Featured Games

Uli Fahrenberg[a], Axel Legay[b]

[a] *EPITA Research and Development Laboratory (LRDE), France*
[b] *Université catholique de Louvain, Belgium*

Abstract

Feature-based analysis of software product lines 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.

As part of a program to make game-based model checking available for software product lines, we introduce featured reachability games, featured minimum reachability games, featured discounted games, featured energy games, and featured parity games. We show that all these admit optimal featured strategies, which project to optimal strategies for any product, and how to compute winners and values of such games in a family-based manner.

Keywords: featured transition system, two-player game, family-based model checking, reachability game, discounted game, energy game, parity game

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.* [17], 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 [12–14, 17, 18, 37, 42, 44, 45].

The number of products in an SPL grows exponentially with the number of features, hence model checking each individual product is prohibitive. Thus, *family-based model checking* has been introduced in [17], allowing for the simultaneous verification of all products. The family-based approach has seen rapid development [1, 13–16, 42, 46] and has been extended to conformance model checking [19], abstraction-based model checking [21, 23, 24], real-time formalisms [5, 22], probabilistic systems [12, 41, 43], and quantitative model checking [2, 26, 27, 38]; see [20] for a recent survey.

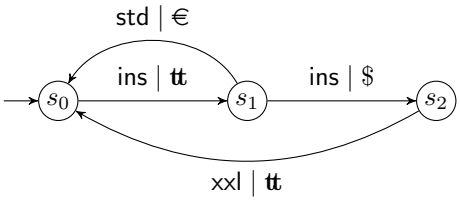

Figure 1: FTS model S of a simple coffee machine SPL.

$$\varphi = \nu X.\mu Y.\overbrace{\Big(\underbrace{(\langle \mathsf{ins}\rangle Y \vee \langle \mathsf{xxl}\rangle Y)}_{\varphi_3} \vee \langle \mathsf{std}\rangle X\Big)}^{\varphi_2}$$

Figure 2: μ-calculus specification for S.

1.1. Model Checking

Many model checking problems can be reduced to two-player *games* on finite graphs. A prominent example is μ-calculus model checking, which is generally done by translating to parity games [7], but also many quantitative model-checking problems can be reduced to (quantitative) games [25, 29–31].

In their recent paper [46], ter Beek *et al.* introduce a procedure for family-based μ-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 [46]. Figure 1 shows a toy model S of a coffee machine with feature set $\{\in, \$\}$ and three products $\{\in\}$, $\{\$\}$, and $\{\in, \$\}$. 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 $\in$ feature is present, and the second ins transition exists only if the $ feature is present.

In Figure 2 we define a μ-calculus formula φ which expresses the property that there exists an infinite run of the system along which infinitely many regular coffees are delivered. Using the translation introduced in [46], which is a feature-enriched version of the standard translation [7] from μ-calculus model checking to parity games, we arrive at the featured parity game depicted in Figure 3 (only the reachable part is shown). 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 [46], player 1 wins the game for a product p iff the projection $\mathsf{proj}_p(S)$ satisfies φ; in our case iff $\in \in p$.

The authors of [46] then continue to develop a family-based algorithm for solving featured parity games. They introduce so-called *variability parity games* in which edges are annotated with subsets of the set of products and then use a version of Zielonka's algorithm [47] to solve these. Here we take a different approach: we employ syntactic edge annotations, using logical formulae rather

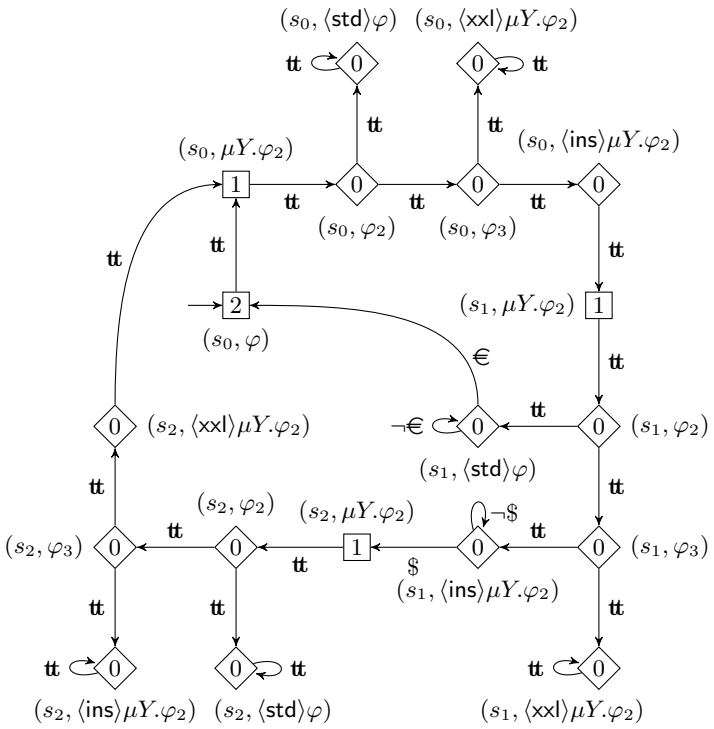

Figure 3: Featured parity game for checking whether $S \models \varphi$.

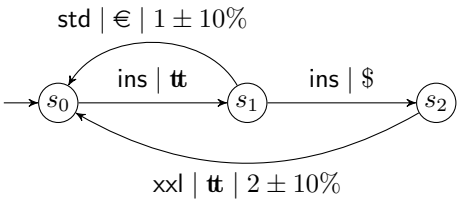

Figure 4: Coffee machine model S with energy annotations.

than sets of products; and we base our algorithm on Jurdzinski's algorithm [35] rather than Zielonka's. The former works by computing *attractors*, which fits well into our general framework which lifts attractors (for different types of games) to *featured attractors*.

1.2. Discounted Games

For another example of the use of games, we turn to the *quantitative* setting. Figure 4 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%; 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 $(\mathsf{ins}, \mathsf{std}, \mathsf{ins}, \mathsf{std}, \dots)$ the two machines would accumulate a difference in energy consumption of 0.2 every second step.

Taking the point of view that *the future is discounted*, we fix a discounting factor $\lambda < 1$ and multiply differences by λ 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 [25], robustness of a model for a product p may be computed using the λ-*discounted bisimulation distance*: let S_1 and S_2 be the versions of the projection $\mathsf{proj}_p(S)$ 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 $d : S_1 \times S_2 \to \mathbb{R}$ is the unique solution to the equation system given by

$$d(s^1, s^2) = \max \begin{cases} \max\limits_{s^1 \to_x^a t^1} \min\limits_{s^2 \to_y^a t^2} |x - y| + \lambda d(t^1, t^2), \\ \max\limits_{s^2 \to_y^a t^2} \min\limits_{s^1 \to_x^a t^1} |x - y| + \lambda d(t^1, t^2) \end{cases}$$

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

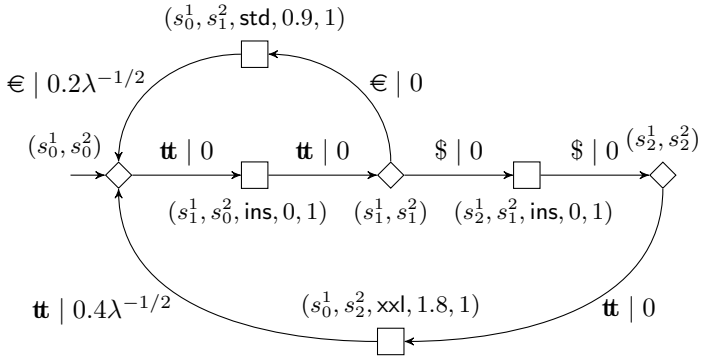

Figure 5: Game for computing discounted distance.

In [30] it is shown that λ-discounted bisimulation distances may be computed by translating to $\sqrt{\lambda}$-discounted games [48]. 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 λ-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:

$$\{(s_1, s_2) \to_\varphi^0 (s_1', s_2, a, x, 1) \mid (s_1, a, x, s_1')_{/\varphi} \in T_1\},$$
$$\{(s_1, s_2) \to_\varphi^0 (s_1, s_2', b, x, 2) \mid (s_2, b, x, s_2')_{/\varphi} \in T_2\},$$
$$\{(s_1', s_2, a, x, 1) \to_\varphi^{\lambda^{-1/2}|a-b|} (s_1', s_2') \mid (s_2, b, x, s_2')_{/\varphi} \in T_2\},$$
$$\{(s_1, s_2', b, x, 2) \to_\varphi^{\lambda^{-1/2}|a-b|} (s_1', s_2') \mid (s_1, a, x, s_1')_{/\varphi} \in T_1\},$$

where we write $t_{i/\varphi}$ for a transition $t_i \in T_i$ with $\gamma_i(t_i) = \varphi$. We show the result of the translation applied to our example in Figure 5, where we have omitted some states and transitions due to symmetry.

We will continue this example in Section 4.3; for $\lambda = 0.99$ and $p = \{€, \$\}$, the distance evaluates to 13.2.

1.3. Energy Games

Games are also important in *controller synthesis*: the problem of generating controllers for discrete event systems [36, 40]. 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 Figure 6, inspired by [3]. 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 deposing 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 deposing 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 has 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 $\neg \mathsf{big} \vee \mathsf{ext}$: 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. We give a more precise analysis of this example in Section 5.3.

1.4. Structure of Paper

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 in a family-based manner. 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 an 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.

