CONCURRENT REACHABILITY GAMES

by

Luca de Alfaro, Thomas A. Henzinger,
and Orna Kupferman

Memorandum No. UCB/ERL M98/33

16 June 1998

CONCURRENT REACHABILITY GAMES

by

Luca de Alfaro, Thomas A. Henzinger,
and Orna Kupferman

Memorandum No. UCB/ERL M98/33

16 June 1998

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Concurrent Reachability Games*

Luca de Alfaro Thomas A. Henzinger Orna Kupferman

Department of EECS,
University of California at Berkeley,
Berkeley, CA 94720-1770, USA
Email:{dealfaro,tah,orna}@eecs.berkeley.edu

Abstract

An open system can be modeled as a two-player game between the system and its environment. At each round of the game, player 1 (the system) and player 2 (the environment) independently and simultaneously choose moves, and the two choices determine the next state of the game. Properties of open systems can be modeled as objectives of these two-player games. For the basic objective of reachability —can player 1 reach a given set of target states?— there are three types of winning states, according to the degree of certainty with which player 1 can reach the target. From type-1 states, player 1 has a deterministic strategy to always reach the target. From type-2 states, player 1 has a randomized strategy to reach the target with probability 1. From type-3 states, player 1 has for every real $\epsilon > 0$ a randomized strategy to reach the target with probability greater than $1 - \epsilon$.

We show that for finite state spaces, all three sets of winning states can be computed in polynomial time: type-1 states in linear time, and type-2 and type-3 states in quadratic time. The algorithms to compute the three sets of winning states also enable the construction of the winning and spoiling strategies. Finally, we apply our results by introducing a temporal logic in which all three kinds of winning conditions can be specified, and which can be model checked in polynomial time. This logic, called randomized ATL, is suitable for reasoning about randomized behavior in open (two-agent) as well as multi-agent systems.

*The work was partially supported by the SRC contract 97-DC-324.041, by ARO under the MURI grant DAAH04-96-1-0341, by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the DARPA/NASA grant NAG-2-1214, and by the NSF grant CCR-9504469.

1 Introduction

One of the central problems in system verification is the *reachability* question: given an initial state s and a target state t, can the system get from s to t? The dynamics of a closed system, which does not interact with its environment, can be modeled by a state-transition graph, and the reachability question reduces to graph reachability, which can be solved in linear time and is complete for NLOGSPACE [Jon75]. By contrast, the dynamics of an open system, which does interact with its environment, is best modeled as a game between the system and the environment.

In some situations, it may suffice to have the system and the environment take turns to make moves, yielding a *turn-based* model. In this case, the game graph is an AND-OR graph. A (deterministic) strategy for the AND player maps every path that ends in an AND state to a successor state, and similarly for the OR player. Thus the reachability question (can the system get from s to t no matter what the environment does?) reduces to AND-OR graph reachability (does the OR player have a strategy so that for all strategies of the AND player, the game, if started in s, reaches t?). This problem can again be solved in linear time and is complete for PTIME [Imm81]. With respect to AND-OR graph reachability, randomized strategies are no more powerful than deterministic strategies. A *randomized* strategy for the AND player maps every path that ends in an AND state to a probability distribution on the successor states, and similarly for the OR player. It can be seen that the reachability question has the same answer as the probabilistic question "does the OR player have a randomized strategy so that for all randomized strategies of the AND player, the game, if started in s, reaches t with probability 1?".

The turn-based model is naive, because in realistic concurrency models, in each state, the system and the environment independently choose moves, and the parallel execution of the moves determines the next state. Such a *simultaneous* game is a natural model for synchronous systems where the moves are chosen truly simultaneously, as well as for distributed systems in which the moves are not revealed until their combined effect (the state transition) is apparent. In particular, the modeling of synchronization between processes often requires the consideration of simultaneous games.

The simultaneous case is more general than the turn-based one, and deterministic strategies no longer tell the whole story about the reachability question. The fact that randomized strategies can be more powerful than deterministic ones is illustrated by the game LEFT-OR-RIGHT, depicted in Figure 1. Initially, the game is at state t_{throw}. At each round, player 1 can choose to throw a snowball either at the left window (move *throwL*) or at the right window (move *throwR*). Independently and simultaneously, player 2 must choose to stand behind either the left window (move *standL*) or the right window (move *standR*). If the snowball hits player 2, the game proceeds to the target state t_{hit}; otherwise, another round of the game is played from t_{throw}.

For each move of player 1, player 2 has a countermeasure. If we consider only deterministic strategies, then for every strategy of player 1, there is (exactly one) strategy of player 2 such that t_{hit} is never reached. Hence, if we base our definitions on deterministic strategies, we obtain to answer NO to the reachability question. The situation of player 2, however, is not nearly as safe as this negative answer implies. If player 1 chooses at each

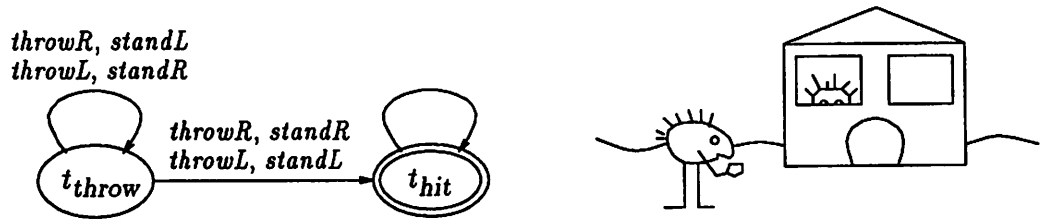

Figure 1: Game LEFT-OR-RIGHT.

round the window at which to throw the snowball by tossing a coin, then player 2 will be hit with probability 1, regardless of her strategy.

The coin-tossing criterion used by player 1 to select the move is an example of randomized strategy, and the game illustrates the value of randomized strategies for winning reachability games. If player 1 adopts a deterministic strategy, the moves he plays during the game are completely determined by the history of the game, which is visible also to player 2. Once player 1 has chosen a deterministic strategy, player 2 can choose her strategy to counteract every move of player 1, as if she were able see it before choosing her own move. Randomized strategies postpone the choice of the move until the game is being played, precluding this type of spying behavior. Another way of thinking about randomized strategies is through the concept of *initial randomization.* The choice of a randomized strategy is equivalent to the choice of a probability distribution over the set of deterministic strategies [Der70]. By choosing such a distribution, rather than a single strategy, player 1 prevents player 2 from tailoring her strategy to counteract the strategy chosen by player 1. The greater power of randomized strategies is a well-known fact in game theory, and it has its roots in von Neumann's minimax theorem [vN28].

Once we consider randomized strategies, we can answer the reachability question with three kinds of affirmative answers. The first kind of answer is the answer SURE:

> Player 1 has a strategy so that for all strategies of player 2, the game, if started in s, always reaches t.

To establish this type of answer, it suffices to consider deterministic strategies only. The second, weaker kind of answer is the answer ALMOST-SURE:

> Player 1 has a strategy so that for all strategies of player 2, the game, if started in s, reaches t with probability 1.

To establish this type of answer, it is necessary to consider randomized strategies, as previously discussed. The third, yet weaker kind of answer is the answer LIMIT-SURE:

> For every real $\varepsilon > 0$, player 1 has a strategy so that for all strategies of player 2, the game, if started in s, reaches t with probability greater than $1 - \varepsilon$.

The three kinds of answers form a proper hierarchy, in the sense that there are cases in which ALMOST-SURE reachability holds whereas SURE reachability does not, and cases in which LIMIT-SURE reachability holds whereas ALMOST-SURE reachability does not hold. Note that the second gap does not appear in reachability problems over Markov chains,

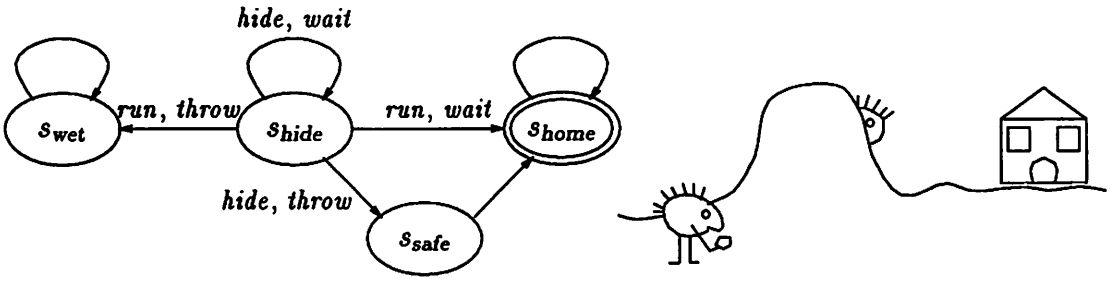

Figure 2: Game HIDE-OR-RUN.

or Markov decision processes [KSK66, BT91]. While the game LEFT-OR-RIGHT witnesses the first gap, the second gap is witnessed by the game HIDE-OR-RUN, adapted from [KS81] and depicted in Figure 2. The target state is s_{home}, and the interesting part of the game happens at state s_{hide}. At this state, player 1 is hiding behind a small hill, while player 2 is trying to hit him with a snowball. Player 1 can choose between hiding or running, and player 2 can choose between waiting and throwing her only snowball. If player 1 runs and player 2 throws the snowball, then player 2 is hit, and the game proceeds to state s_{wet}. If player 1 runs and player 2 waits, then player 1 gets home, and the game proceeds to state s_{home}. If player 1 hides and player 2 throws the snowball, then player 1 is no more in danger, and the game proceeds to state s_{safe}. Finally, if player 1 hides and player 2 waits, the game stays at s_{hide}.

In this game, from state s_{hide} player 1 does not have a strategy (randomized or deterministic) that ensures reaching s_{home} with probability 1: in order to reach home regardless of the strategy of player 2, player 1 may have to take a chance and run while player 2 is still in possession of the snowball. On the other hand, by choosing an appropriate strategy, player 1 can be sure of reaching s_{home} with probability arbitrarily close to 1 [KS81]. In fact, if player 1 runs with very small probability at each round, it becomes very difficult for player 2 to time her snowball to coincide with the running of player 1 —and a badly timed snowball enables player 1 to reach s_{home}. Thus, if player 1 runs at each round with probability p, when p goes to 0, he is able to reach s_{home} with probability approaching 1. Hence, the answer to the reachability question is LIMIT-SURE but not ALMOST-SURE.

In this paper, we consider simultaneous reachability games, and we consider strategies for the players that can be both randomized and history-dependent. The game itself can be either *deterministic,* if the current state and the players' moves uniquely determine the successor state, or *probabilistic,* if the current state and the players' moves determine a probability distribution on the successor state.

The contributions of this paper are as follows. First, we provide efficient algorithms that, given a finite simultaneous game and a set of target states, determine the set of initial states for which the answer to the reachability question is SURE, ALMOST-SURE, and LIMIT-SURE. The set from which the answer is SURE can be determined in linear time using the methods of [AHK97]. By contrast, the sets corresponding to answers ALMOST-SURE and LIMIT-SURE require quadratic time. All three algorithms are formulated as nested fixed-point computations, and can be implemented using symbolic state-space traversal methods [BCM+92]. Our algorithms can also be used to compute the set of states of a finite-

state discrete-time *pursuit-evasion game* from which the pursuer can catch the evader with SURE, ALMOST-SURE, and LIMIT-SURE confidence [Isa65]. Our algorithms also enable the effective construction of winning strategies for player 1, and spoiling strategies for player 2, for the three types of answers.

Polynomial-time algorithms for the reachability problem are known for *turn-based* games [Yan98], as well as for one-player games such as Markov decision processes [HSP83, Var85, CY88, dA97]. Moreover, by associating rewards to the transitions that lead to the target set, or to the states in the target set, our reachability questions can be phrased as questions about the total reward or the average reward of stochastic games. This observation yields successive approximation methods for the computation of the maximal probability of reaching the target [TV87, FV97]. A review of the classes of games for which polynomial-time algorithms are known is presented in Section 2.4.

Second, we characterize the three kinds of reachability in terms of the time (i.e., the number of rounds) required to reach the target state, and in terms of the types of winning and spoiling strategies available to the two players. In particular, while the time to target is bounded if the answer to the reachability question is SURE, only the expected time to target can be bounded if the answer is ALMOST-SURE but not SURE. If the answer is LIMIT-SURE but not ALMOST-SURE, neither the time to target nor the expected time to target are bounded. We also prove prove that the spoiling strategies for ALMOST-SURE reachability must in general have infinite memory, in contrast with the common situation for Markov decision processes [Der70, HSP83, Var85, BT91] and for limit-sure reachability [KS81, Sec97].

Third, we introduce a temporal logic for the specification of open systems, which can be used both for two-agent systems (system vs. environment) and for more general, multi-agent systems. The logic, called *randomized ATL* (RATL), is an extension of the logic ATL of [AHK97]. Both logics let us specify that a set of agents has strategies to ensure that the paths of the global system satisfy given temporal properties. The logic ATL follows the pragmatic approach of considering only deterministic strategies, with the motivation that deterministic controllers are easier to implement than randomized ones. The semantics of ATL is defined on the basis of the SURE answer for reachability questions. The logic RATL considers instead the general class of randomized strategies, and it distinguishes between three kinds of satisfaction for path properties: sure satisfaction (as in ATL), almost-sure satisfaction, and limit-sure satisfaction. The proper hierarchy between the SURE, ALMOST-SURE and LIMIT-SURE answers to the reachability question implies a proper hierarchy for the three kinds of satisfaction. We show that this hierarchy collapses in the special case of safety properties, such as invariance. Our algorithms for solving the reachability question for simultaneous games lead to a symbolic, quadratic-time model-checking algorithm for RATL.

2 Reachability Games

For a finite set A, a *probability distribution* on A is a function $p : A \mapsto [0,1]$ such that $\sum_{a \in A} p(a) = 1$. We denote the set of probability distributions on A by $\mathcal{D}(A)$. Given a distribution $p \in \mathcal{D}(A)$, we denote by $\mathrm{Supp}(p) = \{x \in A \mid p(x) > 0\}$ the *support* of p.

