{feattr}^*(I)(\xi_1(s)(\varphi)_2)(\gamma((s, \xi_1(s)(\varphi)_1, \xi_1(s)(\varphi)_2)) \wedge \varphi) \ominus \xi_1(s)(\varphi)_1$ .

**THEOREM 7.8.** *Let  $G$  be a featured weighted game structure, then there exists a locally optimal player-1 strategy. Further, if  $\xi_1 \in \Xi_1$  is locally optimal, then  $\xi_1(\gamma_p)$  is optimal in  $\text{proj}_p(G)$  for every  $p \in px$ .*

## 7.5 Featured Parity Games

Let  $G = (S_1, S_2, i, T, p)$  be a priority game structure,  $d = \max\{p(s) \mid s \in S\}$  and  $M \subseteq \mathbb{N}^d \cup \{\top\}$  as in Section 6, and define  $I : S \rightarrow M$  by  $I(s) = (0, \dots, 0)$  for all  $s \in S$ . A memoryless player-1 strategy  $\theta_1 : S_1 \rightarrow S$  is locally optimal if  $\text{pattr}^*(I)(s) = \text{prog}(\text{pattr}^*(I), s, \theta_1(s))$  for all  $s \in S_1$ . If player 1 wins the parity game on  $G$ , then she can do so using a locally optimal strategy [27].

Let  $G = (S_1, S_2, i, T, p, \gamma)$  be a featured priority game structure and  $\varphi \in \mathbb{B}(N)$ . Player 1 wins the  $\varphi$ -parity game on  $G$  if there exists a featured strategy  $\xi_1 \in \Xi_1$  such that for all  $\xi_2 \in \Xi_2$ ,  $\text{prio}(\text{out}(\xi_1, \xi_2)(\varphi))$  is an even number.

**LEMMA 7.9.** *Let  $G = (S_1, S_2, i, T, p, \gamma)$  be a featured priority game structure and  $p \in px$ . Player 1 wins the parity game in  $\text{proj}_p(G)$  iff player 1 wins the  $\gamma_p$ -parity game in  $G$ .*

Define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow M)$  by  $I(s)(\varphi) = (0, \dots, 0)$  for all  $s \in S$ ,  $\varphi \in \mathbb{B}(N)$ . A memoryless featured player-1 strategy  $\xi_1 : S_1 \rightarrow (\mathbb{B}(N) \rightarrow S)$  is locally optimal if, for all  $s \in S_1$  and  $\varphi \in \mathbb{B}(N)$ ,  $\text{fpattr}^*(I)(s)(\varphi) = \text{fprog}(\text{fpattr}^*(I), s, \xi_1(s)(\varphi))(\gamma((s, \xi_1(s)(\varphi)) \wedge \varphi))$ .

**THEOREM 7.10.** *Let  $G$  be a featured weighted game structure, then there exists a locally optimal player-1 strategy. Further, if  $\xi_1 \in \Xi_1$  is locally optimal, then  $\xi_1(\gamma_p)$  is optimal in  $\text{proj}_p(G)$  for every  $p \in px$ .*

## 8 SYMBOLIC COMPUTATION

The goal of feature-based analysis is to compute properties of an FTS representation of a SPL for all products at once, and to do so in a family-based way. We have seen that for the various types of games we have treated, values and optimal strategies may be

```

1: function REDUCE( $f : P \rightarrow X$ ):  $P' \rightarrow X$ 
2:    $P', f' \leftarrow \emptyset$ 
3:   while  $P \neq \emptyset$  do
4:     Pick and remove  $\varphi$  from  $P$ 
5:      $x \leftarrow f(\varphi)$ 
6:     for all  $\psi \in P$  do
7:       if  $f(\psi) = x$  then
8:          $\varphi \leftarrow \varphi \vee \psi$ 
9:          $P \leftarrow P \setminus \{\psi\}$ 
10:     $P' \leftarrow P' \cup \{\varphi\}$ 
11:     $f'(\varphi) \leftarrow x$ 
12:   return  $f' : P' \rightarrow X$ 

```

Figure 8: Algorithm which computes canonicalization.

```

1: function LAND( $f_1 : P_1 \rightarrow \mathbb{B}, f_2 : P_2 \rightarrow \mathbb{B}$ ):  $P \rightarrow \mathbb{B}$ 
2:    $P, f \leftarrow \emptyset$ 
3:   for all  $\varphi_1 \in P_1$  do
4:     for all  $\varphi_2 \in P_2$  do
5:       if  $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket \neq \emptyset$  then
6:          $P \leftarrow P \cup \{\varphi_1 \wedge \varphi_2\}$ 
7:          $f(\gamma_1 \wedge \gamma_2) \leftarrow f_1(\gamma_1) \wedge f_2(\gamma_2)$ 
8:   return REDUCE( $f$ )

```

Figure 9: Algorithm for logical and.

```

1: function LOR( $f_1 : P_1 \rightarrow \mathbb{B}, f_2 : P_2 \rightarrow \mathbb{B}$ ):  $P \rightarrow \mathbb{B}$ 
2:    $P, f \leftarrow \emptyset$ 
3:   for all  $\varphi_1 \in P_1$  do
4:     for all  $\varphi_2 \in P_2$  do
5:       if  $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket \neq \emptyset$  then
6:          $P \leftarrow P \cup \{\varphi_1 \wedge \varphi_2\}$ 
7:          $f(\gamma_1 \wedge \gamma_2) \leftarrow f_1(\gamma_1) \vee f_2(\gamma_2)$ 
8:   return REDUCE( $f$ )

```

Figure 10: Algorithm for logical or.

computed by calculating closures of attractors. Hence we expose below feature-based algorithms for calculating these closures.

### 8.1 Featured Reachability Games

Let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured game structure and define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{B})$  by  $I(s)(\varphi) = \mathbf{tt}$  if  $s \in F$ ;  $\mathbf{ff}$  if  $s \notin F$ . Conceptually, the procedure for calculating  $J = \text{fattr}^*(I)$  is a fixed-point algorithm: initialize  $J := I$  and update  $J := J \vee \text{fattr}(J)$  until  $J$  stabilizes.