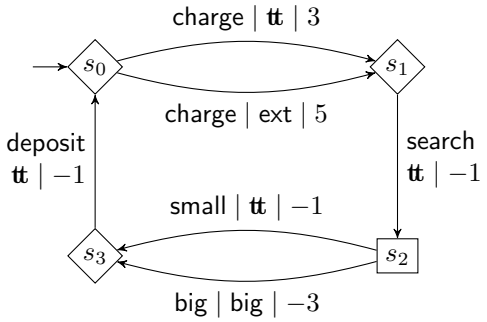

Figure 6: A simple energy game.

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 guard partitions [27] and late splitting [1], to compute attractors for all the featured games we have introduced.

The astute reader may wonder whether there is not a general setting in which our contributions to the different types of games could be unified. A partial answer is given by Gimbert and Zielonka's [32, 33] which show that optimal strategies are *memoryless* in a general class of games which encompasses all games considered here. However, no extension of this work exists which would allow a general procedure for computing attractors in all games with memoryless optimal strategies. (For a recent extension to *finite* memory, see [6].) Hence we are forced to develop our theory separately for each type of games; we have included all proofs for completeness but will point out similarities and differences in order to ease digestion.

This paper is based on [28] which was presented at the 15th Theoretical Aspects of Software Engineering Conference (TASE 2021) in Shanghai, China. Compared to this conference abstract, the present paper contains some more examples to justify our approach, proofs of all results, and the new Section 8 exposing family-based algorithms to compute featured attractors. Further, we correct an error regarding the definition of initial configurations which was pointed out by a reviewer.

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 $s \to s'$ 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 make this intuition precise.

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

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

Any pair of strategies $\theta_1 \in \Theta_1$, $\theta_2 \in \Theta_2$ induces a unique infinite path $\mathsf{out}(\theta_1, \theta_2) = (s_1, s_2, \dots) \in \mathsf{iPaths}(G)$, called the *outcome* of the pair (θ_1, θ_2) and defined inductively by

$$s_1 = i \qquad s_{k+1} = \begin{cases} \theta_1(s_1, \dots, s_k) & \text{if } s_k \in S_1, \\ \theta_2(s_1, \dots, s_k) & \text{if } s_k \in S_2. \end{cases}$$

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 $\mathsf{out}(\theta_1, \theta_2) = (s_1, s_2, \dots)$, 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* $\mathsf{attr} : (S \to \mathbb{B}) \to (S \to \mathbb{B})$, where $\mathbb{B} = \{\mathbf{ff}, \mathbf{tt}\}$ is the boolean lattice, defined for any $U : S \to \mathbb{B}$ by

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

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

Above, $\equiv$ is used to denote *semantic* equivalence of logical formulae. We will generally stay at the syntactic level below and only evaluate logical expressions when needed; in practice, some simplification algorithms such as Espresso [8] should be used to keep the expressions manageable.

Let $\mathsf{attr}^* = \mathsf{id} \vee \mathsf{attr} \vee \mathsf{attr}^2 \vee \cdots$, where id is the identity function $x \mapsto x$, exponentiation denotes repeated function composition, and $\vee$ is the supremum operator on the complete lattice $S \to \mathbb{B}$. Let $I : S \to \mathbb{B}$ be the *initial configuration* given by $I(s) = \mathbf{tt}$ iff $s \in F$, then the following is easy to see.

Lemma 1. *Let $G = (S_1, S_2, i, F, T)$ be a game structure. Player 1 wins the reachability game in G iff $\mathsf{attr}^*(I)(i) \equiv \mathbf{tt}$.* □

The operator attr is monotone on the complete lattice $S \to \mathbb{B}$, thus 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 $\gamma : T \to \mathbb{B}(N)$. 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 $\mathsf{proj}_p(G) = (S_1, S_2, i, F, T')$ 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 the projections to each product $p \in px$, but in a family-based manner. We will thus compute a function $\mathbb{B}(N) \to \mathbb{B}$ which for each feature expression indicates whether player 1 wins the reachability game on $\mathsf{proj}_p(G)$. Note that, as a set of functions into a complete lattice, $\mathbb{B}(N) \to \mathbb{B}$ is itself a complete lattice.

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

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

and let $\mathsf{fattr}^* = \mathsf{id} \vee \mathsf{fattr} \vee \mathsf{fattr}^2 \vee \cdots$, the supremum in the complete lattice $S \to (\mathbb{B}(N) \to \mathbb{B})$. We will evaluate fattr^* on the *initial configuration* $I : S \to (\mathbb{B}(N) \to \mathbb{B})$ given by

$$I(s)(\varphi) = \begin{cases} \mathbf{tt} & \text{if } s \in F \text{ and } \varphi \not\equiv \mathbf{ff}, \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

Theorem 2. *Let $G = (S_1, S_2, i, F, T, \gamma)$ be a featured game structure and $p \in px$, then Player 1 wins the reachability game in $\mathsf{proj}_p(G)$ iff $\mathsf{fattr}^*(I)(i)(\gamma_p) \equiv \mathbf{tt}$.*

Proof. Let $H = \mathsf{proj}_p(G) = (S_1, S_2, i, F, T')$ and, for clarity, write attr_H for the attractor in H and fattr_G for the one in G. We need to show that $\mathsf{fattr}^*_G(I)(i)(\gamma_p) = \mathsf{attr}^*_H(I)(i)$. (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 $s \in S$, $\mathsf{fattr}^n_G(I)(s)(\gamma_p) = \mathsf{attr}^n_H(I)(s)$. We do so by induction on n. For $n = 0$ both sides of the equation become $\mathbf{tt}$ iff $s \in F$, so this is clear.

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

$$\begin{aligned} \mathsf{fattr}^{n+1}_G(I)(s)(\gamma_p) &= \bigvee_{s \to^G s'} \mathsf{fattr}^n_G(I)(s')(\gamma((s, s')) \wedge \gamma_p) \\ &= \bigvee_{s \to^H s'} \mathsf{fattr}^n_G(I)(s')(\gamma_p) \\ &= \bigvee_{s \to^H s'} \mathsf{attr}^n_H(I)(s') = \mathsf{attr}^{n+1}_H(I)(s); \end{aligned}$$

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

The operator fattr is monotone on the complete lattice $S \to (\mathbb{B}(N) \to \mathbb{B})$, thus fattr^* can be computed using a fixed-point algorithm. In Section 8 we will give an algorithm which uses *guard partitions* [27] 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 $s \to_x s'$ 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 $s_i \to_{x_i} s_{i+1}$ for all $i \geq 1$. The notion of configuration is unchanged, and a strategy for player i is now a function $\theta : \mathsf{Conf}_i \to \mathbb{N} \times S$ such that for all $\pi \in \mathsf{Conf}_i$, $\mathsf{end}(\pi) \to_{\theta(\pi)_1} \theta(\pi)_2$, where $\theta(\pi) = (\theta(\pi)_1, \theta(\pi)_2)$. The outcome of a strategy pair is an infinite path $(s_1, x_1, s_2, x_2, \dots) \in \mathsf{iPaths}(G)$ defined as expected.

The *reachability value* of an infinite path $\pi = (s_1, x_1, s_2, x_2, \dots)$ is defined to be $\mathsf{val}_\mathsf{R}(\pi) = \min\{x_1 + \cdots + x_{k-1} \mid s_k \in F\}$, where $\min \emptyset = \infty$ by convention, and the *minimum reachability value* of G is

$$\mathsf{val}_\mathsf{R}(G) = \inf_{\theta_1 \in \Theta_1} \sup_{\theta_2 \in \Theta_2} \mathsf{val}_\mathsf{R}(\mathsf{out}(\theta_1, \theta_2)).$$

That is, $\mathsf{val}_\mathsf{R}(\mathsf{out}(\theta_1, \theta_2))$ is the minimum sum of weights along any accepting finite prefix of the 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* $\mathsf{wattr} : (S \to \mathbb{N} \cup \{\infty\}) \to (S \to \mathbb{N} \cup \{\infty\})$ by

$$\mathsf{wattr}(U)(s) = \begin{cases} \min\limits_{s \to_x s'} x + U(s') & \text{if } s \in S_1, \\ \max\limits_{s \to_x s'} x + U(s') & \text{if } s \in S_2 \end{cases}$$

and let $\mathsf{wattr}^* = \min(\mathsf{id}, \mathsf{wattr}, \mathsf{wattr}^2, \dots)$. The *initial configuration* is $I : S \to \mathbb{N} \cup \{\infty\}$ given by

$$I(s) = \begin{cases} 0 & \text{if } s \in F, \\ \infty & \text{otherwise.} \end{cases}$$

The following seems to be folklore; note that it only holds under our assumption that all weights are non-negative. (See [9] for an extension to negative weights.)