A (two-player) *game structure* $G = \langle S, \mathit{Moves}, \Gamma_1, \Gamma_2, p \rangle$ consists of the following components:

- A finite state space S.
- A finite set *Moves* of moves.
- Two move assignments $\Gamma_1, \Gamma_2 : S \mapsto 2^{\mathit{Moves}} \setminus \emptyset$. For $i \in \{1,2\}$, assignment Γ_i associates with each state $s \in S$ the non-empty set $\Gamma_i(s) \subseteq \mathit{Moves}$ of moves available to player i at state s. For technical convenience, we assume that $\Gamma_i(s) \cap \Gamma_j(t) = \emptyset$ unless $i = j$ and $s = t$, for all $i, j \in \{1,2\}$ and $s, t \in S$.
- A probabilistic transition function $p\colon S \times \mathit{Moves} \times \mathit{Moves} \mapsto \mathcal{D}(S)$, which associates with every state $s \in S$ and moves $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$ a probability distribution $p(s, a_1, a_2) \in \mathcal{D}(S)$ for the successor state.

At every state $s \in S$, player 1 chooses a move $a_1 \in \Gamma_1(s)$, and simultaneously and independently player 2 chooses a move $a_2 \in \Gamma_2(s)$. The game then proceeds to the successor state t with probability $p(s, a_1, a_2)(t)$, for all $t \in S$. For all states $s \in S$ and moves $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$, we indicate by

$$\delta(s, a_1, a_2) = \mathrm{Supp}(p(s, a_1, a_2))$$

the set of possible successors of s when moves a_1, a_2 are selected. A *path* of G is an infinite sequence $\overline{s} = s_0, s_1, s_2, \ldots$ of states in S such that for all $k \geq 0$, there are moves $a_1^k \in \Gamma_1(s_k)$ and $a_2^k \in \Gamma_2(s_k)$ such that $s_{k+1} \in \delta(s_k, a_1^k, a_2^k)$. We denote by Ω the set of all paths.

A *reachability game* (or *game*, for short) $\mathcal{G} = \langle \langle S, \mathit{Moves}, \Gamma_1, \Gamma_2, p \rangle, R \rangle$ consists of a game structure G and a set $R \subseteq S$ of *target states;* the set R itself is called the *target set.* The goal of player 1 in the game $\mathcal{G}$ is to reach a state in the target set R, and the goal of player 2 is to prevent this. Thus, a reachability game is a special case of a *recursive game*, in which all absorbing states are equivalent from the point of view of the reward [Eve57, Sec97]. In the following, we consider a game $\mathcal{G} = \langle \langle S, \mathit{Moves}, \Gamma_1, \Gamma_2, p \rangle, R \rangle$, unless otherwise noted.

To simplify the presentation of the results, we assume that the target set R is *absorbing*; that is, we assume that for every state $s \in R$ and for all moves $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$, we have $\delta(s, a_1, a_2) \subseteq R$. If R is not absorbing, it is trivial to obtain an equivalent game with an absorbing target set.

We define the *size* of the game $\mathcal{G}$ to be equal to the number of entries of the transition function p: specifically,

$$|\mathcal{G}| = \sum_{s \in S} \sum_{a_1 \in \Gamma_1(s)} \sum_{a_2 \in \Gamma_2(s)} |\delta(s, a_1, a_2)| \,.$$

Note that this definition of size assumes that each transition probability can be represented in a constant amount of space. Note also that this definition of size is not affected by our assumption that the moves available to different players or at different states are distinct.

Special classes of reachability games. We distinguish the following subclasses of game structures (and, accordingly, of games):

- A game structure G is *deterministic* if $|\delta(s, a_1, a_2)| = 1$ for all $s \in S$ and all $a_1 \in \Gamma_1(s)$, $a_2 \in \Gamma_2(s)$. To emphasize the fact that in the general case p yields a probability distribution, rather than a single state, we refer to general game structures as *probabilistic* game structures.
- A game structure G is *turn-based* if at every state at most one player can choose among multiple moves; that is, for every state $s \in S$ there exists at most one $i \in \{1,2\}$ with $|\Gamma_i(s)| > 1$.
- A game structure G is *one-player* if one of the two players has only one possible move at every state, i.e. if for some $i \in \{1,2\}$ we have $|\Gamma_i(s)| = 1$ at all $s \in S$.

2.1 Strategies

A *strategy* for player $i \in \{1,2\}$ is a mapping $\pi_i \colon S^+ \mapsto \mathcal{D}(Moves)$ that associates with every nonempty finite sequence $\sigma \in S^+$ of states, representing the past history of the game, a probability distribution $\pi_1(\sigma)$ used to select the next move. Thus, the choice of the next move can be history-dependent and randomized. The strategy π_i can prescribe only moves that are available to player i; that is, for all sequences $\sigma \in S^*$ and states $s \in S$, we require that $\mathrm{Supp}(\pi_i(\sigma s)) \subseteq \Gamma_i(s)$. We denote by Π_i the set of all strategies for player $i \in \{1,2\}$.

Given a state $s \in S$ and two strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, we define $Paths(s, \pi_1, \pi_2) \subseteq \Omega$ to be the set of paths that can be followed by the game, when the game starts from s and the players use the strategies π_1 and π_2. Formally, $s_0, s_1, s_2, \ldots \in Paths(s, \pi_1, \pi_2)$ if $s_0 = s$ and if for all $k \geq 0$ there exist moves $a_1^k \in \Gamma_1(s_k)$ and $a_2^k \in \Gamma_2(s_k)$ such that

$$\pi_1(s_0, \ldots, s_k)(a_1^k) > 0, \qquad \pi_2(s_0, \ldots, s_k)(a_2^k) > 0, \qquad p(s_k, a_1^k, a_2^k)(s_{k+1}) > 0.$$

Once the starting state s and the strategies π_1 and π_2 for the two players have been chosen, the game is reduced to an ordinary stochastic process. Hence, the probabilities of events are uniquely defined, where an *event* $\mathcal{A} \subseteq \Omega$ is a measurable set of paths[1]. For an event $\mathcal{A} \subseteq \Omega$, we denote by $\Pr_s^{\pi_1,\pi_2}(\mathcal{A})$ the probability that a path belongs to $\mathcal{A}$ when the game starts from s and the players use the strategies π_1 and π_2. Similarly, for a measurable function f that associates a number in $\mathbb{R} \cup \{\infty\}$ with each path, we denote by $\mathrm{E}_s^{\pi_1,\pi_2}\{f\}$ the expected value of f when the game starts from s and the strategies π_1 and π_2 are used. For $k \geq 0$, we also let X_k be the random variable denoting the k-th state along a path. Formally, $X_k \colon \Omega \mapsto S$ is the (measurable) function that associates with each path $\overline{s} = s_0, s_1, s_2, \ldots$ the state s_k. Given a subset $U \subseteq S$ of states, we denote the event of reaching U by

$$(\Diamond U) = \{s_0, s_1, s_2, \ldots \in \Omega \mid \exists k \,.\, s_k \in U\}\,,$$

and we denote the random time of first passage in U by $T_{\Diamond U} = \min\{k \mid X_k \in U\}$.

[1] To be precise, we should define events as measurable sets of paths *sharing the same initial state*, and we should replace our events with families of events, indexed by their initial state [KSK66]. However, our (slightly) improper definition leads to more concise notation.

Types of strategies. We distinguish the following types of strategies:

- A strategy π is *deterministic* if for all $\sigma \in S^+$ there exists $a \in \mathit{Moves}$ such that $\pi(\sigma)(a) = 1$. Thus, deterministic strategies are equivalent to functions $S \mapsto \mathit{Moves}$.
- A strategy π is *counting* if $\pi(\sigma_1 s) = \pi(\sigma_2 s)$ for all $s \in S$ and all $\sigma_1, \sigma_2 \in S^*$ with $|\sigma_1| = |\sigma_2|$; that is, the strategy depends only on the current state and the number of past rounds of the game.
- A strategy π is *finite-memory* if the distribution chosen at every state $s \in S$ depends only on s itself, and on a finite number of bits of information about the past history of the game.
- A strategy π is *memoryless* if $\pi(\sigma s) = \pi(s)$ for all $s \in S$ and all $\sigma \in S^+$.

2.2 Classification of Winning States

A *winning state* of game $\mathcal{G}$ is a state from which player 1 can reach the target set R with probability arbitrarily close to 1. We distinguish three classes of winning states: 0

- The class $\mathit{Sure}(R)$ of *sure-reachability states* consists of the states from which player 1 has a strategy to reach R:

$$\mathit{Sure}(R) = \left\{ s \in S \;\middle|\; \exists \pi_1 \in \Pi_1 \,.\, \forall \pi_2 \in \Pi_2 \,.\, \mathit{Paths}(s, \pi_1, \pi_2) \subseteq (\Diamond R) \right\}.$$

- The class $\mathit{Almost}(R)$ of *almost-sure-reachability states* consists of the states from which player 1 has a strategy to reach R with probability 1:

$$\mathit{Almost}(R) = \left\{ s \in S \;\middle|\; \exists \pi_1 \in \Pi_1 \,.\, \forall \pi_2 \in \Pi_2 \,.\, \Pr_s^{\pi_1,\pi_2}(\Diamond R) = 1 \right\}.$$

- The class $\mathit{Limit}(R)$ of *limit-sure-reachability states* consists of the states such that for every real $\epsilon > 0$, player 1 has a strategy to reach R with probability at least $1 - \epsilon$:

$$\mathit{Limit}(R) = \left\{ s \in S \;\middle|\; \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1,\pi_2}(\Diamond R) = 1 \right\}.$$

Clearly, $\mathit{Sure}(R) \subseteq \mathit{Almost}(R) \subseteq \mathit{Limit}(R)$. There are games for which both inclusions are strict. The strictness of the inclusion $\mathit{Sure}(R) \subseteq \mathit{Almost}(R)$ follows from the well-known fact that randomized strategies are more powerful than deterministic strategies [vN28, BO82], and is witnessed by the state t_{throw} of the game LEFT-OR-RIGHT. The strictness of the inclusion $\mathit{Almost}(R) \subseteq \mathit{Limit}(R)$ is witnessed by the state s_{hide} of the game HIDE-OR-RUN [KS81].

Winning and spoiling strategies. The *winning strategies* of a reachability game are the strategies that enable player 1 to win the game whenever possible. We define three types of winning strategies, corresponding to the three classes of winning states:

- A *winning strategy for sure reachability* is a strategy π_1 for player 1 such that, for all states $s \in \mathit{Sure}(R)$ and all strategies π_2 of player 2, we have $\mathit{Paths}(s, \pi_1, \pi_2) \subseteq (\Diamond R)$.
- A *winning strategy for almost-sure reachability* is a strategy π_1 for player 1 such that for all states $s \in \mathit{Almost}(R)$ and all strategies π_2 of player 2, we have $\Pr_s^{\pi_1,\pi_2}(\Diamond R) = 1$.
- A *winning strategy family for limit-sure reachability* is a family $\{\pi_1(\varepsilon) \mid \varepsilon > 0\}$ of strategies for player 1 such that for all reals $\varepsilon > 0$, all states $s \in \mathit{Limit}(R)$, and all strategies π_2 of player 2, we have $\Pr_s^{\pi_1(\varepsilon),\pi_2}(\Diamond R) \geq 1 - \varepsilon$.

The *spoiling strategies* of a reachability game are the strategies that enable player 2 to prevent player 1 from winning the game whenever it cannot be won. Again, we distinguish three types of spoiling strategies:

- A *spoiling strategy for sure reachability* is a strategy π_2 for player 2 such that, for all states $s \notin \mathit{Sure}(R)$ and all strategies π_1 of player 1, we have $\mathit{Paths}(s, \pi_1, \pi_2) \not\subseteq (\Diamond R)$.
- A *spoiling strategy for almost-sure reachability* is a strategy π_2 for player 2 such that for all states $s \notin \mathit{Almost}(R)$ and all strategies π_1 of player 1, we have $\Pr_s^{\pi_1,\pi_2}(\Diamond R) < 1$.
- A *spoiling strategy for limit-sure reachability* is a strategy π_2 for player 2 such that there exists a real $q > 0$ such that for all states $s \notin \mathit{Limit}(R)$ and all strategies π_1 of player 1, we have $\Pr_s^{\pi_1,\pi_2}(\Diamond R) \leq 1 - q$.

We will show that for all three types of reachability, winning and spoiling strategies always exist.

2.3 Time to Reachability

For a state $s \in S$ and an integer $t \geq 0$, we say that the *time from s to target R is bounded by t* if there exists a strategy π_1 for player 1 such that for all strategies π_2 of player 2, $\sup\{T_{\Diamond R}(\overline{s}) \mid \overline{s} \in \mathit{Paths}(s, \pi_1, \pi_2)\} \leq t$. If the time from s to R is not bounded by any integer t, we say that the *time from s to R is unbounded.* We say that the *expected time from s to R is bounded* if there exists a strategy π_1 for player 1 such that for all strategies π_2 of player 2, we have $\mathrm{E}_s^{\pi_1,\pi_2}\{T_{\Diamond R}\} < \infty$. Given a subset $U \subseteq S$ of states, we generalize these definitions to U: the time (or the expected time) to R is bounded from U iff it is bounded from all $s \in U$.

2.4 Previous Results on Reachability Games

Since SURE reachability can be studied by considering deterministic strategies only, the algorithms of [AHK97] enable the computation of the set $\mathit{Sure}(R)$ in linear time in the size of the game. Algorithms to compute the sets $\mathit{Almost}(R)$ and $\mathit{Limit}(R)$ are known

for one-player games (or Markov decision processes) and turn-based games. Moreover, the maximal probability of winning a general reachability game can be computed using successive approximation methods [FV97].

For every state $s \in S$, denote the maximal probability of reaching the target by

$$p^+(s) = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1,\pi_2}(\Diamond R) .$$

Clearly, $s \in \mathit{Limit}(R)$ iff $p^+(s) = 1$. Some of the results and algorithms are obtained through the observation that $p^+(s)$ corresponds to the maximal *average reward* or to the maximal *total reward* of stochastic games constructed from our reachability games. Stochastic games associate with each $s \in S$ and each $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$, a *reward* $r(s, a_1, a_2) \in \mathbb{R}$. For every $s \in S$, the (undiscounted) *average value* $v_{avg}(s)$ and the (undiscounted) *total value* $v_{tot}(s)$ of the game at s are defined by:

$$v_{avg}(s) = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \liminf_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \mathrm{E}_s^{\pi_1,\pi_2}\{r(X_k, Y_k^1, Y_k^2)\} ,$$

$$v_{tot}(s) = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \liminf_{n \to \infty} \sum_{k=0}^{n-1} \mathrm{E}_s^{\pi_1,\pi_2}\{r(X_k, Y_k^1, Y_k^2)\} ,$$

where Y_k^i is the random variable denoting the k-th move played by player i, with $k \geq 0$ and $i \in \{1, 2\}$. Consider a reachability game $\mathcal{G} = \langle G, R \rangle$. If we let

$$r(s, a_1, a_2) = \begin{cases} 1 & \text{if } s \in R \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

for all $s \in S$ and $a_1 \in \Gamma_1(s)$, $a_2 \in \Gamma_2(s)$, then the average value of the stochastic game is equal to the maximal probability of reaching the target, or $v_{avg}(s) = p^+(s)$ for all $s \in S$. If we let

$$r(s, a_1, a_2) = \begin{cases} \sum_{t \in R} p(s, a_1, a_2)(t) & \text{if } s \notin R \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

for all $s \in S$ and $a_1 \in \Gamma_1(s)$, $a_2 \in \Gamma_2(s)$, then on $S \setminus R$ the total value of the stochastic game is equal to the maximal probability of reaching the target, or $v_{tot}(s) = p^+(s)$ for all $s \in S \setminus R$.