In order to symbolically represent functions from  $\mathbb{B}(N)$ , we use guard partitions, see also [23]. A *guard partition* of  $px$  is a set  $P \subseteq \mathbb{B}(N)$  such that  $\llbracket \bigvee P \rrbracket = px$ ,  $\llbracket \varphi \rrbracket \neq \emptyset$  for all  $\varphi \in P$ , and  $\llbracket \varphi_1 \rrbracket \cap \llbracket \varphi_2 \rrbracket = \emptyset$  for all  $\varphi_1, \varphi_2 \in P$  with  $\varphi_1 \neq \varphi_2$ . The set of all guard partitions of  $px$  is denoted  $GP \subseteq 2^{\mathbb{B}(N)}$ .

A function  $f : P \rightarrow X$ , for  $P \in GP$  and  $X$  any set, is *canonical* if  $f(\varphi_1) = f(\varphi_2)$  implies  $\varphi_1 = \varphi_2$  for all  $\varphi_1, \varphi_2 \in P$ . A function  $f : P \rightarrow X$  which is *not* canonical may be reduced into an equivalent



```

1: function FATTR( $U : S \rightarrow (P \rightarrow \mathbb{B})$ ):  $S \rightarrow (P' \rightarrow \mathbb{B})$ 
2:    $U' \leftarrow \emptyset$ 
3:   for all  $s \in S$  do
4:      $P'_s, U'_s \leftarrow \emptyset$ 
5:     for all  $s \rightarrow s'$  do
6:        $Q_{s'}, V_{s'} \leftarrow \emptyset$ 
7:       while  $P_{s'} \neq \emptyset$  do
8:         Pick and remove  $\varphi$  from  $P_{s'}$ 
9:          $\psi \leftarrow \gamma((s, s')) \wedge \varphi$ 
10:        if  $\llbracket \psi \rrbracket \neq \emptyset$  then
11:           $Q_{s'} \leftarrow Q_{s'} \cup \{\psi\}$ 
12:           $V_{s'}(\psi) \leftarrow U_{s'}(\varphi)$ 
13:         $Q_{s'} \leftarrow Q_{s'} \cup \{\neg \gamma((s, s'))\}$ 
14:         $V_{s'}(\neg \gamma((s, s'))) \leftarrow \mathbf{ff}$ 
15:        if  $s \in S_1$  then
16:           $U'_s \leftarrow \text{LOR}(U'_s, V_{s'})$ 
17:        if  $s \in S_2$  then
18:           $U'_s \leftarrow \text{LAND}(U'_s, V_{s'})$ 
19:         $U'_s \leftarrow \text{REDUCE}(U'_s)$ 
20:   return  $U' : S \rightarrow (P' \rightarrow \mathbb{B})$ 

```

Figure 11: Computation of fattr.

```

1: function FATTR*( $G = (S_1, S_2, i, F, T, \gamma)$ ):  $S \rightarrow (P \rightarrow \mathbb{B})$ 
2:    $J = \emptyset$ 
3:   for all  $s \in S$  do
4:      $U_s \leftarrow \{\mathbf{tt}\}$ 
5:     if  $s \in F$  then
6:        $J_s(\mathbf{tt}) \leftarrow \mathbf{tt}$ 
7:     else
8:        $J_s(\mathbf{tt}) \leftarrow \mathbf{ff}$ 
9:   repeat
10:      $J_{\text{old}} \leftarrow J$ 
11:      $J \leftarrow \text{LOR}(J, \text{FATTR}(J))$ 
12:   until  $J = J_{\text{old}}$ 
13:   return  $J$ 

```

Figure 12: Fixed-point iteration for fattr\*.

canonical function  $f' : P' \rightarrow X$  using the algorithm shown in Fig. 8. Every function  $\mathbb{B}(N) \rightarrow X$  has a unique representation as a canonical function  $P \rightarrow X$  for some  $P \in GP$ .

The function for featured computation of attractors is shown in Fig. 11. It uses the functions LAND and LOR, shown in Figs. 9 and 10, which compute logical operations on functions  $P \rightarrow \mathbb{B}$ : for  $f_1 : P_1 \rightarrow \mathbb{B}$  and  $f_2 : P_2 \rightarrow \mathbb{B}$ , LAND returns  $f' = f_1 \wedge f_2$ , and LOR returns  $f' = f_1 \vee f_2$ .

The function FATTR in Fig. 11 computes one iteration of fattr for all states  $s \in S$ . It does so by traversing all transitions  $s \rightarrow s'$  (note that  $s'$  might be equal to  $s$ ), restricting the partitions at  $s'$  to  $\gamma((s, s'))$  (line 9), and then computing  $U'_s = \bigvee_{s \rightarrow s'} V_{s'}$  or  $U'_s = \bigwedge_{s \rightarrow s'} V_{s'}$ , depending on whether  $s \in S_1$  or  $s \in S_2$ , in lines 15f. The algorithm for the fixed-point iteration to compute fattr\* is, then, shown in Fig. 12.

```

1: function MIN( $f_1 : P_1 \rightarrow \mathbb{N} \cup \{\infty\}, f_2 : P_2 \rightarrow \mathbb{N} \cup \{\infty\}$ ):
    $P \rightarrow \mathbb{N} \cup \{\infty\}$ 
2:    $P, f \leftarrow \emptyset$ 
3:   for all  $\varphi_1 \in P_1$  do
4:     for all  $\varphi_2 \in P_2$  do
5:       if  $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket \neq \emptyset$  then
6:          $P \leftarrow P \cup \{\varphi_1 \wedge \varphi_2\}$ 
7:          $f(\gamma_1 \wedge \gamma_2) \leftarrow \min(f_1(\gamma_1), f_2(\gamma_2))$ 
8:   return REDUCE( $f$ )

```

Figure 13: Algorithm for minimum.

```

1: function MAX( $f_1 : P_1 \rightarrow \mathbb{N} \cup \{\infty\}, f_2 : P_2 \rightarrow \mathbb{N} \cup \{\infty\}$ ):
    $P \rightarrow \mathbb{N} \cup \{\infty\}$ 
2:    $P, f \leftarrow \emptyset$ 
3:   for all  $\varphi_1 \in P_1$  do
4:     for all  $\varphi_2 \in P_2$  do
5:       if  $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket \neq \emptyset$  then
6:          $P \leftarrow P \cup \{\varphi_1 \wedge \varphi_2\}$ 
7:          $f(\gamma_1 \wedge \gamma_2) \leftarrow \max(f_1(\gamma_1), f_2(\gamma_2))$ 
8:   return REDUCE( $f$ )

```

Figure 14: Algorithm for maximum.