Lemma 3. *The minimum reachability value of a weighted game structure* $G = (S_1, S_2, i, F, T)$ *is* $\mathsf{val}_{\mathsf{R}}(G) = \mathsf{wattr}^*(I)(i)$. □

The operator wattr is monotone on the complete lattice of functions $S \to \mathbb{N} \cup \{\infty\}$, thus 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 $\gamma : T \to \mathbb{B}(N)$. 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 $\mathsf{fwattr} : (S \to (\mathbb{B}(N) \to \mathbb{N} \cup \{\infty\})) \to (S \to (\mathbb{B}(N) \to \mathbb{N} \cup \{\infty\}))$ by

$$\mathsf{fwattr}(U)(s)(\varphi) = \begin{cases} \min\limits_{s \to_x s'} x + U(s')(\gamma((s,x,s')) \wedge \varphi) & \text{if } s \in S_1, \\ \max\limits_{s \to_x s'} x + U(s')(\gamma((s,x,s')) \wedge \varphi) & \text{if } s \in S_2 \end{cases}$$

and let $\mathsf{fwattr}^* = \min(\mathsf{id}, \mathsf{fwattr}, \mathsf{fwattr}^2, \dots)$. Let $I : S \to (\mathbb{B}(N) \to \mathbb{N} \cup \{\infty\})$ be the *initial configuration* given by

$$I(s)(\varphi) = \begin{cases} 0 & \text{if } s \in F \text{ and } \varphi \not\equiv \mathbf{ff}, \\ \infty & \text{otherwise.} \end{cases}$$

Theorem 4. *Let* $G = (S_1, S_2, i, F, T, \gamma)$ *be a featured weighted game structure and* $p \in px$*, then the minimum reachability value of* $\mathsf{proj}_p(G)$ *is* $\mathsf{val}_{\mathsf{R}}(\mathsf{proj}_p(G)) = \mathsf{fwattr}^*(I)(i)(\gamma_p)$.

Proof. Let $H = \mathsf{proj}_p(G) = (S_1, S_2, i, F, T')$; we need to prove that

$$\mathsf{fwattr}^*_G(I)(i)(\gamma_p) = \mathsf{wattr}^*_H(I)(i).$$

We show inductively that for all $n \geq 0$ and all $s \in S$, $\mathsf{fwattr}^n_G(I)(s)(\gamma_p) = \mathsf{wattr}^n_H(I)(s)$, which will imply the conclusion. For $n = 0$ both sides of the equation become 0 if $s \in F$ and ∞ otherwise, so the base case is clear.

Now let $n \geq 0$ and assume that for all $s' \in S$, $\mathsf{fwattr}^n_G(I)(s')(\gamma_p) = \mathsf{wattr}^n_H(I)(s')$. Let $s \in S_1$, then

$$\begin{aligned} \mathsf{fwattr}^{n+1}_G(I)(s)(\gamma_p) &= \min_{s \to^G_x s'} x + \mathsf{fwattr}^n_G(I)(s')(\gamma((s,x,s')) \wedge \gamma_p) \\ &= \min_{s \to^H_x s'} x + \mathsf{fwattr}^n_G(I)(s')(\gamma_p) \\ &= \min_{s \to^H_x s'} x + \mathsf{wattr}^n_H(I)(s') = \mathsf{wattr}^{n+1}_H(I)(s); \end{aligned}$$

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

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 $\mathsf{val}_\lambda(\pi) = x_1 + \lambda x_2 + \lambda^2 x_3 + \cdots$, and the discounted value of a game G is $\mathsf{val}_\lambda(G) = \sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} \mathsf{val}_\lambda(\mathsf{out}(\theta_1, \theta_2))$. 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 [48] in terms of attractors. Define the *discounted attractor* $\mathsf{dattr} : (S \to \mathbb{R}) \to (S \to \mathbb{R})$ by

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

Lemma 5. *Let $G = (S_1, S_2, i, T)$ be a weighted game structure. The equation system $V = \mathsf{dattr}(V)$ has a unique solution dattr^*, and the discounted value of G is $\mathsf{val}_\lambda(G) = \mathsf{dattr}^*(i)$.* □

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* $\mathsf{fdattr} : (S \to (\mathbb{B}(N) \to \mathbb{R})) \to (S \to (\mathbb{B}(N) \to \mathbb{R}))$ by

$$\mathsf{fdattr}(U)(s)(\varphi) = \begin{cases} \max\limits_{s \to_x s'} x + \lambda U(s')(\gamma((s, x, s')) \wedge \varphi) & \text{if } s \in S_1, \\ \min\limits_{s \to_x s'} x + \lambda U(s')(\gamma((s, x, s')) \wedge \varphi) & \text{if } s \in S_2. \end{cases}$$

Theorem 6. *Let $G = (S_1, S_2, i, T, \gamma)$ be a featured weighted game structure. The equation system $V = \mathsf{fdattr}(V)$ has a unique solution fdattr^*, and for any $p \in px$, the discounted value of $\mathsf{proj}_p(G)$ is $\mathsf{val}_\lambda(\mathsf{proj}_p(G)) = \mathsf{fdattr}^*(i)(\gamma_p)$.*

Proof. Define a metric on $S \to (\mathbb{B}(N) \to \mathbb{R})$ 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 $d(\mathsf{fdattr}(U_1), \mathsf{fdattr}(U_2)) \leq \lambda d(U_1, U_2)$ for any two functions U_1, U_2, that is, fdattr is a *contraction* on the complete metric space $S \to \mathbb{R}^{\mathbb{B}(N)}$. By the Banach fixed-point theorem, fdattr has a unique fixed point which is fdattr^*.

Let $p \in px$ and $H = \mathsf{proj}_p(G) = (S_1, S_2, i, T')$; we need to show that $\mathsf{fattr}^*_H(i) = \mathsf{fdattr}^*_G(i)(\gamma_p)$. Now for any $U : S \to (\mathbb{B}(N) \to \mathbb{R})$ and $s \in S_1$,

$\mathsf{fdattr}(U)(s)(\gamma_p) = \max_{s \to^G_x s'} x + \lambda U(s')(\gamma((s,x,s')) \wedge \gamma_p) = \max_{s \to^H_x s'} x + \lambda U(s')(\gamma_p)$, and the same can be shown if $s \in S_2$ instead. Hence the equation systems defining dattr^*_H and $\mathsf{fdattr}^*_G(\cdot)(\gamma_p)$ are the same; consequently, also their unique fixed points are equal. □

4.3. Example

We show the computation of fdattr^* for the example from Figure 5; recall that this is a $\sqrt{\lambda}$-discounted game. For any $\varphi \in \mathbb{B}(N)$, and writing $i = ((s^1_0, s^2_0))$, we have

$$\begin{aligned}\mathsf{fdattr}^*(i)(\varphi) &= \sqrt{\lambda}\mathsf{fdattr}^*((s^1_1, s^2_0, \mathsf{ins}))(\varphi)\\ &= \sqrt{\lambda}^2\mathsf{fdattr}^*((s^1_1, s^2_1))(\varphi)\end{aligned}$$

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

$$\begin{aligned}&= \max\begin{cases}\sqrt{\lambda}^3 \cdot 0.2\frac{1}{\sqrt{\lambda}} + \sqrt{\lambda}^4\mathsf{fdattr}^*(i)(\varphi \wedge €)\\ \sqrt{\lambda}^4\mathsf{fdattr}^*((s^1_2, s^2_2))(\varphi \wedge \$)\end{cases}\\ &= \max\begin{cases}\sqrt{\lambda}^3 \cdot 0.2\frac{1}{\sqrt{\lambda}} + \sqrt{\lambda}^4\mathsf{fdattr}^*(i)(\varphi \wedge €)\\ \sqrt{\lambda}^5 \cdot 0.4\frac{1}{\sqrt{\lambda}} + \sqrt{\lambda}^6\mathsf{fdattr}^*(i)(\varphi \wedge \$)\end{cases}\\ &= \max\begin{cases}0.2\lambda + \lambda^2\mathsf{fdattr}^*(i)(\varphi \wedge €)\\ 0.4\lambda^2 + \lambda^3\mathsf{fdattr}^*(i)(\varphi \wedge \$)\end{cases}.\end{aligned}$$

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

For $p = \{\$\}$, the equation simplifies to

$$\mathsf{fdattr}^*(i)(\gamma_{\{\$\}}) = 0.4\lambda^2 + \lambda^3\mathsf{fdattr}^*(i)(\gamma_{\{\$\}}),$$

hence $\mathsf{val}_\lambda(\mathsf{proj}_{\{\$\}}(G)) = 0.4\frac{\lambda^2}{1-\lambda^3}$. For $p = \{€, \$\}$, no simplifications are possible; for $\lambda = 0.99$ a standard fixed-point iteration yields $\mathsf{val}_\lambda(\mathsf{proj}_{\{€,\$\}}(G)) = 13.2$.

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 (with negative weights permitted), and also the notions of configurations, strategies, and outcome remain unchanged.

Let $v_0 \in \mathbb{N}$. An infinite path $(s_1, x_1, s_2, x_2, \dots) \in \mathsf{iPaths}(G)$ 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 $\theta_2 \in \Theta_2$, $\mathsf{out}(\theta_1, \theta_2,)$ is energy positive with initial credit v_0.

The following procedure, first discovered in [10], 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 $W = \{0, \ldots, M, \top\}$, where $\top$ is the greatest element, and define an operation $\ominus : W \times \mathbb{Z} \to W$ by $x \ominus y = \max(0, x-y)$ if $x \neq \top$ and $x - y \leq M$; $\top$ otherwise.

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

$$\mathsf{eattr}(U)(s) = \begin{cases} \min\limits_{s \to_x s'} U(s') \ominus x & \text{if } s \in S_1, \\ \max\limits_{s \to_x s'} U(s') \ominus x & \text{if } s \in S_2 \end{cases}$$

and let $\mathsf{eattr}^* = \max(\mathsf{id}, \mathsf{eattr}, \mathsf{eattr}^2, \ldots)$. The *initial configuration* is $I : S \to W$ given by $I(s) = 0$ for all $s \in S$. The following is proven in [10] which also shows that energy games can be decided in pseudo-polynomial time.