2.4.1 One-Player Games

A one-player game is equivalent to a Markov decision process [Der70], in which the controller has as objective to maximize or minimize the probability of reaching the target set R. Under the reward structure (1), the problem of computing the maximum and minimum probability of reaching R is equivalent to the problem of computing the maximum and minimum undiscounted average rewards of the Markov decision process. This latter problem can be solved in polynomial time by a reduction to linear programming, providing an algorithm[2] for the computation of set $\mathit{Limit}(R)$ [Der70, Ber95]. The existence of optimal

[2]It is possible to obtain a polynomial-time algorithm for the computation of set $\mathit{Limit}(R)$ also by considering the reward structure (2) and the problem of computing the total reward of the Markov decision process [Ber95].

strategies for the single player then implies $Almost(R) = Limit(R)$. Additionally, it is known that there are deterministic optimal strategies [Der70, Ber95].

We can obtain more efficient algorithms for our reachability questions on one-player games as follows. If player 1 is the only player having non-singleton move sets, the problem of computing $Almost(R)$ is equivalent to the problem of computing the set of states of a Markov decision process from which R can be reached with maximal probability equal to 1. This problem can be solved using the algorithms described in [dA97]. If player 2 is the only player having non-singleton move sets, the problem of computing $Almost(R)$ is equivalent to the problem of computing the set of states of a Markov decision process from which R is reached with probability 1 under any strategy. This problem can be solved using the algorithms of [HSP83, Var85, CY88].

2.4.2 Turn-based Games

Due to their simpler structure and their ability to model interleaving, turn-based games are commonly considered in computer science and game theory (see for example [Fil81]).

Deterministic turn-based games. As we prove later, for deterministic turn-based games the three types of winning states coincide: that is, $Sure(R) = Almost(R) = Limit(R)$. As mentioned earlier, the problem of computing $Sure(R)$ is equivalent to the AND-OR reachability problem, which can be solved in linear time and is complete for PTIME [Imm81]. The existence of memoryless deterministic winning and spoiling strategies follows from an analysis of the algorithms.

Deterministic turn-based reachability games have "0-1 laws"; that is, for all states $s \in S$ of a turn-based game,

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1,\pi_2}(\Diamond R) \in \{0,1\}. \tag{3}$$

This 0-1 law only applies to deterministic, turn-based games. As an example of a (non-turn-based) deterministic game without a 0-1 law, consider a one-round version of the game LEFT-OR-RIGHT. After the only round, the game moves from the state t_{throw} either to the state t_{hit} or to the state t_{missed}. Then,

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_{t_{throw}}^{\pi_1,\pi_2}(\Diamond\{t_{hit}\}) = \frac{1}{2}.$$

In the case of general reward structures, [ZP96] showed that the average value of a deterministic turn-based game can be computed in pseudo-polynomial time.

Probabilistic turn-based games. In the case of probabilistic turn-based games, our results indicate that $Almost(R) = Limit(R)$. The set $Almost(R) = Limit(R)$ can be computed in polynomial time [Yan98], and the problem of deciding which player has the greatest probability of winning is in NP $\cap$ CO-NP [Con92]. Under the reward structure (1), probabilistic turn-based reachability games are a special case of *switching-controller undiscounted games*. The algorithm of [VTRF83a] enables the computation of $v_{avg}(s) = p^+(s)$ at all $s \in S$, and hence the determination of $Almost(R) = Limit(R)$. The algorithm may require an exponential number of iterations to converge.

	SURE REACHABILITY	ALMOST-SURE REACHABILITY	LIMIT-SURE REACHABILITY
Complexity	linear	quadratic	quadratic
Winning strategies	deterministic and memoryless	memoryless	memoryless
Spoiling strategies	memoryless	counting	memoryless
Time to target	bounded	unbounded	unbounded
Expected time to target	bounded	bounded	unbounded

Table 1: Overview of results about sure, almost-sure, and limit-sure reachability.

2.4.3 General Reachability Games

For general reachability games, the existence of memoryless ε-optimal strategies was shown by [KS81, Sec97]. These results imply the existence of memoryless winning and spoiling strategies for limit-sure reachability. The proofs of existence of these strategies do not provide methods for the effective construction of winning and spoiling strategies.

The total reward $v_{tot}(s)$ of a stochastic game with non-negative rewards can be computed using a successive approximation method [TV87, FV97]. Under the reward structure (2), this method enables the computation of successive approximations for $p^+(s)$ at all $s \in S$. Used jointly with a convergence criterion for establishing whether a sequence of successive approximations $p_0^+(s), p_1^+(s), p_2^+(s), \ldots$ for $p^+(s)$ converges to 1, this method could be used to compute the set $Limit(R)$. Such a convergence criterion has not been studied, to the best of our knowledge.

2.5 Overview of Our Results

In Table 1 we present an overview of the main results on reachability games that are presented in this paper. The first row lists the complexity of the algorithms for computing the sets of winning states with respect to the three types of reachability. The second and the third row list the types of winning and spoiling strategies available to the players. For each type of reachability, we list the tightest class of strategies that surely contains at least one such winning and spoiling strategy (according to the classification of Section 2.1). The last two rows state whether the time to the target, and the expected time to the target, are in general bounded on the sets of winning states. In the paper, we also present several refinements of the results given in the table, corresponding to special classes of games. We also show that, for games that are both deterministic and turn-based, we have

$$Limit(R) = Almost(R) = Sure(R)$$

while for turn-based (but not necessarily deterministic) games we have

$$Limit(R) = Almost(R) \subseteq Sure(R) .$$

3 Computing the Winning States

In this section we present three algorithms for computing, respectively, the three sets *Sure*(R), *Almost*(R), and *Limit*(R). The correctness proofs for the algorithms, as well as the proofs of the theorems presented in this section, will be given in Section 5.

3.1 Building Blocks for the Algorithms

A *move subassignment* γ for player $i \in \{1,2\}$ is a mapping $\gamma: S \mapsto 2^{Moves}$ that associates with each state $s \in S$ a subset $\gamma(s) \subseteq \Gamma_i(s)$ of moves. We use move subassignments to limit the set of moves from which the players can choose when trying to accomplish a goal. We denote by Δ_i the set of all move subassignments for player i.

The function $Pre_1: 2^S \times \Delta_1 \times \Delta_2 \mapsto 2^S$ is defined by

$$Pre_1(U, \gamma_1, \gamma_2) = \left\{ s \in S \;\middle|\; \exists a_1 \in \gamma_1(s) \,.\, \forall a_2 \in \gamma_2(s) \,.\, \delta(s, a_1, a_2) \subseteq U \right\}.$$

Intuitively, $Pre_1(U, \gamma_1, \gamma_2)$ is the set of states from which player 1 can be sure of entering U in one round, regardless of the move chosen by player 2, given that player i chooses moves only according to γ_i, for $i \in \{1,2\}$. The function $Pre_2: 2^S \times \Delta_1 \times \Delta_2 \mapsto 2^S$ is defined in a symmetrical way. The function $Stay_1 : 2^S \times \Delta_1 \times \Delta_2 \mapsto \Delta_1$ is defined such that for all states $s \in S$,

$$Stay_1(U, \gamma_1, \gamma_2)(s) = \left\{ a_1 \in \gamma_1(s) \;\middle|\; \forall a_2 \in \gamma_2(s) \,.\, \delta(s, a_1, a_2) \subseteq U \right\}.$$

Note that if we regard both move subassignments as set of pairs in $S \times Moves$, then $Stay_1(U, \gamma_1, \gamma_2) \subseteq \gamma_1$. Intuitively, $Stay_1(U, \gamma_1, \gamma_2)$ is the largest move subassignment for player 1 that guarantees that the game stays in U for at least one round, regardless of the move chosen by player 2, given that player i chooses moves only according to γ_i, for $i \in \{1,2\}$. The function $Stay_2 : 2^S \times \Delta_1 \times \Delta_2 \mapsto \Delta_1$ is defined in a symmetrical way.

For $i \in \{1,2\}$, the function $Safe_i: 2^S \times \Delta_1 \times \Delta_2 \mapsto 2^S$ associates with each $U \subseteq S$ and each $\gamma_1 \subseteq \Delta_1$, $\gamma_2 \subseteq \Delta_2$ the largest subset $V \subseteq U$ such that $Pre_i(V, \gamma_1, \gamma_2) \subseteq V$. Set V represents the largest subset of U that player i can be sure of not leaving at any time in the future, regardless of the moves chosen by the other player, given that player i chooses moves only according to γ_i, for $i \in \{1,2\}$. This set can be computed in time linear in the size of the game using the following well-known algorithm.

Algorithm 1
Input: Game structure $\mathcal{G}$, subset $U \subseteq S$, two move sub-assignments γ_1 and γ_2 for players 1 and 2, and $i \in \{1,2\}$.
Output: $Safe_i(U, \gamma_1, \gamma_2)$.

> **Initialization:** Let $V_0 = U$.
> **Repeat** For $k \geq 0$, let $V_{k+1} = V_k \cap Pre_i(V_k, \gamma_1, \gamma_2)$.
> **Until** $V_{k+1} = V_k$.
> **Return:** V_k.

A naïve application of this algorithm runs in time quadratic in the size of the game. However, using an appropriate data structure, as suggested in [Bee80, CS91], it can be implemented to run in linear time. The algorithm can also be implemented symbolically as a nested fixed-point iteration.

3.2 Sure-Reachability States

The set $Sure(R)$ satisfies the fixed-point characterization given by the following theorem.

Theorem 1 *$Sure(R)$ is equal to the smallest subset $U \subseteq S$ such that $R \subseteq U$ and $Pre_1(U, \Gamma_1, \Gamma_2) \subseteq U$.*

The set $Sure(R)$ can be computed using the following algorithm.

Algorithm 2
Input: Reachability game $\mathcal{G} = \langle G, R \rangle$.
Output: Sure-reachability set $Sure(R)$.

> **Initialization:** Let $U_0 = R$.
> **Repeat** For $k \geq 0$, let $U_{k+1} = U_k \cup Pre_1(U, \Gamma_1, \Gamma_2)$.
> **Until** $U_{k+1} = U_k$.
> **Return:** U_k.

The algorithm can be implemented to run in time linear in the size of the game [AHK97]. The algorithm can also be implemented symbolically as a fixed-point computation. The theorem below summarizes some basic facts about the set $Sure(R)$.

Theorem 2 *For every reachability game with target set R:*

1. *Algorithm 2 computes set $Sure(R)$. The algorithm can be implemented to run in time linear in the size of the game.*
2. *Player 1 has a memoryless deterministic winning strategy for sure reachability.*
3. *Player 2 has a memoryless spoiling strategy for sure reachability. This spoiling strategy cannot in general be deterministic.*
4. *For every state $s \in Sure(R)$, the time from s to R is bounded by the size of the state space.*

Theorem 2(2) indicates that the consideration of deterministic strategies only is appropriate for the logic ATL, whose semantics is based on sure reachability [AHK97]. For deterministic games, the existence of a memoryless deterministic winning strategy for almost-sure or limit-sure reachability indicates that these two notions of reachability coincide with sure reachability. This result can be interpreted as a converse of Theorem 2(2).

Theorem 3 *Consider a deterministic reachability game with target set R.*

1. *If player 1 has a memoryless deterministic strategy π for almost-sure reachability, then $Sure(R) = Almost(R)$, and π is also a winning strategy for sure reachability.*
2. *If player 1 has a family of deterministic winning strategies for limit-sure reachability, then $Sure(R) = Limit(R) = Almost(R)$.*

If the game is both deterministic and turn-based, then it is possible to strengthen Theorem 2(3), obtaining the 0-1 law (3).

Theorem 4 *If a reachability game with target set R is both deterministic and turn-based, then player 2 has a deterministic spoiling strategy π_2 such that $\Pr_s^{\pi_1,\pi_2}(\Diamond R) = 0$ for all strategies $\pi_1 \in \Pi_1$ for player 1 and all states $s \notin Sure(R)$.*

As an immediate corollary, we obtain the equivalence of the three reachability criteria for deterministic turn-based games.

Corollary 1 *If a reachability game with target set R is both deterministic and turn-based, then $Sure(R) = Almost(R) = Limit(R)$.*

The following theorem provides us with winning and spoiling strategies for sure reachability.

Theorem 5 *Given a reachability game $\mathcal{G} = \langle G, R \rangle$, we can compute winning and spoiling strategies for sure reachability as follows:*

1. *Assume that Algorithm 2 terminates at iteration m, and let $U_0, \ldots, U_m$ be the sets of states computed during the execution of the algorithm.*

   Define $h : U_m \setminus R \mapsto \mathbb{N}$ by $h(s) = \min\{j \in \{1, \ldots, m\} \mid s \in U_j\}$ for each $s \in U_m \setminus R$, and define $\gamma : U_m \setminus R \mapsto Moves$ such that, for all states $s \in U_m \setminus R$, we have

   $$\gamma(s) = Stay_1(U_{h(s)-1}, \Gamma_1, \Gamma_2)(s) .$$

   Let π_1^* be a memoryless deterministic strategy for player 1 that at all $s \in U_m \setminus R$ deterministically chooses a move from $\gamma(s)$ (note that $\gamma(s) \neq \emptyset$). At other states, π_1^* is defined arbitrarily. Then, π_1^* is a winning strategy for sure reachability.

