Logical Methods in Computer Science
Volume 21, Issue 1, 2025, pp. 26:1–26:39
https://lmcs.episciences.org/

Submitted Jan. 26, 2024
Published Mar. 18, 2025

WITH A LITTLE HELP FROM YOUR FRIENDS: SEMI-COOPERATIVE GAMES VIA JOKER MOVES

PETRA VAN DEN BOS [a] AND MARIELLE STOELINGA [a,b]

[a] Formal Methods & Tools, University of Twente, the Netherlands
e-mail address: {p.vandenbos, m.i.a.stoelinga}@utwente.nl

[b] Department of Software Science, Radboud University, the Netherlands

Abstract. This paper coins the notion of Joker games, a variant of concurrent games where the players are not strictly adversarial. Instead, Player 1 can get help from Player 2 by playing a Joker move. We formalize these games as cost games and develop strategies that minimize the use of Jokers – viewed as costs – to secure a win with the least possible help. Our investigation studies the theoretical underpinnings of these games and their associated Joker strategies. In particular, when comparing our cost-minimal strategies with admissible strategies, we find out that they differ. Moreover, while randomization can be beneficial in conventional concurrent games, it does not aid in winning Joker games, although it can help reduce the number of needed Jokers. We also enhance our framework by introducing a secondary objective, namely by minimizing the number of moves executed by a Joker strategy. Finally, we demonstrate the practical advantages of our approach by applying it to test generation in model-based testing.

1. Introduction

Winning strategies. We study 2 player concurrent games played on a game graph with reachability objectives, i.e., the goal for Player 1 is to reach a set of goal states R. A central notion in such games is the concept of a *winning strategy*, which assigns —in states where this is possible —moves to Player 1 so that she always reaches the goal R, no matter how Player 2 plays.

Concurrent reachability games have abundant applications, e.g. controller synthesis [Ber05], assume/guarantee reasoning [CH07], interface theory [dAS03], security [ZAP+16] and test case optimization [HLM+08, DLLN08]. However, it is widely acknowledged [Ber07, Fae09, CH07] that the concept of a winning strategy is often too strong in these applications: unlike games like chess, the opponent does not try to make Player 1's life as hard as possible, but rather, the behaviour of Player 2 is unknown. Moreover, winning strategies do not prescribe any move in states where Player 1 cannot win for sure. This is a major drawback: even if Player 1 cannot win, some strategies are still better than others. Such better strategies select best-effort moves that maximize the possibility of winning in some next state, even though the game cannot be won for sure. Several concepts have been proposed to formalize

DOI:10.46298/LMCS-21(1:26)2025

best-effort strategies [NWZ20, DNT16, BBM+21, AMNS23]. This paper proposes a new approach, using *Jokers*.

Our approach. In this paper, we introduce the notion of *Joker games*, which fruitfully builds upon techniques for robust strategy synthesis from Neider et al. [NWZ20, DNT16]. We define a *Joker strategy* as a best-effort strategy in cases where Player 1 cannot win: Joker games allow Player 1, in addition to regular moves with her own actions, to play a so-called *Joker*. When Player 1 plays a Joker, she not only choose her own move, but also the opponent's move. In this way, Player 2 helps Player 1 in reaching the goal. Next, we minimize the number of Jokers needed to win the game, so that Player 1 only calls for help when needed, using an attractor construction. The obtained Joker-minimal strategies are naturally formalized as cost-minimal strategies in a priced game.

We note that in applications, where it is not possible to actually play a Joker, *Joker-inspired* strategies can be used. Such a strategy is defined the same as a Joker strategy, except that when the Joker strategy would they play a Joker action, the Joker-inspired strategy plays the Player 1 action of that Joker action. When a Player 1 action is played instead of a Joker, Player 1 can still hope that Player 2 chooses a cooperative action, e.g. the Player 2 action from the Joker action. By using a Joker-inspired strategy, Player 1 requires the minimum number of cooperative Player 2 actions for winning, thus minimizing the 'luck' required for winning. This is fruitful in cases where the opponent is not strictly adversarial. Also we note that even when Player 2 plays a non-cooperative action, our strategy will steer back to the goal from the reached state, if winning is still possible.

Our Joker games are concurrent and nondeterministic. Concurrency allows us to faithfully model and handle strategy synthesis for applications with concurrent interactions, where waiting for your turn cannot be assumed. Moreover, nondeterminism allows that outcome of the moves may lead to several next states. In this way, our games express that outcomes cannot be controlled. We do not assign probabilities to quantify nondeterminism, because especially for opponent actions, the probabilities are often unknown for real applications.

Concurrency and nondeterminism are essential in several applications. In game-based testing, where test cases are constructed from synthesized strategies, nondeterminism is needed to faithfully model the interaction between the tester and the system-under-test [vdBS18]. Our games can also model other concurrent and distributed applications, e.g. where two services or agents interact with each other via non-synchronized communication channels, or where a plant operates in its (uncontrollable!) environment.

One of the contributions of this paper is that we present the link of our Joker strategies with cost-minimal strategies. We show that cost-minimal strategies are more general than attractor strategies: all attractor strategies are cost-minimal, but not all cost-minimal strategies are attractor strategies. In particular, non-attractor strategies may outperform attractor strategies with respect to other objectives like the number of moves needed to reach the goal. In this paper, we study and define multi-objective strategies that minimize both the number of Jokers and moves.

Furthermore, we establish several new results for Joker games: (1) While concurrent games require randomized strategies to win a game, this is not true for the specific Joker moves: these can be played deterministically. (2) If we play a strategy with n Jokers, each play contains exactly n Joker moves. (3) Even with deterministic strategies, Joker games are determined. (4) The classes of Joker strategies and admissible strategies do not coincide.

As an extension to result (1) is that although randomization is not necessary to win in Joker games, randomization can be used to reduce the number of required Jokers for winning (with probability 1). Also, result (4) is extended by explaining the differences between Joker strategies and admissible strategies, with examples, and by showing that against a restricted opponent, without memory, Joker strategies and admissible strategies do coincide.

Finally, we illustrate how Joker strategies can be applied in practice, by extracting test cases from Joker-inspired strategies. Here we use techniques from previous work [vdBS18], to translate game strategies to test cases. In particular, a test case provides Player 1 actions as input to the System-Under-Test. We refer to [vdBS18] for related work on game-based approaches for testing. The experiments of this paper on four classes of case studies show that obtained test cases outperform the standard testing approach of random testing. Specifically, Joker-inspired test cases reach the goal more often than random ones, and require fewer steps to do so.

Contributions. Summarizing, the main contributions of this paper are:

(1) We formalize the minimum help Player 1 needs from Player 2 to win as cost-minimal strategies in a Joker game.
(2) We establish several properties: the minimum number of Jokers equals minimum cost, each play of a Joker strategy uses n Jokers, Joker game determinacy, Jokers can be played deterministically, in randomized setting, and admissible strategies do not coincide with cost-minimal strategies.
(3) We refine our Joker approach with second objective of the number of moves.
(4) We illustrate the benefits of our approach for test case generation.

We note that, compared to conference paper [vdBS23], we include the proofs of our theorems (in the appendix), we provide a constructive definition for our multi-objective strategies, we provide a completed comparison of Joker strategies with randomized strategies (with all required definitions) , and we solve the open conjecture of the conference paper [vdBS23] on admissible strategies.

Paper organization. Section 2 compares our work with related work. Section 3 recapitulates concurrent games. Section 4 introduces Joker games, and Section 5 investigates their properties. In section 6 we study the relation between cost-minimal and admissible strategies, and in Sect 7 we study multi-objective Joker strategies Furthermore, we study randomization in section 8 Then, in Section 9 we apply Joker strategies to test case generation. Finally, Section 10 concludes the paper. Proofs of the theorems of this paper are given in the Appendix The artifact of our experimental results of section 9 is provided in [Art].

2. Related work

Dominance. Berwanger [Ber07] coins the notion of strategy dominance, where one strategy is better than another if it wins against more opponent strategies. The maximal elements in the lattice of Player-1 strategies, called *admissible* strategies, are proposed as best strategies in losing states. We show in Section 6 that Joker strategies correspond to admissible strategies, against a restricted opponent, without memory.

Attractor strategies. Dallal et al. consider 'better' actions outside the winning set [DNT16], and resilience against disturbances of the opponent [NWZ20]. In addition, Bartocci et al. [BBM+21] perform numeric optimization on cooperative reachability games to obtain test cases. Recently, Anand et al. [AMNS23] introduced adequately permissive assumptions for strategy synthesis for ω-regular games.

While the construction for Joker strategies closely follows the (attractor) constructions given in [DNT16, BBM+21, AMNS23], our paper provides different and additional results. First, the games in [NWZ20, DNT16, BBM+21, AMNS23] are turn-based and deterministic, whereas ours are concurrent and nondeterministic.

Secondly, while [DNT16, BBM+21, AMNS23] compute optimal strategies in the form of attractor strategies, the works [NWZ20, DNT16, BBM+21, AMNS23] do not present the link with cost-minimal strategies, like we do for our Joker strategies.

Thirdly, existing works [NWZ20, AMNS23] consider more advanced winning conditions for Büchi and parity games than reachability, but they do this for a single winning condition Φ. We also study strategies for the multi-objective of minimizing both the number of Jokers and moves.

A fourth difference with existing works [DNT16, NWZ20, BBM+21, AMNS23] is that they did not focus on defining and proving foundational properties as our theorems (1) to (4), on randomized strategies, the number of Joker moves, determinacy, and admissibility, except that the authors of [AMNS23] show that their strategies are incomparable with admissible strategies.

Probabilities. A different setting where strategies are optimized is the area of stochastic games, where an opponent can take nondeterministic choices, but probabilities for the opponent's actions are known. For example, Nachmanson et al. [NVS+04] consider such games with probabilities, and additionally also costs for actions. They optimize this for obtaining tests that cover all edges.

We note that our Joker games deal differently with nondeterminism than stochastic games like Markov Decision Processes (MDP). Assume the following game, where the opponent has two moves from state q: move x leads to the winning state, e.g., $q \xrightarrow{x} \text{☺}$, while the other move y is a self-loop $q \xrightarrow{y} q$. The MDP approach assumes a probability distribution over the moves x, y. If this distribution assigns a non-zero probability to move y, then Player 1 can always win this game, and needs no Jokers. However, in Joker games, we do not assume non-zero probabilities, i.e. the opponent may choose to not play y at all at his own initiative. Also, in Joker games, we do not assume that we know probabilities for opponent actions. Thus, the MDP probability-maximization approach [NVS+04] is different from the Joker approach.

Cooperation. Different forms of cooperation are considered in related work. For example, in controller synthesis: [CH07] the synthesis problem is refined into a co-synthesis problem, where components compete conditionally: their first aim is to satisfy their own specification, and their second aim is to violate the other component's specification. In [BEK15], a hierarchy of collaboration levels is defined, formalizing how cooperative a controller for an LTL specification can be. Furthermore, Almagor et al. [AK20] define several notions of hopefulness, to perform 'good-enough' synthesis of strategies. Finally, Kupferman et al. [KS23] consider games where players can pay each other to perform a preferred action.

3. Concurrent Games

We consider concurrent games played by two players on a game graph. In each state, Player 1 and 2 *concurrently* choose an action, leading the game in one of the (nondeterministically chosen) next states.

Definition 3.1. A *concurrent game* is a tuple $G = (Q, q^0, \mathrm{Act}_1, \mathrm{Act}_2, \Gamma_1, \Gamma_2, \mathit{Moves})$ where:

- Q is a finite set of states,
- $q^0 \in Q$ is the initial state,
- For $i \in \{1, 2\}$, Act_i is a finite and non-empty set of Player i actions,
- For $i \in \{1, 2\}$, $\Gamma_i : Q \to 2^{\mathrm{Act}_i} \setminus \emptyset$ is an enabling condition, which assigns to each state q a non-empty set $\Gamma_i(q)$ of actions available to Player i in q,
- $\mathit{Moves} : Q \times \mathrm{Act}_1 \times \mathrm{Act}_2 \to 2^Q$ is a function that given the actions of Player 1 and 2 determines the set of next states $Q' \subseteq Q$ the game can be in. We require that $\mathit{Moves}(q, a, x) = \emptyset$ iff $a \notin \Gamma_1(q) \vee x \notin \Gamma_2(q)$.

For the rest of the paper, we fix concurrent game $G = (Q, q^0, \mathrm{Act}_1, \mathrm{Act}_2, \Gamma_1, \Gamma_2, \mathit{Moves})$. We will use game G in all definitions and theorems. Furthermore, we will use game $G_{a \wedge b}$ from Figure 1 as running example for this and next section.

Next, we define a *play* as an infinite sequence of states and actions, and a *play prefix* as a finite prefix of a play. An example of a play prefix in the game $G_{a \wedge b}$ is: $\pi_{\odot} = 1\langle a, x\rangle 2\langle b, x\rangle 4\langle a, x\rangle \odot$.

Definition 3.2. An *infinite play* is an infinite sequence

$$\pi = q_0\langle a_0, x_0\rangle q_1\langle a_1, x_1\rangle q_2 \ldots$$

with $a_j \in \Gamma_1(q_j)$, $x_j \in \Gamma_2(q_j)$, and $q_{j+1} \in \mathit{Moves}(q_j, a_j, x_j)$ for all $j \in \mathbb{N}$. We write $\pi^q_j = q_j$, $\pi^a_j = a_j$, and $\pi^x_j = x_j$ for the j-th state, Player 1 action, and Player 2 action respectively. The set of infinite plays with $\pi^q_0 = q$ is denoted $\Pi^\infty(q)$. We define $\Pi^\infty(G) = \Pi^\infty(q^0)$.

Definition 3.3. A *play* $\pi_{0:j} = q_0\langle a_0, x_0\rangle q_1 \ldots \langle a_{j-1}, x_{j-1}\rangle q_j$ is a (finite) play prefix of infinite play π. We write $\pi^a_{end} = a_{j-1}$ for the last Player 1 action of the play prefix.

The set of all (finite) plays of a set of infinite plays $P \subseteq \Pi^\infty(q)$ is denoted $\mathit{Pref}(P) = \{\pi_{0:j} \mid \pi \in P, j \in \mathbb{N}\}$. We define $\Pi(G) = \mathit{Pref}(\Pi^\infty(G))$, and for any $q \in Q$: $\Pi(q) = \mathit{Pref}(\Pi^\infty(q))$.

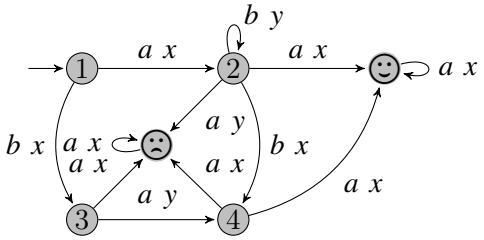

Figure 1. Concurrent game $G_{a \wedge b}$ with states $\{1, 2, 3, 4, \odot, \ominus\}$, Player 1 actions $\{a, b\}$, Player 2 actions $\{x, y\}$, and initial state 1. The *Moves* function and the enabling conditions Γ_1, and Γ_2 of $G_{a \wedge b}$ are represented by the edges. For example, the edge from state 1 to 2 indicates that $a \in \Gamma_1(1)$, $x \in \Gamma_2(1)$, and $2 \in \mathit{Moves}(1, a, x)$.

For the rest of the paper, we fix $R \subseteq Q$ as the set of goal states. Player 1's objective is to reach a state in R. In the examples of this paper, we use a state ☺ as the (single) goal state, i.e. $R = \{☺\}$.

As defined below, a play (prefix) is winning if it reaches some state in R; its winning index records the first time the play visits R. In game $G_{a \wedge b}$, the play prefix mentioned before $\pi_{☺} = 1\langle a, x\rangle 2\langle b, x\rangle 4\langle a, x\rangle ☺$ is clearly winning, and has winning index 4.