Lemma 7. *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 $v_0 \geq \mathsf{eattr}^*(I)(i)$.* □

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* $\mathsf{feattr} : (S \to (\mathbb{B}(N) \to W)) \to (S \to (\mathbb{B}(N) \to W))$ by

$$\mathsf{feattr}(U)(s)(\varphi) = \begin{cases} \min\limits_{s \to_x s'} U(s')(\gamma((s, x, s')) \wedge \varphi) \ominus x & \text{if } s \in S_1, \\ \max\limits_{s \to_x s'} U(s')(\gamma((s, x, s')) \wedge \varphi) \ominus x & \text{if } s \in S_2 \end{cases}$$

and let $\mathsf{feattr}^* = \max(\mathsf{id}, \mathsf{feattr}, \mathsf{feattr}^2, \ldots)$. The *initial configuration* is $I : S \to (\mathbb{B}(N) \to W)$ given by

$$I(s)(\varphi) = \begin{cases} 0 & \text{if } \varphi \not\equiv \mathbf{ff}, \\ \top & \text{otherwise.} \end{cases}$$

Theorem 8. *Let $G = (S_1, S_2, i, T, \gamma)$ be a featured weighted game structure and $v_0 : \mathbb{B}(N) \to \mathbb{N}$. Let $p \in px$, then player 1 wins the energy game on $\mathsf{proj}_p(G)$ with initial credit $v_0(\gamma_p)$ iff $v_0(\gamma_p) \geq \mathsf{feattr}^*(I)(i)(\gamma_p)$.*

Proof. Let $H = \mathsf{proj}_p(G) = (S_1, S_2, i, T')$; we need to prove that

$$\mathsf{feattr}^*_G(I)(i)(\gamma_p) = \mathsf{eattr}^*_H(I)(i).$$

We show inductively that for all $n \geq 0$ and all $s \in S$, $\mathsf{feattr}^n_G(I)(s)(\gamma_p) = \mathsf{eattr}^n_H(I)(s)$, 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 $s' \in S$,

$$\mathsf{feattr}^n_G(I)(s')(\gamma_p) = \mathsf{eattr}^n_H(I)(s').$$

Let $s \in S_1$, then

$$\begin{aligned}\mathsf{feattr}_G^{n+1}(I)(s)(\gamma_p) &= \min_{s \to_x^G s'} \mathsf{feattr}_G^n(I)(s')(\gamma((s,x,s')) \wedge \gamma_p) \ominus x \\ &= \min_{s \to_x^H s'} \mathsf{feattr}_G^n(I)(s')(\gamma_p) \ominus x \\ &= \min_{s \to_x^H s'} \mathsf{eattr}_H^n(I)(s') \ominus x = \mathsf{eattr}_H^{n+1}(I)(s);\end{aligned}$$

similarly for $s \in S_2$. □

5.3. Example

We show the computation of feattr^* for the example from Figure 6. 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 $\mathsf{feattr}^*(I) = f^*$, then for any $\varphi \in \mathbb{B}(N)$,

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

For $p = \emptyset$, only the first of these four lines contributes to the fixed point, which thus becomes $f^*(i)(\gamma_\emptyset) = \mathsf{feattr}^*(I)(i)(\gamma_\emptyset) = 0$. 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 $f^*(i)(\gamma_{\{\mathsf{big}\}}) = \top$ (player 1 cannot win this game) and $f^*(i)(\gamma_{\{\mathsf{ext}\}}) = f^*(i)(\gamma_{\{\mathsf{ext},\mathsf{big}\}}) = 0$.

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 $p : S \to \mathbb{N}$; we again assume these to be non-blocking. The notions of configurations, strategies and outcomes remain unchanged.

For an infinite path $\pi = (s_1, s_2, \dots) \in \mathsf{iPaths}(G)$ let $\mathsf{prio}(\pi) = \liminf_{n\to\infty} p(n)$ be the lowest priority which occurs infinitely often in π. The *parity game* on G is to decide whether there exists a strategy $\theta_1 \in \Theta_1$ such that for all $\theta_2 \in \Theta_2$, $\mathsf{prio}(\mathsf{out}(\theta_1, \theta_2))$ 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 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 [35]. 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 $M' = \{0\} \times \{0, \dots, p_1\} \times \{0\} \times \{0, \dots, p_3\} \times \cdots \times \{0, \dots, p_d\}$; if d is even, then $M' = \{0\} \times \{0, \dots, p_1\} \times \{0\} \times \{0, \dots, p_3\} \times \cdots \times \{0\}$.

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 $\preceq_k$ on M by $x \preceq_k y$ iff $x \leq_k y$ if k is odd; $x <_k y$ or $x = y = \top$ if k is even. Define a function $\mathsf{prog} : (S \to M) \times S \times S \to M$ by $\mathsf{prog}(U, s, s') = \min\{m \in M \mid m \succeq_{p(s)+1} U(s')\}$.

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

$$\mathsf{pattr}(U)(s) = \begin{cases} \min\limits_{s\to s'} \mathsf{prog}(U, s, s') & \text{if } s \in S_1, \\ \max\limits_{s\to s'} \mathsf{prog}(U, s, s') & \text{if } s \in S_2 \end{cases}$$

and let $\mathsf{pattr}^* = \max(\mathsf{id}, \mathsf{pattr}, \mathsf{pattr}^2, \dots)$. The *initial configuration* is $I : S \to M$ given by $I(s) = (0, \dots, 0)$ for all $s \in S$. The following is shown in [35], together with the fact that parity games are decidable in pseudo-polynomial time.

Lemma 9. *Let $G = (S_1, S_2, i, T, p)$ be a priority game structure, then player 1 wins the parity game on G iff $\mathsf{pattr}^*(I)(i) \neq \top$.* □

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 $\gamma : T \to \mathbb{B}(N)$. We again assume these to be non-blocking. Let $d = \max\{p(s) \mid s \in S\}$ and M be defined as above.

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

$$\mathsf{fpattr}(U)(s)(\varphi) = \begin{cases} \min\limits_{s\to s'} \mathsf{fprog}(U, s, s')(\gamma((s, s')) \wedge \varphi) & \text{if } s \in S_1, \\ \max\limits_{s\to s'} \mathsf{fprog}(U, s, s')(\gamma((s, s')) \wedge \varphi) & \text{if } s \in S_2 \end{cases}$$

and let $\mathsf{fpattr}^* = \max(\mathsf{id}, \mathsf{fpattr}, \mathsf{fpattr}^2, \dots)$. The *initial configuration* is $I : S \to (\mathbb{B}(N) \to M)$ given by

$$I(s)(\varphi) = \begin{cases} (0, \dots, 0) & \text{if } \varphi \not\equiv \mathbf{ff}, \\ \top & \text{otherwise.} \end{cases}$$

Theorem 10. *Let $G = (S_1, S_2, i, T, p, \gamma)$ be a featured priority game structure and $p \in px$, then player 1 wins the parity game on $\mathsf{proj}_p(G)$ iff $\mathsf{fpattr}^*(I)(i)(\gamma_p) \neq \top$.*

Proof. Let $H = \mathsf{proj}_p(G) = (S_1, S_2, i, T', p)$; we need to prove that

$$\mathsf{fpattr}^*_G(I)(i)(\gamma_p) = \mathsf{pattr}^*_H(I)(i).$$

We show inductively that for all $n \geq 0$ and all $s \in S$, $\mathsf{fpattr}^n_G(I)(s)(\gamma_p) = \mathsf{pattr}^n_H(I)(s)$, 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 $s' \in S$,

$$\mathsf{fpattr}^n_G(I)(s')(\gamma_p) = \mathsf{pattr}^n_H(I)(s').$$

Let $s \in S_1$, then

$$\begin{aligned}
\mathsf{fpattr}^{n+1}_G(I)(s)(\gamma_p) &= \max \begin{cases} \mathsf{fpattr}^n_G(I)(s)(\gamma_p) \\ \min\limits_{s \to^G s'} \mathsf{fprog}(\mathsf{fpattr}^n_G(I), s, s')(\gamma((s, s')) \wedge \gamma_p) \end{cases} \\
&= \max \begin{cases} \mathsf{pattr}^n_H(I)(s) \\ \min\limits_{s \to^H s'} \mathsf{fprog}(\mathsf{fpattr}^n_G(I), s, s')(\gamma_p) \end{cases} \\
&= \max \begin{cases} \mathsf{pattr}^n_H(I)(s) \\ \min\limits_{s \to^H s'} \min\{m \in M \mid m \succeq_{p(s)} \mathsf{fpattr}^n_G(I)(s')(\gamma_p)\} \end{cases} \\
&= \max \begin{cases} \mathsf{pattr}^n_H(I)(s) \\ \min\limits_{s \to^H s'} \min\{m \in M \mid m \succeq_{p(s)} \mathsf{pattr}^n_H(I)(s')\} \end{cases} \\
&= \max \begin{cases} \mathsf{pattr}^n_H(I)(s) \\ \min\limits_{s \to^H s'} \mathsf{prog}(\mathsf{pattr}^n_H(I), s, s') \end{cases} \\
&= \mathsf{pattr}^{n+1}_H(I)(s).
\end{aligned}$$

For $s \in S_2$ the reasoning is similar. □

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 $\sup_{\theta_2 \in \Theta_2} \mathsf{val}_{\mathsf{R}}(\mathsf{out}(\tilde{\theta}_1, \theta_2)) = \inf_{\theta_1 \in \Theta_1} \sup_{\theta_2 \in \Theta_2} \mathsf{val}_{\mathsf{R}}(\mathsf{out}(\theta_1, \theta_2))$ for reachability games; $\inf_{\theta_2 \in \Theta_2} \mathsf{val}_{\lambda}(\mathsf{out}(\tilde{\theta}_1, \theta_2)) = \sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} \mathsf{val}_{\lambda}(\mathsf{out}(\theta_1, \theta_2))$ 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 $\mathsf{end}(\pi) = \mathsf{end}(\pi')$ implies $\theta_1(\pi) = \theta_1(\pi')$ for all $\pi, \pi' \in \mathsf{Conf}_1$. Hence memoryless player-1 strategies are mappings $S_1 \to S$. It is well-known that it suffices to consider memoryless strategies for reachability games.