2. *Let π_2^* be the memoryless strategy for player 2 that at every $s \in S$ chooses a move from $\Gamma_2(s)$ uniformly at random. Then, π_2^* is a spoiling strategy for sure reachability.*

3.3 Almost-Sure-Reachability States

Given a subset $U \subseteq S$ of states, denote by $\theta_1^U = Stay_1(U, \Gamma_1, \Gamma_2)$ the move sub-assignment for player 1 that guarantees remaining in U for one round (note that it may be $\theta_1^U(s) = \emptyset$ for some $s \in S$). The set $Almost(R)$ satisfies the fixed-point characterization given by the following theorem.

Theorem 6 *Almost(R) is equal to the largest subset $U \subseteq S$ such that:*

$$Safe_1(U, \Gamma_1, \Gamma_2) = U \ , \qquad Safe_2(U \setminus R, \theta_1^U, \Gamma_2) = \emptyset \ . \tag{4}$$

The set *Almost*(R) can be computed using the following algorithm. The algorithm has running time quadratic in the size of the game, and it can be implemented symbolically as a nested fixed-point computation.

Algorithm 3
Input: Reachability game $\mathcal{G} = \langle G, R \rangle$.
Output: Almost-sure-reachability set *Almost*(R).

Initialization: Let $U_0 = S$, $\gamma_0 = \Gamma_1$.
Repeat For $k \geq 0$, let

$$\begin{aligned} C_k &= Safe_2(U_k \setminus R, \gamma_k, \Gamma_2), \\ U_{k+1} &= Safe_1(U_k \setminus C_k, \gamma_k, \Gamma_2), \\ \gamma_{k+1} &= Stay_1(U_{k+1}, \gamma_k, \Gamma_2) \ . \end{aligned}$$

Until $U_{k+1} = U_k$.
Return: U_k.

The algorithm can be understood as follows. The set C_0 is the largest subset of $S \setminus R$ to which player 2 can confine the game. Player 1 must avoid entering C_0 at all costs: if C_0 is entered with positive probability, R will not be reached with probability 1. The set U_1 is the largest set of states from which player 1 can avoid entering C_0. The move subassignment γ_1 then associates with each state the set of moves that player 1 can select in order to avoid leaving U_1. Since $\gamma_1 \subseteq \Gamma_1$, by choosing only moves from γ_1, player 1 may lose some of the ability to resist confinement. The set C_1 is the largest subset of $U_1 \setminus R$ to which player 2 can confine the game, under the assumption that player 1 uses only moves from γ_1. The set U_2 is then the largest subset of U_1 from which player 1 can avoid entering C_1, and the subassignment $\gamma_2 \subseteq \gamma_1$ guarantees that player 1 never leaves U_2. The computation of C_k, U_{k+1}, and γ_{k+1}, for $k \geq 0$, continues in this way, until we reach $m > 0$ such that:

- if player 1 chooses moves only from γ_m, the game will never leave U_m;
- player 2 cannot confine the game to $U_m \setminus R$, even if player 1 chooses moves only from γ_m.

At this point, we have $U_m = Almost(R)$.

Theorem 7 *For every reachability game with target set R:*

1. *Algorithm 3 computes the set Almost(R). The algorithm can be implemented to run in time quadratic in the size of the game.*

2. *(a) Player 1 has a memoryless winning strategy for almost-sure reachability.*
   (b) This winning strategy cannot in general be deterministic.
3. *(a) Player 2 has a counting spoiling strategy for almost-sure reachability.*
   (b) This spoiling strategy cannot in general be deterministic, nor finite-memory.
4. *For every state $s \in Almost(R)$, the expected time from s to target R is bounded.*

Results 1, 2, and 3a follow from the correctness proof of of Algorithm 3, given in Section 5.2. Result 3b is proved by an analysis of the game HIDE-OR-RUN, considering the strategies available to the players at the state $s_{hide} \notin Almost(R)$. Result 4 then follows from result 2a, and from results about the *stochastic shortest-path problem* [BT91]. Note also that

- For every state $s \notin Sure(R)$, the time to R is unbounded, since not all paths reach R.
- For every state $s \notin Almost(R)$, the expected time to R is unbounded, since R is reached with probability always smaller than 1.

If the game is turn-based, then by analyzing the spoiling strategies for player 2 we can prove that $Almost(R) = Limit(R)$. Moreover, in turn-based games deterministic strategies are as powerful as randomized ones.

Theorem 8 *If a reachability game with target set R is turn-based, then:*

1. *$Almost(R) = Limit(R)$.*
2. *There is a memoryless and deterministic strategy that is winning for both almost-sure and limit-sure reachability, and there is a memoryless and deterministic strategy that is spoiling for both almost-sure and limit-sure reachability.*

The following theorem provides us with winning strategies for almost-sure reachability. The construction of spoiling strategies for almost-sure reachability is more involved, and is presented in Section 5.2.

Theorem 9 *Assume that Algorithm 3 terminates at iteration m, and let $U_0, \ldots, U_m$ and $\gamma_1, \ldots, \gamma_m$ be the sequences of sets and move sub-assignments computed by the algorithm. Let π_1^* be the memoryless strategy for player 1 that at each state $s \in U_m$ chooses uniformly at random a move in $\gamma_m(s)$, and at each state $s \in S \setminus U_m$ is defined arbitrarily. Then π_1^* is a winning strategy for almost-sure reachability.*

3.4 Limit-Sure-Reachability States

In this section we describe an algorithm for the computation of limit-sure reachability states. Given a reachability game, both Algorithm 3 for almost-sure reachability and the algorithm for limit-sure reachability iteratively compute two sequences of sets C_0, C_1, ..., C_m and U_0, U_1, ..., U_m. The difference between the two algorithms lies in the way the sets C_k are computed, for $0 \leq k \leq m$: in Algorithm 3 for almost-sure reachability, these sets are computed with respect to *safe escape;* in the algorithm for limit-sure reachability, they are computed with respect to *limit escape.*

3.4.1 Safe Escape

To illustrate the concept of safe escape, assume that Algorithm 3 terminates at iteration m, after computing the sets $C_0, C_1, \ldots, C_m$ and $U_0, U_1, \ldots, U_m$. Each set C_k, for $0 \leq k \leq m$, is computed in two steps. First, the algorithm computes the sub-assignment

$$\gamma_k = \theta_1^{U_k} = Stay_1(U_k, \Gamma_1, \Gamma_2)\,,$$

consisting of all the moves that enable player 1 to remain in U_k for one round. Then, to compute

$$C_k = Safe_2(U_k \setminus R, \theta_1^{U_k}, \Gamma_2) \tag{5}$$

the algorithm sets $V_0 = U_k \setminus R$, and for $j \geq 0$, it iteratively removes from V_j all the states $s \in V_j$ such that

$$s \notin Pre_2(V_j, \gamma_k, \Gamma_2)\,. \tag{6}$$

If (6) holds, so that state s is removed, it means that player 2 has no single move at s that can keep the game in V_k for all moves in $\gamma_k(s)$ of player 1. Hence, if player 1 plays at s all moves of $\gamma_k(s)$ uniformly at random, he can leave V_j with positive probability, regardless of the move chosen by player 2. Moreover, the escape from V_k is *safe:* it involves no risk of leaving U_k, since it is achieved using only the moves in γ_k. We say that a state s as above is *safe-escape* with respect to V_j and U_k.

We now define safe-escape states formally. Given a state s, two probability distributions $\xi_1 \in \mathcal{D}(\Gamma_1(s))$ and $\xi_2 \in \mathcal{D}(\Gamma_2(s))$, and a subset V of states, indicate by $\widetilde{p}(s, \xi_1, \xi_2)(V)$ the one-round probability of going from s to V when players 1 and 2 select the moves according to distributions ξ_1 and ξ_2, respectively. This probability can be computed as

$$\widetilde{p}(s, \xi_1, \xi_2)(V) = \sum_{a_1 \in \Gamma_1(s)} \sum_{a_2 \in \Gamma_2(s)} \sum_{t \in V} \Big[\xi_1(a_1)\, \xi_2(a_2)\, p(s, a_1, a_2)(t)\Big]\,.$$

Given two subsets of states C and U such that $C \subseteq U$ and a state $s \in C$, we say that s is *safe-escape* with respect to C and U iff there is a distribution $\xi_1 \in \mathcal{D}(\Gamma_1(s))$ such that:

$$\inf_{\xi_2 \in \mathcal{D}(\Gamma_2(s))} \widetilde{p}(s, \xi_1, \xi_2)(S \setminus C) > 0 \tag{7}$$

$$\sup_{\xi_2 \in \mathcal{D}(\Gamma_2(s))} \widetilde{p}(s, \xi_1, \xi_2)(S \setminus U) = 0\,. \tag{8}$$

If we think of C as the set from which we must escape, and of $S \setminus U$ as a set where capture occurs, then *safe-escape* states are the ones from which it is possible to escape with positive one-round probability (bounded away from 0), while incurring no risk of capture. From (7) and (8) we can check that s is safe-escape with respect to C and U iff

$$s \notin Pre_2(C, \theta_1^U, \Gamma_2)\,. \tag{9}$$

From this characterization of safe-escape states, by comparison between (6) and (9) we see that for each $1 \leq k \leq m$, the set C_k computed in (5) is the largest subset of $U_k \setminus R$ that does not contain any safe-escape state with respect to C_k and U_k.

3.4.2 Limit Escape

Safe escape is at the basis of the algorithm for almost-sure reachability because, in order to reach the target with probability 1, no risk, however small, can be taken. On the other hand, if the goal is to reach the target with probability arbitrarily close to 1, as is the case for limit reachability, then a small amount of risk of capture can be tolerated, provided the ratio between the one-round probabilities of escape and capture can be made arbitrarily high. We call this type of escape *limit* escape.

Before discussing limit escape in general, let us consider the situation of state s_{hide} of game HIDE-OR-RUN. As we mentioned in the introduction, $s_{hide} \in \mathit{Limit}(R) \setminus \mathit{Almost}(R)$, where $R = \{s_{home}\}$ [KS81]. If we consider the execution of Algorithm 3 on game HIDE-OR-RUN, we see that $C_0 = \{s_{wet}\}$, $C_1 = \{s_{hide}\}$, and $U_1 = \{s_{hide}, s_{safe}, s_{home}\}$. While player 1 cannot escape from C_0, he can escape from C_1 and reach s_{home} with arbitrarily high probability by being "patient enough" and playing move *run* with sufficiently low probability at each round. Precisely, for every $0 < \varepsilon < 1$, define the distribution $\xi_1[\varepsilon] \in \mathcal{D}(\Gamma_1(s))$ by:

$$\xi_1[\varepsilon](run) = \varepsilon, \qquad \xi_1[\varepsilon](hide) = 1 - \varepsilon \,. \tag{10}$$

By using distribution $\xi_1[\varepsilon]$ and letting $\varepsilon \to 0$, player 1 can make the ratio between the probability of escape from C_1 and the probability of capture in $S \setminus U_1$ diverge: in fact,

$$\begin{aligned} \lim_{\varepsilon \to 0} \inf_{\xi_2 \in \mathcal{D}(\Gamma_2(s))} \frac{\widetilde{p}(s, \xi_1[\varepsilon], \xi_2)(S \setminus C_1)}{\widetilde{p}(s, \xi_1[\varepsilon], \xi_2)(S \setminus U_1)} &= \lim_{\varepsilon \to 0} \inf_{0 \le q \le 1} \frac{\varepsilon(1-q) + (1-\varepsilon)q}{\varepsilon q} \\ &= \lim_{\varepsilon \to 0} \frac{1-\varepsilon}{\varepsilon} = \infty \,. \end{aligned} \tag{11}$$

The divergence of the ratio between the one-round probability of escape and the one-round probability of capture enables player 1 to eventually escape with probability arbitrarily close to 1. To verify this, let $\pi_1[\varepsilon]$ be the memoryless strategy for player 1 that uses distribution $\xi_1[\varepsilon]$ at state s_{hide}. Once $\pi_1[\varepsilon]$ is fixed, results on Markov decision processes ensure that the optimal strategy for player 2 to avoid reaching R is memoryless (and also deterministic) [Der70, Ber95]. Hence, simple calculations show that [KS81]:

$$\inf_{\pi_2 \in \Pi_2} \Pr_{s_{hide}}^{\pi_1[\varepsilon], \pi_2}(\Diamond\{s_{home}\}) = 1 - \varepsilon \,,$$

so that

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_{s_{hide}}^{\pi_1[\varepsilon], \pi_2}(\Diamond\{s_{home}\}) = \lim_{\varepsilon \to 0}(1 - \varepsilon) = 1 \,.$$

In the general case, limit escape is defined as follows. Consider two sets of states C and U such that $C \subseteq U$, and a state $s \in C$. We say that s is *limit-escape* with respect to C and U iff

$$\sup_{\xi_1 \in \mathcal{D}(\Gamma_1(s))} \inf_{\xi_2 \in \mathcal{D}(\Gamma_2(s))} \frac{\widetilde{p}(s, \xi_1, \xi_2)(S \setminus C)}{\widetilde{p}(s, \xi_1, \xi_2)(S \setminus U)} = \infty \,. \tag{12}$$

Comparing this definition with (11), we see that state s_{hide} is limit-escape with respect to $C_1 = \{s_{hide}\}$ and $U_1 = \{s_{hide}, s_{safe}, s_{home}\}$.

The key idea to obtain an algorithm for limit-sure reachability is to replace safe escape with limit escape in the computation of the various sets C_k, for $k \geq 0$. In the algorithm for limit-sure reachability, for each $k \geq 0$ we compute C_k as the largest subset of $U_k \setminus R$ that does not contain any *limit-escape* state with respect to C_k and U_k. This intuition will be justified by the correctness proof for the algorithm, presented in Section 5.3.

3.4.3 Computing Limit-Escape States

The following lemma provides an alternative characterization of limit-escape states, which leads to an algorithm for their determination.

Lemma 1 *Given a state s and two sets of states C and U, with $s \in C \subseteq U$, let*

$$E_1 = \Big\{(a,b) \in \Gamma_1(s) \times \Gamma_2(s) \;\Big|\; \delta(s,a,b) \not\subseteq C\Big\}, \tag{13}$$

$$E_2 = \Big\{(b,a) \in \Gamma_2(s) \times \Gamma_1(s) \;\Big|\; \delta(s,a,b) \not\subseteq U\Big\}, \tag{14}$$

and let $\mathcal{A} \subseteq \Gamma_1(s)$ and $\mathcal{B} \subseteq \Gamma_2(s)$ be the least sets such that:

1. *for all $a \in \Gamma_1(s)$, if $\{b \mid (b,a) \in E_2\} \subseteq \mathcal{B}$, then $a \in \mathcal{A}$;*
2. *for all $b \in \Gamma_2(s)$, if there is $a \in \mathcal{A}$ with $(a,b) \in E_1$, then $b \in \mathcal{B}$.*

Then, s is limit-escape with respect to C and U iff $\mathcal{B} = \Gamma_2(s)$.