## 8.2 Featured Minimum Reachability Games

Let  $G = (S_1, S_2, i, F, T, \gamma)$  be a featured weighted game structure define  $I : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{N} \cup \{\infty\})$  by  $I(s)(\varphi) = 0$  if  $s \in F$ ;  $\infty$  if  $s \notin F$ . The computation of the fixed point  $\text{fwattr}^*(I)$  is similar to the one in the previous section and shown in Figs. 13 through 16. In the algorithm for fwattr (Fig. 15), line 12 now adds the weights of the respective transitions, and the logical operations have been replaced by maximum and minimum (Figs. 13 and 14).

## 8.3 Featured Discounted Games

The algorithms for computing values of featured discounted games are shown in Figs. 17 and 18. They use functions MIN and MAX similar to the ones in Figs. 13 and 14. The function  $\text{FDATTR}^*$  in Fig. 18 takes a discounting factor  $\lambda$  and a precision  $\varepsilon$  as inputs;  $\lambda$  is used for the iteration in  $\text{FDATTR}$ , and  $\varepsilon$  is used to terminate the computation of  $\text{fdattr}^*$  once a desired level of precision has been reached.

## 8.4 Featured Energy and Parity Games

The algorithms for computing the attractors of featured energy games and of featured parity games are very similar to the ones already shown and not depicted due to space restrictions. They can be found in the long version [24].

## 9 CONCLUSION

We have in this work lifted most of the two-player games which are used in model checking and controller synthesis to software product lines. We have introduced featured versions of reachability games, minimum reachability games, discounted games, energy games, and parity games. We have shown how to compute featured attractors for these games, using family-based algorithms with late

```

1: function FWATTR( $U : S \rightarrow (P \rightarrow \mathbb{N} \cup \{\infty\})$ ):  $S \rightarrow (P' \rightarrow \mathbb{N} \cup \{\infty\})$ 
2:    $U' \leftarrow \emptyset$ 
3:   for all  $s \in S$  do
4:      $P'_s, U'_s \leftarrow \emptyset$ 
5:     for all  $s \rightarrow_x s'$  do
6:        $Q_{s'}, V_{s'} \leftarrow \emptyset$ 
7:       while  $P_{s'} \neq \emptyset$  do
8:         Pick and remove  $\varphi$  from  $P_{s'}$ 
9:          $\psi \leftarrow \gamma((s, s')) \wedge \varphi$ 
10:        if  $\llbracket \psi \rrbracket \neq \emptyset$  then
11:           $Q_{s'} \leftarrow Q_{s'} \cup \{\psi\}$ 
12:           $V_{s'}(\psi) \leftarrow x + U_{s'}(\varphi)$ 
13:         $Q_{s'} \leftarrow Q_{s'} \cup \{\neg\gamma((s, s'))\}$ 
14:         $V_{s'}(\neg\gamma((s, s'))) \leftarrow \mathbf{ff}$ 
15:        if  $s \in S_1$  then
16:           $U'_s \leftarrow \text{MIN}(U'_s, V_{s'})$ 
17:        if  $s \in S_2$  then
18:           $U'_s \leftarrow \text{MAX}(U'_s, V_{s'})$ 
19:       $U'_s \leftarrow \text{REDUCE}(U'_s)$ 
20:   return  $U' : S \rightarrow (P' \rightarrow \mathbb{N} \cup \{\infty\})$ 

```

Figure 15: Computation of fwattr.

```

1: function FWATTR*( $G = (S_1, S_2, i, F, T, \gamma)$ ):  $S \rightarrow (P \rightarrow \mathbb{N} \cup \{\infty\})$ 
2:    $J = \emptyset$ 
3:   for all  $s \in S$  do
4:      $U_s \leftarrow \{\mathbf{tt}\}$ 
5:     if  $s \in F$  then
6:        $J_s(\mathbf{tt}) \leftarrow 0$ 
7:     else
8:        $J_s(\mathbf{tt}) \leftarrow \infty$ 
9:   repeat
10:     $J_{\text{old}} \leftarrow J$ 
11:     $J \leftarrow \text{MIN}(J, \text{FWATTR}(J))$ 
12:  until  $J = J_{\text{old}}$ 
13:  return  $J$ 

```

Figure 16: Fixed-point iteration for fwattr\*.

splitting, and how to use these featured attractors to compute winners, values, and optimal strategies for all products at once.

The astute reader may have noticed that *mean-payoff games* are conspicuously absent from this paper. The immediate reason for this absence is that mean-payoff games do not admit attractors; instead they are solved by computing loops [40]. [7] show an easy reduction from mean-payoff to energy games which may be used to compute winners in featured mean-payoff games. To compute values and optimal strategies, the reduction to discounted games in [26], building on earlier work in [40], may be used.