Definition 3.4. A play $\pi \in \Pi^\infty(q) \cup \Pi(q)$ for $q \in Q$ is *winning* for reachability goal $R \subseteq Q$, if there exist a $j \in \mathbb{N}$ such that $\pi^q_j \in R$. A play prefix $\pi_{0:j}$ is winning, if there exists a $k \in \mathbb{N}$ such that $k \leq j$ and $\pi^q_k \in R$. The *winning index* of a play $\pi \in \Pi^\infty(q) \cup \Pi(q)$ is the first index where R is reached:
$WinInd(\pi, R) = \min\{j \in \mathbb{N} \mid \pi^q_j \in R\}$, where $\min \emptyset = \infty$.

Players choose their actions according to a *strategy*: given the play until now, each player determines their next move. Below, we define deterministic strategies; randomized ones are deferred until Section 5.

A strategy is *positional*, if the choice for an action only depends on the last state of the play. A positional Player 1 strategy for game $G_{a \wedge b}$ is a strategy where Player 1 chooses b in state 1, and a in other states. The game outcomes for a given Player 1 strategy, are the possible plays in the game against any Player 2 strategy.

Definition 3.5. A *strategy* for Player $i \in \{1, 2\}$ starting in state $q \in Q$ is a function $\sigma_i : \Pi(q) \to \mathrm{Act}_i$, such that $\sigma_i(\pi) \in \Gamma_i(\pi^q_{end})$ for any $\pi \in \Pi(q)$. We write $\Sigma_i(q)$ for the set of all Player i strategies starting in q, and set $\Sigma_i(G) = \Sigma_i(q_0)$.

A strategy $\sigma_i \in \Sigma_i(q)$ is *positional* if for all plays $\forall \pi, \tau \in \Pi(q)$ we have $\pi^q_{end} = \tau^q_{end} \implies \sigma_i(\pi) = \sigma_i(\tau)$.

The *outcome* of a Player 1 strategy $\sigma_1 \in \Sigma_1(q)$ is the set of infinite plays that occur if Player 1 plays according to σ_1:

$$Outc(\sigma_1) = \{\pi \in \Pi^\infty(q) \mid \forall j \in \mathbb{N} : \ \sigma_1(\pi_{0:j}) = \pi^a_{j+1}\}$$

A Player 1 strategy is *winning* if all its game outcomes are winning. The *winning region* collects all states where Player 1 can win, i.e., from which she has a winning strategy. In $G_{a \wedge b}$, Player 1 has no winning strategy from any state except ☺.

Definition 3.6. Let $q \in Q$ be a game state. A Player 1 strategy $\sigma_1 \in \Sigma(q)$ is *winning*, if all plays from $Outc(\sigma_1)$ are winning. The Player 1 *winning region* $WinReg(G, R)$ for game G and goal R is the set of all states $Q' \subseteq Q$ such that for each $q \in Q'$, Player 1 has a winning strategy $\sigma_1 \in \Sigma_1(q)$.

4. Joker games

4.1. **Joker games as cost games.** Joker strategies formalize the notion of best-effort strategy by allowing Player 1 to choose, apart from her regular moves, a so-called Joker move. We associate to each concurrent game G a Joker game $G^♢$. Joker strategies are (regular) strategies in $G^♢$. With a Joker move, Player 1 chooses both a move a of her own, and a move x of Player 2, as well as the next state reached when executing these moves. In this way, Player 1 controls both Player 2 and the nondeterministic choice that determines the

next state $q' \in \mathit{Moves}(q,a,x)$. In the Joker game, a Player 1 strategy may use both Player 1 moves and Joker moves.

Since we are interested in strategies where Player 1 needs a minimum number of Jokers to win, we associate cost 1 to Joker moves, and cost 0 to other moves. Cost-minimal strategies are then the strategies that are most independent from the opponent, but defined from non-winning states. Below, $\spadesuit$ refers to parts of a Joker game that differ from the regular game G, and $\spadesuit$ to states, moves, etc. where Jokers are actually played.

Definition 4.1. We associate to each concurrent game G a Joker game $G^{\spadesuit} = (Q, q^0, \mathrm{Act}_1 \cup (\mathrm{Act}_1 \times \mathrm{Act}_2 \times Q), \mathrm{Act}_2, \Gamma_1^{\spadesuit}, \Gamma_2, \mathit{Moves}^{\spadesuit}, \mathit{Cost})$ where:

$$\Gamma_1^{\spadesuit}(q) = \{(a,x,q') \in \Gamma_1(q) \times \Gamma_2(q) \times Q \mid q' \in \mathit{Moves}(q,a,x)\}$$

$$\Gamma_1^{\spadesuit}(q) = \Gamma_1(q) \cup \Gamma_1^{\spadesuit}(q)$$

$$\mathit{Moves}^{\spadesuit}(q,a,x) = \begin{cases} \{q'\} & \text{if } a = (a',x',q') \in \Gamma_1^{\spadesuit}(q) \\ \mathit{Moves}(q,a,x) & \text{otherwise} \end{cases}$$

$$\mathit{Cost}(q,a) = \begin{cases} 1 & \text{if } a \in \Gamma_1^{\spadesuit}(q) \\ 0 & \text{otherwise} \end{cases}$$

We write $\Sigma_1^{\spadesuit}(q)$ to denote the set of strategies in $G^{\spadesuit}$ starting at state q.

As an example, consider the Joker game of $G_{a \wedge b}$. We then have that $(a,x,\text{☺}) \in \Gamma_1^{\spadesuit}(2)$, and that $\mathit{Cost}(2,(a,x,\text{☺})) = 1$.

For the rest of the paper we fix Joker game $G^{\spadesuit}$. In the remainder of the paper we often use the following two observations about Definition 4.1:

(1) The states of a concurrent game are the same as its associated Joker game.
(2) All plays of G are also plays of $G^{\spadesuit}$.

Definition 4.2 defines the cost of plays, strategies, and states. The cost of a play π arises by adding the costs of all moves until the goal R is reached. If R is never reached, the cost of π is ∞. The cost of a strategy σ_1 considers the worst case resolution of Player 2 actions and nondeterministic choices. The cost of a state q is the cost of Player 1's cost-minimal strategy from q.

For example, in Figure 1, we have $\mathit{Cost}(1\langle a,x\rangle 2\langle a,x\rangle\text{☺}) = 0$. However, against a Player 1 strategy σ_1 that chooses Player 1 action a from both state 1 and 2, Player 2 has a strategy that chooses y in state 2. The game then progresses to ☹, from where state ☺ can never be reached, so the Player 1 strategy σ_1 has cost ∞. On the other hand, if Player 1 uses the strategy $\sigma_1^{\spadesuit}$ that chooses Player 1 action a from state 1, and Joker action $(a,x,\text{☺})$ from state 2, the game progresses to state ☺ in two moves, no matter what strategy Player 2 uses. Hence, the cost of strategy $\sigma_1^{\spadesuit}$ is 1.

Definition 4.2. Let $q \in Q$ be a state, $\pi \in \Pi^{\infty}(q) \cup \Pi(q)$ a play, and $\sigma_1 \in \Sigma_1^{\spadesuit}(q)$ a strategy in Joker game $G^{\spadesuit}$. For goal states R, define their cost as follows:

$$\mathit{Cost}(\pi) = \begin{cases} \Sigma_{j=0}^{\mathit{WinInd}(\pi,R)-1} \mathit{Cost}(q_j,a_j) & \text{if } \pi \text{is winning} \\ \infty & \text{otherwise} \end{cases}$$

$$\mathit{Cost}(\sigma_1) = \sup_{\pi \in \mathit{Outc}(\sigma_1)} \mathit{Cost}(\pi)$$

$$\mathit{Cost}(q) = \inf_{\sigma_1 \in \Sigma_1^{\spadesuit}(q)} \mathit{Cost}(\sigma_1)$$

A strategy $\sigma \in \Sigma_1^{\diamondsuit}(q)$ is *cost-minimal* if $\sigma \in \operatorname{arginf}_{\sigma_1 \in \Sigma_1^{\diamondsuit}(q)} \{Cost(\sigma_1)\}$

4.2. **Winning states in Joker games.** For cost games, there is a standard fixed point computation to compute the minimum cost for reaching a goal state from state q. This computation relies on the following equations, where v_n denotes the cost computed in iteration n:

$$\begin{aligned} v_0(q) &= 0 \text{ if } q \in R \\ v_0(q) &= \infty \text{ if } q \notin R \\ v_{k+1}(q) &= \min_{a \in \Gamma_1(q)} \max_{x \in \Gamma_2(q)} \max_{q' \in Moves(q,a,x)} Cost(q,a) + v_k(q') \end{aligned}$$

For finite games, a fixed point will be reached where $v_N = v_{N+1}$, for which it holds that $v_N(q) = Cost(q)$ for all q.

Winning Joker games via attractors. Joker games allow the fixed point computation to be simplified, by exploiting their specific structure, with all costs being either 1 for Joker actions, or 0 otherwise. We adapt the classical attractor construction (see e.g., [Zie98]) on the original game G. Though we compute winning states for the Joker game, this attractor can be computed on the concurrent game.

The construction relies on two concepts: the predecessor $Pre(Q')$ contains all states with some move into Q'; the controllable predecessor $CPre_1(Q')$ contains those states where Player 1 can force the game into Q', no matter how Player 2 plays and how the nondeterminism is resolved. We note that $Pre(Q')$ can be equivalently defined as the states $q \in Q$ with $(a, x, q') \in \Gamma_1^{\spadesuit}(q)$ (for $q' \in Q'$). Hence, via $Pre(Q')$, we consider all Joker moves into Q'. For example, in game $G_{a \wedge b}$, we have $Pre(\{\text{☺}\}) = \{2, 4\}$, and $CPre_1(\{\text{☺}\}) = \emptyset$.

Definition 4.3. Let $Q' \subseteq Q$ be a set of states. The *predecessor* $Pre(Q')$ of Q', and the Player 1 *controllable predecessor* $CPre_1(Q')$ of Q' are:

$$\begin{aligned} Pre(Q') &= \{q \in Q \mid \exists a \in \Gamma_1(q), \exists x \in \Gamma_2(q), \exists q' \in Moves(q,a,x) : q' \in Q'\} \\ CPre_1(Q') &= \{q \in Q \mid \exists a \in \Gamma_1(q), \forall x \in \Gamma_2(q) : Moves(q,a,x) \subseteq Q'\} \end{aligned}$$

The classical *attractor* is the set of states from which Player 1 can force the game to reach R, winning the game. It is constructed by expanding the goal states via $CPre_1$, until a fixed point is reached [dAHK98]; the rank k indicates in which computation step a state was added [Zie98]. Thus, the lower k, the fewer moves Player 1 needs to reach its goal. As example, observe that $Attr(G_{a \wedge b}, \{\text{☺}\}) = \{\text{☺}\}$.

Definition 4.4. The Player 1 *attractor* is $Attr(G, R)$, where:

$$\begin{aligned} Attr^0(G,R) &= R \\ Attr^{k+1}(G,R) &= Attr^k(G,R) \ \cup CPre_1(Attr^k(G,R)) \\ Attr(G,R) &= \bigcup_{k \in \mathbb{N}} Attr^k(G,R) \end{aligned}$$

The function $ARank : Q \to \mathbb{N}$ associates to each state $q \in Q$ a rank $ARank(q) = \min\{k \in \mathbb{N} \mid q \in Attr^k(G,R)\}$. Recall that $\min \emptyset = \infty$.

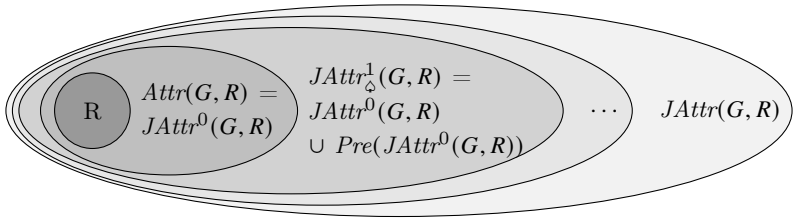

FIGURE 2. Illustration of the Joker attractor computation: it starts with states in R initially and then adds states using the attractor and Joker attractor operations, until a fixpoint is reached.

We define the *Joker attractor* to interleave the standard attractor and the predecessor operator. Figure 2 illustrates the computation. Since *Attr* and *Pre* only use elements from G, the computation is performed in G. We will see that for defining the Joker attractor strategy (Definition 4.11) we do need Joker game $G^\diamondsuit$. With $JRank(q)$ we denote after how many *Pre* operations q was first added; we will see that this is the number of used Jokers.

Definition 4.5. The Player 1 Joker attractor is $JAttr(G,R)$, where:

$$\begin{aligned} JAttr^0(G,R) &= Attr(G,R) \\ JAttr^{k+1}_\diamondsuit(G,R) &= JAttr^k(G,R) \cup Pre(JAttr^k(G,R)) \\ JAttr^{k+1}(G,R) &= Attr(G, JAttr^{k+1}_\diamondsuit(G,R)) \\ JAttr(G,R) &= \bigcup_{k\in\mathbb{N}} JAttr^k(G,R) \end{aligned}$$

We call $JAttr(G,R)$ the Joker attractor of G. The Joker states are $JAttr_\spadesuit(G,R) = \bigcup_{k\in\mathbb{N}} JAttr^{k+1}_\diamondsuit(G,R) \setminus JAttr^k(G,R)$. To each Joker attractor $JAttr(G,R)$ we associate a Joker rank function $JRank : Q \to \mathbb{N}$, where for each state $q \in Q$ we define $JRank(q) = \min\{k \in \mathbb{N} \mid q \in JAttr^k(G,R)\}$.

Example 4.6. As an example, we show the computation of the Joker attractor for game $G_{a\wedge b}$ with goal ☺ below. We also indicate for each state what its $JRank$ is, and whether it is a Joker state or not.

$$\begin{aligned} JAttr^0(G_{a\wedge b}, ☺) &= \{☺\} \\ JAttr^1_\diamondsuit(G_{a\wedge b}, ☺) &= \{2, 4, ☺\} \\ JAttr^1(G_{a\wedge b}, ☺) &= \{1, 2, 4, ☺\} \\ JAttr^2_\diamondsuit(G_{a\wedge b}, ☺) &= \{1, 2, 3, 4, ☺\} \\ JAttr^2(G_{a\wedge b}, ☺) &= \{1, 2, 3, 4, ☺\} \\ JAttr^3(G_{a\wedge b}, ☺) &= \{1, 2, 3, 4, ☺\} \end{aligned}$$

$q \in Q$	$JRank$	Joker state
1	1	no
2	1	yes
3	2	yes
4	1	yes
☺	0	no
☹	∞	no

Theorem 4.7 states that the cost of state q in $G^\diamondsuit$, i.e. the minimum number of Jokers needed from q to reach a goal state, is equal to $JRank(q)$. The proof follows from the following observations:

- Clearly, states that can be won by Player 1 in G have cost 0, so these fall into $JAttr^0(G,R)$.

- Similarly, if all states in Q' can be won with at most k jokers, then so can states in $Attr(Q')$.
- The *Joker states* of set $JAttr^{k+1}(G, R)$, are the states where Player 1 can only win from any opponent if she uses a Joker. By playing a Joker, the game moves to a state in $JAttr^k(G, R)$. Joker states are the predecessors of $JAttr^k(G, R)$.

Together, these observations establish that in all states of $JAttr^k(G, R)$, the game can be won with at most k Jokers, i.e., with cost k.

Theorem 4.7. *For all $q \in Q$, we have $JRank(q) = Cost(q)$.*

See proof in appendix.

4.3. **Winning strategies in Joker games.** The attractor construction cannot only be used to compute the states of the Joker attractor, but also to construct the corresponding winning strategy. As is common in strategy construction, we do so by adding some extra administration to the (controllable) predecessor operations *Pre* and *CPre*. The witnessed predecessor $wPre(Q')$ returns the states *and* their Joker moves into Q'. Similarly, the witnessed controllable predecessor $wCPre(Q')$ returns the states, *and* their Player 1 actions that force the game into Q'.

We have seen that in game $G_{a \wedge b}$, $Pre(\{☺\}) = \{2, 4\}$. With $wPre$ we compute the corresponding Joker moves of these states: $wPre(\{☺\}) = \{(2, (a, x, ☺)), (4, (a, x, ☺))\}$. And instead of $CPre_1(\{2\}) = \{1\}$ we compute the controlled predecessor with Player 1 actions: $wCPre_1(\{2\}) = \{(1, a)\}$.