From the lemma, we obtain the following algorithm for the determination of limit-escape states.

Algorithm 4 (test for limit-escape states)
Input: Game structure G, two sets $C \subseteq U \subseteq S$ of states, and a state $s \in C$.
Output: YES if s is limit-escape with respect to C and U, NO otherwise.

> **Initialization:** Let $\mathcal{B}_{-1} = \emptyset$, and let E_1 and E_2 be defined as in (13) and (14).
> **Repeat** For $k \geq 0$, let
>
> $$\mathcal{A}_k = \Big\{a \in \Gamma_1(s) \;\Big|\; \forall b \in \Gamma_2(s) \,.\, \text{if } (b,a) \in E_2 \text{ then } b \in \mathcal{B}_{k-1}\Big\},$$
>
> $$\mathcal{B}_k = \Big\{b \in \Gamma_2(s) \;\Big|\; \exists a \in \mathcal{A}_k \,.\, (a,b) \in E_1\Big\}.$$
>
> **Until** $\mathcal{A}_{k+1} = \mathcal{A}_k$ and $\mathcal{B}_{k+1} = \mathcal{B}_k$.
> **Return:** YES if $\mathcal{B}_k = \Gamma_2(s)$, NO otherwise.

If the above algorithm returns an affirmative answer with input s, C, and U, we write *lim-esc*(s,C,U) = YES; similarly, we write *lim-esc*(s,C,U) = NO in case of negative answer. The algorithm, and the lemma, can be understood as follows. First, we construct a bipartite graph with sets of vertices $\Gamma_1(s)$ and $\Gamma_2(s)$ and sets of edges E_1 and E_2. The sets of vertices correspond to the moves available to players 1 and 2 at s. There is an edge in E_1 from

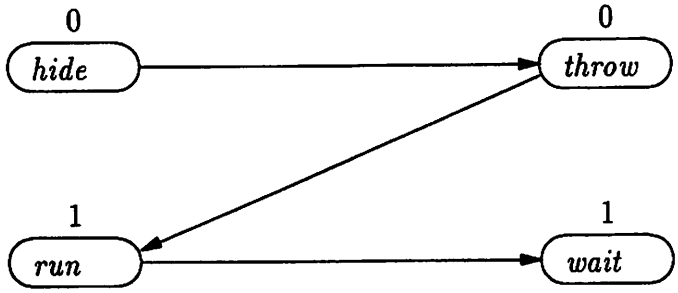

Figure 3: Bipartite graph generated by Algorithm 4 for state s_{hide} of game HIDE-OR-RUN, with respect to $C = \{s_{hide}\}$ and $U = \{s_{hide}, s_{safe}, s_{home}\}$. The labels $\ell(\cdot)$ of the moves are written above the corresponding vertices.

$a \in \Gamma_1(s)$ to $b \in \Gamma_2(s)$ if a, b played together lead to an escape from C with positive probability; there is an edge in E_2 from $b \in \Gamma_2(s)$ to $a \in \Gamma_1(s)$ if a, b played together lead outside U, i.e. to capture, with positive probability. The graph corresponding to state s_{hide} of game HIDE-OR-RUN, and sets $C = \{s_{hide}\}$, $U = \{s_{hide}, s_{safe}, s_{home}\}$ is depicted in Figure 3.

Once the graph is constructed, we let $\mathcal{A}_0 \subseteq \Gamma_1(s)$ be the set of moves for player 1 that are safe with respect to capture, i.e. that lead inside U regardless of the move played by player 2. We let $\mathcal{B}_0$ be the set of moves for player 2 that, if played together with some move in $\mathcal{A}_0$, enable the escape from C with non-zero one-round probability (and zero risk of capture). From this, we see by comparison with (6) and (9) that s is safe-escape with respect to C and U iff $\mathcal{B}_0 = \Gamma_2(s)$: we will later return to this point. The construction of the sequences of sets $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \ldots$ and $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \ldots$ continues then as follows. At round $i > 0$, we let $\mathcal{A}_i \subseteq \Gamma_1(s)$ be the set of moves for player 1 whose incoming edges all originate from $\mathcal{B}_{i-1}$. We then let $\mathcal{B}_i \subseteq \Gamma_2(s)$ be the set of moves for player 2 that have at least one incoming edge originating from $\mathcal{A}_i$. The construction continues until, for some $k \geq 0$, no more moves can be added to $\mathcal{A}_k$ and $\mathcal{B}_k$.

We say that a move $a \in \Gamma_1(s)$ has been *labeled* if $a \in \mathcal{A}_k$; if a has been labeled we define $\ell(a) = \min\{i \mid a \in \mathcal{A}_i\}$ to be its *label*. Similarly, $b \in \Gamma_2(s)$ has been labeled if $b \in \mathcal{B}_k$, in which case its label is $\ell(b) = \min\{i \mid b \in \mathcal{B}_i\}$. The algorithm declares state s limit-escape w.r.t. U and C iff all the moves $\Gamma_2(s)$ for player 2 at s have been labeled. The labeled graph for state s_{hide} of game HIDE-OR-RUN is depicted in Figure 3.

To understand why the algorithm is correct, assume first that s is declared limit-escape. By definition, this means that all moves of player 2 at s have been labeled, implying that also all moves of player 1 have been labeled. The labels of the moves for player 1 provide us with an ε-indexed family of distributions that make the ratio (12) diverge. Given $0 < \varepsilon < 1/(2|\Gamma_1(s)|)$, let $\xi_1[\varepsilon]$ be the distribution that plays move $a \in \Gamma_1(s)$ with probability $\varepsilon^{\ell(a)}$ if $\ell(a) > 0$, and that plays all the moves in $\{a \in \Gamma_1(s) \mid \ell(a) = 0\}$ uniformly at random with the remaining probability. From Figure 3, we see that the distribution constructed in this fashion for state s_{hide} of game HIDE-OR-RUN coincides with the one given in (10). To see

that (12) holds, we show that

$$\lim_{\varepsilon \to 0} \inf_{\xi_2 \in \mathcal{D}(\Gamma_2(s))} \frac{\tilde{p}(s, \xi_1[\varepsilon], \xi_2)(S \setminus C)}{\tilde{p}(s, \xi_1[\varepsilon], \xi_2)(S \setminus U)} = \infty \,. \tag{15}$$

In fact, consider any move $b \in \Gamma_2(s)$ for player 2. Since b is labeled, there is a move $a \in \Gamma_1(s)$ labeled with $\ell(a) = \ell(b)$ with an edge from a to b. Hence, playing b will cause to leave C with one-round probability at least proportional to the probability with which a is played, or $\varepsilon^{\ell(b)}$. On the other hand, all the moves a that played together with b leave U have a label strictly greater than $\ell(b)$, since there is an edge from b to all these moves. Hence, the one-round probability of leaving U will be proportional at most to $|\Gamma_1(s)|\varepsilon^{\ell(b)+1}$. Since this reasoning can be repeated for all the moves of player 2 at s, the ratio between the one round probabilities of leaving C and of leaving U diverges as $\varepsilon \to 0$, and (15) holds (a rigorous proof is presented in Section 5.3).

Conversely, assume that that s is not declared limit-escape. This implies that some of the moves of player 2 at s have not been labeled. To see that in this case (12) does not hold, suppose that player 2 plays the unlabeled moves in $\Gamma_2(s)$ uniformly at random. For each move $a \in \Gamma_1(s)$ of player 1, there are two cases, depending on whether a has been labeled or not:

- If a has been labeled, then playing a will keep the game in C: in fact, if a move b of player 2 leads outside of C when played with a, then $(a, b) \in E_1$, so that b is labeled and hence not played. Thus, (12) will not hold.
- If a has not been labeled, then there must be an unlabeled move $b \in \Gamma_2(s)$ with an edge from a to b (or else a would have been labeled). Since b is played with constant probability, the one-round probability of leaving U is proportional to the probability of playing a; and of course the one-round probability of leaving C is either 0, or proportional to the probability of playing a. In either case, we see that the ratio between the probability of leaving C and that of leaving U cannot diverge, so that (12) will again not hold.

The correctness of the algorithm then implies that of the lemma.

As we remarked earlier, the method (9) for the determination of safe-escape states is equivalent to executing only the first round of Algorithm 4, and checking whether $\mathcal{B}_0 = \Gamma_2(s)$. Executing only the first round corresponds to computing only $\mathcal{A}_0$ and $\mathcal{B}_0$, and using only the label 0. This equivalence is not a coincidence. For safe escape, player 1 must keep the probability of risk equal to 0. Thus, playing moves in $\Gamma_1(s)$ with probability that tends to 0 is not useful to player 1: either a move incurs no risk, and it can be played at will, or it incurs some risk, and it cannot be played at all. Hence, to establish whether a state is safe-escape, player 1 does not need to consider distributions that play moves with probability ε^i with $i > 0$ as $\varepsilon \to 0$, and only the exponent 0 for ε must be considered.

3.4.4 Computing Limit-Sure Reachability States

Given two subsets of states W, U with $W \subseteq U$, we denote by *Lim-safe*(W, U) the largest subset $V \subseteq W$ that does not contain any limit-escape state with respect to V and U. This set can be computed with the following algorithm.

Algorithm 5
Input: Game structure G, and two sets $W \subseteq U \subseteq S$ of states.
Output: *Lim-safe*$(W, U) \subseteq S$.

Initialization: Let $V_0 = W$.
Repeat For $k \geq 0$, let $V_{k+1} = \{s \in V_k \mid s$ not limit-escape w.r.t. V_k and $U\}$.
Until $V_{k+1} = V_k$.
Return: V_k.

As mentioned above, the set *Limit*(R) satisfies the fixed-point characterization given by the following theorem.

Theorem 10 *Limit*(R) *is equal to the largest subset* $U \subseteq S$ *such that*

$$Safe_1(U, \Gamma_1, \Gamma_2) = U \qquad Lim\text{-}safe(U \setminus R, U) = \emptyset\,. \tag{16}$$

The set *Limit*(R) can be computed using the following algorithm, obtained from Algorithm 3 by replacing safe escape with limit escape in the computation of the sets C_k, for $k \geq 0$.

Algorithm 6
Input: Reachability game $\mathcal{G} = \langle G, R \rangle$.
Output: Limit-sure-reachability set *Almost*(R).

Initialization: Let $U_0 = S$, $\gamma_0 = \Gamma_1$.
Repeat For $k \geq 0$, let

$$C_k = Lim\text{-}safe(U_k \setminus R, U_k),$$
$$U_{k+1} = Safe_1(U_k \setminus C_k, \Gamma_1, \Gamma_2).$$

Until $U_{k+1} = U_k$.
Return: U_k.

For example, in the game HIDE-OR-RUN Algorithm 6 computes $C_0 = \{s_{wet}\}$, $U_1 = \{s_{hide}, s_{safe}, s_{home}\}$, $C_1 = \emptyset$, and finally, *Limit*$(R) = U_2 = U_1 = \{s_{hide}, s_{safe}, s_{home}\}$, in agreement with our previous analysis of the game.

Theorem 11 *For every reachability game with target set* R*:*

1. *Algorithm 6 computes set Limit*(R)*. The algorithm can be implemented to run in time quadratic in the size of the game.*
2. *Player 1 has a family of memoryless winning strategies for limit-sure reachability. These winning strategies cannot in general be deterministic.*
3. *Player 2 has a memoryless spoiling strategy for limit-sure reachability. This spoiling strategy cannot in general be deterministic.*

Result 1 is proved through a detailed analysis of Algorithms 4, 5, and 6. In particular, to obtain a version of the algorithm that runs in quadratic time it is necessary to optimize the implementation of Algorithm 5. The optimized version is given as Algorithm 8 of Section 5.3.

Results 2 and 3 are from [KS81]. However, while previous results were concerned only with the existence of particular types of winning and spoiling strategies [Eve57, KS81, Sec97], our algorithms provide methods for the effective computation of such strategies. These methods are presented in Theorems 15 and 16 of Section 5.3.

4 Randomized ATL

For the specification and verification of open systems, [AHK97] introduced the temporal logic ATL. The logic ATL is interpreted over multi-player game structures, and includes formulas of the form $\langle\langle A\rangle\rangle\psi$, which asserts that a team A of players (called *agents*) have a strategy to ensure that all paths of the game satisfy the temporal property ψ. The semantics of the logic ATL is defined with respect to sure reachability, For example, in a two-player game structure, if φ_R is a formula that is true exactly for all states in R, then the ATL formula $\langle\langle\text{Player1}\rangle\rangle\Diamond\varphi_R$ is true exactly in the sure-reachability states. The definition of semantics of ATL formulas considers only deterministic strategies for player 1:[3] from Theorem 2(2), we see that this suffices for the analysis of sure reachability.

In this section, we introduce the logic *randomized ATL* (RATL). The logic RATL is defined with respect to randomized strategies, and distinguishes between three kinds of satisfaction for path properties: sure satisfaction, almost-sure satisfaction, and limit-sure satisfaction; correspondingly, the single quantifier $\langle\langle\ \rangle\rangle$ of ATL is replaced by the three quantifiers $\langle\langle\ \rangle\rangle_{sure}$, $\langle\langle\ \rangle\rangle_{almost}$, and $\langle\langle\ \rangle\rangle_{limit}$. The RATL formulas $\langle\langle A\rangle\rangle_{sure}\varphi$ (resp. $\langle\langle A\rangle\rangle_{almost}\varphi$ or $\langle\langle A\rangle\rangle_{limit}\varphi$) asserts that the set A of agents have strategies to ensure that the specification φ holds surely (resp. almost-surely or limit-surely). For example, the formula $\langle\langle\text{Player1}\rangle\rangle_{almost}\Diamond\varphi_R$ will be true exactly in the almost-sure-reachability states.

4.1 Systems

Formulas of randomized ATL are interpreted over systems with multiple agents. A *system* $\mathcal{S} = \langle n, S, Moves, \Gamma, p, \mathcal{Q}, L\rangle$ consists of the following components:

- A number $n > 0$ of agents.
- A finite state space S.
- A finite set *Moves* of moves.
- A move assignment $\Gamma\colon S \times \{1,\ldots,n\} \mapsto 2^{Moves} \setminus \emptyset$ that associates with each agent $i \in \{1,\ldots,n\}$ and each state $s \in S$ the set $\Gamma(s,i) \subseteq Moves$ of moves available to agent i at s. Again, for technical convenience we assume that $\Gamma(s,i) \cap \Gamma(t,j) = \emptyset$ unless $i = j$ and $s = t$, for all $i,j \in \{1,\ldots,n\}$ and $s,t \in S$.