Define again $I : S \to \mathbb{B}$ by $I(s) = \mathbf{tt}$ iff $s \in F$. A memoryless player-1 strategy $\theta_1 : S_1 \to S$ is *locally optimal* if, for all $s \in S_1$, $\mathsf{attr}^*(I)(s) = \mathsf{attr}^*(I)(\theta_1(s))$; that is, among all options $s \to s'$, it $\theta_1(s)$ is such that $\mathsf{attr}^*(I)(s) = \bigvee_{s \to s'} \mathsf{attr}^*(I)(s')$ 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 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 $\gamma : T \to \mathbb{B}(N)$ 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 $\xi_i : \mathsf{Conf}_i \to (\mathbb{B}(N) \to S)$ such that for all $\pi \in \mathsf{Conf}_i$ and $\varphi \in \mathbb{B}(N)$, $\mathsf{end}(\pi) \to \xi_i(\pi)(\varphi)$. The set of featured strategies for player i is denoted Ξ_i. We define mappings $\Xi_i \times \mathbb{B}(N) \to \Theta_i$, denoted $(\xi_i, \varphi) \mapsto \xi_i(\varphi)$ and defined by $\xi_i(\varphi)(\pi) = \xi_i(\pi)(\varphi)$ for all $\pi \in \mathsf{Conf}_i$.

A pair of featured strategies $\xi_1 \in \Xi_1$, $\xi_2 \in \Xi_2$ defines a mapping

$$\mathsf{out}(\xi_1, \xi_2) : \mathbb{B}(N) \to \mathsf{iPaths}(G)$$

from feature guards to infinite paths in G, where $\mathsf{out}(\xi_1, \xi_2)(\varphi) = (s_1, s_2, \dots)$ is given by $s_1 = i$, $s_{2k} = \xi_1(s_1, \dots, s_{2k-1})(\varphi)$, and $s_{2k+1} = \xi_2(s_1, \dots, s_{2k})(\varphi)$.

Let $\varphi \in \mathbb{B}(N)$. Player 1 *wins the φ-reachability game* if there exists a strategy $\xi_1 \in \Xi_1$ such that for all $\xi_2 \in \Xi_2$, with $\mathsf{out}(\xi_1, \xi_2)(\varphi) = (s_1, s_2, \dots)$, there is an index $k \geq 1$ for which $s_k \in F$ and $\varphi \wedge \gamma((s_1, \dots, s_k)) \not\equiv \mathbf{ff}$.

Lemma 11. *Let $G = (S_1, S_2, i, F, T, \gamma)$ be a featured game structure and $p \in px$. Player 1 wins the reachability game in $\mathsf{proj}_p(G)$ iff she wins the γ_p-reachability game in G.*

Proof. Assume that player 1 wins the reachability game in $H = \mathsf{proj}_p(G)$. Then there is $\theta_1 \in \Theta_1$ such that for all $\theta_2 \in \Theta_2$, writing $\mathsf{out}(\theta_1, \theta_2) = (s_1, s_2, \dots)$, 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.,* $\gamma_p \wedge \gamma((s_1, \dots, s_k)) \not\equiv \mathbf{ff}$. 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 $\mathsf{out}(\xi_1, \xi_2)(\varphi) = (s_1, s_2, \dots)$, there is an index $k \geq 1$ for which $s_k \in F$ and $\varphi \wedge \gamma((s_1, \dots, s_k)) \not\equiv \mathbf{ff}$; that is, player 1 wins the γ_p-reachability game in G.

For the converse, assume that player 1 wins the γ_p-reachability game in G, and let $\xi_1 \in \Xi_1$ be such that for any $\xi_2 \in \Xi_2$, writing $\mathsf{out}(\xi_1, \xi_2)(\gamma_p) = (s_1, s_2, \dots)$, there is an index $k \geq 1$ for which $s_k \in F$ and $\gamma_p \wedge \gamma((s_1, \dots, s_k)) \not\equiv \mathbf{ff}$, *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 $(s_1, s_2), \cdots, (s_{k-1}, s_k)$ are present in H. Let $\theta_1 = \xi_1(\gamma_p)$. We have shown that for all $\theta_2 \in \Theta_2$, writing $\mathsf{out}(\theta_1, \theta_2) = (s_1, s_2, \dots)$, there is an index $k \geq 1$ for which $s_k \in F$; that is, player 1 wins the reachability game in H. □

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

Define, as before, $I : S \to (\mathbb{B}(N) \to \mathbb{B})$ by

$$I(s)(\varphi) = \begin{cases} \mathbf{tt} & \text{if } s \in F \text{ and } \varphi \not\equiv \mathbf{ff}, \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

A memoryless featured strategy $\xi_1 : S_1 \to (\mathbb{B}(N) \to S)$ is *locally optimal* if, for all $s \in S_1$ and $\varphi \in \mathbb{B}(N)$, $\mathsf{fattr}^*(I)(s)(\varphi) = \mathsf{fattr}^*(I)(\xi_1(s)(\varphi))(\gamma((s, \xi_1(s)(\varphi))) \wedge \varphi)$.

Theorem 12. *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 $\mathsf{proj}_p(G)$ for every $p \in px$.*

Proof. We show the second claim first. Write $H = \mathsf{proj}_p(G)$, assume ξ_1 to be locally optimal, write $\theta_1 = \xi_1(\gamma_p)$, and let $s \in S_1$. Then $\mathsf{attr}^*_H(I)(s) = \mathsf{fattr}^*_G(I)(s)(\gamma_p) = \mathsf{fattr}^*_G(I)(\theta_1(s))\ (\gamma((s, \theta_1(s))) \wedge \gamma_p) = \mathsf{fattr}^*_G(I)(\theta_1(s))(\gamma_p) = \mathsf{attr}^*_H(I)(\theta_1(s))$, thus θ_1 is locally optimal in $\mathsf{proj}_p(G)$.

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

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 $\theta_1 : S_1 \to \mathbb{N} \times S$. Such a strategy is locally optimal if $\mathsf{wattr}^*(I)(s) = \theta_1(s)_1 + \mathsf{wattr}^*(I)(\theta_1(s)_2)$ for all $s \in S_1$, where $I : S \to \mathbb{N}$ is defined by $I(s) = 0$ if $s \in F$; ∞ 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 $\xi_i : \mathsf{Conf}_i \to (\mathbb{B}(N) \to \mathbb{N} \times S)$. The outcome of a pair ξ_1, ξ_2 of featured strategies is again a mapping $\mathsf{out}(\xi_1, \xi_2) : \mathbb{B}(N) \to \mathsf{iPaths}(G)$ defined as expected.

The *featured reachability value* of a mapping $\pi : \mathbb{B}(N) \to \mathsf{iPaths}(G)$ is the function $\mathsf{fval}_\mathsf{R}(\pi) : \mathbb{B}(N) \to \mathbb{N} \cup \{\infty\}$ given by $\mathsf{fval}_\mathsf{R}(\pi)(\varphi) = \mathsf{val}_\mathsf{R}(\pi(\varphi))$, and the *featured minimum reachability value* of G is

$$\mathsf{fval}_\mathsf{R}(G) = \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \mathsf{fval}_\mathsf{R}(\mathsf{out}(\xi_1, \xi_2)),$$

where the order in $\mathbb{B}(N) \to \mathbb{N} \cup \{\infty\}$ is point-wise.

