With a Little Help from Your Friends: Semi-cooperative Games via Joker Moves

Petra van den Bos[1(✉)] and Marielle Stoelinga[1,2]

[1] Formal Methods and Tools, University of Twente, Enschede, The Netherlands
p.vandenbos@utwente.nl
[2] Department of Software Science, Radboud University, Nijmegen, The Netherlands

Abstract. This paper coins the notion of Joker games where Player 2 is not strictly adversarial: Player 1 gets help from Player 2 by playing a Joker. We formalize these games as cost games, and study their theoretical properties. Finally, we illustrate their use 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 [3], assume/guarantee reasoning [11], interface theory [16], security [25] and test case optimization [14,20]. However, it is also widely acknowledged [4,11,17] 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. This is a major drawback: even if Player 1 cannot win, some strategies are better than others.

Several solutions have been proposed to remedy this problem, via *best effort strategies*, i.e. strategies that determine which move to play in losing states. For example, Berwanger [4] 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. Further, specific solutions have been proposed to refine the notion of winning in specific areas, especially in controller synthesis: [11] refines the synthesis problem 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 [5], a hierarchy of collaboration levels is defined, formalizing how cooperative a controller for an LTL specification can be.

M. Huisman and A. Ravara (Eds.): FORTE 2023, LNCS 13910, pp. 155–172, 2023.
https://doi.org/10.1007/978-3-031-35355-0_10

Our Approach: Joker Games. The above notions are all qualitative, i.e., they refine the notion of winning strategy, but do not quantify how much collaboration from the opponent is needed. To fill this gap, we introduce *Joker games*.

By fruitfully building upon robust strategy synthesis results by Neider et al. [13,23], a *Joker strategy* provides a best effort strategy in cases where Player 1 cannot win. Concretely, we allow Player 1, in addition to her own moves, to play a so-called *Joker move*. With such a Joker move, Player 1 can choose her own move *and* the opponent's move, so that Player 2 helps Player 1 reaching the goal. Then, we minimize the number of Jokers needed to win the game, thereby minimizing the help needed from Player 2 Player to win. We formalize such Joker-minimal strategies as cost-minimal strategies in a priced game.

While the construction for Joker strategies closely follows the construction in [13,23] (and later in [2]), our games extend these results in several ways: First, the games in [2,13,23] are turn-based and deterministic, whereas ours are concurrent and nondeterministic. That is, outcome of the moves may lead to several next states. Nondeterminism is essential in game-based testing, to model a faithful interaction between the tester and the system-under-test [6].

An important difference is that Neider et al. focus on finding optimal strategies in the form of attractor strategies, while we present our Joker strategies as general cost-minimal strategies. In fact, we show that 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.

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.

Finally, we illustrate how Joker strategies can be applied in practice, by extracting test cases from Joker strategies. Here we use techniques from previous work [6], to translate game strategies to test cases. We refer to [6] for related work on game-based approaches for testing. In the experiments of this paper we show on four classes of case studies 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:

- We formalize the minimum help Player 1 needs from Player 2 to win as cost-minimal strategies in a Joker game.
- 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.
- We refine our Joker approach with second objective of the number of moves.
- We illustrate the benefits of our approach for test case generation.

Paper Organization. Section 2 recapitulates concurrent games. Section 3 introduces Joker games, and Sect. 4 investigates their properties. In Sect. 6 we study multi-objective Joker strategies, and in Sect. 6 admissible strategies. In Sect. 7 we apply Joker strategies to test case generation. Section 8 concludes the paper. Proofs for the theorems of this paper are given in [7], and the artefact of our experimental results of Sect. 7 is provided in [1].

2 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 1. *A* concurrent game *is a tuple* $G = (Q, q^0, Act_1, Act_2, \Gamma_1, \Gamma_2, Moves)$ *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* $i \in \{1,2\}$, $\Gamma_i : Q \to 2^{Act_i} \setminus \emptyset$ *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*,*
- $Moves : Q \times Act_1 \times Act_2 \to 2^Q$ *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* $Moves(q,a,x) = \emptyset$ *iff* $a \notin \Gamma_1(q) \vee x \notin \Gamma_2(q)$*.*

For the rest of the paper, we fix concurrent game $G = (Q, q^0, \mathrm{Act}_1, \mathrm{Act}_2, \Gamma_1, \Gamma_2, Moves)$. Next, we define a *play* as a sequence of states and actions.

Definition 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* $q_{j+1} \in Moves(q_j, a_j, x_j)$ *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* $\Pi^\infty(G) = \Pi^\infty(q^0)$*.*

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* $\pi^a_{end} = a_{j-1}$, $\pi^x_{end} = x_{j-1}$, *and* $\pi^q_{end} = q_j$ *for the last Player 1 action, last Player 2 action and last state of a finite play* π*. All states in* $\pi_{0:j}$ *are collected in* $States(\pi_{0:j}) = \{\pi^q_k \mid 0 \le k \le j\}$*. The set of all (finite) plays of a set of infinite plays* $P \subseteq \Pi^\infty(q)$ *is denoted* $Pref(P) = \{\pi_{0:j} \mid \pi \in P, j \in \mathbb{N}\}$*. We define* $\Pi(G) = Pref(\Pi^\infty(G))$*, and for any* $q \in Q$*:* $\Pi(q) = Pref(\Pi^\infty(q))$*.*

We consider plays where Player 1 wins, i.e. reaches a state in $R \subseteq Q$.