[3] In ATL, the behavior of the second player is not defined using the notion of strategy.

- A transition probability function $p\colon S \times \mathit{Moves}^n \mapsto \mathcal{D}(S)$, that associates with each $s \in S$ and each list of moves $a_1 \in \Gamma(s,1), \ldots, a_n \in \Gamma(s,n)$ a probability distribution $p(s,a_1,\ldots,a_n) \in \mathcal{D}(S)$ for the successor state.
- A finite set $\mathcal{Q}$ of propositions.
- A function $L\colon S \mapsto 2^{\mathcal{Q}}$ that labels each state with the propositions that are true in the state.

Thus, a system with n agents is a labeled n-player game structure: at every state $s \in S$, each agent $i \in \{1,\ldots,n\}$ chooses a move $a_i \in \Gamma(s,i)$, and the game proceeds to a successor state chosen according to distribution $p(s,a_1,\ldots,a_n)$. Typically, the agents model individual processes or components or the environment of a reactive program, and the game structure is deterministic.

The *paths* of $\mathcal{S}$ are defined in analogy to two-player game structures. A *strategy* π_A for a (possibly empty) set $A = \{i_1,\ldots,i_k\} \subseteq \{1,\ldots,n\}$ of agents is a mapping $\pi_A\colon S^+ \mapsto \mathcal{D}(\mathit{Moves}^k)$ such that $\pi_A(\sigma s)(a_1,\ldots,a_k) > 0$ implies $a_j \in \Gamma(s,i_j)$ for all $1 \le j \le k$. Given a set A of agents, we denote by Π_A the set of strategies for A, and we let $B = \{1,\ldots,n\} \setminus A$.

4.2 Randomized ATL Syntax and Semantics

Syntax. The temporal logic RATL is defined with respect to a set $\mathcal{Q}$ of propositions and a set $\{1,\ldots,n\}$ of agents. A randomized ATL formula is one of the following:

- q, for propositions $q \in \mathcal{Q}$.
- $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1 and φ_2 are randomized ATL formulas.
- $\langle\!\langle A \rangle\!\rangle_{\mathit{win}} \bigcirc\varphi$ or $\langle\!\langle A \rangle\!\rangle_{\mathit{win}} \Box\varphi$ or $\langle\!\langle A \rangle\!\rangle_{\mathit{win}} \varphi_1 \mathcal{U} \varphi_2$, where $A \subseteq \{1,\ldots,n\}$ is a set of agents, $\mathit{win} \in \{\mathit{sure}, \mathit{almost}, \mathit{limit}\}$ is a type of winning condition, and φ, φ_1 and φ_2 are randomized ATL formulas.

The operators $\langle\!\langle\ \rangle\!\rangle_{\mathit{win}}$ are *path quantifiers*, and $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") are the usual *temporal operators* [Eme90, MP91].

Semantics. We interpret randomized ATL formulas over the states of a system $\mathcal{S}$ that has the same sets of agents and propositions used to define the formulas.

The subformulas of randomized ATL of the form $\bigcirc\varphi$, $\Box\varphi$, or $\varphi_1 \mathcal{U} \varphi_2$ are called *path subformulas*, and they are interpreted over the paths of $\mathcal{S}$. For a path subformula ψ, we denote by $[\![\psi]\!]$ the event consisting of all the paths that satisfy ψ. For path subformulas φ, φ_1, φ_2, this event is defined using the standard semantics of the temporal operators:

- $s_0, s_1, s_2, \ldots \in [\![\bigcirc\varphi]\!]$ iff $s_1 \models \varphi$.
- $s_0, s_1, s_2, \ldots \in [\![\Box\varphi]\!]$ iff $s_i \models \varphi$ for all $i \ge 0$.
- $s_0, s_1, s_2, \ldots \in [\![\varphi_1 \mathcal{U} \varphi_2]\!]$ iff there is $k \ge 0$ such that $s_k \models \varphi_2$ and $s_i \models \varphi_1$ for all $0 \le i < k$.

It is not difficult to show that $[\![\psi]\!]$ is measurable for any path subformula ψ, under any strategies [Var85].

Subformulas of randomized ATL that have the form p, $\neg\varphi$, $\varphi_1 \vee \varphi_2$, or $\langle\langle A\rangle\rangle_{win}\psi$ are called *state subformulas*, and they are interpreted over the states of $\mathcal{S}$. For a state subformula φ, we write $s \models \varphi$ to indicate that the state s satisfies φ. For $q \in \mathcal{Q}$, for state subformulas φ, φ_1, φ_2, and for path subformulas ψ, we define:

- $s \models q$ iff $q \in L(s)$.
- $s \models \neg\varphi$ iff $s \not\models \varphi$.
- $s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$.
- $s \models \langle\langle A\rangle\rangle_{sure}\psi$ iff there exists $\pi_A \in \Pi_A$ such that for all $\pi_B \in \Pi_B$ we have $Paths(s, \pi_A, \pi_B) \subseteq [\![\psi]\!]$.
- $s \models \langle\langle A\rangle\rangle_{almost}\psi$ iff there exists $\pi_A \in \Pi_A$ such that for all $\pi_B \in \Pi_B$ we have $\Pr_s^{\pi_A,\pi_B}([\![\psi]\!]) = 1$.
- $s \models \langle\langle A\rangle\rangle_{limit}\psi$ iff $\sup_{\pi_A\in\Pi_A} \inf_{\pi_B\in\Pi_B} \Pr_s^{\pi_A,\pi_B}([\![\psi]\!]) = 1$.

From these definitions, we see that the logic ATL corresponds to the fragment of randomized ATL that includes only the path quantifier $\langle\langle A\rangle\rangle_{sure}$.

From the classification of winning states in Section 2, it follows that $s \models \langle\langle A\rangle\rangle_{sure}\psi$ implies $s \models \langle\langle A\rangle\rangle_{almost}\psi$, which in turn implies $s \models \langle\langle A\rangle\rangle_{limit}\psi$; the reverse implications do not necessarily hold. Interestingly, the implications can be strict only for path subformulas ψ of the form $\varphi\,\mathcal{U}\,\psi$, which specify liveness properties (such as reaching a target R, $true\,\mathcal{U}\,\varphi_R$). By contrast, for path subformulas ψ of the form $\bigcirc\varphi$ and $\Box\varphi$, which specify safety properties, the three winning conditions are equivalent, as stated by the following theorem.

Theorem 12 *Consider a path formula ψ of the form $\bigcirc\varphi$ or $\Box\varphi$. Then, for every state s of a system $\mathcal{S}$, we have $s \models \langle\langle A\rangle\rangle_{sure}\psi$ iff $s \models \langle\langle A\rangle\rangle_{almost}\psi$ iff $s \models \langle\langle A\rangle\rangle_{limit}\psi$.*

4.3 Model Checking for Randomized ATL

The *model-checking problem* for randomized ATL asks, given a system $\mathcal{S}$ and a randomized ATL formula φ, for the set of states of $\mathcal{S}$ that satisfy φ. A model-checking algorithm for randomized ATL can proceed bottom-up on the state subformulas of φ, as in CTL and ATL model checking [CE81, QS81, AHK97]. There are two cases of state subformulas that have no equivalent expression in ATL: $\langle\langle A\rangle\rangle_{almost}\varphi_1\,\mathcal{U}\,\varphi_2$ and $\langle\langle A\rangle\rangle_{limit}\varphi_1\,\mathcal{U}\,\varphi_2$. In order to check these subformulas, we first construct a two-player game structure, in which player 1 corresponds to the set A of agents, and player 2 corresponds to the set B. We define the target set R to be the set $R = \{s \in S \mid s \models \varphi_2\}$ consisting of the states that satisfy the eventuality φ_2. If this set is not absorbing, as required by the reachability algorithms, then the game structure should be locally modified to make R absorbing. Then, we modify Algorithms 3 and 6, substituting the ordinary computation of set C_0 with

$$C_0 = Safe_2(S \setminus R, \Gamma_1, \Gamma_2) \cup \{s \in S \mid s \not\models \varphi_1 \vee \varphi_2\}\,. \tag{17}$$

Intuitively, while in the $\Diamond R$ reachability game player 1 only has to avoid states in which player 2 can keep him away from the target set R, in the $\varphi_1 \mathcal{U} \varphi_2$ game player 1 also has to avoid states that satisfy neither φ_1 nor φ_2. The following theorem and corollary summarize the relevant results on the model checking of randomized ATL specifications.

Theorem 13 *The model-checking problem for randomized ATL specifications can be solved by combining the algorithms of [AHK97] for* $\langle\langle\ \rangle\rangle_{\text{sure}}$ *with the modification (17) of Algorithms 3 and 6 for* $\langle\langle\ \rangle\rangle_{\text{almost}}$ *and* $\langle\langle\ \rangle\rangle_{\text{limit}}$.

Corollary 2 *The model checking problem can be solved in time quadratic in the size of the system.*

5 Proofs of the Results

In this section we provide the correctness proofs of the algorithms for the computation of the sets $Sure(R)$, $Almost(R)$, and $Limit(R)$, as well as the proofs of the theorems presented in the previous sections. While proving the correctness of the algorithms, we also describe how to construct the winning and spoiling strategies for the various types of reachability. To simplify the notation, given a subset $U \subseteq S$ of states, we denote by $\overline{U} = S \setminus U$ its complement with respect to S.

5.1 Sure Reachability

Proof of Theorems 1, 2 and 5. Assume that Algorithm 2 terminates at iteration m, and let $U_0, \ldots, U_m$ be the sets of states computed during the execution of the algorithm.

Define $h : U_m \setminus R \mapsto \mathbb{N}$ by $h(s) = \min\{j \in \{1, \ldots, m\} \mid s \in U_j\}$ for each $s \in U_m \setminus R$, and let π_1^*, π_2^* be the winning and spoiling strategies described in Theorem 5.

For $s \in U_m$, consider any $\pi_2 \in \Pi_2$ and any path $\overline{s} = s_0, s_1, s_2, \ldots \in Paths(s, \pi_1^*, \pi_2)$, with $s_0 = s$. From the definition of π_1^*, it is immediate to see that for all $j \geq 0$, if $s_j \in U_m \setminus R$ then $s_{j+1} \in U_m$ and either $s_{j+1} \in R$, or $h(s_j) > h(s_{j+1})$. This shows that $\overline{s} \in (\Diamond R)$. Theorem 2(2,4) and Theorem 5(1) follow from this analysis.

In the other direction, if $s \notin U_m$, then for all $a \in \Gamma_1(s)$ there is $b \in \Gamma_2(s)$ such that $\delta(s, a, b) \not\subseteq U_m$. Hence, for all $s \notin U_m$ and all strategies $\pi_1 \in \Pi_1$ there is a path $\overline{s} \in Paths(s, \pi_1, \pi_2^*)$ such that $\overline{s} \notin (\Diamond U_m)$, and therefore $\overline{s} \notin (\Diamond R)$. This proves Theorem 5(2), and together with the above argument, also Theorem 2(1). The correctness of Algorithm 2 also leads to the fixed-point characterization expressed by Theorem 1.

To see that player 2 may not have a deterministic spoiling strategy, it suffices to consider state t_{throw} of the LEFT-OR-RIGHT game. Clearly, $t_{throw} \notin Sure(R)$; yet, given any deterministic strategy π_2 for player 2, we can construct a deterministic strategy π_1 for player 1 so that the target t_{hit} is reached surely in one round. This proves Theorem 2(4). ∎

Proof of Theorem 3. Consider a deterministic reachability game. For the first part of the theorem, assume there is a memoryless deterministic winning strategy π_1^* for almost-sure

reachability. From the point of view of player 2, the game under strategy π_1^* is equivalent to a directed graph (S, E) with set of edges

$$E = \{(s,t) \mid \exists b \in \Gamma_2(s) \,.\, t \in \delta(s, a_s, b)\},$$

where $a_s \in \Gamma_1(s)$ is the single move such that $\pi_1^*(s)(a_s) = 1$. Consider an arbitrary state s; there are two cases:

- If there is an infinite path in (S, E) that originates from s and never enters R, then player 2 has a (memoryless deterministic) strategy π_2 to ensure that this path is followed. Hence, $\Pr_s^{\pi_1^*,\pi_2}(\Diamond R) = 0$. Since π_1^* is a winning strategy for almost-sure reachability, $s \notin \mathit{Almost}(R)$, and $s \notin \mathit{Sure}(R)$.
- If all infinite paths in (S, E) that originate from s eventually reach R, then all the paths originating from s of length greater than $|S|$ have a state in R. Using this fact, it is not difficult to prove by comparison with Algorithm 2 that $s \in \mathit{Sure}(R)$, and hence $s \in \mathit{Almost}(R)$.

These two cases together prove $\mathit{Sure}(R) = \mathit{Almost}(R)$.

For the second part of the theorem, note that there is only a finite number of memoryless deterministic strategies. Hence, there must be at least one of the winning strategies for limit-sure reachability that is also a winning strategy for almost-sure reachability. The result then follows from the first part of the theorem. ∎

Proof of Theorem 4. Assume that the reachability game is deterministic and turn-based, and let $m \geq 0$ and $U_0, \ldots, U_m$ be as in the previous proof. Consider $s \notin U_m$. There are two cases, depending on which player's turn it is at s. If it is player 1's turn, i.e. if $|\Gamma_2(s)| = 1$, then it must be $\delta(s, a, b) \cap U_m = \emptyset$ for all $a \in \Gamma_1(s)$ and for the single $b \in \Gamma_2(s)$, or else s would be included in U_{m+1} and the algorithm would not terminate at iteration m. Similarly, if it is player 2's turn, i.e. if $|\Gamma_1(s)| = 1$, then there must be at least one $b \in \Gamma_2(s)$ such that $\delta(s, a, b) \cap U_m = \emptyset$, for the single $a \in \Gamma_1(s)$. In both cases, there is $b \in \Gamma_2(s)$ such that $\delta(s, a, b) \cap U_m = \emptyset$ for all $a \in \Gamma_1(s)$, and this leads immediately to the existence of a memoryless deterministic spoiling strategy π_2 for player 2 having the properties stated in the theorem. ∎

5.2 Almost-Sure Reachability

Before proving the correctness of Algorithm 3, we need the following technical lemma.