Definition 4.8. Let $Q' \subseteq Q$ be a set of states. The *witnessed predecessor* $wPre(Q')$ of Q', and the Player 1 *witnessed controllable predecessor* $wCPre_1(Q')$ of Q' are:

$$wPre(Q') = \{(q, (a, x, q')) \in Q \times (\mathrm{Act}_1 \times \mathrm{Act}_2 \times Q) \mid q' \in Moves(q, a, x) \cap Q'\}$$
$$wCPre_1(Q') = \{(q, a) \in Q \times \mathrm{Act}_1 \mid a \in \Gamma_1(q) \wedge (\forall x \in \Gamma_2(q) : Moves(q, a, x) \subseteq Q')\}$$

Now, we obtain the sets of winning actions, by replacing, in the construction of $JAttr(G, R)$, *Pre* and *CPre* by $wPre$ and $wCPre_1$, respectively. We only select actions for moving from newly added states of the $k+1$-th attractor, or the $k+1$-th Joker attractor set, respectivly, to states of the m-th attractor, or m-th Joker attractor set, respectively, such that $m \neq k$. This way, always a state of a previous attractor or previous Joker set is reached, ensuring progress towards reaching a goal state.

Example 4.9. For game $G_{a \wedge b}$ we compute the following witnessed Joker attractor sets:

$$wJAttr^0(G_{a \wedge b}, \{☺\}) = \emptyset$$
$$wJAttr^1_{♤}(G_{a \wedge b}, \{☺\}) = \{(2, (a, x, ☺)), (4, (a, x, ☺))\}$$
$$wJAttr^1(G_{a \wedge b}, \{☺\}) = \{(1, a), (2, (a, x, ☺)), (4, (a, x, ☺))\}$$
$$wJAttr^2_{♤}(G_{a \wedge b}, \{☺\}) = \{(1, a), (2, (a, x, ☺)), (3, (a, x, 4)), (4, (a, x, ☺))\}$$

Definition 4.10. We define the *witnessed* attractor $wAttr(G,R)$ and *witnessed* Joker attractor $wJAttr(G,R)$:

$$
\begin{aligned}
wAttr^0(G,R) &= \emptyset \\
wAttr^{k+1}(G,R) &= wAttr^k(G,R) \cup \{(q,a) \in wCPre_1(Attr^k(G,R)) \mid q \notin Attr^k(G,R)\} \\
wAttr(G,R) &= \bigcup_{k\in\mathbb{N}} wAttr^k(G,R) \\
wJAttr^0(G,R) &= \emptyset \\
wJAttr^{k+1}_{\diamondsuit}(G,R) &= wJAttr^k(G,R) \cup \{(q,(a,x,q')) \in wPre(JAttr^k(G,R)) \mid q \notin JAttr^k(G,R)\} \\
wJAttr^{k+1}(G,R) &= wAttr(G, wJAttr^{k+1}_{\diamondsuit}(G,R)) \\
wJAttr(G,R) &= \bigcup_{k\in\mathbb{N}} wJAttr^k(G,R)
\end{aligned}
$$

A *Joker attractor strategy* in $G^{\diamondsuit}$ is a strategy that plays according to the witnesses: in Joker states, a Joker action from $wJAttr(G,R)$ is played. In non-Joker states q, the strategy takes its action from an attractor witness $wAttr(G,R)$ that is also included in $wJAttr(G,R)$. The Joker attractor strategy is not defined in states not included in $JAttr(G,R)$, since no goal state can be reached anymore (as we will see in Theorem 5.2). It is also undefined in goal states R, where the goal has already been reached. Figure 3 shows the Joker strategy of game $G_{a\wedge b}$.

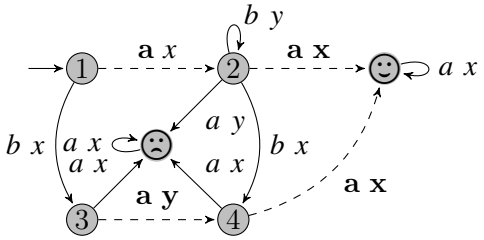

FIGURE 3. Game $G_{a\wedge b}$, as in Figure 1, but with dashed edges for moves of the Joker attractor strategy. On these dashed edges, if both the Player 1 and the Player 2 action are bold, then a Joker action is played. Otherwise, if only the Player 1 action is bold, then a (normal) Player 1 action is played. Note that non-dashed moves will never be played with this Joker strategy, no matter what Player 2 strategy or resolution of nondeterminism is used. Also, the dashed move from state 3 will not be played, because the strategy does not go there from the initial state.

Definition 4.11. A strategy $\sigma_1 \in \Sigma_1^{\diamondsuit}(q)$ in $G^{\diamondsuit}$ is a *Joker attractor strategy*, if for any $\pi \in \Pi(G^{\diamondsuit})$ we have $\pi^q_{end} \in JAttr(G,R) \setminus R \implies (\pi^q_{end}, \sigma_1(\pi)) \in wJAttr(G,R)$.

5. Properties of Joker games

We establish four fundamental properties of Joker games.

1. All outcomes use same the number of Jokers. A first, and perhaps surprising result (Theorem 5.1) is that all outcomes of a Joker attractor strategy use exactly the same number of Joker actions: all plays starting in state q contain exactly $JRank(q)$ Jokers. This property follows by induction from the union-like computation illustrated in Figure 2. Cost-minimal strategies in general cost games do not have this property, because some outcomes may use lower costs. In particular, cost-minimal strategies that are not obtained via the attactor construction may also lead to plays which differ in the number of Jokers they use. This is illustrated in Figure 4 and shows that attractor strategies are special among the cost-minimal strategies.

Theorem 5.1. *Let $q \in JAttr(G, R)$. Then:*

(1) *Let $\sigma_1^J \in \Sigma_1^{\diamondsuit}(q)$ be a Joker attractor strategy in $G^{\diamondsuit}$. Then any play $\pi \in Outc(\sigma_1^J)$ has* exactly $JRank(q)$ *Joker actions in winning prefix $\pi_{0:WinInd(\pi)}$.*

(2) *Let $\sigma_1 \in \Sigma_1^{\diamondsuit}(q)$ be a cost-minimal strategy in $G^{\diamondsuit}$. Then any play $\pi \in Outc(\sigma_1)$ has* at most $JRank(q)$ *Joker actions in the winning prefix $\pi_{0:WinInd(\pi)}$.*

See proof in appendix.

The intuition of the proof of Theorem 5.1 is illustrated in Figure 4: a Joker is always played in a Joker state. By construction of the Joker attractor strategy, the Joker action moves the game from a state q to a state q', such that the $JRank$ of q is one higher than the $JRank$ of q'. In non-Joker states no Joker is used to reach a next Joker state.

2. Characterization of winning states of Joker games. We establish that: (1) for every goal R, the winning region in Joker game $G^{\diamondsuit}$ (i.e., states where Player 1 can win with any strategy, not necessarily cost-minimal) coincides with the Joker attractor $JAttr(G, R)$, and (2) the set of all states having a play reaching a state from R coincide with Joker attractor $JAttr(G, R)$. The intuition for this is that, by playing a Joker in any state, Player 1 can force the game to any state that is reachable. Consequently, states from where a goal state can be reached, are winning, and in the Joker attractor.

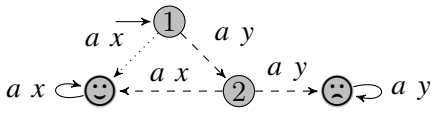

Figure 4. A cost-minimal strategy is depicted by the dashed edges. It plays a cost-0 a action in state 1, and hopes Player 2 plays x to arrive in ☺ with cost 0. If Player 2 plays y, the strategy plays a Joker in state 2 to win nevertheless. The computation of the Joker attractor yields that state 1 and 2 both have $JRank$ 1. Since initial state 1 is always reached first by the predecessor, the Joker attractor strategy (dotted) plays a Joker in state 1 immediately, and reaches ☺ directly from state 1. This example shows that cost-minimal strategies from state 1 may use fewer than $JRank(1)$ Jokers (against some but not all opponents), and that Joker attractor strategies from state 1 always use $JRank(1)$ Jokers.

The states from where Player 1 cannot ever win the game, because no goal state is reachable, is thus the set $JAttr(G, R) \setminus R$. In the examples of this paper we represent these states with the single state ☹.

Theorem 5.2. *Let* $Reach(G, R) = \{q \in Q \mid q$ *can reach a state* $q' \in R\}$. *Then*

$$WinReg(G^{\diamondsuit}, R) = JAttr(G, R) = Reach(G, R)$$

See proof in appendix.

3. Joker attractor strategies and minimality. Theorem 5.3 states correctness of Joker attractor strategies: they are indeed cost-minimal. The converse, cost-minimal strategies are Joker attractor strategies, is not true, as shown by Figure 4 and 5. The game of Figure 5 also shows that Joker attractor strategies need not take the shortest route to the goal (see also section 7). Note that a consequence of Theorem 5.3 is that Joker strategies are indeed winning in Joker games.

Theorem 5.3. *Any Joker attractor strategy is cost-minimal (a), but not every cost-minimal strategy is a Joker attractor strategy (b).*

See proof in appendix.

4. Determinacy. Unlike concurrent games, Joker games are *determined*. That is, in each state of the Joker game, either Player 1 has a winning strategy or Player 2 can keep the game outside R forever. Determinacy (Theorem 5.4) follows from Theorem 5.2: by $WinReg(G^{\diamondsuit}, R) = JAttr(G, R)$ we have a winning strategy in any state from $JAttr(G, R)$, and by $JAttr(G, R) = Reach(G, R)$ we have that states not in $JAttr(G, R)$ have no play to reach R, so Player 2 wins with any strategy.

Theorem 5.4. *Joker games are determined.*

See proof in appendix.

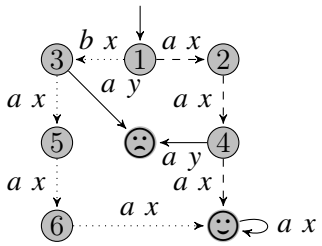

FIGURE 5. This game G_{dist} has a cost-minimal strategy (dashed; cost 1 for the Joker used in state 4) that is not a Joker attractor strategy. For this game, there is a unique Joker attractor strategy (dotted). It selects Player 1 action b from state 1 (to Joker state 3), since states 1 and 2 are added at the same iteration of the attractor over the set with Joker states 3 and 4. The witnessed attractor in iteration k only selects Player 1 actions to states from an iteration $\neq k$. The dashed cost-minimal strategy requires fewer moves to reach ☺ than the dotted Joker strategy, namely 3 versus 4 moves.

6. Admissible strategies

Various papers [BPRS16, BCH+16, Fae09] advocate that a player should play *admissible* strategies if there is no winning strategy. Admissible strategies (Definition 6.2) are the maximal elements in the lattice of all strategies equipped with the dominance order $<_d$. Here, a Player 1 strategy σ_1 dominates strategy σ_1', denoted $\sigma_1' <_d \sigma_1$, iff whenever σ_1' wins from opponent strategy ρ, so does σ_1. We hence then define dominant and admissible strategies, in concurrent games, using the set of opponent strategies a Player 1 strategy can win from (Definition 6.2). Since the nondeterministic choice for the next state of the game affects whether Player 1 wins, we introduce Player 3 (Definition 6.1) for making this choice. Specifically, given a play prefix π, and moves a and x chosen by Player 1 and 2, respectively, a Player 3 strategy chooses one of the next states in $Moves(\pi^q_{end}, a, x)$. The set of winning opponent strategies is then defined as the pairs of Player 2 *and* Player 3 strategies.

Definition 6.1. A Player 3 strategy is a function $\sigma_3 : \Pi(q) \times \mathrm{Act}_1 \times \mathrm{Act}_2 \to Q$, such that $\sigma_3(\pi, a, x) \in Moves(\pi^q_{end}, a, x)$ for all $\pi \in \Pi(q), a \in \Gamma_1(\pi^q_{end}), x \in \Gamma_2(\pi^q_{end})$. We write $\Sigma_3(q)$ for the set of all Player 3 strategies from q.

The *outcome* $Outc(\sigma_1, \sigma_2, \sigma_3)$ of strategies $\sigma_1 \in \Sigma_1(q)$, $\sigma_2 \in \Sigma_2(q)$, and $\sigma_3 \in \Sigma_3(q)$ is the single play $\pi \in \Pi^\infty(q)$ such that:

$$\forall j \in \mathbb{N} : \ \sigma_1(\pi_{0:j}) = \pi^a_j \wedge \sigma_2(\pi_{0:j}) = \pi^x_j \wedge \sigma_3(\pi_{0:j}, \sigma_1(\pi_{0:j}), \sigma_2(\pi_{0:j})) = \pi^q_{j+1}$$

Definition 6.2. The Player 2 and 3 strategy pairs that are winning in G for a strategy $\sigma_1 \in \Sigma_1(G)$ are defined as:

$$Win\Sigma_{2,3}(\sigma_1, R) = \{(\sigma_2, \sigma_3) \in \Sigma_2(G) \times \Sigma_3(G) \mid Outc(\sigma_1, \sigma_2, \sigma_3) \in Win\Pi(G, R)\}$$

For any $q \in Q$, a strategy $\sigma_1 \in \Sigma_1(q)$ is *dominated by* a strategy $\sigma_1' \in \Sigma_1(q)$, denoted $\sigma_1 <_d \sigma_1'$, if $Win\Sigma_{2,3}(\sigma_1, R) \subset Win\Sigma_{2,3}(\sigma_1', R)$. Strategy $\sigma_1 \in \Sigma_1(q)$ is *admissible* if there is no strategy $\sigma_1' \in \Sigma_1(q)$ with $\sigma_1 <_d \sigma_1'$.

To compare cost-minimal strategies, including Joker attractor strategies, with admissible strategies, we note that cost-minimal strategies are played in Joker games, where Joker actions have full control over the opponent. Admissible strategies, however, are played in regular concurrent games, without Joker actions. To make the comparison, Definition 6.3 therefore associates to any Player 1 strategy σ in $G^\diamondsuit$, a Joker-inspired strategy σ_{insp} in G: if σ chooses Joker action (a, x, q), then σ_{insp} plays Player 1 action a.

Definition 6.3. Let $\sigma \in \Sigma_1(q)$ be a Player 1 strategy in Joker game $G^\diamondsuit$. Define the *Joker-inspired strategy* $\sigma_{insp} \in \Sigma_1(q)$ of σ in concurrent game G for any $\pi \in \Pi(G)$ as:

$$\sigma_{insp}(\pi) = \begin{cases} a & \text{if } \sigma(\pi) = (a, x, q) \\ \sigma(\pi) & \text{otherwise} \end{cases}$$

In the next two subsections, we study whether admissible strategies are always Joker-inspired cost-minimal stratgies, and whether Joker-inspired cost-minimal strategies are always admissible.

6.1. Admissible strategies not necessarily cost-minimal.

The game of Figure 6 shows that admissible strategies need not be a Joker-inspired strategy of a cost-minimal strategy, as stated by Theorem 6.4. While cost-minimal strategies ensure that the minimum number of Jokers is used in a Joker game, admissible strategies are not defined to satisfy this condition.

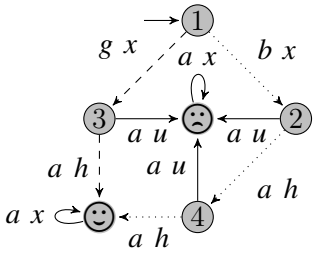

FIGURE 6. The dotted strategy chooses b in state 1, and after that plays a in the two Joker states 2 and 4. In the Joker game, strategies that pass those two states need to play a Joker action to be winning, since Player 2 can prevent Player 1 from ever reaching ☺ by playing a u in state 2 or 4. Hence, the dotted strategy spends two Jokers in the Joker game. The dotted strategy is admissible in the concurrent game though, since it wins from the Player 2 strategy that chooses u in state 3 and h or x anywhere else, while the dashed Joker-inspired strategy looses from this Player 2 strategy.