Definition 3. *A play* $\pi \in \Pi^\infty(q) \cup \Pi(q)$ *for* $q \in Q$ *is* winning *for reachability goal* R*, if there exist a* $j \in \mathbb{N}$ *such that* $\pi^q_j \in R$*. The* winning index *of a play* $\pi \in \Pi^\infty(q) \cup \Pi(G)$ *is:* $WinInd(\pi, R) = \min\{j \in \mathbb{N} \mid \pi^q_j \in R\}$*, where* $\min \emptyset = \infty$*.*

Given a play prefix, players choose their actions according to a strategy. A strategy is positional, if the choice for an action only depends on the last state of the play. The game outcomes are the possible plays of the game when using the strategy. We define deterministic strategies here; see Sect. 4 for randomization.

Definition 4. *A* strategy *for Player $i \in \{1,2\}$ starting in state $q \in Q$ is a function $\sigma_i : \Pi(q) \to Act_i$, 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. Player 1 can win the game by using a winning strategy from a state of its winning region.

Definition 5. *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 $\sigma_1 \in \Sigma_1(q)$.*

3 Joker Games

We formalize the notion of help by Player 2 by associating to each concurrent game G a Joker game $G^{\diamondsuit}$. In $G^{\diamondsuit}$, Player 1 can always get control, i.e. choose any enabled move, at the cost of using a Joker. Thus, in any state q of the game, Player 1 may either choose a regular Player 1 action $a \in \Gamma_1(q)$, or a Joker action $(a, x, q') \in \Gamma_1(q) \times \Gamma_2(q) \times Q$ with $q' \in Moves(q, a, x)$. In this way, a Joker action encodes getting help from Player 2 and 'the system' (the nondeterministic choice of a state from $Moves(q, a, x)$).

We are interested in strategies using the minimum number of Jokers that Player 1 needs to win, because we will use this later in model-based testing settings, where the test execution is neither cooperative nor adversarial. Thus, we set up a cost game where Joker actions have cost 1 and regular Player 1 actions cost 0. We use $\diamondsuit$ to define the parts of a Joker game that differ from those of a regular game. With $\spadesuit$, we specifically refer to states, moves, etc. where Jokers are actually played.

Definition 6. *We associate to each concurrent game G a Joker game $G^{\diamondsuit} = (Q, q^0, Act_1 \cup (Act_1 \times Act_2 \times Q), Act_2, \Gamma_1^{\diamondsuit}, \Gamma_2, Moves^{\diamondsuit}, Cost(G^{\diamondsuit}, R))$ where:*

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

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

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

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

For the rest of the paper we fix Joker game $G^{\diamondsuit}$. We will write $\Sigma_1^{\diamondsuit}(q)$ for the set of strategies in $G^{\diamondsuit}$ starting at state q. Note that the states of both games are the same, and that all plays of G are also plays of $G^{\diamondsuit}$.

Definition 7 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.

Definition 7. *Let $q \in Q$ be a state, $\pi \in \Pi^{\infty}(q) \cup \Pi(q)$ a play, and $\sigma_1 \in \Sigma_1^{\diamondsuit}(q)$ a strategy in Joker game $G^{\diamondsuit}$. For goal states R, define their cost as follows:*

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

$$Cost(G^{\diamondsuit}, R)(\sigma_1) = \sup_{\pi \in Outc(\sigma_1)} Cost(G^{\diamondsuit}, R)(\pi)$$

$$Cost(G^{\diamondsuit}, R)(q) = \inf_{\sigma_1 \in \Sigma_1^{\diamondsuit}(q)} Cost(G^{\diamondsuit}, R)(\sigma_1)$$

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

Winning States in Joker Games. Normally, the value of a cost game is computed by a fixed point computation.

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

Joker games allow this computation to be simplified, by exploiting their specific structure with cost 0 for competitive and cost 1 for cooperative moves. We adapt the classical attractor construction (see e.g., [26]) on the original game G. The construction relies on two concepts: the predecessor $Pre(Q')$ contains all states with some move into Q'; the controllable predecessor 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 $q' \in Q'$).

Definition 8. *Let $Q' \subseteq Q$ be a set of states. The* predecessor $Pre(Q')$ *of Q', and the* controllable predecessor $CPre_1(Q')$ *of Q' are:*

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

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 [15]; the rank k indicates in which computation step a state was added [26]. Thus, the lower k, the fewer moves Player 1 needs to reach its goal.