Two-player games are an established technique for model checking and control synthesis, and our work shows that this technology may be lifted to featured model checking and featured control synthesis. In future work we plan to implement our algorithms and

```

1: function FDATEATTR( $U : S \rightarrow (P \rightarrow \mathbb{R}), \lambda$ ):  $S \rightarrow (P' \rightarrow \mathbb{R})$ 
2:    $U' \leftarrow \emptyset$ 
3:   for all  $s \in S_1$  do
4:      $P'_s, U'_s \leftarrow \emptyset$ 
5:     for all  $s \rightarrow_x s'$  do
6:        $Q_{s'}, V_{s'} \leftarrow \emptyset$ 
7:       while  $P_{s'} \neq \emptyset$  do
8:         Pick and remove  $\varphi$  from  $P_{s'}$ 
9:          $\psi \leftarrow \gamma((s, s')) \wedge \varphi$ 
10:        if  $\llbracket \psi \rrbracket \neq \emptyset$  then
11:           $Q_{s'} \leftarrow Q_{s'} \cup \{\psi\}$ 
12:           $V_{s'}(\psi) \leftarrow x + \lambda U_{s'}(\varphi)$ 
13:         $Q_{s'} \leftarrow Q_{s'} \cup \{\neg\gamma((s, s'))\}$ 
14:         $V_{s'}(\neg\gamma((s, s'))) \leftarrow \mathbf{ff}$ 
15:        if  $s \in S_1$  then
16:           $U'_s \leftarrow \text{MAX}(U'_s, V_{s'})$ 
17:        if  $s \in S_2$  then
18:           $U'_s \leftarrow \text{MIN}(U'_s, V_{s'})$ 
19:       $U'_s \leftarrow \text{REDUCE}(U'_s)$ 
20:   return  $U' : S \rightarrow (P' \rightarrow \mathbb{R})$ 

```

Figure 17: Computation of fdattr.

```

1: function FDATEATTR*( $G = (S_1, S_2, i, T, \gamma), \lambda, \varepsilon$ ):  $S \rightarrow (P \rightarrow \mathbb{R})$ 
2:    $J = \emptyset$ 
3:   for all  $s \in S$  do
4:      $U_s \leftarrow \{\mathbf{tt}\}$ 
5:      $J_s(\mathbf{tt}) \leftarrow 0$ 
6:   repeat
7:      $J_{\text{old}} \leftarrow J$ 
8:      $J \leftarrow \text{FDATEATTR}(J, \lambda)$ 
9:   until  $\|J - J_{\text{old}}\| < \varepsilon$ 
10:  return  $J$ 

```

Figure 18: Fixed-point iteration for fdattr\*.

integrate them into the mCRL2 toolset [8, 34], using BDD representations of product families, in order to evaluate our work on benchmark models.

We also plan to extend our work into the probabilistic and timed settings. Controller synthesis often deals with real-time or hybrid systems, and SPL models of such systems are by now well-established [19, 33, 35]. For real-time systems, we are looking into extending *timed games* [3] with features, analogously to the featured timed automata of [19]; for probabilistic systems, a featured extension of stochastic games [31] appears straight-forward.

## REFERENCES

- [1] Sven Apel, Alexander von Rhein, Philipp Wendler, Armin Größlinger, and Dirk Beyer. Strategies for product-line verification: case studies and experiments. In David Notkin, Betty H. C. Cheng, and Klaus Pohl, editors, *ICSE*, pages 482–491. IEEE Computer Society, 2013.
- [2] Sebastian S. Bauer, Line Juhl, Kim G. Larsen, Jiří Srba, and Axel Legay. A logic for accumulated-weight reasoning on multiweighted modal automata. In Tiziana Margaria, Zongyan Qiu, and Hongli Yang, editors, *TASE*, pages 77–84. IEEE Computer Society, 2012.