The dotted strategy of Figure 6 uses that it already dominates another strategy if there is one opponent strategy in a non-visited state the dotted strategy trivially wins from, while the other looses. Hence, domination does not prevent using too many Jokers in the Joker game.

Theorem 6.4. *An admissble strategy is not always a Joker-inspired strategy of a cost-minimal strategy.*

See proof in appendix.

6.2. **Admissibility of Joker-inspired, cost-minimal strategies.** It turns out that Joker-inspired strategies of cost-minimal strategies need not be admissible, for a rather trivial reason: a cost-minimal strategy that chooses a losing move in a non-visited state is not admissible. See Figure 7. Because actions in non-visited states do not influence the outcome of a game, we want to exclude this case. Therefore, we will only consider the admissibility of *global* cost-minimal strategies (Definition 6.5): strategies that are cost-minimal for any initial state of the Joker game. We note here that Joker attractor strategies are global-cost-minimal by construction.

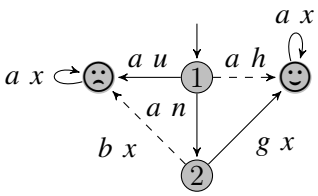

FIGURE 7. The dashed strategy uses bad action b in state 2 instead of good action g, because using Joker (a,h,☺) in state 1 will make Player 1 win the game. Hence, the Joker-inspired strategy of this winning cost-minimal strategy is not admissible, as it is dominated by strategies choosing g in state 2.

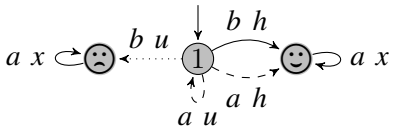

FIGURE 8. The dashed Player 1 strategy, that choosing a in state 1, dominates the dotted Player 1 strategy, that choosing b in state 1, because only the first strategy wins from a Player 2 strategy that plays u on the first visit of state 1, but h for all later visits. Both Player 1 strategies are Joker-inspired.

Definition 6.5. Let $\sigma \in \Sigma_1^{\diamondsuit}(q)$ be a strategy in Joker game $G^{\diamondsuit}$. Then σ is *global cost-minimal* if σ is cost-minimal from any $q' \in \mathit{Reach}(G, R)$.

Even when only considering *global* cost-minimal strategies, it turns out some of these strategies may not be admissible when playing against opponent strategies using memory (Theorem 6.6(1)). Figure 8 shows an example where a Player 2 strategy uses memory. It has two Player 1 strategies that both need to pass by a Joker state. In this state Player 2 can force the game to the ☹ state against one of the Player 1 strategies, such that Player 1 can never win any more. Against the other Player 1 strategy, Player 2 only prevent progress towards ☺, for one move only, via a loop to the same state.

Without memory for Player 2 and 3, Joker-inspired strategies of global cost-minimal are truly admissible (Theorem 6.6(2)). The inituition is as follows. From each Joker state, there is a winning and a loosing Player 2 and 3 strategy pair for a given Player 1 action. Here we note that global cost-minimal strategies may only suffer from such loosing strategies in Joker states, as they are in control in the other states. Without memory, Player 2 and 3 fix the choice for every Joker state and Player 1 action of a strategy. In particular, a cyclic move as in Figure 8 is taken forever. This leads to the following two conclusions:

- Some Joker-inspired global cost-minimal Player 1 strategies have the same winning pairs of Player 2 and 3 strategies, i.e., in particular all strategies that move through the same Joker states.
- For the other Joker-inspired global cost-minimal Player 1 strategies, we have for each two Player 1 strategies σ and σ', a pair of Player 2 and 3 strategies (σ_2, σ_3), such that σ wins from σ_2 and σ_3, and σ' not, *and vice versa*. In particular, this property holds for strategies that move through different Joker states, and for some strategies (depending on the game topology), that use different actions in the same Joker states (because Player 3 can inspect the Player 1 action played in a state).

In either case, the Joker-inspired global cost-minimal Player 1 strategies do not dominate each other (and strategies that are not cost-minimal will not perform better). Consequently, Joker-inspired cost-minimal strategies are admissible against positional playing Player 2 and 3.

Theorem 6.6.

(1) *A Joker-inspired strategy of a global cost-minimal strategy is not always admissible.*
(2) *A Joker-inspired strategy of a global cost-minimal strategy is admissible for positional Player 2 and 3 strategies.*

See proof in appendix.

7. Joker strategies using fewer moves

Although multiple Joker attractor strategies can be constructed via Definition 4.11, their number of moves may not be minimal in the Joker game. The cause for this is that Joker attractor strategies prefer reaching a Joker state in few moves and using more moves afterwards, over reaching a Joker state in more moves, but using less moves afterwards. Figure 5 shows that this is not always beneficial for the total number of moves taken towards the goal: Joker attractor strategies may need more moves in total than other cost-minimal strategies.

To address this, we define the *Joker distance attractor* in Definition 7.2. It optimizes for the minimum number of Jokers, as first objective, and prefers fewer moves, as a second objective. The derived strategies (Definition 7.6) will use the minimum number of Jokers, like ordinary Joker strategies, but aim to use fewer moves from initial state to goal state.

The Joker distance attractor re-uses the structure of the ordinary Joker attractor sets in its computation. To ensure usage of the minimum number of Jokers, only moves within a Joker attractor set, and moves from a Joker state to the next Joker attractor set are considered. Within the subgraph of this restricted set of moves, the total distance from initial state to goal state is minimized. Specifically, the computation proceeds as follows. Goal states R are the states $dJAttr^0(G,R)$ with distance 0. Then this set is expanded iteratively via the Joker distance predecessor $dPre$ to $dJAttr^k(G,R)$. We use this predecessor to select a state q that has a move to a state $q' \in dJAttr^{k-1}(G,R)$. Only states q that satisfy either of the following conditions (1) or (2) are selected:

(1) State q is a Joker state, and:
   - There is a Joker move from q to q', and
   - State q is a state of the next Joker attractor set, i.e. $JRank(q) = JRank(q') + 1$

(2) State q is a non-Joker state of the Joker attractor, and
   - There is a move from q to some state $q' \in dJAttr^{k-1}(G,R)$, against any opponent, i.e., a controllable Player 1 action makes the game move from q to q'.
   - State q is state of the same Joker attractor set as q', i.e., $JRank(q) = JRank(q')$

Definition 7.1. Let $Q' \subseteq Q$ be a set of states. The *Joker distance predecessor* $dPre(Q')$ of Q' is:

$$\begin{aligned} dPre(Q') = \{q \in JAttr_{\spadesuit}(G,R) \mid \exists a \in \Gamma_1(q), \exists x \in \Gamma_2(q), \exists q' \in Moves(q,a,x) : \\ q' \in Q' \wedge JRank(q) = JRank(q') + 1\} \\ \cup \{q \in JAttr(G,R) \setminus JAttr_{\spadesuit}(G,R) \mid \exists a \in \Gamma_1(q), \forall x \in \Gamma_2(q) : \\ Moves(q,a,x) \subseteq Q' \wedge JRank(q) = JRank(q')\} \end{aligned}$$

Definition 7.2. The Player 1 *Joker distance attractor* is $dJAttr(G,R)$, where:

$$\begin{aligned} dJAttr^0(G,R) &= R \\ dJAttr^{k+1}(G,R) &= dJAttr^k(G,R) \cup dPre(dJAttr^k(G,R)) \\ dJAttr(G,R) &= \bigcup_{k \in \mathbb{N}} dJAttr^k(G,R) \end{aligned}$$

The function $dJRank : Q \to \mathbb{N}$ associates to each state $q \in Q$ its *Joker distance* $dJRank(q) = \min\{k \in \mathbb{N} \mid q \in dJAttr^k(G,R)\}$.

Example 7.3. We compute the Joker distances for states of Joker game of Figure 5. Its Joker attractor computation is as follows:

$$JAttr^0(G_{dist}, ☺) = \{5, 6, ☺\}$$
$$JAttr^1_{♤}(G_{dist}, ☺) = \{3, 4, 5, 6, ☺\}$$
$$JAttr^1(G_{dist}, ☺) = \{1, 2, 3, 4, 5, 6, ☺\}$$

The Joker distance attractor is then computed as follows:

$$dJAttr^0(G_{dist}, ☺) = \{☺\}$$
$$dJAttr^1(G_{dist}, ☺) = \{4, 6, ☺\}$$
$$dJAttr^2(G_{dist}, ☺) = \{2, 4, 5, 6, ☺\}$$
$$dJAttr^3(G_{dist}, ☺) = \{1, 2, 3, 4, 5, 6, ☺\}$$

We remark the following about the definition of the Joker distance attractor. Firstly, we note that the Joker distance attractor has the same structure as the standard attractor (Definition 4.4), except that it uses the special Joker distance predecessor instead of the controlled predecessor of Definition 4.3. Secondly, because the Joker distance predecessor requires that the Joker rank decreases by 1, every time a move through a Joker state is selected, the Joker distance attractor ensures that the minimum number of Jokers is used. Thirdly, the distance is computed from the number of moves taken, by not preferring moves through non-Joker states over moves through Joker states. Finally, we note that the computed distance may not be the smallest distance possible. A game may have plays with less moves, that require the use of more Jokers than the minimum. See Figure 9 for an example. A summary of previous remarks is that the Joker distance attractor is a fixpoint computation (like a standard attractor), that minimizes on (J*Rank*, dJ*Rank*), where minimal *JRank* is preferred over minimal *dJRank*.

Similar to the witnessed attractors of Definition 4.10, we define a witnessed Joker distance attractor (Definition 7.5), to obtain a corresponding distance-minimizing Joker strategy (Definition 7.6). The witnessed Joker distance attractor collects Joker actions (in Joker states) and Player 1 actions (in non-Joker states) that will be composed into a strategy that uses a minimum Jokers and minimum number of moves.

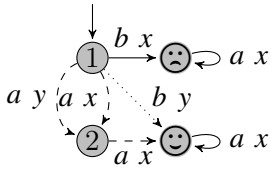

Figure 9. The dashed Joker distance strategy uses 0 Jokers and arrives at ☺ in 2 moves against any opponent strategy. The dotted strategy arrives at ☺ in 1 move, if it plays a Joker in state 1 to prevent Player 2 from playing x and reaching ☹. State 1 is not a Joker state, and Player 1 action b in state 1 is not controllable, so the Joker distance attractor will not add state 1 via this b action with distance 1. Instead it will add state 1 via the a action with distance 2.

Example 7.4. As an example, consider game G_{dist} of Figure 5 again. We compute the following witnessed Joker distance attractor:

$$wdJAttr^0(G_{dist}, ☺) = \emptyset$$
$$wdJAttr^1(G_{dist}, ☺) = \{(4, (a, x, ☺)), (6, a)\}$$
$$wdJAttr^2(G_{dist}, ☺) = \{(2, a), (4, (a, x, ☺)), (5, a), (6, a)\}$$
$$wdJAttr^3(G_{dist}, ☺) = \{(1, a), (2, a), (3, (a, x, 5)), (4, (a, x, ☺)), (5, a), (6, a)\}$$

In particular, we note that the final set has the move $(1, a)$ to state 2, and not $(1, b)$ to state 3, because $2 \in dJAttr^2(G_{dist}, ☺)$ and $3 \notin dJAttr^2(G_{dist}, ☺)$. Hence, the corresponding Joker distance strategy chooses action a in state 1, such that ☺ is reached in 3 moves.

Definition 7.5. The *witnessed Joker distance predecessor* $wdPre(Q')$ of Q' is:

$$\begin{aligned} wdPre(Q') = \{(q, (a, x, q')) \in JAttr_{♠}(G, R) \times (\mathrm{Act}_1 \times \mathrm{Act}_2 \times Q) \mid \\ q' \in Moves(q, a, x) \cap Q' \wedge JRank(q) = JRank(q') + 1\} \\ \cup \{(q, a) \in JAttr(G, R) \setminus JAttr_{♠}(G, R) \times \mathrm{Act}_1 \mid \forall x \in \Gamma_2(q) : \\ Moves(q, a, x) \subseteq Q' \wedge JRank(q) = JRank(q')\} \end{aligned}$$

The Player 1 *witnessed Joker distance attractor* is $wdJAttr(G, R)$, where:

$$\begin{aligned} wdJAttr^0(G, R) &= \emptyset \\ wdJAttr^{k+1}(G, R) &= wdJAttr^k(G, R) \\ &\quad \cup \{(q, a) \in wdPre(dJAttr^k(G, R)) \mid q \notin dJAttr^k(G, R)\} \\ wdJAttr(G, R) &= \bigcup_{k \in \mathbb{N}} wdJAttr^k(G, R) \end{aligned}$$

Definition 7.6. A strategy $\sigma_1 \in \Sigma_1^{♢}(q)$ in $G^{♢}$ is a *Joker distance strategy*, if for any $\pi \in \Pi(G^{♢})$ we have:

$$\pi^q_{end} \in dJAttr(G, R) \setminus R \implies (\pi^q_{end}, \sigma_1(\pi)) \in wdJAttr(G, R)$$

Theorem 7.7 states that Joker distance strategies indeed are cost-minimal. Additionally, Joker distance strategies will use a lower or equal number of moves than any Joker attractor strategy. The intuition is that a Joker distance strategy considers more cost-optimal plays than Joker attractor strategies, and then select the plays with the minimum number of moves. Against some opponents, however, cost-minimal strategies, that are not Joker attractor strategies, may use less moves than Joker distance strategies. An example is shown in Figure 10. Here, the cost-minimal strategy takes a Player 1 move from a Joker state to another Joker state with the same Joker rank. Joker distance strategies only use Joker moves to Joker states, that decrease the Joker rank (see Definition 7.1), to ensure cost-minimality. It is future work to find a construction for *all* cost-minimal strategies of Joker games, and then optimize this to find strategies with a minimal number of moves.

Theorem 7.7. *Let σ be a Joker distance strategy, and σ' a Joker attractor strategy, both in Joker game $G^{♢}$. Then σ is cost-minimal (a), and for any play $\pi \in Outc(\sigma)$ and $\pi' \in Outc(\sigma')$ we have $Cost(\pi) \leq Cost(\pi')$ (b).*

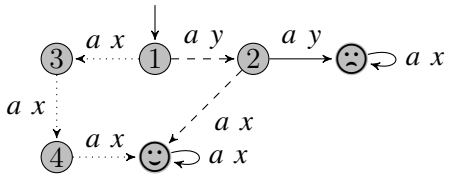

FIGURE 10. A cost-minimal strategy with the minimum number of moves is depicted by the dashed edges. It plays a cost-0 a action in state 1, and hopes Player 2 plays y to arrive in state 2 with cost 0. Then it plays a Joker in state 2 to arrive in state ☺ in only 2 moves. If Player 2 plays x in state 1, then the game state ☺ will be reached with 0 Jokers, in 3 moves. Because state 1 and 2 have the same rank, a Joker distance strategy will use a Joker action from state 1 to state 3, and 3 moves in total to reach the goal.

See proof in appendix.

8. Randomized Joker strategies

A distinguishing feature of concurrent games is that, unlike in turn-based games, randomized strategies may help to win a game. A classical example is the Penny Matching Game of Figure 11.

We show that randomization is not needed for winning Joker games, but does help to reduce the number of Joker moves. However, randomization in Joker states is never needed.

In subsection 8.1, we first set up the required machinery leading to the definition of the probabilistic attractor. We use this in subsection 8.2 to define the Joker attractor for randomized strategies: it simply substitutes the standard attractor by the probabilistic attractor. Readers only interested in (novel) results may skip to subsection 8.3: here we revisit the theorems of section 5, but then for randomized Joker strategies. Finally, in subsection 8.4 we discuss the (non)benefits of adding randomization to Joker games.

8.1. **Probabilistic attractor.**

In this section we follow definitions of [dAHK98], but we adapt them to our concurrent games where *Moves* returns a set of states.