Definition 9. *The Player 1* attractor *is* $Attr(G, R)$*, where:*

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

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

The cost of state q in $G^{\diamondsuit}$ is obtained by interleaving the attractor and predecessor operators, computing sets of states that we call the *Joker attractor*. See Fig. 1 for an illustration of 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 12) we do need Joker game $G^{\diamondsuit}$. In states of set $JAttr^k(G, R)$, the game can be won with at most k Jokers, i.e., with cost k. 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 $Attr(Q')$. In Joker states, 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 $JAttr^k(G, R)$.

Definition 10. *The Player 1 Joker attractor is* $JAttr(G, R)$*, where:*

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

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* $JRank(G, R) : Q \to \mathbb{N}$*, where for each state* $q \in Q$ *we define* $JRank(G, R)(q) = \min\{k \in \mathbb{N} \mid q \in JAttr^k(G, R)\}$*.*

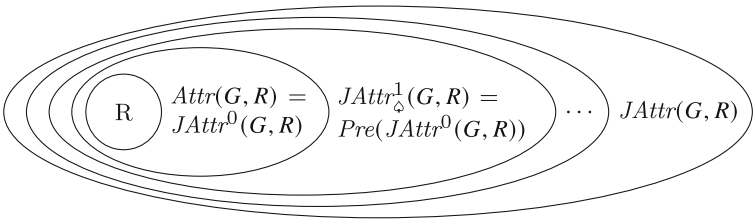


Fig. 1. Illustration of the Joker attractor computation: it starts with states in R initially and then adds states using the attractor and Joker attractor operations, until a fixpoint is reached.

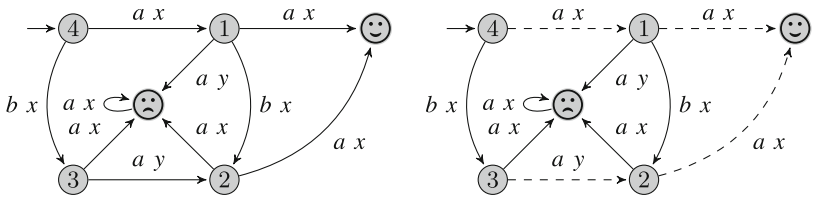


Fig. 2. Left: concurrent game $G_{a \vee b}$ with goal state ☺ and initial state 4. Right: same game, with dashed edges for moves of a Joker attractor strategy.

Example 1. We compute the Joker attractor for goal ☺ in for concurrent game $G_{a \vee b}$ of Fig. 2, together with the $JRank(G, R)$. We indicate whether the state is a Joker state or not, i.e., whether Player 1 plays a Joker.

$$JAttr^0(G_{a\vee b}, ☺) = \{☺\}$$
$$JAttr^1_{♤}(G, R)(G_{a\vee b}, ☺) = \{1, 2, ☺\}$$
$$JAttr^1(G_{a\vee b}, ☺) = \{1, 2, 4, ☺\}$$
$$JAttr^2_{♤}(G, R)(G_{a\vee b}, ☺) =$$
$$JAttr^2(G_{a\vee b}, ☺) = JAttr^3(G_{a\vee b}, ☺) = \{1, 2, 3, 4, ☺\}$$

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

Theorem 1 states the correctness of equations in Definition 10: the number of Jokers needed to reach R, i.e. $JRank(G, R)(q)$, equals $Cost(G^{♤}, R)(q)$.

Theorem 1. *For all $q \in Q$, we have $JRank(G, R)(q) = Cost(G^{♤}, R)(q)$.*

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. To do so, we need to update the definitions of the (controllable) predecessor. This requires some extra administration, recording which actions used in Pre and $CPre_1$ are *witnesses* for moving to states Q'.

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

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. With $wPre$ and $wCPre_1$ we select actions for moving from the states that are newly added to the $k+1$-th attractor, or the $k+1$-th Joker attractor set, respectively, to move to states of the k-th attractor, or k-th Joker attractor set, respectively.

Definition 11. *We define the* witnessed *attractor $wAttr(G, R)$ and Joker attractor $wJAttr(G, R)$:*

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

A *Joker attractor strategy* in $G^{\diamondsuit}$ is a strategy that plays according to the witnesses: in Joker states, a Joker action from $wJAttr^k(G,R)$ is played. In non-Joker states q, the strategy takes its action from $wAttr(G,R)$ if the state has $JRank(G,R)(q)$=0, and from $wAttr(JAttr^k_{\diamondsuit}(G,R))$ if the state has $JRank(G,R)(q) = k$ for some $k > 0$.

Definition 12. *A strategy $\sigma_1 \in \Sigma_1^{\diamondsuit}(q)$ in $G^{\diamondsuit}$ is a* Joker attractor strategy*, if for any $\pi \in \Pi(G^{\diamondsuit})$ with $\pi^q_{end} \in JAttr(G,R)$ and $JRank(G,R)(\pi^q_{end}) = k$ we have:*

$$
\begin{aligned}
&(\pi^q_{end} \in JAttr_{\spadesuit}(G,R) \implies \sigma_1(\pi) \in wJAttr(G,R)) \qquad \wedge \\
&(k = 0 \wedge \pi^q_{end} \notin R \implies (\pi^q_{end}, \sigma_1(\pi)) \in wAttr(G,R)) \qquad \wedge \\
&(k > 0 \wedge \pi^q_{end} \notin JAttr_{\spadesuit}(G,R) \implies (\pi^q_{end}, \sigma_1(\pi)) \in wAttr(JAttr^k_{\diamondsuit}(G,R)))
\end{aligned}
$$

4 Properties of Joker Games

We establish five fundamental properties of Joker games.