- [3] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. UPPAAL-Tiga: Time for playing games! In Werner Damm and Holger Hermanns, editors, *CAV*, volume 4590 of *Lect. Notes Comput. Sci.*, pages 121–125. Springer-Verlag, 2007.
- [4] Harsh Beohar and Mohammad Reza Mousavi. Input-output conformance testing for software product lines. *J. Log. Alg. Meth. Prog.*, 85(6):1131–1153, 2016.
- [5] Julian C. Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 871–919. Springer-Verlag, 2018.
- [6] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Inf.*, 54(1):85–125, 2017.
- [7] Luboš Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Meth. Syst. Design*, 38:97–118, 2011.
- [8] Olav Bunte, Jan Friso Groote, Jeroen J. A. Keiren, Maurice Laveaux, Thomas Neele, Erik P. de Vink, Wieger Wesselink, Anton Wijs, and Tim A. C. Willemse. The mCRL2 toolset for analysing concurrent systems - improvements in expressivity and usability. In Tomáš Vojnar and Lijun Zhang, editors, *TACAS (II)*, volume 11428 of *Lect. Notes Comput. Sci.*, pages 21–39. Springer-Verlag, 2019.
- [9] Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Asp. Comput.*, 30(1):45–75, 2018.
- [10] Andreas Classen, Maxime Cordy, Patrick Heymans, Axel Legay, and Pierre-Yves Schobbens. Model checking software product lines with SNIP. *Softw. Tools Technol. Transf.*, 14(5):589–612, 2012.
- [11] Andreas Classen, Maxime Cordy, Patrick Heymans, Axel Legay, and Pierre-Yves Schobbens. Formal semantics, modular specification, and symbolic verification of product-line behaviour. *Sci. Comput. Program.*, 80:416–439, 2014.
- [12] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-François Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking. *IEEE Trans. Software Eng.*, 39(8):1069–1089, 2013.
- [13] Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay. Symbolic model checking of software product lines. In Richard N. Taylor, Harald C. Gall, and Nenad Medvidovic, editors, *ICSE*, pages 321–330. ACM, 2011.
- [14] Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, Axel Legay, and Jean-François Raskin. Model checking lots of systems: Efficient verification of temporal properties in software product lines. In Jeff Kramer, Judith Bishop, Premkumar T. Devanbu, and Sebastián Uchitel, editors, *ICSE (1)*, pages 335–344. ACM, 2010.
- [15] Maxime Cordy, Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay. ProVeLines: a product line of verifiers for software product lines. In *SPLC Workshops*, pages 141–146. ACM, 2013.
- [16] Maxime Cordy, Andreas Classen, Gilles Perrouin, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. Simulation-based abstractions for software product-line model checking. In Martin Glinz, Gail C. Murphy, and Mauro Pezzè, editors, *ICSE*, pages 672–682. IEEE Computer Society, 2012.
- [17] Maxime Cordy, Xavier Devroey, Axel Legay, Gilles Perrouin, Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Jean-François Raskin. A decade of featured transition systems. In Maurice H. ter Beek, Alessandro Fantechi, and Laura Semini, editors, *From Software Engineering to Formal Methods and Tools, and Back*, volume 11865 of *Lect. Notes Comput. Sci.*, pages 285–312. Springer-Verlag, 2019.
- [18] Maxime Cordy, Patrick Heymans, Axel Legay, Pierre-Yves Schobbens, Bruno Dawagne, and Martin Leucker. Counterexample guided abstraction refinement of product-line behavioural models. In Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey, editors, *FSE*, pages 190–201. ACM, 2014.
- [19] Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. Behavioural modelling and verification of real-time software product lines. In Eduardo Santana de Almeida, Christa Schwanninger, and David Benavides, editors, *SPLC*, pages 66–75. ACM, 2012.
- [20] Aleksandar S. Dimovski, Axel Legay, and Andrzej Wąsowski. Variability abstraction and refinement for game-based lifted model checking of full CTL. In Reiner Hahnle and Wil M. P. van der Aalst, editors, *FASE*, volume 11424 of *Lect. Notes Comput. Sci.*, pages 192–209. Springer-Verlag, 2019.
- [21] Aleksandar S. Dimovski and Andrzej Wąsowski. Variability-specific abstraction refinement for family-based model checking. In Marieke Huisman and Julia Rubin, editors, *FASE*, volume 10202 of *Lect. Notes Comput. Sci.*, pages 406–423. Springer-Verlag, 2017.
- [22] Uli Fahrenberg and Axel Legay. The quantitative linear-time–branching-time spectrum. *Theor. Comput. Sci.*, 538:54–69, 2014.
- [23] Uli Fahrenberg and Axel Legay. Quantitative properties of featured automata. *Softw. Tools Technol. Transf.*, 21(6):667–677, 2019.
- [24] Uli Fahrenberg and Axel Legay. Featured games. *CoRR*, 2020.
- [25] Uli Fahrenberg, Axel Legay, and Karin Quaas. Computing branching distances using quantitative games. In Robert M. Hierons and Mohamed Mosbah, editors, *ICTAC*, volume 11884 of *Lect. Notes Comput. Sci.*, pages 59–75. Springer-Verlag, 2019.
- [26] Hugo Gimbert and Wiesław Zielonka. Applying Blackwell optimality: priority mean-payoff games as limits of multi-discounted games. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, pages 331–356. Amsterdam University Press, 2008.
- [27] Marcin Jurdzinski. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS*, volume 1770 of *Lect. Notes Comput. Sci.*, pages 290–301. Springer-Verlag, 2000.
- [28] Ratnesh Kumar and Vijay K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Springer-Verlag, 2012.
- [29] Malte Lochau, Stephan Mennicke, Hauke Baller, and Lars Ribbeck. Incremental model checking of delta-oriented software product lines. *J. Log. Alg. Meth. Prog.*, 85(1):245–267, 2016.
- [30] Rafael Olacchia, Uli Fahrenberg, Joanne M. Atlee, and Axel Legay. Long-term average cost in featured transition systems. In Hong Mei, editor, *SPLC*, pages 109–118. ACM, 2016.
- [31] Hans J. M. Peters and O. J. Vrieze. *Surveys in Game Theory and Related Topics*, volume 39 of *CWI Tract*. Centrum voor Wiskunde en Informatica, 1987.
- [32] Peter J. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.
- [33] Genaina N. Rodrigues, Vander Alves, Vinicius Nunes, André Lanna, Maxime Cordy, Pierre-Yves Schobbens, Amir Molzam Sharifloo, and Axel Legay. Modeling and verification for probabilistic properties in software product lines. In *HASE*, pages 173–180. IEEE Computer Society, 2015.
- [34] Maurice H. ter Beek, Erik P. de Vink, and Tim A. C. Willemse. Family-based model checking with mCRL2. In Marieke Huisman and Julia Rubin, editors, *FASE*, volume 10202 of *Lect. Notes Comput. Sci.*, pages 387–405. Springer-Verlag, 2017.
- [35] Maurice H. ter Beek, Axel Legay, Alberto Lluch-Lafuente, and Andrea Vandin. Statistical analysis of probabilistic models of software product lines with quantitative constraints. In Douglas C. Schmidt, editor, *SPLC*, pages 11–15. ACM, 2015.
- [36] Maurice H. ter Beek, Axel Legay, Alberto Lluch-Lafuente, and Andrea Vandin. A framework for quantitative modeling and analysis of highly (re)configurable systems. *IEEE Trans. Software Eng.*, 46(3):321–345, 2020.
- [37] Maurice H. ter Beek and Franco Mazzanti. VMC: recent advances and challenges ahead. In Stefania Gnesi, Alessandro Fantechi, Maurice H. ter Beek, Goetz Botterweck, and Martin Becker, editors, *SPLC Workshops*, pages 70–77. ACM, 2014.
- [38] Maurice H. ter Beek, Sef van Loo, Erik P. de Vink, and Tim A. C. Willemse. Family-based SPL model checking using parity games with variability. In Heike Wehrheim and Jordi Cabot, editors, *FASE*, volume 12076 of *Lect. Notes Comput. Sci.*, pages 245–265. Springer-Verlag, 2020.
- [39] Tom van Dijk. Oink: An implementation and evaluation of modern parity game solvers. In Dirk Beyer and Marieke Huisman, editors, *TACAS*, volume 10805 of *Lect. Notes Comput. Sci.*, pages 291–308. Springer-Verlag, 2018.
- [40] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.

## APPENDIX: PROOFS

PROOF OF THM. 2.2. Let  $H = \text{proj}_p(G) = (S_1, S_2, i, F, T')$  and, for clarity, write  $\text{attr}_H$  for the attractor in  $H$  and  $\text{fattr}_G$  for the one in  $G$ . We need to show that  $\text{fattr}_G^*(I)(i)(\gamma_p) = \text{attr}_H^*(I)(i)$ . (Note that we are using the same notation  $I$  for both  $G$  and  $H$ ; this should cause no confusion.)

The conclusion will follow once we can show that for all  $n \geq 0$  and all  $s \in S$ ,  $\text{fattr}_G^n(I)(s)(\gamma_p) = \text{attr}_H^n(I)(s)$ . We do so by induction on  $n$ . For  $n = 0$  both sides of the equation become **tt** iff  $s \in F$ , so this is clear.