Lemma 2 *Let $\gamma_1, \gamma_2 : S \mapsto 2^{Moves} \setminus \emptyset$ be two non-empty move sub-assignments for players 1 and 2. Let $\pi_2 \in \Pi_2$ be the memoryless strategy for player 2 that chooses at every state $s \in S$ a move from $\gamma_2(s)$ uniformly at random. Denote also with $\Pi_1(\gamma_1) \subseteq \Pi_1$ the set of strategies for player 1 that at each $s \in S$ choose only moves from $\gamma_1(s)$. For any $U \subseteq S$, let $V = \mathit{Safe}_1(U, \gamma_1, \gamma_2)$. The following statements hold:*

1. *There is $q > 0$ such that for all $s \in U \setminus V$ and all strategies $\pi_1 \in \Pi_1(\gamma_1)$ for player 1, we have*

$$\Pr_s^{\pi_1,\pi_2}\Big(\bigvee_{i=0}^{|U|} X_k \notin U\Big) \geq q\,.$$

2. *If $V = \emptyset$, then $\Pr_s^{\pi_1,\pi_2}(\Diamond\overline{U}) = 1$ for all $s \in U$ and all $\pi_1 \in \Pi_1(\gamma_1)$.*

Similar statements hold if the roles of player 1 and player 2 are exchanged.

Proof. Under strategy π_2, the game from the point of view of player 1 is a Markov decision process [Der70]. The first statement can be proved by induction on the number of the iteration at which s has been removed from U during the execution of Algorithm 1. The second result follows by noting that the probability that a path from $s \in U$ has not left U in the first i rounds is no greater than $(1-q)^{\lfloor i/|U| \rfloor}$, and by taking the limit for $i \to \infty$. ∎

Next, we describe how to construct spoiling strategies for limit-sure reachability. The construction is slightly involved, since these strategies cannot be finite-memory, as stated by Theorem 7(3b).

Theorem 14 *Assume that Algorithm 3 terminates at iteration m, and let $U_0, \ldots, U_m$ and $\gamma_1, \ldots, \gamma_m$ be the sequences of sets and move sub-assignments computed by the algorithm. Let $q_0, q_1, q_2, \ldots$ be an infinite sequence of real numbers such that $0 < q_j < 1$ for all $j \geq 0$, and $\prod_{j=0}^{\infty} q_j = 1/2$. Such a sequence can be constructed by taking $q_i = 2^{(-1/2^{i+1})}$, for $i \geq 0$. Construct the counting strategy π_2^* for player 2 as follows:*

1. *At $s \in C_i$, for $0 \leq i < m$ (note that $C_m = Safe_2(U_m \setminus R, \gamma_m, \Gamma_2) = \emptyset$), strategy π_2^* plays according to the number l of rounds played since the start of the game. At round l, π_2^* plays as follows:*

   (a) *with probability q_l, strategy π_2^* plays uniformly at random a move from $Stay_2(C_i, \gamma_i, \Gamma_2)(s)$;*

   (b) *with probability $1 - q_l$, strategy π_2^* plays uniformly at random a move from $\Gamma_2(s)$.*

2. *At $s \in S \setminus \bigcup_{i=0}^{m-1} C_i$, strategy π_2^* plays uniformly at random a move from $\Gamma_2(s)$.*

Then, π_2^* is a spoiling strategy for almost-sure reachability.

Proof of Theorem 7(1,2,3a), Theorem 9, and Theorem 14. Assume that the algorithm terminates at iteration m, and let $U_0, \ldots, U_m$ and $\gamma_1, \ldots, \gamma_m$ be the sequences of sets and move sub-assignments computed by the algorithm. Let π_1^* be the memoryless strategy for player 1 described in Theorem 9, and let π_2^* be the counting spoiling strategy described in Theorem 14. Let also $q_0, q_1, q_2, \ldots$ be the sequence of probabilities used to construct π_2^* in Theorem 14.

First, we prove that $U_m \subseteq Almost(R)$. Since the algorithm terminates at iteration m, we have $Safe_2(U_m \setminus R, \gamma_m, \Gamma_2) = \emptyset$. Hence, by the second part of Lemma 2, for $s \in U_m$

and all $\pi_2 \in \Pi_2$ we have $\Pr_s^{\pi_1,\pi_2}(\Diamond(\overline{U}_m \cup R)) = 1$. Note that, under strategy π_1^*, once the game is in U_m it will never leave U_m, regardless of the strategy used by player 2. Hence, we conclude that $\Pr_s^{\pi_1,\pi_2}(\Diamond R) = 1$ for all $s \in U_m$ and $\pi_2 \in \Pi_2$, as was to be proved.

To prove that $Almost(R) \subseteq U_m$, we prove by complete induction on i, for $0 \leq i < m$, that if $s \in U_i \setminus U_{i+1}$ then for all $\pi_1 \in \Pi_1$ we have

$$\Pr_s^{\pi_1,\pi_2^*}(\Diamond R) < 1 .$$

Consider an arbitrary strategy π_1 for player 1. For each $0 \leq i < m$ there are two cases, depending whether $s \in C_i$ or $s \in U_i \setminus (C_i \cup U_{i+1})$.

- If $s \in C_i$, then let

$$A_i = \left\{ t_0, t_1, t_2, \ldots \in \Omega \;\middle|\; \exists k \geq 0 \,.\, \Big[\bigwedge_{j=0}^{k} t_j \in C_i \wedge \mathrm{Supp}(\pi_1(t_0, t_1, \ldots t_k)) \not\subseteq \gamma_i(t_k) \Big] \right\}$$

  be the event of player 1 playing with non-zero probability a move selected outside of γ_i while still in C_i.

  Assume first $\Pr_s^{\pi_1,\pi_2^*}(A_i) > 0$. Then, there is a finite sequence $\sigma : s = t_0, t_1, \ldots, t_k$ of states of C_i such that:

$$\Pr_s^{\pi_1,\pi_2^*}\Big(\bigwedge_{j=0}^{k} X_j = t_j \Big) > 0, \qquad \mathrm{Supp}(\pi_1(\sigma)) \not\subseteq \gamma_i(t_k) .$$

  By definition of γ_i, if player 2 plays according to π_2^* and player 1 at $t_k \in C_i \subseteq U_i$ plays move $a \notin \gamma_i(t_k)$, the game leaves U_i with positive probability, since π_2^* chooses each move in $\Gamma_2(t_k)$ with positive probability. Hence, a behavior from s has a positive probability of leaving U_i, and the induction hypothesis leads to the desired result.

  If $\Pr_s^{\pi_1,\pi_2^*}(A_i) = 0$, let $(\Box C_i) = \{t_0, t_1, t_2, \ldots \mid \forall k \,.\, t_k \in C_i\}$ be the event of being confined to C_i. Since $\Pr_s^{\pi_1,\pi_2^*}(A_i) = 0$, as long as the game is in C_i player 1 never chooses a move outside of γ_i. Hence, by definition of γ_i, we have

$$\Pr_s^{\pi_1,\pi_2^*}(\Box C_i) \;\geq\; \Pr_s^{\pi_1,\pi_2^*}\Big(\forall k \,.\, \mathrm{Supp}(\pi_2^*(X_0, \ldots, X_k)) \subseteq Stay_2(C_i, \gamma_i, \Gamma_2)(X_k)\Big) = \frac{1}{2} ,$$

  where the last equality is a consequence of the definition of π_2^*. This indicates that if $\Pr_s^{\pi_1,\pi_2^*}(A_i) = 0$, then a path from s is confined forever in C_i with positive probability, which leads immediately to the desired result.

- If $s \in U_i \setminus (U_{i+1} \cup C_i)$, then strategy π_2^* in $U_i \setminus (U_{i+1} \cup C_i)$ plays uniformly at random from the sub-assignment Γ_2. Since

$$U_{i+1} = Safe_1(U_i \setminus C_i, \gamma_i, \Gamma_2) = Safe_1(U_i \setminus C_i, \Gamma_1, \Gamma_2)$$

  by Lemma 2 we have for all $\pi_1 \in \Pi_1$ that

$$\Pr_s^{\pi_1,\pi_2^*}\Big(\Diamond(C_i \cup \overline{U}_i)\Big) > 0 .$$

  The induction hypothesis, jointly with the analysis of the previous case, leads then to the result.

The above arguments prove Theorem 9 and Theorem 14, and thus also Theorem 7(2a,3a). The lack of memoryless deterministic winning strategies (Theorem 7(2b)) is witnessed by the behavior of game LEFT-OR-RIGHT from state t_{throw}. Theorem 7(1) also follows from the above arguments, and from an analysis of Algorithm 3. ∎

Proof of Theorem 7(4). Consider again the winning strategy π_1^* for player 1 described in Theorem 5, and let $K = |Almost(R)|$. Under strategy π_1^* the set $Almost(R)$, once entered, is never left, regardless of the strategy chosen by player 2. By Lemma 2, there is $q > 0$ such that for all $s \in Almost(R)$ and all $\pi_2 \in \Pi_2$ we have

$$\Pr_s^{\pi_1^*,\pi_2}(\bigvee_{k=0}^{K} X_k \in R) \geq q\,.$$

Hence, from any s, the probability that the time to R is greater than n is at most $(1-q)^{\lfloor n/K \rfloor}$, and by standard arguments this yields the first part of Theorem 7(4). To prove the second part of Theorem 7(4), note that if the time from s to R is bounded, then by definition there is a strategy $\pi_1 \in \Pi_1$ such that $Paths(s, \pi_1, \pi_2) \subseteq (\Diamond R)$ for all $\pi_2 \in \Pi_2$, and hence $s \in Sure(R)$. ∎

Theorem 6 follows as a direct corollary of these results.

Proof of Theorem 6. Let U^* be the largest set satisfying conditions (4). Assume that Algorithm 3 terminates at iteration m with output U_m. To prove that $U^* \subseteq Almost(R)$ we can repeat the argument used to show that $U_m \subseteq Almost(R)$ in the proof of Theorem 7. Since U_m also satisfies (4), we also have $Almost(R) = U_m \subseteq U^*$, and this concludes the proof. ∎

It is interesting to note that, while we can prove the containment $U^* \subseteq Almost(R)$ without reference to Algorithm 3, we have only been able to prove the reverse containment $Almost(R) \subseteq U^*$ by analyzing Algorithm 3.

To prove Theorem 7(3b), we first restate more precisely the definition of *finite-memory* strategy. We say that a strategy π is *finite-memory* if there is a deterministic automaton (Q, η, q_{in}) with set of states Q, transition function $\eta : Q \times S \mapsto Q$, and initial state $q_{in} \in Q$, and a mapping $\pi' : Q \times S \mapsto \mathcal{D}(Moves)$ such that for all $\sigma \in S^*$ we have

$$\pi(\sigma s) = \pi'(\eta^*(q_{in}, \sigma), s)\,,$$

where $\eta^* : Q \times S^* \mapsto Q$ is the multi-step transition relation of the automaton, defined as usual.

Proof of Theorem 7(3b). Consider the game HIDE-OR-RUN, and a finite memory strategy $\pi_2 \in \Pi_2$ based on the deterministic automaton (Q, η, q_{in}) and on the mapping $\pi_2' : S \times Q \mapsto \mathcal{D}(Moves)$. Define the strategy $\pi_1 \in \Pi_1$ for player 1 by

$$\pi_1(\sigma s_{hide})(hide) = \begin{cases} 1 & \text{if } \pi_2(\sigma s_{hide})(throw) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\pi_1(\sigma s_{hide})(run) = 1 - \pi_1(\sigma s_{hide})(hide)$$

for all $\sigma \in S^*$. At states other than s_{hide}, the strategy is trivial, since it must always choose the only available move. Note that π_1 is a finite-memory strategy based on the same automaton as π_2, so that there is a mapping $\pi_1' : S \times Q \mapsto \mathcal{D}(Moves)$ such that $\pi_1(\sigma s) = \pi_1'(s, \eta^*(q_{in}, \sigma s))$ for all $\sigma \in S^*$ and final states $s \in S$.

We want to show that $\Pr_{s_{hide}}^{\pi_1,\pi_2}(\Diamond\{s_{home}\}) = 1$. By definition of π_1, the game when started from s_{hide} never reaches s_{wet}. Moreover, once π_1 and π_2 are fixed, the game corresponds to a Markov chain with set of states $S \times Q$ and transition probabilities

$$\Pr(\langle s', q' \rangle \mid \langle s, q \rangle) = \sum_{a \in \Gamma_1(s)} \sum_{b \in \Gamma_2(s)} p(s, a, b)(s')\, \pi_1'(s, q)(a)\, \pi_2'(s, q)(b)$$

for all $s, s' \in S$ and $q, q' \in Q$. When the automaton is presented with the infinite input $s_{hide}{}^{\omega}$, it will produce the infinite state sequence

$$q_{in}, q_1, \ldots, q_k, (q_{k+1}, q_{k+2}, \ldots, q_{k+m})^{\omega} ,$$

for some $m > 0$. Whether the game reaches s_{home}, or whether it remains forever confined to s_{hide}, clearly depends on the behavior of the Markov chain on the set of states

$$\left\{ \langle s_{hide}, q_{k+1} \rangle, \langle s_{hide}, q_{k+2} \rangle, \ldots, \langle s_{hide}, q_{k+m} \rangle \right\} .$$

By construction of π_1, this set of states is not a closed recurrent class. Hence, the game is confined to s_{hide} with probability 0, and reaches s_{home} with probability 1. This concludes the argument. ∎

The results on almost-sure reachability for turn-based games can be proved as follows.

Proof of Theorem 8. Suppose that the game is turn-based, assume that Algorithm 3 terminates at iteration m, and let $U_0, \ldots, U_m$ and $\gamma_1, \ldots, \gamma_m$ be the sequences of sets and move sub-assignments computed by the algorithm.

First, we prove that player 1 has a memoryless deterministic winning strategy for almost-sure reachability by constructing a memoryless deterministic strategy π_1^* for player 1 as follows. At $s \in U_m \setminus R$, strategy π_1^* plays deterministically one of the moves that caused the elimination of s from $Safe_2(U_m \setminus R, \gamma_m, \Gamma_2) = \emptyset$ during the execution of Algorithm 1. At $s \in R \cup \overline{U}_m$, strategy π_1^* is defined arbitrarily. Define the move sub-assignment θ corresponding to π_1^* by $\theta(s) = \mathrm{Supp}(\pi_1^*(s))$ for all $s \in S$. By construction of π_1^*, we see that $Safe_2(U_m \setminus R, \theta, \Gamma_2) = \emptyset$. Hence, by the second part of Lemma 2, for all $s \in U_m \setminus R$ and all $\pi_2 \in \Pi_2$ we have

$$\Pr_s^{\pi_1^*,\pi_2}(\Diamond(R \cup \overline{U}_m)) = 1 .$$

From this, and from the fact that $\theta \subseteq \gamma_m = Stay_1(U_m, \Gamma_1, \Gamma_2)$, we conclude $\Pr_s^{\pi_1^*,\pi_2}(\Diamond R) = 1$ for all $s \in U_m \setminus R$ and all $\pi_2 \in \Pi_2$. This indicates that π_1^* is a winning strategy for almost-sure reachability.