1. All outcomes use same number of Jokers. A first, and perhaps surprising result is that, all outcomes in Joker attractor strategy σ, use exactly the same number of Joker actions, namely $JRank(G,R)(q)$ Jokers, if σ starts in state q. This follows from the union-like computation illustrated in Fig. 1. This is unlike cost-minimal strategies in general cost games, where some outcomes may use lower costs. This is also unlike cost-minimal strategies in $JAttr(G,R)$ that are not obtained via the attactor construction, see Fig. 3.

Theorem 2. *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 *$JRank(G,R)(q)$ 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 *$JRank(G,R)(q)$ Joker actions in the winning prefix $\pi_{0:WinInd(\pi)}$.*

The intuition of the proof of Theorem 2 can be derived from Fig. 3: a Joker is always played in a Joker state. By construction of the Joker attractor strategy, the Joker action moves the game from q to a state q' with $JRank(G,R)(q')+1 = JRank(G,R)(q)$. In non-Joker states no Joker is used to reach a next Joker state.

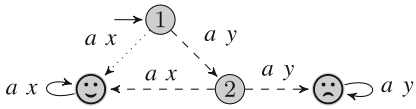


Fig. 3. A cost-minimal strategy is depicted by the dashed edges. It plays a cost-0 a action in state 1, and hopes Player 2 plays x to arrive in ☺ with cost 0. If Player 2 plays y, the strategy plays a Joker in state 2 to win nevertheless. The computation of the Joker attractor yields that state 1 and 2 both have $JRank$ 1. Since state 1 is reached first, the Joker attractor strategy (dotted) plays a Joker in this state immediately, and reaches ☺ directly from state 1. This example shows that cost-minimal strategies from state 1 may use less than $JRank(G, R)(1)$ Jokers, and Joker attractor strategies from 1 always use $JRank(G, R)(1)$ Jokers.

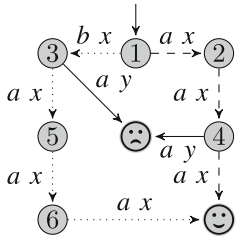


Fig. 4. This game G_{cost} has a cost-minimal strategy (dashed; cost 1 for the Joker used in state 4) that is not a Joker attractor strategy. The unique Joker attractor strategy (dotted) selects Player 1 action b from state 1 (to Joker state 3), since $1, 2 \in Attr^1(G_{cost}, JAttr^1_{\spadesuit}(G_{cost}, \{☺\}))$. The dashed strategy requires fewer moves to reach ☺ (3 vs 4).

2. Characterization of winning states. For the set of states $JAttr(G, R)$, 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 $JAttr(G, R)$.

Theorem 3. *Let* $Reach(G, R) = \{q \in Q \mid q$ *can reach a state* $q' \in R\}$. *Then*

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

3. Joker attractor strategies and minimality. We show some fundamental results relating Joker attractor strategies to Joker-minimal strategies: Theorem 4 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 Fig. 3 and 4. The game of Fig. 4 also shows that Joker attractor strategies need not take the shortest route to the goal (see also Sect. 6).

Theorem 4. *Any Joker attractor strategy is cost-minimal.*

4. Determinacy. Unlike arbitrary concurrent games with deterministic strategies, Joker games are *determined*. That is, in each state of $JAttr(G, R)$, either

Player 1 has a winning strategy or Player 2 can keep the game outside R forever. Determinacy (Theorem 5) follows from Theorem 3: 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. *Joker games are determined.*

5 Joker Games with Randomized Strategies

A distinguishing feature of concurrent games is that, unlike turn-based games, randomized strategies may help to win a game. A classical example is the Penny Matching Game of Fig. 5. 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. We first set up the required machinery, following [15].

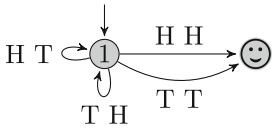


Fig. 5. Penny Matching Game: Each Player chooses a side of a coin. If they both choose heads, or both tails, then Player 1 wins, otherwise they play again. Player 1 wins this game with probability 1, by flipping the coin (each side has probability $\frac{1}{2}$).

Definition 13. *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 $q \in Q$. A randomized Player i strategy from q is a function $\sigma_i : Pref(q) \to Distr(Act_i)$, such that $\sigma_i(\pi)(a) > 0$ implies $a \in \Gamma_i(\pi^q_{end})$. Let $\Sigma^r_i(q)$ 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 nonderminism is resolved. Given moves a and x by Player 1 and 2, the Player 3 strategy $\sigma_3(\pi, a, x)$ chooses, probabilistically, one of the next states in $Moves(\pi^q_{end}, a, x)$.