In Definition 8.2, a randomized strategy for Player 1 or 2 assigns probabilities to a play π and next action $a \in \mathrm{Act}_1$ or $x \in \mathrm{Act}_2$, respectively. A Player 1 strategy σ_p for the game

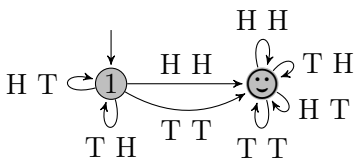

FIGURE 11. The penny matching game $G_{\in}$: each Player chooses a side of a coin. If they both choose heads (H), or both tails (T), then Player 1 wins, otherwise they play again. Player 1 wins this game with probability 1, by flipping the coin, such that each side has probability $\frac{1}{2}$.

$G_{\in}$ of Figure 11 for example assigns probability $\frac{1}{2}$ both to H and T. Definition 8.1 defines probability distributions and uniform randomness.

Definition 8.1. A *(probability) distribution* over a finite set X is a function $\nu : X \to [0,1]$ such that $\sum_{x \in X} \nu(x) = 1$. The set of all probability distributions over X is denoted $Distr(X)$. Let $\#X$ denote the number of elements of set X. A probability function $\nu : X \to [0,1]$ is uniform random for $Y \subseteq X$, if for all $y \in Y$, $\nu(y) = \frac{1}{\#Y}$.

Definition 8.2. Let $q \in Q$. A randomized Player i (for $i \in \{1,2\}$) strategy from q is a function $\sigma_i : Pref(q) \to Distr(\mathrm{Act}_i)$, such that $\sigma_i(\pi)(a) > 0$ implies $a \in \Gamma_i(\pi^q_{end})$. With $\Sigma^r_i(q)$ we denote the randomized strategies for Player $i \in \{1,2\}$ from q.

To define the probability of an outcome, we must not only know how player Players 1 and 2 play, but also how the nondeterminism for choosing from the set of states $Moves(q,a,x)$ is resolved. We introduce Player 3 for resolving this nondeterminism. Definition 8.3 defines randomized Player 3 strategies, similar to its non-randomized variant in Definition 6.1. The outcome of a game for a Player 1, 2 and 3 strategy then consists of all plays with actions that have a non-zero probability.

Definition 8.3. A randomized Player 3 strategy is a function $\sigma_3 : \Pi(q) \times \mathrm{Act}_1 \times \mathrm{Act}_2 \to Distr(Q)$, such that $\sigma_3(\pi,a,x)(q') > 0$ implies $q' \in Moves(\pi^q_{end},a,x)$ for all $\pi \in \Pi(q), a \in \Gamma_1(q), x \in \Gamma_2(q)$. We write $\Sigma^r_3(q)$ for the set of all randomized Player 3 strategies from q.

The *outcome* $Outc(\sigma_1,\sigma_2,\sigma_3)$ of randomized strategies $\sigma_1 \in \Sigma^r_1(q)$, $\sigma_2 \in \Sigma^r_2(q)$, and $\sigma_3 \in \Sigma^r_3(q)$ are the plays $\pi \in \Pi^\infty(q)$ such that for all $j \in \mathbb{N}$:

$$\sigma_1(\pi_{0:j})(\pi^a_j) > 0 \ \wedge \ \sigma_2(\pi_{0:j})(\pi^x_j) > 0 \ \wedge \ \sigma_3(\pi_{0:j},\sigma_1(\pi_{0:j}),\sigma_2(\pi_{0:j}))(\pi^q_{j+1}) > 0$$

Given randomized strategies for Player 1, 2, and 3, Definition 8.4 defines the probability of a play prefix of the game. This probability is computed as the multiplication of the probabilities given by the strategies for the play prefix. A Player 1 strategy is then an *almost sure winning strategy* if it can win from any Player 2 and Player 3 strategy, with probability 1. Note that for the randomized strategy σ_p, that chooses H and T with probability $\frac{1}{2}$ in the game $G_{\in}$, Player 2 (and 3) cannot prevent with any strategy to not choose the same side of the coin, since Player 1 chooses at random. Since Player 1 may always try again, eventually, always state ☺ will be reached (i.e. with probability 1). Consequently σ_p is an almost sure winning strategy.

Definition 8.4. Let $\sigma_1 \in \Sigma^r_1(q_0)$, $\sigma_2 \in \Sigma^r_2(q_0)$, and $\sigma_3 \in \Sigma^r_3(q_0)$ be randomized strategies, and let $\pi = q_0\langle a_0,x_0\rangle q_1 \ldots \langle a_{j-1},x_{j-1}\rangle q_j$ be a finite play prefix. We define its probability as:

$$P(\pi) = \prod_{i=0}^{j-1} \sigma_1(\pi_{0:i})(a_i) \cdot \sigma_2(\pi_{0:i})(x_i) \cdot \sigma_3\left(\pi_{0:i},\sigma_1(\pi_{0:i}),\sigma_2(\pi_{0:i})\right)(q_{i+1})$$

The strategies $\bar{\sigma} = (\sigma_1,\sigma_2,\sigma_3)$ define a probability space $(\Omega,\mathcal{F},\mathcal{P}^{\bar{\sigma}})$ over the set of outcomes. A Player 1 strategy $\sigma_1 \in \Sigma^r_1(q)$ is *almost sure winning* for reachability goal R if for all $\sigma_2 \in \Sigma^r_2(q)$, and $\sigma_3 \in \Sigma^r_3(q)$, we have: $\mathcal{P}^{\bar{\sigma}}[\{\pi \in Outc(\sigma_1,\sigma_2,\sigma_3) \mid \pi \text{ is winning}\}] = 1$. Strategy σ_1 is *sure winning* if for all $\sigma_2 \in \Sigma^r_2(q)$, and $\sigma_3 \in \Sigma^r_3(q)$, and for any $\pi \in Outc(\sigma_1,\sigma_2,\sigma_3)$, π is winning.

Definition 8.7 restates the definition of the probabilistic attractor [dAHK98], for our concurrent games. The Player 1 probabilistic attractor $pAttr_1(G,R)$ is computed iteratively to find the largest set of states that Player 1 can confine the game to. To ensure this

confinement, Player 1 restricts the set of actions she chooses from, for each state. These Player 1 actions for states correspond to the witness $CPre_1^w$ as defined in Definition 4.10. This witness gets refined in each iteration $pAttr_1^k(G,R)$, by computing the states Player 1 and 2 can confine the game to for the current witness, namely the set of states $Safe_i(Q')$ (Definition 8.6). As noted in [dAHK98], efficient data structures can be used for efficient computation [Bee80]. In Definition 8.6 we give the (simple) naive fixpoint computation.

Example 8.5. As an example, consider the game $G_{\in}$ of Figure 11 again. The computation of the probabilistic attractor is then as follows:

$$
\begin{aligned}
pAttr_1^0(G_{\in},\{☺\}) &= \{1,☺\}\\
pAttr_2^0(G_{\in},\{☺\}) &= Safe_2(pAttr_1^0(G_{\in},\{☺\}) \setminus \{☺\}, CPre_1^w(pAttr_1^0(G_{\in},\{☺\})))\\
&= Safe_2(\{1,☺\} \setminus \{☺\}, CPre(\{1,☺\}))\\
&= Safe_2(\{1\},\{(1,H),(1,T),(2,H),(2,T)\})\\
&= \emptyset \text{ (see below)}\\
Safe_2^0(\{1\},\{(1,H),(1,T),(2,H),(2,T)\}) &= \{1\}\\
Safe_2^1(\{1\},\{(1,H),(1,T),(2,H),(2,T)\}) &= \{1\} \cap pCPre_2(\{1\}) = \{1\} \cap \emptyset = \emptyset\\
&= Safe_2^2(\{1\},\{(1,H),(1,T),(2,H),(2,T)\})\\
pAttr_1^1(G_{\in},\{☺\}) &= Safe_1(pAttr_1^0(G_{\in},\{☺\}) \setminus pAttr_2^0(G_{\in},\{☺\}), CPre_1^w(pAttr_1^0(G_{\in},\{☺\})))\\
&= Safe_1(\{1,☺\},\{(1,H),(1,T),(2,H),(2,T)\}) = \{1,☺\} \text{ (see below)}\\
Safe_1^0(\{1,☺\},\{(1,H),(1,T),(2,H),(2,T)\}) &= \{1,☺\}\\
Safe_1^1(\{1,☺\},\{(1,H),(1,T),(2,H),(2,T)\}) &= \{1,☺\} \cap pCPre_1(\{1,☺\}) = \{1,☺\}
\end{aligned}
$$

Definition 8.6. The *probabilistic controlled predecessor* $pCPre_i(Q',W)$, and *confinement sets* $Safe_i$ for Player $i \in \{1,2\}$ are defined as follows:

$$
\begin{aligned}
&pCPre_1(Q',W) = \{\ q \in Q \mid \exists a \in \Gamma_1(q) : (q,a) \in W \wedge \forall x \in \Gamma_2(q) : Moves(q,a,x) \subseteq Q'\}\\
&pCPre_2(Q',W) = \{\ q \in Q \mid \exists x \in \Gamma_2(q), \forall a \in \Gamma_1(q) : (q,a) \in W \implies Moves(q,a,x) \subseteq Q'\}\\
&Safe_i^0(Q',W) = Q'\\
&Safe_i^{k+1}(Q',W) = Q' \cap pCPre_i(Safe_i^k(Q',W),W)\\
&Safe_i(Q',W) = \bigcap_{k\in\mathbb{N}} Safe_i^k(Q',W)
\end{aligned}
$$

Definition 8.7. The Player 1 probabilistic attractor is $pAttr_1(G,R)$, where:

$$
\begin{aligned}
&pAttr_1^0(G,R) = Q\\
&pAttr_2^k(G,R) = Safe_2(pAttr_1^k(G,R) \setminus R, CPre_1^w(pAttr_1^k(G,R)))\\
&pAttr_1^{k+1}(G,R) = Safe_1(pAttr_1^k(G,R) \setminus pAttr_2^k(G,R), CPre_1^w(pAttr_1^k(G,R)))\\
&pAttr_1(G,R) = \bigcap_{k\in\mathbb{N}} pAttr_1^k(G,R)
\end{aligned}
$$

The function $pARank : Q \times 2^Q \to \mathbb{N}$ associates to each state $q \in Q$ a rank $pARank(q,R) = \min\{k \in \mathbb{N} \mid q \in pAttr_1^k(G,R)\}$.

A randomized Joker strategy $\sigma \in \Sigma_1^r(q)$ is defined from the witness computed in the last iteration of the probabilistic attractor: the strategy chooses a Player 1 action a from state q uniformly at random for any $(q,a) \in CPre_1^w(pAttr_1(G,R) \setminus R)$. From the computation of Example 8.5, we have that $(1,H),(1,T) \in CPre_1^w(pAttr_1(G,R))$. Hence, the strategy σ_p for game $G_\in$ is a probabilistic attractor strategy, since it assigns probability $\frac{1}{2}$ to H and T each from state 1.

Definition 8.8. A randomized Joker strategy is a strategy that is uniform random for $CPre_1^w(pAttr_1(G,R))$, i.e. for any $\pi \in \Pi(G)$ such that $\pi^q_{end} \in pAttr_1(G,R) \setminus R$ we have $\sigma(\pi)(a) = \dfrac{1}{\#\{(\pi^q_{end},a) \in CPre_1^w(pAttr_1(G,R) \setminus R)\}}$.

8.2. **Probabilistic Joker attractor.** In section 5, several fundamental properties on Joker attractor strategies were given. In this subsection, we will see that these theorems hold for randomized strategies too. In particular, we consider randomized Player 2 and 3 strategies, against randomized Joker strategies. We construct the latter strategies by adapting the Joker attractor (Definition 4.5): we replace the standard attractor $Attr(G,R)$ by the probabilistic attractor $pAttr_1(G,R)$. The result is a probabilistic Joker attractor (Definition 8.9). The corresponding randomized Joker strategies (Definition 8.12) use a Joker action in a Joker state, as usual, but use randomization, like randomized strategies, in other states.

Definition 8.9. The Player 1 probablistic Joker attractor is $pJAttr(G,R)$, where:

$$\begin{aligned} pJAttr^0(G,R) &= pAttr_1(G,R) \\ pJAttr^{k+1}_\clubsuit(G,R) &= pJAttr^k(G,R) \ \cup Pre(pJAttr^k(G,R)) \\ pJAttr^{k+1}(G,R) &= pAttr(pJAttr^{k+1}_\clubsuit(G,R)) \\ pJAttr(G,R) &= \bigcup_{k\in\mathbb{N}} pJAttr^k(G,R) \end{aligned}$$

We call $pJAttr(G,R)$ the probabilistic Joker attractor of G. The probabilistic Joker states are $pJAttr_\spadesuit(G,R) = \bigcup_{k\in\mathbb{N}} pJAttr^k_\clubsuit(G,R) \setminus pJAttr^k(G,R)$. To each probabilistic Joker attractor $pJAttr(G,R)$ we associate a probabilistic Joker rank function $pJRank : Q \to \mathbb{N}$, where for each state $q \in Q$ we define $pJRank(q) = \min\{k \in \mathbb{N} \mid q \in pJAttr^k(G,R)\}$.

Example 8.10. Consider for example the game in Figure 12. Here we added states 0 and ☹ to the penny matching game of Figure 11, and state 0 is the initial state. From state 0, Player 2 is in full control: if Player 2 chooses H, the game moves to state 1, while if he chooses T, the game moves to ☹, nomatter what Player 1 chooses. In the computation the probabilistic Joker attractor, we then find that $\{1,☺\} \subseteq pAttr_1(G,R)$, $0 \in pJAttr^1_\clubsuit(G,R)$, and that $☹ \notin pJAttr(G,R)$.

For randomized Joker strategies (Definition 8.12), we first need to define the probabilistic version of the witnessed Joker attractor (Definition 8.11). The difference with the witnesses of Definition 4.10 is that we extract the witness of a probabilistic attractor, by directly applying $CPre_1^w$ on this set of states, exactly as the definition of the randomized Joker strategy (Definition 8.8). Definition 8.12 defines that in Joker states, a randomized Joker strategy uses the witness (i.e. the Joker action) of the probabilistic Joker attractor, and in other states the strategy uses the witness (i.e. controlled predecessor) of the probabilistic Joker attractor.

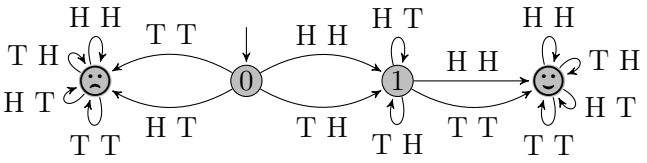

FIGURE 12. The penny matching game extended with two additional states. A randomized Joker strategy uses either of Joker actions (0,H,H) and/or (0,T,H) with probability 1 from state 0, and actions H and T with probability $\frac{1}{2}$ each from state 1. A non-randomized Joker strategy would use Jokers both in state 0 and 1.

Definition 8.11. We define the *witnessed* probabilistic Joker attractor $wpJAttr(G,R)$ as follows:

$$
\begin{aligned}
wpJAttr^0(G,R) &= \emptyset \\
wpJAttr^{k+1}_{\diamondsuit}(G,R) &= wpJAttr^k(G,R) \cup \{(q,a) \in wPre(pJAttr^k(G,R)) \mid q \notin pJAttr^k(G,R)\} \\
wpJAttr^{k+1}(G,R) &= CPre^w_1(pAttr(G, wpJAttr^{k+1}_{\diamondsuit}(G,R))) \\
wpJAttr(G,R) &= \bigcup_{k\in\mathbb{N}} wpJAttr^k(G,R)
\end{aligned}
$$

Definition 8.12. A randomized strategy $\sigma_1 \in \Sigma_1^{\diamondsuit,r}(q)$ in $G^{\diamondsuit}$ is a *randomized Joker strategy*, if for any $\pi \in \Pi(G^{\diamondsuit})$ with $\pi^q_{end} \in pJAttr(G,R)$ and any $a \in \Gamma_1(\pi^q_{end})$ we have:

- $\sigma_1(\pi)(a) > 0 \implies (\pi^q_{end}, a) \in wpJAttr(G,R)$,
- $\pi^q_{end} \in JAttr_{\spadesuit}(G,R) \implies a \in \Gamma_1^{\spadesuit}(\pi^q_{end}) \wedge \sigma_1(\pi)(a) = 1$, and
- $\pi^q_{end} \notin JAttr_{\spadesuit}(G,R) \implies a \in \Gamma_1(\pi^q_{end})$, and σ_1 is uniform random in π^q_{end} for its non-zero probability actions a

The extended penny matching game of Figure 12 shows the advantage of randomized Joker attractor strategies over non-randomized Joker attractor strategies: randomization may help to use less Jokers (Theorem 8.13).