Lemma 13. *For $G = (S_1, S_2, i, F, T, \gamma)$ any featured weighted game structure and $p \in px$, $\mathsf{val}_\mathsf{R}(\mathsf{proj}_p(G)) = \mathsf{fval}_\mathsf{R}(G)(\gamma_p)$.*

Proof. Denoting strategy sets in G by Ξ_i and in $\mathsf{proj}_p(G)$ by Θ_i, we see that $\Theta_i = \Xi_i(\gamma_p)$. Then

$$\begin{aligned}
\mathsf{fval}_\mathsf{R}(G)(\gamma_p) &= \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \mathsf{fval}_\mathsf{R}(\mathsf{out}(\xi_1, \xi_2))(\gamma_p) \\
&= \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \mathsf{val}_\mathsf{R}(\mathsf{out}(\xi_1, \xi_2)(\gamma_p)) \\
&= \inf_{\xi_1 \in \Xi_1} \sup_{\xi_2 \in \Xi_2} \mathsf{val}_\mathsf{R}(\mathsf{out}(\xi_1(\gamma_p), \xi_2(\gamma_p))) \\
&= \inf_{\theta_1 \in \Xi_1(\gamma_p)} \sup_{\theta_2 \in \Xi_2(\gamma_p)} \mathsf{val}_\mathsf{R}(\mathsf{out}(\theta_1, \theta_2)) \\
&= \inf_{\theta_1 \in \Theta_1} \sup_{\theta_2 \in \Theta_2} \mathsf{val}_\mathsf{R}(\mathsf{out}(\theta_1, \theta_2)) = \mathsf{val}_\mathsf{R}(\mathsf{proj}_p(G)).
\end{aligned}$$

□

Define again $I : S \to (\mathbb{B}(N) \to \mathbb{N} \cup \{\infty\})$ by

$$I(s)(\varphi) = \begin{cases} 0 & \text{if } s \in F \text{ and } \varphi \not\equiv \mathbf{ff}, \\ \infty & \text{otherwise.} \end{cases}$$

Memoryless featured player-1 strategies are mappings $\xi_1 : S_1 \to (\mathbb{B}(N) \to \mathbb{N} \times S)$. Such a strategy is locally optimal if, for all $s \in S_1$ and $\varphi \in \mathbb{B}(N)$, $\mathsf{fwattr}^*(I)(s)(\varphi) = \xi_1(s)(\varphi)_1 + \mathsf{fwattr}^*(I)(\xi_1(s)(\varphi)_2)(\gamma((s, \xi_1(s)(\varphi)_1, \xi_1(s)(\varphi)_2)) \wedge \varphi)$.

Theorem 14. *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 $\mathsf{proj}_p(G)$ for every $p \in px$.*

Proof. Similar to the proof of Theorem 12. □

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 $\theta_1 : S_1 \to \mathbb{Z} \times S$ is locally optimal if $\mathsf{dattr}^*(s) = \theta_1(s)_1 + \lambda\mathsf{dattr}^*(\theta_1(s)_2)$ for all $s \in S_1$. Locally optimal strategies always exist and are optimal [48].

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

Lemma 15. *For $G = (S_1, S_2, i, T, \gamma)$ any featured weighted game structure and $p \in px$, $\mathsf{val}_\lambda(\mathsf{proj}_p(G)) = \mathsf{fval}_\lambda(G)(\gamma_p)$.*

Proof. Similar to the proof of Lemma 13. □

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

Theorem 16. *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 $\mathsf{proj}_p(G)$ for every $p \in px$.*

Proof. Similar to the proof of Theorem 12. □

7.4. Featured Energy Games

Let $G = (S_1, S_2, i, T)$ be a weighted game structure and define $M = \sum_{s \in S} \max(\{0\} \cup \{-x \mid (s, x, s') \in T\})$, $W = \{0, \ldots, M, \top\}$, and $I : S \to W$ by $I(s) = 0$ for all $s \in S$ as before. A memoryless player-1 strategy $\theta_1 : S_1 \to \mathbb{Z} \times S$ is locally optimal if $\mathsf{eattr}^*(I)(s) = \mathsf{eattr}^*(I)(\theta_1(s)_2) \ominus \theta_1(s)_1$ 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 [10].

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

Lemma 17. *Let $G = (S_1, S_2, i, T, \gamma)$ be a featured weighted game structure, $v_0 : \mathbb{B}(N) \to \mathbb{N}$, and $p \in px$. Player 1 wins the energy game with initial credit $v_0(\gamma_p)$ in $\mathsf{proj}_p(G)$ iff player 1 wins the γ_p-energy game in G with initial credit v_0.*

PROOF. Similar to the proof of Lemma 13. □

Define, as before, $I : S \to (\mathbb{B}(N) \to W)$ by

$$I(s)(\varphi) = \begin{cases} 0 & \text{if } \varphi \not\equiv \mathbf{ff}, \\ \top & \text{otherwise.} \end{cases}$$

A memoryless featured player-1 strategy $\xi_1 : S_1 \to (\mathbb{B}(N) \to \mathbb{Z} \times S)$ is locally optimal if, for all $s \in S_1$ and $\varphi \in \mathbb{B}(N)$,

$$\mathsf{feattr}^*(I)(s)(\varphi) = \\ \mathsf{feattr}^*(I)(\xi_1(s)(\varphi)_2)(\gamma((s, \xi_1(s)(\varphi)_1, \xi_1(s)(\varphi)_2)) \wedge \varphi) \ominus \xi_1(s)(\varphi)_1.$$

Theorem 18. *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 $\mathsf{proj}_p(G)$ for every $p \in px$.*

PROOF. Similar to the proof of Theorem 12. □

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 $I : S \to M$ by $I(s) = (0, \ldots, 0)$ for all $s \in S$. A memoryless player-1 strategy $\theta_1 : S_1 \to S$ is locally optimal if $\mathsf{pattr}^*(I)(s) = \mathsf{prog}(\mathsf{pattr}^*(I), s, \theta_1(s))$ 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 [35].

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 φ-parity game on G if there exists a featured strategy $\xi_1 \in \Xi_1$ such that for all $\xi_2 \in \Xi_2$, $\mathsf{prio}(\mathsf{out}(\xi_1, \xi_2)(\varphi))$ is an even number.

Lemma 19. *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 $\mathsf{proj}_p(G)$ iff player 1 wins the γ_p-parity game in G.*

PROOF. Similar to the proof of Lemma 13. □

Define again $I : S \to (\mathbb{B}(N) \to M)$ by

$$I(s)(\varphi) = \begin{cases} (0, \ldots, 0) & \text{if } \varphi \not\equiv \mathbf{ff}, \\ \top & \text{otherwise.} \end{cases}$$

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

Theorem 20. *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 $\mathsf{proj}_p(G)$ for every $p \in px$.*

PROOF. Similar to the proof of Theorem 12. □

8. Symbolic Computation

The goal of feature-based analysis is to compute properties pertaining to an FTS representation of an 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 computed by calculating closures of attractors. Hence we expose below feature-based algorithms for calculating these closures.

As usual for the field [15], the power of our feature-based algorithms lies not in their worst-case complexity, which is the same as for algorithms which would analyze each product separately; instead the advantage of feature-based algorithms in general is that partial similarities between products are exploited to reduce computation times.

8.1. Featured Reachability Games

Let $G = (S_1, S_2, i, F, T, \gamma)$ be a featured game structure and define $I : S \to (\mathbb{B}(N) \to \mathbb{B})$ by

$$I(s)(\varphi) = \begin{cases} \mathbf{tt} & \text{if } s \in F \text{ and } \varphi \not\equiv \mathbf{ff}, \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

Conceptually, the procedure for calculating $J = \mathsf{fattr}^*(I)$ is a fixed-point algorithm: initialize $J \leftarrow I$ and update $J \leftarrow J \vee \mathsf{fattr}(J)$ until J stabilizes.

In order to symbolically represent functions from $\mathbb{B}(N)$, we use guard partitions, see also [27]. A *guard partition* of px is a set $P \subseteq \mathbb{B}(N)$ such that $[\![\bigvee P]\!] = px$, $[\![\varphi]\!] \neq \emptyset$ for all $\varphi \in P$, and $[\![\varphi_1]\!] \cap [\![\varphi_2]\!] = \emptyset$ for all $\varphi_1, \varphi_2 \in P$ with $\varphi_1 \neq \varphi_2$. The set of all guard partitions of px is denoted $GP \subseteq 2^{\mathbb{B}(N)}$. The interested reader is referred to [27] for further details.

A function $f : P \to X$, 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 $f : P \to X$ which is *not* canonical may be reduced into an equivalent canonical function $f' : P' \to X$ using the function Reduce shown in Figure 7, which reduces the partition P so that any formulae $\varphi, \psi \in P$ on which f takes the same value are combined into $\varphi \vee \psi$. Every function $\mathbb{B}(N) \to X$ has a unique representation as a canonical function $P \to X$ for some $P \in GP$.

The function for featured computation of attractors is shown in Figure 9. It uses the functions Land and Lor, shown in Figure 8, which compute logical operations on functions $P \to \mathbb{B}$: for $f_1 : P_1 \to \mathbb{B}$ and $f_2 : P_2 \to \mathbb{B}$, Land returns $f' = f_1 \wedge f_2$, and Lor returns $f' = f_1 \vee f_2$. Both work by constructing a common refinement of the partitions P_1 and P_2 and then assigning appropriate values to the formulae in the refinement; note that the only difference between Land and Lor is this assignment in line 7.