Definition 14. *A randomized Player 3 strategy is a function $\sigma_3 : \Pi(q) \times Act_1 \times Act_2 \to Distr(Q)$, such that $\sigma_3(\pi, a, x)(q') > 0$ implies $q' \in Moves(\pi^q_{end}, a, x)$ for all $\pi \in \Pi(q)$. 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, Player 2, and Player 3, Definition 15 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.

Definition 15. *Let $\sigma_1 \in \Sigma_1^r(q_0)$, $\sigma_2 \in \Sigma_2^r(q_0)$, and $\sigma_3 \in \Sigma_3^r(q_0)$ 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 $\sigma_1 \in \Sigma_1^r(q)$ is almost sure winning *for reachability goal R if for all $\sigma_2 \in \Sigma_2^r(q)$, and $\sigma_3 \in \Sigma_3^r(q)$, 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 $\sigma_2 \in \Sigma_2^r(q)$, and $\sigma_3 \in \Sigma_3^r(q)$, and for any $\pi \in Outc(\sigma_1, \sigma_2, \sigma_3)$, π is winning.*

Properties of Joker Games with Randomization. In Sect. 4, several fundamental properties were given, which hold for randomized strategies too (Theorem 6). By replacing the standard attractor in the definition of the Joker attractor (Definition 10), by the probabilistic atractor [15], we obtain the definition of the probabilistic Joker attractor. Next, by using this probabilistic Joker attractor instead of the Joker attractor, the definitions for the witnissed probablistic Joker attractor (Definition 11), and probablistic Joker strategies (Definition 12) can be reformulated. From these probablistic definitions, Theorem 6 then follows.

Theorem 6. *Theorem 2, 3, 4 and 5 hold for Joker games with randomized strategies.*

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 deterministic strategy. This result is less surprising than it may seem (Theorem 7(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, 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 7(2)). We can use a probabilistic attractor [15] 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 7(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.

Theorem 7. *If a state $q \in Q$ of Joker game $G^{♤}$ has an almost sure winning strategy $\sigma_1^r \in \Sigma_1^{r,♤}(q)$, then*

1. *she also has a winning deterministic 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.*

6 Better Help from Your Friends: Multi-objective and Admissible Strategies

Short Joker Strategies. Although multiple Joker attractor strategies may be constructed via Definition 12, their number of moves may not be minimal. The cause for this is that Joker attractor strategies have to reduce the $ARank(G, R)$ or $JRank(G, R)$ by 1 each step. Figure 4 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 take a shortest path while spending the minimum number of Jokers, we use the structure of the Joker attractor sets to compute a distance function. Definition 17 defines distance for the following four types of states: goal states R, Joker states $JAttr_{\spadesuit}(G, R)$, non-Joker states part of some attractor $JAttr_{\not\spadesuit}(G, R)$, and unreachable states (states not in $JAttr(G, R)$). In a goal state, the distance is 0, and in an unreachable state it is infinite. In a Joker state, we use the Joker action to the state of the next Joker attractor set that has the smallest distance to the goal. In a non-Joker state from some attractor, we choose a Player 1 action that results in reaching a state within the current Joker attractor set, such that this action minimizes the distance to the goal from the reached state. In the latter case we assume that Player 2 and 3 cooperate with decreasing the distance, by using min in the definition. We chose this because we use the definition in Theorem 8, where we consider the situation that Player 2 and 3 cooperate. Less cooperation can be assumed by e.g. replacing min by max. Definition 16 states auxilary definitions used in Definition 17. Most importantly it defines the Joker restricted enabling condition $\Gamma_1^J(q)$ that returns Player 1 actions that surely lead to states within the current Joker attractor set.

Definition 16. *Let $q \in Q$, and $Q' \subseteq Q$. Define the states reachable in one move $Post(q)$, the k-th Joker states $JAttr^k_{\spadesuit}(G, R)$, the non-Joker, attractor states $JAttr_{\not\spadesuit}(G, R)$, the k-th non-Joker states, the restricted enabling condition $\Gamma_1(Q')$, and the Joker restricted enabling condition Γ_1^J as:*

$$Post(q) = \{q' \in Q \mid \exists a \in \Gamma_1(q), \exists x \in \Gamma_2(q) : q' \in Moves(q, a, x)\}$$

$$JAttr^k_{\spadesuit}(G, R) = JAttr^k_{\clubsuit}(G, R) \setminus JAttr^{k-1}(G, R)$$

$$JAttr_{\not\spadesuit}(G, R) = JAttr(G, R) \setminus JAttr_{\spadesuit}(G, R)$$

$$JAttr^k_{\not\spadesuit}(G, R) = JAttr^k(G, R) \setminus JAttr^k_{\clubsuit}(G, R)$$

$$\Gamma_1(Q')(q) = \{a \in \Gamma_1(q) \mid \forall x \in \Gamma_2(q) : Moves(q, a, x) \subseteq Q'\}$$

$$\Gamma_1^J(q) = \begin{cases} \Gamma_1(Attr(G, R))(q) & \text{if } q \in Attr(G, R) \\ \Gamma_1(JAttr^{k+1}(G, R) \setminus JAttr^k(G, R))(q) & \text{if } q \in JAttr^{k+1}_{\not\spadesuit}(G, R) \\ \emptyset & \text{otherwise} \end{cases}$$