Theorem 8.13. *Let σ be a randomized Joker strategy, and σ' a Joker attractor strategy. Then $Cost(\sigma) \leq Cost(\sigma')$.*

See proof in appendix.

8.3. **Properties of Joker games with randomization.** In this subsection, we revisit the theorems of section 5 for randomized strategies. Theorem 8.15 states that all outcomes of a *randomized* Joker strategy use exactly the same number of Jokers. The intuition for this is that the probabilistic Joker attractor $pJAttr(G,R)$ and the Joker attractor $JAttr(G,R)$ both use the predecessor *Pre* in the construction for the Joker action. The difference is the use of $pAttr_1(G,R)$ versus $Attr(G,R)$, i.e. in the penny matching game of Figure 11, Joker strategies use 1 Joker to move from state 1 to ☺, while randomized Joker strategies use 0 Jokers.

Theorem 8.14. *Let $q \in pJAttr(G,R)$, and let $G^{\diamondsuit}$ be the Joker game, where all players may use randomized strategies. Then:*

(1) *Let* $\sigma_1^J \in \Sigma_1^{\diamondsuit,r}(q)$ *be a randomized Joker strategy in* $G^{\diamondsuit}$*. Then any play* $\pi \in Outc(\sigma_1^J)$ *has* exactly $pJRank(q)$ *Joker actions in winning prefix* $\pi_{0:WinInd(\pi)}$.
(2) *Let* $\sigma_1 \in \Sigma_1^{\diamondsuit(q),r}$ *be a randomized, cost-minimal strategy in* $G^{\diamondsuit}$*. Then any play* $\pi \in Outc(\sigma_1)$ *has* at most $pJRank(q)$ *Joker actions in the winning prefix* $\pi_{0:WinInd(\pi)}$.

See proof in appendix.

In Theorem 8.15 we state that in (Joker) games with randomized strategies, the probabilistic Joker attractor coincides with the states from where a goal state can be reached in the concurrent game, and with the states from where Player 1 can win in the Joker game. The intuition for the proof is that Joker attractor strategies may replace the power of randomization (of randomized Joker strategies) by using additional Jokers actions (see also subsection 8.4).

Theorem 8.15. *Consider the Joker games* $G^{\diamondsuit}$*, where all players may use randomized strategies. Let* $Reach(G, R) = \{q \in Q \mid q \text{ can reach a state } q' \in R\}$*. Then*

$$WinReg(G^{\diamondsuit}, R) = pJAttr(G, R) = Reach(G, R)$$

See proof in appendix.

Theorem 8.16 states that randomized Joker strategies are cost-minimal. The intuition is that the probabilistic Joker attractor has the same structure as the Joker attractor.

Theorem 8.16. *Consider the Joker games, where all players may use randomized strategies. Then any randomized Joker strategy is cost-minimal.*

See proof in appendix.

Similar to determinacy for Joker games without randomized strategies (Theorem 5.4), we have that determinacy of Joker games *with* randomized strategies (Theorem 8.17) follows from Theorem 8.15.

Theorem 8.17. *Consider the Joker games, where all players may use randomized strategies. Then these Joker games are determined.*

See proof in appendix.

8.4. **The (non-)benefits of randomization in Joker games.** We show that Joker games do not need randomized strategies: if Player 1 can win a Joker game with a randomized strategy, then she can win this game with a non-randomized strategy. This result is less surprising than it may seem (Theorem 8.18(1)), since Joker actions are very powerful: they can determine the next state of the game to be any state reachable in one move. In the penny matching game of Figure 11, Player 1 may in state 1 just take the Joker move $(H, H, ☺)$ and reach the state ☺ immediately. With randomization, Player 1 can win this game without using Joker moves.

We note however that we only need to use the power of Jokers in Joker states (Theorem 8.18(2)). We can use the probabilistic Joker attractor (Definition 8.9) to attract to Joker states with probability 1, and then use a Joker move in this Joker state, where even using randomization, the game cannot be won with probability 1. The Jokers used in these Joker states then only needs to be played deterministically, i.e. with probability 1 (Theorem 8.18(3)). The intuition for this last statement is that a Joker move determines

the next state completely, so by choosing the 'best' next state there is no need to include a chance for reaching any other state. In Theorem 8.19 we note that, in particular, the randomized Joker strategies of Definition 8.12 satisfy this last property Theorem 8.18(3).

Theorem 8.18. *If a state $q \in Q$ of Joker game $G^{\diamondsuit}$ has an almost sure winning strategy $\sigma_1^r \in \Sigma_1^{r,\diamondsuit}(q)$, then*
(1) *she also has a winning non-randomized strategy.*
(2) *she also has an almost sure winning strategy that only uses Jokers in Joker states.*
(3) *she also has an almost sure winning strategy that only uses Jokers in Joker states, such that these Jokers can all be played with probability 1.*

See proof in appendix.

Theorem 8.19. *Any strategy from Definition 8.12 satisfies Theorem 8.18(3).*

See proof in appendix.

9. Experiments

We illustrate the application of Joker games in model-based testing (MBT).

MBT. In MBT, the model specifies the desired interaction of the system-under-test (SUT) with its environment, e.g. the user or some other system. The tester (i.e. this is a special user) interacts with the SUT to find bugs, i.e. undesired interactions, of the system. A test case specifies the inputs the tester provides, given the outputs that the SUT provides in the interaction. In MBT, test cases are derived from the model. By executing the test cases on the SUT, it can be checked whether the SUT shows the desired interaction, as specified by the model.

Because the number of test cases that can be derived from the model is usually infinite, a selection criteria is used to select a finite number. One standard criteria is to derive test cases such that for each state in the model, a test case reaching this state in the model is selected. When the test cases have been executed on the SUT, we say that all states are covered.

The derivation of test cases that cover all states is non-trivial, since the tester may not have full control over the outputs of the SUT. The SUT may choose its outputs in such a way that prevents reaching some states. On the other hand, most SUTs are not adversarial by nature, but they are also not necessarily cooperative.

Testing as a game. Therefore, we apply our Joker games to find test cases that reach a state with the least help, i.e. the minimum number of Jokers. To do this, we represent the tester as Player 1 and the SUT as Player 2 in our Joker games. Specifically, we translate the model-based testing model to a concurrent game, compute Joker attractor strategies and their Joker-inspired strategies, and translate the latter strategies to test cases. The first and last step are performed via the translations provided by Van den Bos and Stoelinga [vdBS18].

In the translation from model to game, we use that in each game state, Player 1/Tester has three options: stop testing, provide one of her inputs to the SUT, or observe an output of the SUT. Player 2/SUT has two options: provide an output to the tester, or do nothing. Hence Player 1 and 2 decide concurrently what their next action is. The next state is then determined as follows: if the Tester provides an input, and the SUT does nothing, the input

is received and processed by the SUT. If the Tester observes the SUT (virtually also doing no action of its own), and the SUT provides an output, the output was processed by the SUT and sent to the Tester. If the Tester provides an input, while the SUT also provides an output, then several ways of solving this conflict are possible, such as input-eager, output-eager [vdBS18]. We opt for the *nondeterministic* solution, where the picked action is chosen nondeterministically. Note that this corresponds with having a set of states $Moves(q, a, x)$.

Testing experiments. We investigate the effectiveness of Joker strategies in their application in MBT, by comparing the Joker-based testing, as explained above, with randomized testing. In randomized testing, the tester selects an action (i.e. an input, or doing nothing) uniformly at random, or chooses to stop testing. In Joker-based testing the test case executes the actions of the Joker-inspired strategy, or decides to stop testing. By using Joker-inspired strategies as test cases, we allow for a fair and realistic (no use of Joker actions) comparison.

Case studies. We applied our experiments on four case studies from [vdBV21]: the opening and closing behaviour of the TCP protocol (26 states, 53 transitions) [FBJV16], an elaborate drinks vending machine for drinks (269 states,687 transitions) [com, KSHS17], the Echo Algorithm for leader election (105 states, 275 transitions) [Fok18], and the Dropbox functionality for file storage in the cloud (752 states, 1520 transitions) [HPAN16, TL19].

Experimental setup. For each case study, we randomly selected the different goal states: 5 for the (relatively small) TCP case study and 15 for the other cases. For each of these goals, we extract a Joker attractor strategy. Specifically, we compute Joker attractor strategies (Definition 4.11) from the witnessed Joker attractor (Definition 4.10), and take their Joker-inspired strategies (Definition 6.3). If there were multiple resulting strategies for one goal state, we select one at random. Then we translate the obtained Joker-inspired strategies to test cases (one for each goal). Also, we translate the model-based testing model to a concurrent game. The latter two translations are performed as defined by Van den Bos and Stoelinga [vdBS18]. We run each of the obtained Joker test cases 10.000 times. Also, we run 10.000 random test cases. A random test case chooses a possible Player 1 action at uniformly random. All Joker test cases and random test cases choose to stop testing with (the same) probability p, and choose an input action with probability $1 - p$.

Test case execution is done according to the standard model-against-model testing approach [vdBV21]. In this setup, an impartial SUT is simulated from the model, by making the simulation choose any SUT-action uniformly at random, and by resolving any non-determinism by choosing a state from $Moves(q, a, x)$ uniformly at random too. Hence, when a Joker- or random test case proposes a Player1/Tester action, the simulation chooses a Player 2/SUT-action, and then resolves any non-determinism. When the Joker- or random test case proposes to stop testing, the simulation ends. The finite play, with Player 1 and 2 actions, and the visited states is then recorded as the result of the test case execution. This way, we collect the test results of all 10.000 Joker test cases and all 10.000 random test cases, per goal, of each case study. Per goal and case study, and for each of the two sets of 10.000 test cases we then compute the following numbers: (1) the percentage of runs that reach the goal state, and (2) the average number of moves to reach the goal state, if a goal state was reached.

FIGURE 13. Joker-based versus randomized testing: percentage of tests reaching goal.

Results. Figure 13 shows the experimental results for computation (1). In this graph, a point represents one goal state of a case study. The x-coordinate is the computed percentage (1) of the Joker test cases, and the y-coordiante is the computed percentage (1) of the random test cases. This way we compare the percentage of Joker test cases that reach the goal state with the percentage of random test cases that reaches the goal state. Hence, if a point is positioned below the diagonal, the Joker test cases reached the goal state more often than the random test cases. We see that this is the case for all points. Also, a majority of points is close to 0% for the random test cases, while Joker test cases are (much) more successful for most goals.

Figure 14 shows the experimental results for computation (2). To construct the graph, we computed for each goal, and separately for the sets of Joker test cases and random test cases, the average number of actions for reaching the goal. Hence, for each goal we obtain two average numbers: a number r for the random test cases, and a number j for the Joker test cases. For each goal, we then obtain the ratio r/j. If a test case did not reach a goal, the test case is not used to compute the average, so e.g. if x out of the 10.000 test cases reached the goal, the average number of moves is computed from the number of moves of the x test cases.

One bar in Figure 14 represents one case study, with all its goals. The black line depicts the variation between the ratios for the goals. Hence, we see that for all case studies but the Vending Machine, the ratios are close to each other. A possible reason for the high variation of ratios for the Vending Machine case study, is that we can see in Figure 13 that there are especially few random test cases that reach the goals of the Vending Machine case study, such that the average is computed over the number of moves of those few test cases. Moreover, we omitted results for 5 goal states of the Vending Machine case study, as these states were not reached in any of the 10.000 runs of random testing, while there were > 3500 successful Joker runs for each of those goals. In general, for all the case studies, we see that the ratio is above 1, meaning that the Joker test cases use less moves than the random test cases.

The overall conclusion from the graphs of Figure 13 and Figure 14 is that the Joker test cases clearly outperform the random test cases, both in reaching goal states, and the number

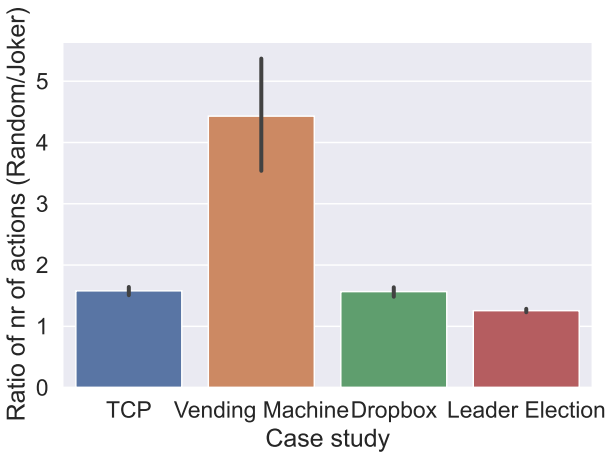

FIGURE 14. Ratio of average number of moves to the goals, where the average number of the random test cases is divided by the average number of the Joker test cases.

of moves needed to reach a goal state. Our experimental results can be reproduced with the artefact [Art].

10. CONCLUSIONS

We introduced the notion of Joker games, showed that its attractor-based Joker strategies use a minimum number of Jokers, and proved properties on determinacy, minimization of number of moves, randomization, and admissible strategies in the context of Joker games. In experiments we showed the effectiveness of Joker strategies when applied in model-based testing.

In future work, we would like to extend the experimental evaluation of Joker strategies on applications, and investigate more cost-minimal strategies. For example, we would like to also find the cost-minimal strategies that are currently not found via the Joker attractor construction. Furthermore, we are interested in other multi-objective strategies than our Joker distance strategies, i.e. strategies that are cost-minimal and also optimized for some other goal. Specifically, inspired by [DNT16], we would like to investigate how to select and construct the Joker strategies that are the most robust against 'bad' opponent actions. Such 'bad' actions could e.g. move the game to a state where the game cannot be won anymore, or to a state from where many Jokers need to be spend to reach the goal. Lastly, we would like to investigate the relation between our work on Joker games and the work of Annand et al. [AMNS23], which comprises adequately permissive assumptions on (turn-based) games with ω-regular winning conditions.

REFERENCES

[AK20] Shaull Almagor and Orna Kupferman. Good-enough synthesis. In Shuvendu K. Lahiri and Chao Wang, editors, *CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, pages 541–563. Springer, 2020. doi:10.1007/978-3-030-53291-8_28.

[AMNS23] Ashwani Anand, Kaushik Mallik, Satya Prakash Nayak, and Anne-Kathrin Schmuck. Computing adequately permissive assumptions for synthesis. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *TACAS 2023, Held as Part of ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II*, volume 13994 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2023. doi:10.1007/978-3-031-30820-8_15.

[Art] The artefact of this paper for reproducing the experimental results of Section 9. https://doi.org/10.5281/zenodo.7712109.

[BBM+21] Ezio Bartocci, Roderick Bloem, Benedikt Maderbacher, Niveditha Manjunath, and Dejan Nickovic. Adaptive testing for specification coverage in CPS models. In Raphaël M. Jungers, Necmiye Ozay, and Alessandro Abate, editors, *7th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2021, Brussels, Belgium, July 7-9, 2021*, volume 54 of *IFAC-PapersOnLine*, pages 229–234. Elsevier, 2021. URL: https://doi.org/10.1016/j.ifacol.2021.08.503, doi:10.1016/J.IFACOL.2021.08.503.

[BBR22] Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. Optimal strategies in concurrent reachability games. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.CSL.2022.7, doi:10.4230/LIPICS.CSL.2022.7.

[BCH+16] Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A. Pérez, Mickael Randour, Jean-François Raskin, Ocan Sankur, and Mathieu Sassolas. Non-zero sum games for reactive synthesis. In Adrian-Horia Dediu, Jan Janousek, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, volume 9618 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2016. doi:10.1007/978-3-319-30000-9_1.

[Bee80] Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. Database Syst.*, 5(3):241–259, 1980. doi:10.1145/320613.320614.