The function Fattr in Figure 9 computes one iteration of fattr for all states $s \in S$. It does so by traversing all transitions $s \to s'$ (note that s' might be equal to s), restricting the partitions at s' to $\gamma((s, s'))$ (line 9), and then computing $U'_s = \bigvee_{s \to s'} V_{s'}$ or $U'_s = \bigwedge_{s \to s'} V_{s'}$, depending on whether $s \in S_1$ or $s \in S_2$, in

```
1: function REDUCE(f : P → X): P′ → X
2:     P′, f′ ← ∅
3:     while P ≠ ∅ do
4:         Pick and remove φ from P
5:         x ← f(φ)
6:         for all ψ ∈ P do
7:             if f(ψ) = x then
8:                 φ ← φ ∨ ψ
9:                 P ← P \ {ψ}
10:        P′ ← P′ ∪ {φ}
11:        f′(φ) ← x
12:    return f′ : P′ → X
```

Figure 7: Algorithm which computes canonicalization.

```
1: function LAND(f_1 : P_1 → 𝔹, f_2 : P_2 → 𝔹): P → 𝔹
2:     P, f ← ∅
3:     for all φ_1 ∈ P_1 do
4:         for all φ_2 ∈ P_2 do
5:             if [[φ_1 ∧ φ_2]] ≠ ∅ then
6:                 P ← P ∪ {φ_1 ∧ φ_2}
7:                 f(γ_1∧γ_2) ← f_1(γ_1)∧f_2(γ_2)
8:     return REDUCE(f)
```

```
1: function LOR(f_1 : P_1 → 𝔹, f_2 : P_2 → 𝔹): P → 𝔹
2:     P, f ← ∅
3:     for all φ_1 ∈ P_1 do
4:         for all φ_2 ∈ P_2 do
5:             if [[φ_1 ∧ φ_2]] ≠ ∅ then
6:                 P ← P ∪ {φ_1 ∧ φ_2}
7:                 f(γ_1∧γ_2) ← f_1(γ_1)∨f_2(γ_2)
8:     return REDUCE(f)
```

Figure 8: Algorithms for logical and and or.

```
1: function FATTR(U : S → (P → 𝔹)): S → (P′ → 𝔹)
2:   U′ ← ∅
3:   for all s ∈ S do
4:     P′_s, U′_s ← ∅
5:     for all s → s′ do
6:       Q_{s′}, V_{s′} ← ∅
7:       while P_{s′} ≠ ∅ do
8:         Pick and remove φ from P_{s′}
9:         ψ ← γ((s, s′)) ∧ φ
10:        if [[ψ]] ≠ ∅ then
11:          Q_{s′} ← Q_{s′} ∪ {ψ}
12:          V_{s′}(ψ) ← U_{s′}(φ)
13:      Q_{s′} ← Q_{s′} ∪ {¬γ((s, s′))}
14:      V_{s′}(¬γ((s, s′))) ← ff
15:      if s ∈ S_1 then
16:        U′_s ← LOR(U′_s, V_{s′})
17:      if s ∈ S_2 then
18:        U′_s ← LAND(U′_s, V_{s′})
19:    U′_s ← REDUCE(U′_s)
20:  return U′ : S → (P′ → 𝔹)
```

Figure 9: Computation of fattr.

lines 15f. The algorithm for the fixed-point iteration to compute fattr^* is, then, shown in Figure 10.

8.2. Featured Minimum Reachability Games

Let $G = (S_1, S_2, i, F, T, \gamma)$ be a featured weighted game structure and

$$I(s)(\varphi) = \begin{cases} 0 & \text{if } s \in F \text{ and } \varphi \not\equiv \mathbf{ff}, \\ \infty & \text{otherwise} \end{cases}$$

as before. The computation of the fixed point $\mathsf{fwattr}^*(I)$ is similar to the one in the previous section and shown in Figures 11 through 13.

The only essential differences between FATTR and FWATTR are to be found in line 12, which now adds the weights of the respective transitions, and lines 16 and 18 where the logical operations have been replaced by maximum and minimum, see Figure 11. Similarly, the only essential differences between FATTR* and FWATTR* are the assignments of bottom and top values in lines 6 and 8 and the use of MIN instead of LOR in line 11.

8.3. Featured Discounted Games

The algorithms for computing values of featured discounted games are shown in Figures 14 and 15. They use functions MIN and MAX similar to the ones in

```
1: function FATTR*(G = (S_1, S_2, i, F, T, γ)): S → (P → 𝔹)
2:   J = ∅
3:   for all s ∈ S do
4:     U_s ← {tt}
5:     if s ∈ F then
6:       J_s(tt) ← tt
7:     else
8:       J_s(tt) ← ff
9:   repeat
10:    J_old ← J
11:    J ← LOR(J, FATTR(J))
12:  until J = J_old
13:  return J
```

Figure 10: Fixed-point iteration for fattr^*.

```
1: function MIN(f_1 : P_1 → ℕ ∪ {∞}, f_2 : P_2 → ℕ ∪ {∞}): P → ℕ ∪ {∞}
2:   P, f ← ∅
3:   for all φ_1 ∈ P_1 do
4:     for all φ_2 ∈ P_2 do
5:       if [[φ_1 ∧ φ_2]] ≠ ∅ then
6:         P ← P ∪ {φ_1 ∧ φ_2}
7:         f(γ_1 ∧ γ_2) ← min(f_1(γ_1), f_2(γ_2))
8:   return REDUCE(f)
```

```
1: function MAX(f_1 : P_1 → ℕ ∪ {∞}, f_2 : P_2 → ℕ ∪ {∞}): P → ℕ ∪ {∞}
2:   P, f ← ∅
3:   for all φ_1 ∈ P_1 do
4:     for all φ_2 ∈ P_2 do
5:       if [[φ_1 ∧ φ_2]] ≠ ∅ then
6:         P ← P ∪ {φ_1 ∧ φ_2}
7:         f(γ_1 ∧ γ_2) ← max(f_1(γ_1), f_2(γ_2))
8:   return REDUCE(f)
```

Figure 11: Algorithms for minimum and maximum.

```
1: function FWATTR(U : S → (P → ℕ ∪ {∞})): S → (P′ → ℕ ∪ {∞})
2:   U′ ← ∅
3:   for all s ∈ S do
4:     P′_s, U′_s ← ∅
5:     for all s →_x s′ do
6:       Q_{s′}, V_{s′} ← ∅
7:       while P_{s′} ≠ ∅ do
8:         Pick and remove φ from P_{s′}
9:         ψ ← γ((s, s′)) ∧ φ
10:        if [[ψ]] ≠ ∅ then
11:          Q_{s′} ← Q_{s′} ∪ {ψ}
12:          V_{s′}(ψ) ← x + U_{s′}(φ)
13:      Q_{s′} ← Q_{s′} ∪ {¬γ((s, s′))}
14:      V_{s′}(¬γ((s, s′))) ← ff
15:      if s ∈ S_1 then
16:        U′_s ← MIN(U′_s, V_{s′})
17:      if s ∈ S_2 then
18:        U′_s ← MAX(U′_s, V_{s′})
19:    U′_s ← REDUCE(U′_s)
20:  return U′ : S → (P′ → ℕ ∪ {∞})
```

Figure 12: Computation of fwattr.

```
1: function FWATTR*(G = (S_1, S_2, i, F, T, γ)): S → (P → ℕ ∪ {∞})
2:   J = ∅
3:   for all s ∈ S do
4:     U_s ← {tt}
5:     if s ∈ F then
6:       J_s(tt) ← 0
7:     else
8:       J_s(tt) ← ∞
9:   repeat
10:    J_old ← J
11:    J ← MIN(J, FWATTR(J))
12:  until J = J_old
13:  return J
```

Figure 13: Fixed-point iteration for fwattr*.

```
1: function FDATTR(U : S → (P → ℝ), λ): S → (P′ → ℝ)
2:   U′ ← ∅
3:   for all s ∈ S_1 do
4:     P′_s, U′_s ← ∅
5:     for all s →_x s′ do
6:       Q_{s′}, V_{s′} ← ∅
7:       while P_{s′} ≠ ∅ do
8:         Pick and remove φ from P_{s′}
9:         ψ ← γ((s, s′)) ∧ φ
10:        if [[ψ]] ≠ ∅ then
11:          Q_{s′} ← Q_{s′} ∪ {ψ}
12:          V_{s′}(ψ) ← x + λU_{s′}(φ)
13:      Q_{s′} ← Q_{s′} ∪ {¬γ((s, s′))}
14:      V_{s′}(¬γ((s, s′))) ← ff
15:      if s ∈ S_1 then
16:        U′_s ← MAX(U′_s, V_{s′})
17:      if s ∈ S_2 then
18:        U′_s ← MIN(U′_s, V_{s′})
19:    U′_s ← REDUCE(U′_s)
20:  return U′ : S → (P′ → ℝ)
```

Figure 14: Computation of fdattr.

Figure 11. The function FDATTR in Figure 14 is essentially like FWATTR, except for the discounting applied in line 12 and the swapping of MAX and MIN in lines 16 and 18.

The function FDATTR* in Figure 15 takes a discounting factor λ and a precision ε as inputs; λ is used for the iteration in FDATTR, and ε is used to terminate the computation of fdattr^* once a desired level of precision has been reached.