Definition 17. *We define the distance function* $d : Q \to \mathbb{N}$ *as follows:*

$$d(q) = \begin{cases} 0 & q \in R \\ 1 + \min_{q' \in Post(q) \cap JAttr^k(G,R)} d(q') & q \in JAttr_{\spadesuit}^{k+1}(G,R) \\ 1 + \min_{a \in \Gamma_1^J(q)} \min_{x \in \Gamma_2(q)} \min_{q' \in Moves(q,a,x)} d(q') & q \in JAttr_{\not\spadesuit}(G,R) \\ \infty & q \notin JAttr(G,R) \end{cases}$$

A short Joker strategy (Definition 18) minimizes the distance while using the minimum number of Joker actions. The construction of such a strategy from the distance function is straightforward: we use a distance minimizing Player 1 action in a non-Joker state, and a distance minimizing Joker action in a Joker state. This distance minimization follows the structure of Definition 17.

Definition 18. *A strategy* $\sigma \in \Sigma_1(q)$ *in Joker game* $G^{\diamondsuit}$ *is a* short Joker strategy *from* q*, if for any play* $\pi \in Outc(\sigma)$ *with* $q' = \pi_{end}$ *the following formulas hold:*

$$q' \in JAttr_{\spadesuit}(G,R) \implies \sigma(\pi) \in \operatorname*{arg\,min}_{(a,x,q'') \in \Gamma_1^{\spadesuit}(q')} \{d(q'')\}$$

$$q' \in JAttr_{\not\spadesuit}(G,R) \implies \sigma(\pi) \in \operatorname*{arg\,min}_{a \in \Gamma_1^J(q')} \{d(q'') \mid x \in \Gamma_2(q'), q'' \in Moves(q',a,x)\}$$

Unfolding Definition 17 for state 1 of Fig. 4 results in $d(1) = 3$, as expected. With Definition 18 we then obtain a short Joker strategy that is cost-minimal (it uses 1 Joker action), and uses the minimum number of moves (namely 3).

Theorem 8 states that Definition 18 indeed defines cost-minimal strategies using the minimum number of moves for the number of used Joker actions, when helped by Player 2 and 3 (i.e. there is a play). A short Joker strategy prefers the minimum number of Joker actions over the minimum number of moves, so it will not take a shorter route if that costs more than the minimum number of Jokers.

Theorem 8. *A short Joker strategy* σ *is cost-minimal (1), and has a play* $\pi \in Outc(\sigma)$ *using the minimum number of moves to win while using the minimum number of Joker actions (2).*

Admissible Strategies. Various papers [9,10,17] advocate that a player should play *admissible* strategies if there is no winning strategy. Admissible strategies (Definition 20) 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'. In Definition 20, the set of winning strategies is defined as the pairs of Player 2 ánd Player 3 strategies, since the nondeterministic choice for the next state affects whether Player 1 wins. Definition 19 defines non-randomized Player 3 strategies, similar to its randomized variant in Definition 14.

Definition 19. *A Player 3 strategy is a function* $\sigma_3 : \Pi(q) \times Act_1 \times Act_2 \to Q$*, such that* $\sigma_3(\pi, a, x) \in Moves(\pi_{end}^q, a, x)$ *for all* $\pi \in \Pi(q)$*. 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 play* $\pi \in \Pi^\infty(q)$ *such that:*

$$\forall j \in \mathbb{N} : \ \sigma_1(\pi_{0:j}) = \pi_j^a \wedge \sigma_2(\pi_{0:j}) = \pi_j^x \wedge \sigma_3(\pi_{0:j}, \sigma_1(\pi_{0:j}), \sigma_2(\pi_{0:j})) = \pi_{j+1}^q$$

Definition 20. *Let* G *be a concurrent game and* R *a reachability goal. Define the Player 2 and 3 strategy pairs that are winning for a strategy* $\sigma_1 \in \Sigma_1(G)$ *as:*

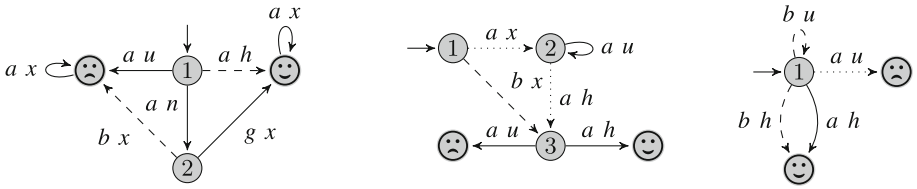
$$Win\Sigma_{2,3}(\sigma_1, R) = \{(\sigma_2, \sigma_3) \in \Sigma_2(G) \times \Sigma_3(G) \mid Outc(\sigma_1, \sigma_2, \sigma_3) \in Win\Pi(G, R)\}$$