Now let  $n \geq 0$  and assume that for all  $s' \in S$ ,  $\text{fattr}_G^n(I)(s')(\gamma_p) = \text{attr}_H^n(I)(s')$ . Let  $s \in S_1$ , then

$$\begin{aligned} \text{fattr}_G^{n+1}(I)(s)(\gamma_p) &= \bigvee_{s \rightarrow_{\mathcal{G}}^x s'} \text{fattr}_G^n(I)(s')(\gamma((s, x, s'))) \wedge \gamma_p \\ &= \bigvee_{s \rightarrow_{\mathcal{H}}^x s'} \text{fattr}_G^n(I)(s')(\gamma_p) \\ &= \bigvee_{s \rightarrow_{\mathcal{H}}^x s'} \text{attr}_H^n(I)(s') = \text{attr}_H^{n+1}(I)(s); \end{aligned}$$

for  $s \in S_2$  the proof is similar.  $\square$

PROOF OF THM. 3.2. Let  $H = \text{proj}_p(G) = (S_1, S_2, i, F, T')$ ; we need to prove that  $\text{fwattr}_G^*(I)(i)(\gamma_p) = \text{wattr}_H^*(I)(i)$ . We show inductively that for all  $n \geq 0$  and all  $s \in S$ ,  $\text{fwattr}_G^n(I)(s)(\gamma_p) = \text{wattr}_H^n(I)(s)$ , which will imply the conclusion. For  $n = 0$  both sides of the equation become 0 if  $s \in F$  and  $\infty$  otherwise, so the base case is clear.

Now let  $n \geq 0$  and assume that for all  $s' \in S$ ,  $\text{fwattr}_G^n(I)(s')(\gamma_p) = \text{wattr}_H^n(I)(s')$ . Let  $s \in S_1$ , then  $\text{fwattr}_G^{n+1}(I)(s)(\gamma_p) = \min_{s \rightarrow_{\mathcal{G}}^x s'} x + \text{fwattr}_G^n(I)(s')(\gamma((s, x, s'))) \wedge \gamma_p = \min_{s \rightarrow_{\mathcal{H}}^x s'} x + \text{fwattr}_G^n(I)(s')(\gamma_p) = \min_{s \rightarrow_{\mathcal{H}}^x s'} x + \text{wattr}_H^n(I)(s') = \text{wattr}_H^{n+1}(I)(s)$ ; for  $s \in S_2$  the proof is similar.  $\square$

PROOF OF THM. 4.2. Define a metric on  $S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{R})$  by  $d(U_1, U_2) = \max_{s \in S} \max_{\varphi \in \mathbb{B}(N)} |U_1(s)(\varphi) - U_2(s)(\varphi)|$ . Then  $d(\text{fdattr}(U_1), \text{fdattr}(U_2)) \leq \lambda d(U_1, U_2)$  for any two functions  $U_1, U_2$ , that is,  $\text{fdattr}$  is a contraction on the complete metric space  $S_1 \rightarrow \mathbb{R}^{\mathbb{B}(N)}$ . By the Banach fixed-point theorem,  $\text{fdattr}$  has a unique fixed point which is  $\text{fdattr}^*$ .

Let  $p \in \text{px}$  and  $H = \text{proj}_p(G) = (S_1, S_2, i, T')$ ; we need to show that  $\text{fattr}_H^*(i) = \text{fdattr}_G^*(i)(\gamma_p)$ . Now for any  $U : S \rightarrow (\mathbb{B}(N) \rightarrow \mathbb{R})$  and  $s \in S_1$ ,  $\text{fdattr}(U)(s)(\gamma_p) = \max_{s \rightarrow_{\mathcal{G}}^x s'} x + \lambda U(s')(\gamma((s, x, s'))) \wedge \gamma_p = \max_{s \rightarrow_{\mathcal{H}}^x s'} x + \lambda U(s')(\gamma_p)$ , and the same can be shown if  $s \in S_2$  instead. Hence the equation systems defining  $\text{dattr}_H^*$  and  $\text{fdattr}_G^*(\cdot)(\gamma_p)$  are the same; consequently, also their unique fixed points are equal.  $\square$

PROOF OF THM. 5.2. Let  $H = \text{proj}_p(G) = (S_1, S_2, i, T')$ ; we need to prove that  $\text{feattr}_G^*(I)(i)(\gamma_p) = \text{eattr}_H^*(I)(i)$ . We show inductively that for all  $n \geq 0$  and all  $s \in S$ ,  $\text{feattr}_G^n(I)(s)(\gamma_p) = \text{eattr}_H^n(I)(s)$ , which will imply the conclusion. For  $n = 0$ , the equation becomes  $I(s)(\gamma_p) = I(s)$  which is clear.

Now let  $n \geq 0$  and assume that for all  $s' \in S$ ,  $\text{feattr}_G^n(I)(s')(\gamma_p) = \text{eattr}_H^n(I)(s')$ . Let  $s \in S_1$ , then  $\text{feattr}_G^{n+1}(I)(s)(\gamma_p) = \min_{s \rightarrow_{\mathcal{G}}^x s'} \text{feattr}_G^n(I)(s')(\gamma((s, x, s'))) \wedge \gamma_p \ominus x = \min_{s \rightarrow_{\mathcal{H}}^x s'} \text{feattr}_G^n(I)(s')(\gamma_p) \ominus x = \min_{s \rightarrow_{\mathcal{H}}^x s'} \text{eattr}_H^n(I)(s') \ominus x = \text{eattr}_H^{n+1}(I)(s)$ ; similarly for  $s \in S_2$ .  $\square$

PROOF OF THM. 6.2. Let  $H = \text{proj}_p(G) = (S_1, S_2, i, T', p)$ ; we need to prove that  $\text{fpattr}_G^*(I)(i)(\gamma_p) = \text{pattr}_H^*(I)(i)$ . We show inductively that for all  $n \geq 0$  and all  $s \in S$ ,  $\text{fpattr}_G^n(I)(s)(\gamma_p) = \text{pattr}_H^n(I)(s)$ , which will imply the conclusion. For  $n = 0$ , the equation becomes  $I(s)(\gamma_p) = I(s)$  which is clear.