[BEK15] Roderick Bloem, Rüdiger Ehlers, and Robert Könighofer. Cooperative reactive synthesis. In Bernd Finkbeiner, Geguang Pu, and Lijun Zhang, editors, *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, volume 9364 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2015. doi:10.1007/978-3-319-24953-7_29.

[Ber05] Dimitri P. Bertsekas. *Dynamic programming and optimal control, 3rd Edition*. Athena Scientific, 2005. URL: https://www.worldcat.org/oclc/314894080.

[Ber07] Dietmar Berwanger. Admissibility in infinite games. In Wolfgang Thomas and Pascal Weil, editors, *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*, volume 4393 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2007. doi:10.1007/978-3-540-70918-3_17.

[BPRS16] Romain Brenguier, Guillermo A. Pérez, Jean-François Raskin, and Ocan Sankur. Admissibility in quantitative graph games. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2016.42, doi:10.4230/LIPICS.FSTTCS.2016.42.

[CH07] Krishnendu Chatterjee and Thomas A. Henzinger. Assume-guarantee synthesis. In Orna Grumberg and Michael Huth, editors, *TACAS 2007, Held as Part of ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 261–275. Springer, 2007. doi:10.1007/978-3-540-71209-1_21.

[com] ComMA: Introductory ComMA tutorial. https://www.eclipse.org/comma/tutorial/intro.html.

[dAHK98] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 564–575. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743507.

[dAS03] Luca de Alfaro and Mariëlle Stoelinga. Interfaces: A game-theoretic framework for reasoning about component-based systems. In Antonio Brogi, Jean-Marie Jacquet, and Ernesto Pimentel,

editors, *Proceedings of FOCLASA 2003, the Foundations of Coordination Languages and Software Architectures, a satellite event of CONCUR 2003, Marseille, France, September 2, 2003*, volume 97 of *Electronic Notes in Theoretical Computer Science*, pages 3–23. Elsevier, 2003. URL: https://doi.org/10.1016/j.entcs.2004.04.030, doi:10.1016/J.ENTCS.2004.04.030.

[DLLN08] Alexandre David, Kim Guldstrand Larsen, Shuhao Li, and Brian Nielsen. A game-theoretic approach to real-time system testing. In Donatella Sciuto, editor, *Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008*, pages 486–491. ACM, 2008. doi:10.1109/DATE.2008.4484728.

[DNT16] Eric Dallal, Daniel Neider, and Paulo Tabuada. Synthesis of safety controllers robust to unmodeled intermittent disturbances. In *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*, pages 7425–7430. IEEE, 2016. doi:10.1109/CDC.2016.7799416.

[Fae09] Marco Faella. Admissible strategies in infinite games over graphs. In Rastislav Královic and Damian Niwinski, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2009. doi:10.1007/978-3-642-03816-7_27.

[FBJV16] Paul Fiterău-Broştean, Ramon Janssen, and Frits W. Vaandrager. Combining model learning and model checking to analyze TCP implementations. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, volume 9780 of *Lecture Notes in Computer Science*, pages 454–471. Springer, 2016. doi:10.1007/978-3-319-41540-6_25.

[Fok18] Wan Fokkink. Distributed algorithms - an intuitive approach, second edition, 2018.

[HLM+08] Anders Hessel, Kim Guldstrand Larsen, Marius Mikucionis, Brian Nielsen, Paul Pettersson, and Arne Skou. Testing real-time systems using UPPAAL. In Robert M. Hierons, Jonathan P. Bowen, and Mark Harman, editors, *Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, volume 4949 of *Lecture Notes in Computer Science*, pages 77–117. Springer, 2008. doi:10.1007/978-3-540-78917-8_3.

[HPAN16] John Hughes, Benjamin C. Pierce, Thomas Arts, and Ulf Norell. Mysteries of dropbox: Property-based testing of a distributed synchronization service. In *2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016, Chicago, IL, USA, April 11-15, 2016*, pages 135–145. IEEE Computer Society, 2016. doi:10.1109/ICST.2016.37.

[KS23] Orna Kupferman and Noam Shenwald. Games with Trading of Control. In Guillermo A. Pérez and Jean-François Raskin, editors, *CONCUR 2023*, volume 279 of *LIPIcs*, pages 19:1–19:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2023.19, doi:10.4230/LIPIcs.CONCUR.2023.19.

[KSHS17] Ivan Kurtev, Mathijs Schuts, Jozef Hooman, and Dirk-Jan Swagerman. Integrating interface modeling and analysis in an industrial setting. In Luís Ferreira Pires, Slimane Hammoudi, and Bran Selic, editors, *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2017, Porto, Portugal, February 19-21, 2017*, pages 345–352. SciTePress, 2017. doi:10.5220/0006133103450352.

[NVS+04] Lev Nachmanson, Margus Veanes, Wolfram Schulte, Nikolai Tillmann, and Wolfgang Grieskamp. Optimal strategies for testing nondeterministic systems. In George S. Avrunin and Gregg Rothermel, editors, *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2004, Boston, Massachusetts, USA, July 11-14, 2004*, pages 55–64. ACM, 2004. doi:10.1145/1007512.1007520.

[NWZ20] Daniel Neider, Alexander Weinert, and Martin Zimmermann. Synthesizing optimally resilient controllers. *Acta Informatica*, 57(1-2):195–221, 2020. URL: https://doi.org/10.1007/s00236-019-00345-7, doi:10.1007/S00236-019-00345-7.

[TL19] Jan Tretmans and Michel van de Laar. Model-based testing with TorXakis: The mysteries of Dropbox revisited. In V. Strahonja, editor, *CECIIS 2019, Varazdin, Croatia. Proceedings*, pages 247–258. Zagreb: Faculty of Organization and Informatics, University of Zagreb, 2019.

[vdBS18] Petra van den Bos and Mariëlle Stoelinga. Tester versus bug: A generic framework for model-based testing via games. In Andrea Orlandini and Martin Zimmermann, editors, *Proceedings Ninth*

International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018, volume 277 of *EPTCS*, pages 118–132, 2018. doi:10.4204/EPTCS.277.9.

[vdBS23] Petra van den Bos and Mariëlle Stoelinga. With a little help from your friends: Semi-cooperative games via joker moves. In Marieke Huisman and António Ravara, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 43rd IFIP WG 6.1 International Conference, FORTE 2023, Held as Part of the 18th International Federated Conference on Distributed Computing Techniques, DisCoTec 2023, Lisbon, Portugal, June 19-23, 2023, Proceedings*, volume 13910 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2023. doi:10.1007/978-3-031-35355-0_10.

[vdBV21] Petra van den Bos and Frits W. Vaandrager. State identification for labeled transition systems with inputs and outputs. *Sci. Comput. Program.*, 209:102678, 2021. URL: https://doi.org/10.1016/j.scico.2021.102678, doi:10.1016/J.SCICO.2021.102678.

[ZAP+16] Quanyan Zhu, Tansu Alpcan, Emmanouil A. Panaousis, Milind Tambe, and William Casey, editors. *Decision and Game Theory for Security - 7th International Conference, GameSec 2016, New York, NY, USA, November 2-4, 2016, Proceedings*, volume 9996 of *Lecture Notes in Computer Science*. Springer, 2016. doi:10.1007/978-3-319-47413-7.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.

Appendix

Appendix A. Proofs

Theorem 4.7. *For all $q \in Q$, we have $JRank(q) = Cost(q)$.*

Proof Theorem 4.7. First consider case $q \notin JAttr(G, R)$. By Definition 4.5 we then have that $JRank(q) = \infty$. By Theorem 5.2 we have $q \notin Reach(G, R)$. Hence q cannot reach goal R with any play. By Definition 4.2, we then have that any play has cost ∞, so any strategy has cost ∞, and then $Cost(q) = \infty$. Hence, $JRank(q) = Cost(q)$.

Now consider $q \in JAttr(G, R)$, so $q \in Reach(G, R)$. By Definition 4.11 we then have a Joker attractor strategy. By Theorem 5.3, this strategy is cost-minimal. By Theorem 5.1 it uses $JRank(q)$ Jokers in any play, so its cost is $JRank(q)$. Since $Cost(q)$ is defined as the cost of cost-minimal strategies (Definition 4.2), we have that $JRank(q) = Cost(q)$. □

Theorem 5.1. *Let $q \in JAttr(G, R)$. Then:*

(1) *Let $\sigma_1^J \in \Sigma_1^{\diamondsuit}(q)$ be a Joker attractor strategy in $G^{\diamondsuit}$. Then any play $\pi \in Outc(\sigma_1^J)$ has* exactly *$JRank(q)$ Joker actions in winning prefix $\pi_{0:WinInd(\pi)}$.*

(2) *Let $\sigma_1 \in \Sigma_1^{\diamondsuit}(q)$ be a cost-minimal strategy in $G^{\diamondsuit}$. Then any play $\pi \in Outc(\sigma_1)$ has* at most *$JRank(q)$ Joker actions in the winning prefix $\pi_{0:WinInd(\pi)}$.*

Proof Theorem 5.1.

(1) We first prove lemma (W): σ_1^J is a winning strategy in Joker game $G^{\diamondsuit}$. Let q' be a state in a play of $Outc(\sigma_1^J)$. Observe the following three cases:

- If $q' \in Attr(G, R)$, then σ_1^J is a classical attractor strategy from q that is winning [dAHK98], so reaches R, without using any Joker actions.
- If $q' \in JAttr^k(G, R) \setminus JAttr_{\spadesuit}(G, R)$, q' is a non-Joker state of the k-th Joker attractor set, such that σ_1^J is a classical attractor strategy from q' with goal $JAttr^k_{\diamondsuit}(G, R)$. Hence, σ_1^J will surely reach one of the Joker states $JAttr^k_{\diamondsuit}(G, R)$, without using any Joker actions.
- If $q' \in JAttr_{\spadesuit}(G, R)$, then by the definition of a Joker attractor strategy (Definition 4.11), σ_1^J uses a Joker action derived from the predecessor Pre, such that the Joker action causes the game to reach state $q'' \in JAttr^k(G, R)$ from $q' \notin JAttr^k(G, R)$. The use of Pre_w in the definition of a Joker attractor strategy (Definition 4.11) exactly corresponds with the use of Pre in the definition of the Joker attractor (Definition 4.5), so we have $JRank(q'') + 1 = JRank(q')$, and spend 1 Joker action making this move.

Consequently, σ_1^J is a winning strategy, so (W) holds. Moreover, using 1 Joker action, reduces the rank of the next state by 1. Using the attractor in non-Joker states, costs no Joker actions. Since each of the $JRank(q)$ encountered Joker states we use 1 Joker action, we spend exactly $JRank(q)$ Joker actions in total.

2) By Theorem 5.3 we have that a Joker attractor strategy is cost-minimal, so any other cost-minimal strategy must have cost $JRank(q)$. That the cost can be less than $JRank(q)$ for some plays follows from Figure 4. □

Theorem 5.2. *Let $Reach(G, R) = \{q \in Q \mid q$ can reach a state $q' \in R\}$. Then*

$$WinReg(G^{\diamondsuit}, R) = JAttr(G, R) = Reach(G, R)$$

Proof Theorem 5.2. To prove $Reach(G, R) \subseteq WinReg(G^{\diamondsuit}, R)$, we observe that a state $q' \in R$ that can be reached from a state q has a play $q_0\langle a_0, x_0\rangle q_1 \dots q_n$ with $q_0 = q$ and $q_n = q'$. This play allows constructing the winning strategy $\sigma \in \Sigma_1^{\diamondsuit}(q)$ in the Joker game,

that only plays Joker actions to imitate the play: $\sigma(q_0 \dots q_k) = (a_k, x_k, q_{k+1})$ for all $0 \leq k < n$ in $G^{\diamondsuit}$.

We conclude $WinReg(G^{\diamondsuit}, R) \subseteq Reach(G, R)$ from the three facts that (1) a state with a winning strategy in $G^{\diamondsuit}$ has a play reaching R, and (2) $G^{\diamondsuit}$ has the same states as G, and (3) playing a Joker action in $G^{\diamondsuit}$ corresponds to a regular move in G.

From lemma (W) from the proof of Theorem 5.1(1) we can derive that any state in $JAttr(G, R)$ has a winning strategy in $G^{\diamondsuit}$, so $q \in WinReg(G^{\diamondsuit}, R)$. Consequently we have: $JAttr(G, R) \subseteq WinReg(G^{\diamondsuit}, R)$.

Lastly, to prove that $Reach(G, R) \subseteq JAttr(G, R)$, we observe that $q \in (Pre)^k(JAttr(G, R)) \iff q \in JAttr(G, R)$ for any $k \in \mathbb{N}$ by the definition of $JAttr(G, R)$ (Definition 4.5). For any $q \in Reach(G, R)$, we must have $q \in (Pre)^k(JAttr(G, R))$ for some $k \in \mathbb{N}$, as q can reach R in a finite number of steps, because Q is finite. Consequently, we have $q \in JAttr(G, R)$ too.

From above $\subseteq$-relations the stated equalities follow trivially. □

Theorem 5.3. *Any Joker attractor strategy is cost-minimal (a), but not every cost-minimal strategy is a Joker attractor strategy (b).*

Proof Theorem 5.3.
(a) Let $q \in Q$ be a state and let $\sigma \in \Sigma_1(q)$ be a Joker attractor strategy according to Definition 4.11.

First suppose that $q \notin JAttr(G, R)$. Then $q \notin Reach(G, R)$ by Theorem 5.2, so q has no winning play. Then any strategy – also σ – has $Cost(\sigma) = \infty$ by Definition 4.2. Consequently, σ is cost-minimal.

Now suppose that $q \in JAttr(G, R)$. We prove that σ is cost-minimal by induction on $k = JRank(q)$.

If $k = 0$, then σ is a standard attractor strategy that does not spend any Joker actions, so $Cost(\sigma) = 0$. A Joker game has no actions with negative costs, so trivially, σ is cost-minimal.

For $k > 0$, we have the induction hypothesis that any Joker strategy $\sigma' \in \Sigma_1(q')$ with $JRank(q') = m$ for $m < k$ is cost-minimal.

If $q \in JAttr_{\spadesuit}(G, R)$, then σ uses a Joker action (a, x, q'') with $(q, (a, x, q'')) \in wJAttr(G, R)$. By Definition 4.10 we have that $q'' \in JAttr^m(G, R)$ for $m < k$, so σ is cost-minimal when starting in q''. By the Joker attractor construction we know that there is no possibility to reach q'' (or any other state in $JAttr^m(G, R)$) from q for sure (i.e. against any Player 2 and 3), because q is not in the controllable predecessor of $JAttr^m(G, R)$. So Player 2 or 3 can prevent Player 1 from winning if she uses no Joker action, which would yield a path, and corresponding strategy, with cost ∞ . Hence, using one Joker action is the minimum cost for reaching a state in $JAttr^m(G, R)$. Consequently, σ is cost-minimal in this case.

If $q \notin JAttr_{\spadesuit}(G, R)$, then σ is a standard attractor strategy attracting to a Joker state of $JAttr^k(G, R)$. These standard attractor actions have cost 0. Upon reaching a Joker state, the same reasoning as above applies: we need to spend one Joker action. Hence, σ is also cost-minimal in this case.
(b) See Figure 5. □

Theorem 5.4. *Joker games are determined.*

Proof Theorem 5.4. This follows from Theorem 5.2:

- In any state $q \in JAttr(G, R)$ we have a winning Player 1 strategy.
- From any $q \notin JAttr(G, R)$, no state of R can be reached, so Player 2 wins with any strategy. □

Theorem 6.4. *An admissble strategy is not always a Joker-inspired strategy of a cost-minimal strategy.*

Proof Theorem 6.4.
See Figure 6. □

Theorem 6.6.