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 Joker attractor strategies and admissible strategies, we note that Joker attractor 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 21 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 21. *Let* $\sigma \in \Sigma(q)$ *be a strategy in* $G^\diamondsuit$. *Define the* Joker-inspired strategy σ_{insp} *of* σ *for any* $\pi \in \Pi(G)$ *as:*

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



$$\sigma_2^a(q_0 \ldots q_k \langle a_k, x_k \rangle q_{k+1}) = \begin{cases} u & \text{if } a_k = b \\ h & \text{otherwise} \end{cases}$$

$$\sigma_2^{uh}(q_0 \ldots q_k \langle a_k, x_k \rangle q_{k+1}) = \begin{cases} u & \text{if } q_k = 1 \\ h & \text{otherwise} \end{cases}$$

Fig. 6. Left: the dashed cost-minimal 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. Middle: the dotted strategy uses 2 Jokers, and is admissible, since it wins from strategy σ_2^a while the dashed cost-minimal strategy (using 1 Joker) does not. Right: the dashed cost-minimal strategy dominates dotted cost-minimal strategy, because dashed strategy wins from σ_2^{uh} while dotted strategy does not.

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 Fig. 6(left). Therefore, Theorem 9 relates admissible and *global* cost-minimal strategies (Definition 22): strategies that are cost-minimal for any initial state of the Joker game. Then still, the classes of admissible and of the Joker-inspired variants of global cost-minimal strategies do not coincide. Figure 6(middle,right) shows two examples, with Player 2 strategies using memory, for both parts of Theorem 9: (1) an admissible strategy need not be cost-minimal, and (2) vice versa. We conjecture that the converse of Theorem 9 holds for memoryless Player 2 and 3 strategies.

Definition 22. *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 $q' \in Reach(G, R)$.*

Theorem 9.

1. *A Joker-inspired strategy of a global cost-minimal strategy is not always admissible.*
2. *A admissible strategy is not always a Joker-inspired strategy of a global cost-minimal strategy.*

7 Experiments

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

Testing as a Game. We translate input-output labeled transition systems describing the desired behaviour of the System Under Test (SUT) to concurrent games, as described in [6]. In each game state, Player 1/Tester has 3 options: stop testing, provide one of the inputs to the SUT, or observe the behaviour of the SUT. Player 2/SUT has 2 options: provide an output to the SUT, 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 processed. If the Tester observes the SUT (virtually also doing no action of its own), the output is processed. If the Tester provides an input, while the SUT provides an output, then several regimes are possible, such as input-eager, output-eager [6]. We opt for the *nondeterministic* regime, where picked action is chosen nondeterministically (this corresponds with having a set of states $Moves(q, a, x)$). In this set up, we investigate the effectiveness of Joker-based testing through a comparison with randomized testing. In Joker-based testing we take the Joker-inspired strategies of Joker attractor strategies, to allow for a fair and realistic (no use of Joker actions) comparison.

Case Studies. We applied our experiments on four case studies from [8]: the opening and closing behaviour of the TCP protocol (26 states, 53 transitions) [18], an elaborate vending machine for drinks (269 states,687 transitions) [12,22] the Echo Algorithm for leader election (105 states, 275 transitions) [19], and a tool for file storage in the cloud (752 states, 1520 transitions) [21,24].

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, and translate it via its corresponding Joker-inspired strategy, without Joker actions, to a test case, as described in [6]. We run this Joker test case 10.000 times. Also, we run 10.000 random test cases. A random test case chooses an action uniformly at random in any state. Test case execution is done according to the standard model-against-model testing approach [8]. In this setup, an impartial SUT is simulated from the model, by making the simulation choose any action with equal probability. From the 10.000 test case runs for each goal of each case study, we compute: (1) the number of runs that reach the goal state, and (2) the average number of actions to reach the goal state, if a goal state was reached.

Results. Figure 7 shows the experimental results. In the left graph, a point represents one goal state of a case study, and compares the averages of the Joker test case, and random test case. For the right graph, for each goal, the number of actions needed for reaching the goal for Random testing has been divided by the number of actions for the Joker test case. Results for all goals of one case study are shown with one bar. 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. Clearly, the graphs show that Joker test cases outperform random testing. The experimental results can be reproduced with the artefact of this paper [1].

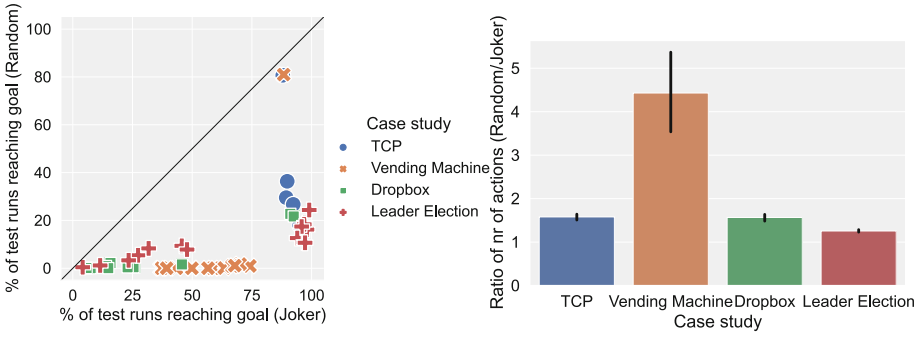


Fig. 7. Experimental results: Joker-based versus randomized testing.