Now let  $n \geq 0$  and assume that for all  $s' \in S$ ,  $\text{fpattr}_G^n(I)(s')(\gamma_p) = \text{pattr}_H^n(I)(s')$ . Let  $s \in S_1$ , then  $\text{fpattr}_G^{n+1}(I)(s)(\gamma_p) = \max\{\text{fpattr}_G^n(I)(s)(\gamma_p), \min_{s \rightarrow_{\mathcal{G}}^x s'} \text{fprog}(\text{fpattr}_G^n(I), s, s')(\gamma((s, x, s'))) \wedge \gamma_p\} = \max\{\text{pattr}_H^n(I)(s), \min_{s \rightarrow_{\mathcal{H}}^x s'} \text{fprog}(\text{pattr}_H^n(I), s, s')(\gamma_p)\} = \max\{\text{pattr}_H^n(I)(s), \min_{s \rightarrow_{\mathcal{H}}^x s'} \min\{m \in M \mid m \geq_{p(s)} \text{fpattr}_G^n(I)(s')(\gamma_p)\}\} = \max\{\text{pattr}_H^n(I)(s), \min_{s \rightarrow_{\mathcal{H}}^x s'} \min\{m \in M \mid m \geq_{p(s)} \text{pattr}_H^n(I)(s')\}\} = \max\{\text{pattr}_H^n(I)(s), \min_{s \rightarrow_{\mathcal{H}}^x s'} \text{prog}(\text{pattr}_H^n(I), s, s')\} = \text{pattr}_H^{n+1}(I)(s)$ . For  $s \in S_2$  the reasoning is similar.  $\square$

PROOF OF LEMMA 7.1. Assume that player 1 wins the reachability game in  $H = \text{proj}_p(G)$ . Then there is  $\theta_1 \in \Theta_1$  such that for all  $\theta_2 \in \Theta_2$ , writing  $\text{out}(\theta_1, \theta_2) = (s_1, s_2, \dots)$ , there is an index  $k \geq 1$  for which  $s_k \in F$ . Let  $\theta_2 \in \Theta_2$ . All transitions  $(s_1, s_2), (s_2, s_3), \dots$  are in  $H$ , hence  $p \models \gamma((s_1, \dots, s_k))$ , i.e.,  $\gamma_p \wedge \gamma((s_1, \dots, s_k)) \neq \mathbf{ff}$ . Let  $\xi_1 \in \Xi_1$  be any strategy for which  $\xi_1(\gamma_p) = \theta_1$ . We have shown that for any  $\xi_2 \in \Xi_2$ , writing  $\text{out}(\xi_1, \xi_2)(\varphi) = (s_1, s_2, \dots)$ , there is an index  $k \geq 1$  for which  $s_k \in F$  and  $\varphi \wedge \gamma((s_1, \dots, s_k)) \neq \mathbf{ff}$ ; that is, player 1 wins the  $\gamma_p$ -reachability game in  $G$ .

For the converse, assume that player 1 wins the  $\gamma_p$ -reachability game in  $G$ , and let  $\xi_1 \in \Xi_1$  be such that for any  $\xi_2 \in \Xi_2$ , writing  $\text{out}(\xi_1, \xi_2)(\gamma_p) = (s_1, s_2, \dots)$ , there is an index  $k \geq 1$  for which  $s_k \in F$  and  $\gamma_p \wedge \gamma((s_1, \dots, s_k)) \neq \mathbf{ff}$ , i.e.,  $p \models \gamma((s_1, \dots, s_k))$ . Then  $p \models \gamma((s_1, s_2)) \wedge \dots \wedge \gamma((s_{k-1}, s_k))$ , so that all the transitions  $(s_1, s_2), \dots, (s_{k-1}, s_k)$  are present in  $H$ . Let  $\theta_1 = \xi_1(\gamma_p)$ . We have shown that for all  $\theta_2 \in \Theta_2$ , writing  $\text{out}(\theta_1, \theta_2) = (s_1, s_2, \dots)$ , there is an index  $k \geq 1$  for which  $s_k \in F$ ; that is, player 1 wins the reachability game in  $H$ .  $\square$

PROOF OF THM. 7.2. We show the second claim first. Write  $H = \text{proj}_p(G)$ , assume  $\xi_1$  to be locally optimal, write  $\theta_1 = \xi_1(\gamma_p)$ , and let  $s \in S_1$ . Then  $\text{attr}_H^*(I)(s) = \text{fattr}_G^*(I)(s)(\gamma_p) = \text{fattr}_G^*(I)(\theta_1(s))(\gamma((s, \theta_1(s))) \wedge \gamma_p) = \text{fattr}_G^*(I)(\theta_1(s))(\gamma_p) = \text{attr}_H^*(I)(\theta_1(s))$ , thus  $\theta_1$  is locally optimal in  $\text{proj}_p(G)$ .

For the first claim of the theorem, let  $s \in S_1$  and  $\varphi \in \mathbb{B}(N)$ . We have  $\text{fattr}_G^*(I)(s)(\varphi) = \bigvee_{s \rightarrow_{\mathcal{G}}^x s'} \text{fattr}_G^*(I)(s')(\gamma((s, x, s'))) \wedge \varphi$ . The set  $\{s' \in S_2 \mid s \rightarrow s'\}$  is finite, hence there is  $\tilde{s}'$  such that  $\text{fattr}_G^*(I)(s)(\varphi) = \text{fattr}_G^*(I)(\tilde{s}')(\gamma((s, \tilde{s}')) \wedge \varphi)$ . Define  $\xi_1(s)(\varphi) = \tilde{s}'$ .  $\square$