8.4. Featured Energy Games

Let $G = (S_1, S_2, i, T, \gamma)$ be a featured weighted game structure, $M = \sum_{s \in S} \max(\{0\} \cup \{-x \mid (s, x, s') \in T\})$, $W = \{0, \ldots, M, \top\}$, and

$$I(s)(\varphi) = \begin{cases} 0 & \text{if } \varphi \not\equiv \mathbf{ff}, \\ \top & \text{otherwise} \end{cases}$$

as before. The algorithms for computing the fixed point $\mathsf{feattr}^*(I)$ are shown in Figures 16 and 17. In FEATTR, the only changes are again in lines 12, 16, and 18; FEATTR* is similar to FWATTR* except for the initial configuration and the swap of MIN and MAX in the iteration.

```
1: function FDATTR*(G = (S_1, S_2, i, T, γ), λ, ε): S → (P → ℝ)
2:   J = ∅
3:   for all s ∈ S do
4:     U_s ← {tt}
5:     J_s(tt) ← 0
6:   repeat
7:     J_old ← J
8:     J ← FDATTR(J, λ)
9:   until ‖J − J_old‖ < ε
10:  return J
```

Figure 15: Fixed-point iteration for fdattr*.

```
1: function FEATTR(U : S → (P → W)): S → (P' → W)
2:   U' ← ∅
3:   for all s ∈ S_1 do
4:     P'_s, U'_s ← ∅
5:     for all s →_x s' do
6:       Q_s', V_s' ← ∅
7:       while P_s' ≠ ∅ do
8:         Pick and remove φ from P_s'
9:         ψ ← γ((s, s')) ∧ φ
10:        if [[ψ]] ≠ ∅ then
11:          Q_s' ← Q_s' ∪ {ψ}
12:          V_s'(ψ) ← U_s'(φ) ⊖ x
13:      Q_s' ← Q_s' ∪ {¬γ((s, s'))}
14:      V_s'(¬γ((s, s'))) ← ff
15:      if s ∈ S_1 then
16:        U'_s ← MIN(U'_s, V_s')
17:      if s ∈ S_2 then
18:        U'_s ← MAX(U'_s, V_s')
19:    U'_s ← REDUCE(U'_s)
20:  return U' : S → (P' → W)
```

Figure 16: Computation of feattr.

```
1: function FEATTR*(G = (S_1, S_2, i, T, γ)): S → (P → ℕ ∪ {∞})
2:   J = ∅
3:   for all s ∈ S do
4:     U_s ← {tt}
5:     J_s(tt) ← 0
6:   repeat
7:     J_old ← J
8:     J ← MAX(J, FEATTR(J))
9:   until J = J_old
10:  return J
```

Figure 17: Fixed-point iteration for feattr^*.

```
1: function FPROG(U : S → (P → M), s, s'): P' → M
2:   U', P' ← ∅
3:   while P_{s'} ≠ ∅ do
4:     Pick and remove φ from P_{s'}
5:     P' ← P' ∪ {φ}
6:     U'(φ) ← min{m ∈ M | m ⪰_{p(s)} U_{s'}(φ)}
7:   return REDUCE(U')
```

Figure 18: Computation of fprog.

8.5. Featured Parity Games

Let $G = (S_1, S_2, i, T, p, \gamma)$ be a featured priority game structure, $d = \max\{p(s) \mid s \in S\}$, $M \subseteq \mathbb{N}^d \cup \{\top\}$, and

$$I(s)(\varphi) = \begin{cases} (0, \ldots, 0) & \text{if } \varphi \not\equiv \mathbf{ff}, \\ \top & \text{otherwise} \end{cases}$$

as in Section 6. The algorithms for computing the fixed point $\mathsf{fpattr}^*(I)$ are shown in Figures 18 through 20. Again, FPATTR is similar to FEATTR except for lines 12, 16, and 18, and FPATTR* is similar to FEATTR*.

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 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

```
1: function FPATTR(U : S → (P → M)): S → (P′ → M)
2:   U′ ← ∅
3:   for all s ∈ S do
4:     P′_s, U′_s ← ∅
5:     for all s → s′ do
6:       Q_{s′}, V_{s′} ← ∅
7:       while P_{s′} ≠ ∅ do
8:         Pick and remove φ from P_{s′}
9:         ψ ← γ((s, s′)) ∧ φ
10:        if [[ψ]] ≠ ∅ then
11:          Q_{s′} ← Q_{s′} ∪ {ψ}
12:          V_{s′}(ψ) ← FPROG(U, s, s′)(φ)
13:      Q_{s′} ← Q_{s′} ∪ {¬γ((s, s′))}
14:      V_{s′}(¬γ((s, s′))) ← ff
15:      if s ∈ S_1 then
16:        U′_s ← MIN(U′_s, V_{s′})
17:      if s ∈ S_2 then
18:        U′_s ← MAX(U′_s, V_{s′})
19:    U′_s ← REDUCE(U′_s)
20:  return U′ : S → (P′ → M)
```

Figure 19: Computation of fpattr.

```
1: function FPATTR*(G = (S_1, S_2, i, T, p, γ)): S → (P → M)
2:   J = ∅
3:   for all s ∈ S do
4:     U_s ← {tt}
5:     J_s(tt) ← 0
6:   repeat
7:     J_old ← J
8:     J ← MAX(J, FPATTR(J))
9:   until J = J_old
10:  return J
```

Figure 20: Fixed-point iteration for fpattr*.

loops [48]. [10] shows 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 [34, 48] 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 the featured setting. In future work we plan to implement our algorithms and integrate them into the mCRL2 toolset [11, 42], using guard partitions, late splitting, and 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 [22, 41, 43]. For real-time systems, we are looking into extending *timed games* [4] with features, analogously to the featured timed automata of [22]; for probabilistic systems, a featured extension of stochastic games [39] 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] Joanne M. Atlee, Uli Fahrenberg, and Axel Legay. Measuring behaviour interactions between product-line features. In *FormaliSE@ICSE*, pages 20–25. IEEE, 2015.

[3] 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.

[4] 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 *Lecture Notes in Computer Science*, pages 121–125. Springer, 2007.

[5] Harsh Beohar and Mohammad Reza Mousavi. Input-output conformance testing for software product lines. *Journal of Logic and Algebraic Methods in Programming*, 85(6):1131–1153, 2016.

[6] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Logical Methods in Computer Science*, 18(1), 2022.

[7] 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, 2018.

[8] Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, and Alberto L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*, volume 2 of *The Kluwer International Series in Engineering and Computer Science*. Springer, 1984.

[9] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 54(1):85–125, 2017.

[10] Luboš Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38:97–118, 2011.

[11] 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. In Tomáš Vojnar and Lijun Zhang, editors, *TACAS (II)*, volume 11428 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2019.

[12] Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. ProFeat: Feature-oriented engineering for family-based probabilistic model checking. *Formal Aspects of Computing*, 30(1):45–75, 2018.

[13] Andreas Classen, Maxime Cordy, Patrick Heymans, Axel Legay, and Pierre-Yves Schobbens. Model checking software product lines with SNIP. *International Journal on Software Tools for Technology Transfer*, 14(5):589–612, 2012.

[14] Andreas Classen, Maxime Cordy, Patrick Heymans, Axel Legay, and Pierre-Yves Schobbens. Formal semantics, modular specification, and symbolic verification of product-line behaviour. *Science of Computer Programming*, 80:416–439, 2014.

[15] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-François Raskin. Featured transition systems: Foundations for verifying variability-intensive systems. *IEEE Transactions on Software Engineering*, 39(8):1069–1089, 2013.

[16] 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.

[17] 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.

[18] Maxime Cordy, Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay. ProVeLines: a product line of verifiers for software product lines. In *SPLC*, pages 141–146. ACM, 2013.

[19] 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.

[20] 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*, volume 11865 of *Lecture Notes in Computer Science*, pages 285–312. Springer, 2019.

[21] 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.

[22] 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.

[23] Aleksandar S. Dimovski, Axel Legay, and Andrzej Wąsowski. Variability abstraction and refinement for game-based lifted model checking of full CTL. In Reiner Hähnle and Wil M. P. van der Aalst, editors, *FASE*, volume 11424 of *Lecture Notes in Computer Science*, pages 192–209. Springer, 2019.

[24] 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 *Lecture Notes in Computer Science*, pages 406–423. Springer, 2017.

[25] Uli Fahrenberg and Axel Legay. The quantitative linear-time–branching-time spectrum. *Theoretical Computer Science*, 538:54–69, 2014.

[26] Uli Fahrenberg and Axel Legay. Featured weighted automata. In *FormaliSE@ICSE*, pages 51–57. IEEE, 2017.

[27] Uli Fahrenberg and Axel Legay. Quantitative properties of featured automata. *International Journal on Software Tools for Technology Transfer*, 21(6):667–677, 2019.

[28] Uli Fahrenberg and Axel Legay. Featured games. In *TASE*, pages 167–174. IEEE Computer Society, 2021.

[29] 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 *Lecture Notes in Computer Science*, pages 59–75. Springer, 2019.

[30] Uli Fahrenberg, Axel Legay, and Karin Quaas. Computing branching distances with quantitative games. *Theoretical Computer Science*, 847:134–146, 2020.

[31] Uli Fahrenberg, Axel Legay, and Claus Thrane. The quantitative linear-time–branching-time spectrum. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 103–114, 2011.

[32] Hugo Gimbert and Wiesław Zielonka. When can you play positionally? In Jiří Fiala, Václav Koubek, and Jan Kratochvíl, editors, *MFCS*, volume 3153 of *Lecture Notes in Computer Science*, pages 686–697. Springer, 2004.

[33] Hugo Gimbert and Wiesław Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005.

[34] 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.

[35] Marcin Jurdzinski. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.

[36] Ratnesh Kumar and Vijay K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Springer, 2012.

[37] Malte Lochau, Stephan Mennicke, Hauke Baller, and Lars Ribbeck. Incremental model checking of delta-oriented software product lines. *Journal of Logic and Algebraic Methods in Programming*, 85(1):245–267, 2016.

[38] Rafael Olaechea, 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.

[39] 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.

[40] Peter J. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[41] Genaína 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.

[42] 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 *Lecture Notes in Computer Science*, pages 387–405. Springer, 2017.

[43] 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.

[44] 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 Transactions on Software Engineering*, 46(3):321–345, 2020.

[45] 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.

[46] Maurice H. ter Beek, Sjef 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 *Lecture Notes in Computer Science*, pages 245–265. Springer, 2020.

[47] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

[48] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1&2):343–359, 1996.