8 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, randomization, and admissible strategies in the context of Joker games.

In future work, we would like to extend the experimental evaluation of Joker strategies on applications, prove (or disprove) that, against memoryless Player 2 and 3 strategies, admissible strategies are global cost-minimal and vice versa, and investigate other multi-objective Joker strategies, a.o. to quantify the 'badness'

of adversarial moves [13], where a bad move is e.g. a move that takes the game to a state where many Jokers need to be spend to reach the goal again.

References

1. The artefact of this paper for reproducing the experimental results of Section 7. https://doi.org/10.5281/zenodo.7712109
2. Bartocci, E., Bloem, R., Maderbacher, B., Manjunath, N., Ničković, D.: Adaptive testing for specification coverage in CPS models. IFAC-PapersOnLine **54**(5), 229–234 (2021). 7th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2021
3. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific (1998)
4. Berwanger, D.: Admissibility in infinite games. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 188–199. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70918-3_17
5. Bloem, R., Ehlers, R., Könighofer, R.: Cooperative reactive synthesis. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) ATVA 2015. LNCS, vol. 9364, pp. 394–410. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24953-7_29
6. van den Bos, P., Stoelinga, M.: Tester versus bug: a generic framework for model-based testing via games, vol. 277, pp. 118–132 (2018)
7. van den Bos, P., Stoelinga, M.: With a little help from your friends: semi-cooperative games via joker moves (2023). Extended version with appendix. arXiv:2304.13417
8. van den Bos, P., Vaandrager, F.: State identification for labeled transition systems with inputs and outputs. Sci. Comput. Program. **209**, 102678 (2021)
9. Brenguier, R., et al.: Non-zero sum games for reactive synthesis. In: Dediu, A.-H., Janoušek, J., Martín-Vide, C., Truthe, B. (eds.) LATA 2016. LNCS, vol. 9618, pp. 3–23. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30000-9_1
10. Brenguier, R., Pérez, G.A., Raskin, J.-F., Sankur, O.: Admissibility in quantitative graph games. CoRR, abs/1611.08677 (2016)
11. Chatterjee, K., Henzinger, T.A.: Assume-guarantee synthesis. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 261–275. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71209-1_21
12. ComMA: Introductory ComMA tutorial. https://www.eclipse.org/comma/tutorial/intro.html
13. Dallal, E., Neider, D., Tabuada, P.: Synthesis of safety controllers robust to unmodeled intermittent disturbances. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 7425–7430. IEEE (2016)
14. David, A., Larsen, K.G., Li, S., Nielsen, B.: A game-theoretic approach to real-time system testing. In: Design, Automation and Test in Europe, DATE 2008, pp. 486–491. IEEE (2008)
15. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. In: FOCS 1998: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, p. 564. IEEE Computer Society (1998)
16. de Alfaro, L., Stoelinga, M.: Interfaces: a game-theoretic framework for reasoning about component-based systems. Electron. Notes Theor. Comput. Sci. **97**, 3–23 (2004)

17. Faella, M.: Admissible strategies in infinite games over graphs. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 307–318. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03816-7_27
18. Fiteră u-Broştean, P., Janssen, R., Vaandrager, F.: Combining model learning and model checking to analyze TCP implementations. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9780, pp. 454–471. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_25
19. Fokkink, W.: Distributed Algorithms - An Intuitive Approach, 2nd edn (2018)
20. Hessel, A., Larsen, K.G., Mikucionis, M., Nielsen, B., Pettersson, P., Skou, A.: Testing real-time systems using UPPAAL. In: Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers, pp. 77–117 (2008)
21. Hughes, J., Pierce, B.C., Arts, T., Norell, U.: Mysteries of dropbox: property-based testing of a distributed synchronization service. In: 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), pp. 135–145. IEEE (2016)
22. Kurtev, I., Schuts, M., Hooman, J., Swagerman, D.-J.: Integrating interface modeling and analysis in an industrial setting. In: Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development - MODELSWARD, pp. 345–352. INSTICC, SciTePress (2017)
23. Neider, D., Weinert, A., Zimmermann, M.: Synthesizing optimally resilient controllers. Acta Informatica **57**(1–2), 195–221 (2020)
24. Tretmans, J., van de Laar, M.: Model-based testing with TorXakis: the mysteries of Dropbox revisited. In: Strahonja, V. (ed.),CECIIS : 30th Central European Conference on Information and Intelligent Systems, 2–4 October 2019, Varazdin, Croatia. Proceedings, pp. 247–258. Faculty of Organization and Informatics, University of Zagreb, Zagreb (2019)
25. Zhu, Q., Alpcan, T., Panaousis, E., Tambe, M., Casey, W. (eds.): GameSec 2016. LNCS, vol. 9996. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47413-7
26. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoret. Comput. Sci. **200**(1), 135–183 (1998)