PROOF OF LEMMA 7.3. Denoting strategy sets in  $G$  by  $\Xi_i$  and in  $\text{proj}_p(G)$  by  $\Theta_i$ , we see that  $\Theta_i = \Xi_i(\gamma_p)$ . Then  $\text{fval}_R(G)(\gamma_p) = \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \text{fval}_R(\text{out}(\xi_1, \xi_2)(\gamma_p)) = \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \text{val}_R(\text{out}(\xi_1, \xi_2)(\gamma_p)) = \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \text{val}_R(\text{out}(\xi_1(\gamma_p), \xi_2(\gamma_p))) = \inf_{\theta_1 \in \Xi_1(\gamma_p)} \sup_{\theta_2 \in \Xi_2(\gamma_p)} \text{val}_R(\text{out}(\theta_1, \theta_2)) = \inf_{\theta_1 \in \Theta_1} \sup_{\theta_2 \in \Theta_2} \text{val}_R(\text{out}(\theta_1, \theta_2)) = \text{val}_R(\text{proj}_p(G))$ .  $\square$

## APPENDIX: OTHER ALGORITHMS

```

1: function FEATTR( $U : S \rightarrow (P \rightarrow W)$ ):  $S \rightarrow (P' \rightarrow W)$ 
2:    $U' \leftarrow \emptyset$ 
3:   for all  $s \in S_1$  do
4:      $P'_s, U'_s \leftarrow \emptyset$ 
5:     for all  $s \rightarrow_x s'$  do
6:        $Q_{s'}, V_{s'} \leftarrow \emptyset$ 
7:       while  $P_{s'} \neq \emptyset$  do
8:         Pick and remove  $\varphi$  from  $P_{s'}$ 
9:          $\psi \leftarrow \gamma((s, s')) \wedge \varphi$ 
10:        if  $\llbracket \psi \rrbracket \neq \emptyset$  then
11:           $Q_{s'} \leftarrow Q_{s'} \cup \{\psi\}$ 
12:           $V_{s'}(\psi) \leftarrow U_{s'}(\varphi) \ominus x$ 
13:           $Q_{s'} \leftarrow Q_{s'} \cup \{\neg\gamma((s, s'))\}$ 
14:           $V_{s'}(\neg\gamma((s, s'))) \leftarrow \mathbf{ff}$ 
15:          if  $s \in S_1$  then
16:             $U'_s \leftarrow \text{MIN}(U'_s, V_{s'})$ 
17:          if  $s \in S_2$  then
18:             $U'_s \leftarrow \text{MAX}(U'_s, V_{s'})$ 
19:           $U'_s \leftarrow \text{REDUCE}(U'_s)$ 
20:   return  $U' : S \rightarrow (P' \rightarrow W)$ 

```

Figure 19: Computation of feattr.

```

1: function FEATTR*( $G = (S_1, S_2, i, T, \gamma)$ ):  $S \rightarrow (P \rightarrow \mathbb{N} \cup \{\infty\})$ 
2:    $J = \emptyset$ 
3:   for all  $s \in S$  do
4:      $U_s \leftarrow \{\mathbf{tt}\}$ 
5:      $J_s(\mathbf{tt}) \leftarrow 0$ 
6:   repeat
7:      $J_{\text{old}} \leftarrow J$ 
8:      $J \leftarrow \text{MAX}(J, \text{FEATTR}(J))$ 
9:   until  $J = J_{\text{old}}$ 
10:  return  $J$ 

```

Figure 20: Fixed-point iteration for feattr\*.

```

1: function FPROG( $U : S \rightarrow (P \rightarrow M), s, s'$ ):  $P' \rightarrow M$ 
2:    $U', P' \leftarrow \emptyset$ 
3:   while  $P_{s'} \neq \emptyset$  do
4:     Pick and remove  $\varphi$  from  $P_{s'}$ 
5:      $P' \leftarrow P' \cup \{\varphi\}$ 
6:      $U'(\varphi) \leftarrow \min\{m \in M \mid m \geq_{p(s)} U_{s'}(\varphi)\}$ 
7:   return  $\text{REDUCE}(U')$ 

```

Figure 21: Computation of fprog.

```

1: function FPATTR( $U : S \rightarrow (P \rightarrow M)$ ):  $S \rightarrow (P' \rightarrow M)$ 
2:    $U' \leftarrow \emptyset$ 
3:   for all  $s \in S$  do
4:      $P'_s, U'_s \leftarrow \emptyset$ 
5:     for all  $s \rightarrow s'$  do
6:        $Q_{s'}, V_{s'} \leftarrow \emptyset$ 
7:       while  $P_{s'} \neq \emptyset$  do
8:         Pick and remove  $\varphi$  from  $P_{s'}$ 
9:          $\psi \leftarrow \gamma((s, s')) \wedge \varphi$ 
10:        if  $\llbracket \psi \rrbracket \neq \emptyset$  then
11:           $Q_{s'} \leftarrow Q_{s'} \cup \{\psi\}$ 
12:           $V_{s'}(\psi) \leftarrow \text{FPROG}(U, s, s')(\varphi)$ 
13:           $Q_{s'} \leftarrow Q_{s'} \cup \{\neg\gamma((s, s'))\}$ 
14:           $V_{s'}(\neg\gamma((s, s'))) \leftarrow \mathbf{ff}$ 
15:          if  $s \in S_1$  then
16:             $U'_s \leftarrow \text{MIN}(U'_s, V_{s'})$ 
17:          if  $s \in S_2$  then
18:             $U'_s \leftarrow \text{MAX}(U'_s, V_{s'})$ 
19:           $U'_s \leftarrow \text{REDUCE}(U'_s)$ 
20:   return  $U' : S \rightarrow (P' \rightarrow M)$ 

```

Figure 22: Computation of fpattr.

```

1: function FPATTR*( $G = (S_1, S_2, i, T, p, \gamma)$ ):  $S \rightarrow (P \rightarrow M)$ 
2:    $J = \emptyset$ 
3:   for all  $s \in S$  do
4:      $U_s \leftarrow \{\mathbf{tt}\}$ 
5:      $J_s(\mathbf{tt}) \leftarrow 0$ 
6:   repeat
7:      $J_{\text{old}} \leftarrow J$ 
8:      $J \leftarrow \text{MAX}(J, \text{FPATTR}(J))$ 
9:   until  $J = J_{\text{old}}$ 
10:  return  $J$ 

```

Figure 23: Fixed-point iteration for fpattr\*.