(1) *A Joker-inspired strategy of a global cost-minimal strategy is not always admissible.*
(2) *A Joker-inspired strategy of a global cost-minimal strategy is admissible for positional Player 2 and 3 strategies.*

Proof Theorem 6.6.
(1) See Figure 8.
(2) Global cost-minimal strategies will, if possible, only use actions such that Player 2 and 3 cannot prevent Player 1 from winning, so we only need to consider states where Player 1 is not in full control: the Joker states. (If there are no Joker states, then any Joker-inspired cost-minimal strategy is winning.) We also note that Player 1 strategies that are not cost-minimal will not be able to do better, since at least the cost-minimal number of Joker states need to be passed by a play to a goal. From any Joker state q, the game continues to a state q', where either three of the following conditions hold:

(1) $q' \notin \mathit{Reach}(G, R)$, i.e. Player 1 can never win the game any more.
(2) $\mathit{JRank}(q) > \mathit{JRank}(q')$, i.e. Player 1 got help from Player 2 and/or 3 to progress towards the goal.
(3) $q' \in \mathit{JAttr}(G, R)$ and $\mathit{JRank}(q) <= \mathit{JRank}(q')$, i.e. Player 1 is prevented to make progress.

Note that, for any Player 1 action, there is a Player 2 and a Player 3 action that forces the game to a state of type (1) or (3), because otherwise q would not have been a Joker state. Joker-inspired global cost-minimal strategies will only propose Player 1 actions that also have a Player 2 and a Player 3 action such that the game moves to a state of type (2). Because the Player 2 and 3 strategies are positional, their choice for the type of the next state, is fixed, given a Player 1 action.

Hence, we draw the following conclusion (C): if two Player 1 strategies choose the same action, the next state of the game is the same. If two Player 1 strategies choose a different Player 1 action, then, depending on the game topology (i.e. the moves from state q), Player 2 and 3 either need to continue to the same type of state for any Player 1 action, or there are Player 2 and 3 strategies to make one Player 1 strategy continue to a type (2) state and the other to a (1) or (3) state, *and vice versa* (because q is a Joker state, so Player 1 cannot force the game to a type (2) state for any enabled Player 1 action).

Consequently, we have a subset of Joker states $Q_1 \subseteq \mathit{JAttr}_\spadesuit(G, R)$, where Player 2 and 3 can force the game to a type (1) state, where Player 1 cannot win (i.e. cannot ever reach a goal state). Furthermore we have a subset of Joker states $Q_3 \subseteq \mathit{JAttr}_\spadesuit(G, R)$, where Player 2 and 3 can only force the game to a type (3) state, but, after a finite number of moves, Player 2 and 3 can force the game to visit a state from Q_1 for any Player 1 strategy that attempts to reach a goal state (i.e. our global cost-optimal strategies).

For the remaining set of Joker states $Q'_3 = \mathit{JAttr}_\spadesuit(G, R) \setminus (Q_1 \cup Q_3)$, we show that Player 2 and 3 can force the game into a cycle, for any Player 1 strategy where Player 2 and 3 cannot force the game to a state of $Q_1 \cup Q_3$. Concretely, in any state $q \in Q'_3$, we have the following observation: Player 2 and 3 must be able to force the game to the same or another state $q' \in Q'_3$. Hence, we can think of this as a graph problem, where we have a finite set

of nodes Q'_3, and each node needs to have an outgoing directed edge to a node of Q'_3. The analysis of this problem is as follows. When trying to add an edge e for each node n, one by one, without making a cycle, one has to connect the edge from n to a node that does not have and outgoing edge yet, because otherwise we create a cycle via e. Because the set Q'_3 is finite, the last added edge needs to go to a node that already has an outgoing edge. Consequently this creates a cycle anyway. Hence, also in Joker states Q'_3 Player 2 and 3 have a strategy such that Player 1 does not ever reach a goal state, as the game is kept in a cycle forever.

Consequently, from any Joker state, Player 2 and 3 have a strategy against any Player 1 strategy, such that a goal state is never reached. Also, Player 2 and 3 have a strategy that allows any global cost-minimal Player 1 strategy to reach a goal state. Because the Player 2 and 3 strategies need to be positional, such a pair fixes the choice of Player 2 action and next state, for a given Player 1 action and current state. Hence, only global cost-minimal Player 1 strategies that choose different actions in Joker states, or steer the game to different Joker states via non-Joker states, may have different winning sets of Player 2 and 3 strategies.

This leads to the generalized conclusion C: two global cost-minimal Player 1 strategies either have the same pairs of winning Player 2 and 3 strategies, or for each of the two Player 1 strategies there is a pair of winning Player 2 and 3 strategies the other strategy cannot win from. Hence, each of the two global cost-minimal Player 1 strategies does not dominate the other strategy. Strategies that are not global cost-minimal will not be able to dominate either, as they also will have to pass a Joker state towards a goal state. Consequently, global cost-optimal strategies are admissible. □

Theorem 7.7. *Let σ be a Joker distance strategy, and σ' a Joker attractor strategy, both in Joker game $G^{\diamondsuit}$. Then σ is cost-minimal (a), and for any play $\pi \in Outc(\sigma)$ and $\pi' \in Outc(\sigma')$ we have $Cost(\pi) \leq Cost(\pi')$ (b).*

Proof Theorem 7.7.
(a)

The following observation provides the foundation for below proof: When a Joker state q is added to a set $dJAttr^k(G, R)$ of the Joker distance attractor (Definition 7.2) by the distance predecessor *dPre* (Definition 7.1), the definition of *dPre* requires the following when adding a state q:

- When $JRank(q) = JRank(q')+1$, q must have been added for some Joker move from q to q', since this part of the definition (i.e. $\exists a \in \Gamma_1(q), \exists x \in \Gamma_2(q), \exists q' \in Moves(q, a, x) : q' \in Q'$) is exactly the predecessor *Pre* operation of Definition 4.3.
- When $JRank(q) = JRank(q')$, q is added for some controlled move from q to q', since this part of the definition (i.e. $\exists a \in \Gamma_1(q), \forall x \in \Gamma_2(q) : Moves(q, a, x) \subseteq Q'$) is exactly the controlled predecessor of Definition 4.3. So , the rank of the states via the controlled predecessor remain the same, and no Joker action is used.

Now we do a proof by induction to show that Joker distance strategies are cost-minimal. Let $\sigma \in \Sigma(q^0)$ be a Joker distance strategy for Joker game $G^{\diamondsuit}$ with initial state $q^0 \in JAttr(G, R)$. Let $JRank(q^0) = n$ for some $n \in \mathbb{N}$.

- First suppose $n = 0$. Because the Joker attractor added q^0 via controlled predecessor operations only, this must be the case for the Joker distance predecessor too. Hence, the resulting strategy will enforce a play with states $q^0, q', ..., r$, where $r \in R$, all states have rank 0, and 0 Joker actions are used. Then trivially σ is cost-minimal.

- Now suppose that $n > 0$. From the induction hypothesis we obtain that for any state q' with $JRank(q') = n-1$, the Joker distance strategy σ will use $JRank(q')$ Jokers for any play starting from q', i.e., against any opponent. There must be a state q' with $JRank(q') = n-1$ such that there is a play from q to q', because $q \in JAttr^n(G,R)$ and $q' \in JAttr^{n-1}(G,R)$. In particular, we know from the Joker attractor construction that we have a finite play π from q to q' with states $q, ..., q'', q_j, q''', ..., q'''', q'$, such that $q''', ..., q''''$ are obtained via the controlled predecessor, then state q_j via the predecessor, i.e. via a Joker move, and states $q, ..q., q''$ via the controlled predecessor again. Because the Joker distance predecessor adds states also via the predecessor or controlled predecessor, while ensuring a rank increase of 1 only occurs when applying the predecessor, and otherwise remains equal, q must also have be added to the Joker distance attractor at some iteration, though the intermediate states added can be different than those in π (and hence the corresponding strategy differs). The consequence is that the Joker distance attractor will have witnesses for moves from q to q' such that only one Joker action is used. By the induction hypothesis we then obtain that the Joker distance strategy uses n Jokers from q^0.

Since Joker strategies are cost-minimal (Theorem 5.3), and since have proven that Joker distance strategies use the same number of Joker actions, we obtain that Joker distance strategies are cost-minimal.

(b) We observe that the Joker distance predecessor selects all states that can be added via both the predecessor and the controlled predecessor. The Joker attractor either uses the controlled predecessor or the predecessor to add states. Consequently, the witnessed Joker distance attractor considers all and more winning plays in the Joker game than the witnessed Joker attractor. Moreover, the Joker distance attractor minimizes the distance of the selected moves because of its attractor structure, where its rank is the distance. Therefore a Joker distance strategy uses the same or less moves than a Joker strategy for any play. □

Theorem 8.13. *Let σ be a randomized Joker strategy, and σ' a Joker attractor strategy. Then $Cost(\sigma) \leq Cost(\sigma')$.*

Proof Theorem 8.13.
This follows from a result of [BBR22]. They define a set of states $\boldsymbol{Sure}(R)$ that have a sure winning strategy (win against any opponent), i.e. this corresponds to $Attr(G,R)$ in our notation. They furthermore define a set of states $\boldsymbol{Almost}(R)$ that have an almost sure winning strategy (win against any opponent with probability 1), i.e. this corresponds to $pAttr_1(G,R)$ in our notation. The result of [BBR22] is then that $\boldsymbol{Sure}(R) \subseteq \boldsymbol{Almost}(R)$. Hence, the probabilistic attractor consists of the same or more states than the standard attractor. The result is that states that are Joker states according to the Joker attractor may be included in the probabilistic attractor. The probabilistic Joker attractor then finds that the initial state has a lower Joker rank, than with the Joker attractor, such that the randomized winning strategies σ in the Joker game use less Jokers than Joker attractor strategies σ'. Figure 12 shows that games exist for which this occurs. In the other cases (since $\boldsymbol{Sure}(R) \subseteq \boldsymbol{Almost}(R)$), the probabilistic attractor and standard attractor yield the same set of states. Hence, we have for randomized Joker strategies σ and Joker attractor strategies σ' that $Cost(\sigma) \leq Cost(\sigma')$. □

Theorem 8.14. *Let $q \in pJAttr(G,R)$, and let $G^{\diamondsuit}$ be the Joker game, where all players may use randomized strategies. Then:*

(1) *Let* $\sigma_1^J \in \Sigma_1^{\diamondsuit,r}(q)$ *be a randomized Joker strategy in* $G^\diamondsuit$*. Then any play* $\pi \in Outc(\sigma_1^J)$ *has* exactly $pJRank(q)$ *Joker actions in winning prefix* $\pi_{0:WinInd(\pi)}$.
(2) *Let* $\sigma_1 \in \Sigma_1^{\diamondsuit(q),r}$ *be a randomized, cost-minimal strategy in* $G^\diamondsuit$*. Then any play* $\pi \in Outc(\sigma_1)$ *has* at most $pJRank(q)$ *Joker actions in the winning prefix* $\pi_{0:WinInd(\pi)}$.

Proof Theorem 8.14.
(1) We can follow the original proof of Theorem 5.1(1), and only make the following two changes to use it in the setting of randomized strategies:

- In case $q' \in Attr(G,R)$, we have that σ_1^J is a *probablistic* attractor strategy from q that is *almost* sure winning, i.e. reach R *with probability 1*, without using any Joker actions.
- In case $q' \in JAttr^k(G,R) \setminus JAttr_\spadesuit(G,R)$, we similarly have a *probablistic* attractor strategy now, that *almost* surely reaches one of the Joker states without using any Joker actions.

(2) follows from Theorem 8.16 as in the original proof of Theorem 5.1(2). □

Theorem 8.15. *Consider the Joker games* $G^\diamondsuit$*, where all players may use randomized strategies. Let* $Reach(G,R) = \{q \in Q \mid q \text{ can reach a state } q' \in R\}$*. Then*

$$WinReg(G^\diamondsuit, R) = pJAttr(G,R) = Reach(G,R)$$

Proof Theorem 8.15. We follow the proof of Theorem 5.2 with the following changes:

- In the proof of case $Reach(G,R) \subseteq WinReg(G^\diamondsuit,R)$ we adapt the strategy σ to play the proposed Joker actions with probability 1.
- In the proof of case $WinReg(G^\diamondsuit,R) \subseteq Reach(G,R)$, we use for fact (1) that there is an almost sure winning strategy, and in fact (3) we use a Joker action that is played with probability 1 .
- The proof of case $pJAttr(G,R) \subseteq WinReg(G^\diamondsuit,R)$ uses lemma (W) of Theorem 5.1(1) as described for this case in Theorem 5.2. But in lemma (W) we substitute $Attr$ by $pAttr$, $JAttr(G,R)$ by $pJAttr(G,R)$, and $JRank$ by $pJRank$.
- In the proof of case $Reach(G,R) \subseteq pJAttr(G,R)$, we need no changes, as the definition of $pJAttr(G,R)$ uses Pre also when $pAttr_1(G,R)$ is used instead of $Attr(G,R)$. □

Theorem 8.16. *Consider the Joker games, where all players may use randomized strategies. Then any randomized Joker strategy is cost-minimal.*

Proof Theorem 8.16. We follow the proof of Theorem 5.3, but use the probabilistic definitions at all places in this proof. Hence we adapt the proof as follows:

- We define $\sigma \in \Sigma_1^r(q)$ as a randomized Joker strategy.
- In the first case we suppose that $q \notin pJAttr(G,R)$, and use Theorem 8.15 instead of Theorem 5.2.
- In the second case we take $q \in pJAttr(G,R)$ and prove that σ is cost-minimal by induction on $k = pJRank(q)$.
  - In case $k = 0$ we use a randomized Joker strategy (Definition 8.7) instead of a standard attractor strategy, and similarly use no Joker actions, so σ is cost-minimal.
  - In case $k > 0$ we have the induction hypothesis that any $\sigma' \in \Sigma_1^r(q')$ with $pJRank(q') = m$ for $m < k$ is cost-minimal.
    * If $q \in pJAttr_\spadesuit(G,R)$, then σ uses a Joker action (a,x,q'') with $(q,(a,x,q'')) \in wpJAttr(G,R)$. By Definition 8.9 we have that $q'' \in pJAttr^m(G,R)$, so σ is cost-minimal when starting in q''. By the probabilistic Joker attractor construction we know that there is no possibility to reach q'' from q with probability 1, because q has

been excluded by $Safe_1$ at some point. For same reasons as in the proof of Theorem 5.3 it is minimum cost to spend 1 Joker action, so σ is cost-minimal in this case.

* If $q \notin pJAttr_{\spadesuit}(G,R)$ then σ is a randomized Joker strategy to a Joker state of $pJAttr^k(G,R)$, using 0 cost actions only. Upon reaching a Joker state we follow the above reasoning. Hence σ is also cost-minimal in this case. □

Theorem 8.17. *Consider the Joker games, where all players may use randomized strategies. Then these Joker games are determined.*

Proof Theorem 8.17. Similar to the proof of Theorem 5.4 this follows from Theorem 8.15. □

Theorem 8.18. *If a state $q \in Q$ of Joker game $G^{\diamondsuit}$ has an almost sure winning strategy $\sigma_1^r \in \Sigma_1^{r,\diamondsuit}(q)$, then*

(1) *she also has a winning non-randomized strategy.*
(2) *she also has an almost sure winning strategy that only uses Jokers in Joker states.*
(3) *she also has an almost sure winning strategy that only uses Jokers in Joker states, such that these Jokers can all be played with probability 1.*

Proof Theorem 8.18. Let $\sigma_1 \in \Sigma_1^r(q^0)$ be an almost sure winning randomized strategy.
1) Then there is a winning play in $Outc(\sigma_1)$. As in the proof of Theorem 5.2, we can use this play for constructing a strategy that plays the Joker actions matching the play.
2) Because we use the probabilistic Joker attractor (Definition 8.7), in each probabilistic Joker attractor set, we can attract with probability 1 to a Joker state [BBR22]. In a Joker state q we choose Joker action (a,x,q') such that $pJRank(q) > pJRank(q')$ with probability 1. By choosing such a move in a Joker state, the game moves to a state with a smaller Joker rank with probability 1. Decreasing the Joker rank in each Joker state leads to reaching a goal state. Because a next Joker state is reached with probability 1 each time, the overall probability of the resulting strategy is 1 too, so it is almost sure winning.
3) See the proof of 2). □

Theorem 8.19. *Any strategy from Definition 8.12 satisfies Theorem 8.18(3).*

Proof Theorem 8.19. This is exactly what we use in the proof of Theorem 8.18(2). □