To show the existence of a memoryless deterministic spoiling strategy for almost-sure reachability, we construct the memoryless deterministic strategy π_2^* for player 2 as follows:

- At $s \in C_i$, for $0 \leq i < m$ (note that $C_m = \mathit{Safe}_2(U_m \setminus R, \gamma_m, \Gamma_2) = \emptyset$), strategy $\pi_2^\bullet$ plays a move selected arbitrarily from $\mathit{Stay}_2(C_i, \gamma_i, \Gamma_2)(s)$.
- At $s \in U_i \setminus (U_{i+1} \cup C_i)$, for all $0 \leq i < m$, strategy $\pi_2^\bullet$ plays deterministically one of the moves that caused the elimination of s from $\mathit{Safe}_1(U_i \setminus C_i, \gamma_i, \Gamma_2)$ during the execution of Algorithm 1.
- At $s \in U_m$, strategy $\pi_2^\bullet$ is defined arbitrarily.

Proceeding as in the proof of Theorem 7(3a), we can prove that $\Pr_s^{\pi_1,\pi_2^\bullet}(\Diamond R) < 1$ for all $s \notin U_m$ and all $\pi_1 \in \Pi_1$. The argument is again an induction by cases, with the same inductive hypothesis used in the proof of Theorem 7(3a). The case for $s \in U_i \setminus (U_{i+1} \cup C_i)$, for $0 \leq i < m$, can be proved essentially in the same way.

If $s \in C_i$, for $0 \leq i < m$, we reason as follows. If player 1 plays a move in $\gamma_i(s)$, then the game will remain in C_i. If player 1 plays a move not in $\gamma_i(s)$, then it must be player 1's turn to move, i.e. $|\Gamma_2(s)| = 1$. By definition of γ_i, we know that the game leaves U_i with non-zero probability. Jointly, these considerations prove that $\pi_2^\bullet$ is a memoryless deterministic spoiling strategy for almost-sure reachability.

Finally, the fact that $\mathit{Almost}(R) = \mathit{Limit}(R)$ is a direct consequence of the existence of memoryless spoiling strategies. In fact, from the point of view of player 1, the game under strategy $\pi_2^\bullet$ is equivalent to a Markov decision process. Hence, if player 2 uses strategy $\pi_2^\bullet$, there is a (memoryless) strategy π_1° for player 1 that maximizes the probability of reaching R from every state [Der70, Ber95]. Therefore, for every $s \in S \setminus \mathit{Almost}(R)$, there is $q_s < 1$ such that

$$\max_{\pi_1 \in \Pi_1} \Pr_s^{\pi_1,\pi_2^\bullet}(\Diamond R) = \Pr_s^{\pi_1^\circ,\pi_2^\bullet}(\Diamond R) = q_s \,.$$

This yields directly that $\mathit{Almost}(R) = \mathit{Limit}(R)$, together with the fact that strategies $\pi_1^\bullet$ and $\pi_2^\bullet$ are winning and spoiling also for limit-sure reachability. ∎

5.3 Limit-Sure Reachability

In order to prove Theorem 11, we must first show that Algorithm 4 correctly determines whether a state is a limit-escape state. In fact, we provide a stronger characterization of limit-escape states than that provided by (12). The proof proceeds in two parts: first, we prove that if $\mathit{lim\text{-}esc}(s, C, U) = \textsc{Yes}$, then s satisfies (12); next, we show that if $\mathit{lim\text{-}esc}(s, C, U) = \textsc{No}$, then the ratio in (12) is bounded away from infinity. While proving these results, we also define some distributions that are useful in the construction of the winning and spoiling strategies.

In these arguments, we are often interested in the behavior of parameterized strategies, for the value of the parameter close to 0. To simplify the notation, we call we call a *right neighborhood* of 0 an interval $[0, d]$ for some $d > 0$. We indicate by ξ a generic right neighborhood of 0, and by Ξ the set of all right neighborhoods of 0. Let also $M = \max\{|\Gamma_i(s)| \mid i \in \{1,2\} \wedge s \in S\}$ be the maximum number of moves available to a player at any state. For each $a \in \mathit{Moves}$, denote also by ξ^a the distribution that selects move a deterministically: these distributions are called *singular* distributions.

Given s, C, and U such that *lim-esc*$(s, C, U) =$ YES and $0 \leq \varepsilon \leq 1/(2M)$, we construct a distribution *evasion*$(s, C, U)[\varepsilon] \in \mathcal{D}(\Gamma_1(s))$ that enables the limit-escape from s as $\varepsilon \to 0$. Let k be the number of iterations required for the call *lim-esc*(s, C, U) to terminate, and let $\mathcal{A}_0, \ldots, \mathcal{A}_k$ and $\mathcal{B}_0, \ldots, \mathcal{B}_k$ be the sets of moves computed during the iterative execution of the algorithm. All moves in $\Gamma_1(s)$ are labeled, since *lim-esc*$(s, C, U) =$ YES: define $\ell(a) = \min\{j \mid a \in \mathcal{A}_j\}$ for each $a \in \Gamma_1(s)$. For all $a \in \Gamma_1(s)$, we define *evasion* by

$$\mathit{evasion}(s, C, U)[\varepsilon](a) = \begin{cases} \varepsilon^{\ell(a)} & \text{if } \ell(a) > 0; \\ \dfrac{1}{|\Gamma_1(s) \setminus \mathcal{A}_0|}\left(1 - \displaystyle\sum_{a \in \Gamma_1(s) \setminus \mathcal{A}_0} \varepsilon^{\ell(a)}\right) & \text{otherwise.} \end{cases}$$

The following lemma uses the above distribution to prove that Algorithm 4 answers YES only for limit-escape states. The lemma provides a stronger characterization of limit-escape states than that provided by (12), which follows as a corollary. The stronger characterization is used to prove Theorem 11.

Lemma 3 *Assume that* *lim-esc*$(s, C, U) =$ YES, *and let* $\xi_1[\varepsilon] =$ *evasion*$(s, C, U)[\varepsilon]$, *for* $0 \leq \varepsilon \leq 1/(2M)$. *Then, there are constants* $\alpha, \beta > 0$ *and a right neighborhood* ξ *of* 0 *such that for every distribution* $\xi_2 \in \mathcal{D}(\Gamma_2(s))$ *there is* $0 \leq i \leq M$ *such that*

$$\widetilde{p}(s, \xi_1[\varepsilon], \xi_2)(\overline{C}) \geq \alpha \varepsilon^i, \tag{18}$$

$$\widetilde{p}(s, \xi_1[\varepsilon], \xi_2)(\overline{U}) \leq \beta \varepsilon^{i+1} \tag{19}$$

for all $0 \leq \varepsilon \leq 1/(2M)$.

Proof. Let k be the number of iterations required for Algorithm 4 to terminate, let E_1, E_2 be as computed in the initialization step of the algorithm, and let $\mathcal{A}_0, \ldots, \mathcal{A}_k$ and $\mathcal{B}_0, \ldots, \mathcal{B}_k$ be the sets of moves computed during the iteration. Since the algorithm terminates with an affirmative answer, we have $\Gamma_1(s) = \mathcal{A}_k$ and $\Gamma_2(s) = \mathcal{B}_k$. To establish the result, note that every distribution $\xi_2 \in \mathcal{D}(\Gamma_2(s))$ can be written as the convex combination of singular distributions:

$$\xi_2 = \sum_{b \in \Gamma_2(s)} \xi_2(b)\, \xi^b .$$

We first prove that the lemma holds for these singular distributions. Consider any move $b \in \Gamma_2(s)$. Since b has been labeled, there is at least one $a \in \Gamma_1(s)$ with $(a, b) \in E_1$ and $\ell(a) = \ell(b)$. Since $(a, b) \in E_1$, when both a and b are played the game leaves C with probability $\widetilde{p}(s, a, b)(\overline{C})$. Move a is played with one-round probability $\varepsilon^{\ell(a)} = \varepsilon^{\ell(b)}$ if $\ell(a) > 0$, and with probability at least $1/(2M)$ if $\ell(a) = 0$. Taking

$$\alpha_b = \begin{cases} \frac{1}{2M}\widetilde{p}(s, a, b)(\overline{C}) & \text{if } \ell(b) = 0 \\ \widetilde{p}(s, a, b)(\overline{C}) & \text{otherwise} \end{cases}$$

and noting that $\alpha_b > 0$, for all $0 \leq \varepsilon \leq 1/(2M)$ we have

$$\widetilde{p}(s, \xi_1[\varepsilon], \xi^b)(\overline{C}) \geq \alpha_b \varepsilon^{\ell(b)} .$$

Next, we consider the possibility of leaving U when move b is played. For $a \in \Gamma_1(s)$, if $\delta(s,a,b) \not\subseteq U$, then $(b,a) \in E_2$, which implies $\ell(a) > \ell(b)$, so that $\xi_1[\varepsilon](a) \leq \varepsilon^{\ell(b)+1}$. Summing over all moves in $\Gamma_1(s)$, for all $0 \leq \varepsilon \leq 1/(2M)$ we obtain

$$\widetilde{p}(s,\xi_1[\varepsilon],\xi^b)(\overline{U}) \leq M\varepsilon^{\ell(b)+1}$$

Let $\alpha = \min\{\alpha_b \mid b \in \Gamma_2(s)\}/2$, $\beta = 2M$ and, for each $\xi_2 \in \mathcal{D}(\Gamma_2(s))$, let $i = \min\{\ell(b) \mid \xi_2(b) > 0\}$. The inequalities (18) and (19) follow by noting that

$$\widetilde{p}(s,\xi_1[\varepsilon],\xi_2)(\overline{C}) = \sum_{b\in\Gamma_2(s)} \xi_2(b)\widetilde{p}(s,\xi_1[\varepsilon],\xi^b)(\overline{C}) \geq \alpha\varepsilon^i$$

$$\widetilde{p}(s,\xi_1[\varepsilon],\xi_2)(\overline{U}) = \sum_{b\in\Gamma_2(s)} \xi_2(b)\widetilde{p}(s,\xi_1[\varepsilon],\xi^b)(\overline{U}) \leq \beta\varepsilon^{i+1}$$

for ε in a sufficiently small right neighborhood of 0. ∎

Corollary 3 *If lim-esc(s,C,U) = YES, then (12) holds.*

Proof. Assume that *lim-esc*(s,C,U) = YES, and let $\xi_1[\varepsilon]$ = *evasion*$(s,C,U)[\varepsilon]$, for $0 \leq \varepsilon \leq 1/(2M)$. By Lemma 3, there is $\kappa = \alpha/\beta > 0$ and a right neighborhood ξ of 0 such that for all $\xi_2 \in \mathcal{D}(\Gamma_2(s))$ and all $\varepsilon \in \xi$ we have

$$\frac{\widetilde{p}(s,\xi_1[\varepsilon],\xi_2)(\overline{C})}{\widetilde{p}(s,\xi_1[\varepsilon],\xi_2)(\overline{U})} \geq \frac{\kappa}{\varepsilon}\,.$$

The result follows by taking the limit $\varepsilon \to 0$. ∎

Given s, C, and U such that *lim-esc*(s,C,U) = NO, we construct a distribution *imprison*$(s,C,U,\mathcal{A},\mathcal{B}) \in \mathcal{D}(\Gamma_2(s))$ that enables player 2 to prevent a limit-escape from state s. Let k be the number of iterations required for the *lim-esc* call to terminate, and $\mathcal{B}_0,\ldots,\mathcal{B}_k$ be the subsets of $\mathcal{B}$ moves computed during the call. For all $b \in \Gamma_2(s)$, define

$$\mathit{imprison}(s,C,U)(b) = \begin{cases} \dfrac{1}{|\Gamma_2(s) \setminus \mathcal{B}_k|} & \text{if } b \in \Gamma_2(s) \setminus \mathcal{B}_k \\ 0 & \text{otherwise} \end{cases}$$

Since *lim-esc*(s,C,U) = NO implies $\mathcal{B}_k \subset \mathcal{B}$, the above is a well-defined distribution. The following lemma is the counterpart of Lemma 3, and shows that if Algorithm 4 answers negatively the limit-escape question, then indeed the ratio in (12) is bounded away from infinity.

Lemma 4 *Assume that lim-esc(s,C,U) = NO, and let ξ_2 = imprison(s,C,U). Then, there is $\kappa > 0$ such that for all $\xi_1 \in \mathcal{D}(\Gamma_1(s))$ we have*

$$\widetilde{p}(s,\xi_1,\xi_2)(\overline{U}) \geq \kappa\,\widetilde{p}(s,\xi_1,\xi_2)(\overline{C})\,.$$

Proof. Let k be the number of iterations required for the call *lim-esc*(s, C, U) to terminate, let E_1, E_2 be as computed in the initialization step of the algorithm, and let $\mathcal{A}_0, \ldots, \mathcal{A}_k$ and $\mathcal{B}_0, \ldots, \mathcal{B}_k$ be the sets of moves computed during the iteration.

Consider a move $a \in \Gamma_1(s)$ that has been labeled (i.e. $a \in \mathcal{A}_k$). For any $b \in \Gamma_2(s)$, if $\delta(s, a, b) \notin C$, then there is $(a, b) \in E_1$, and b has been labeled by the algorithm. Since ξ_2 does not play any move that has been labeled, we have $\widetilde{p}(s, \xi^a, \xi_2)(\overline{C}) = 0$. Hence, for a general distribution $\xi_1 \in \mathcal{D}(\Gamma_1(s))$, we have

$$\widetilde{p}(s, \xi_1, \xi_2)(\overline{C}) \leq \sum_{a \notin \mathcal{A}_k} \xi_1(a) \,. \tag{20}$$

Conversely, consider a move $a \in \Gamma_1(s)$ that has not been labeled. There must be an unlabeled $b \in \Gamma_2(s)$ with $(b, a) \in E_2$. This b is played with probability at least $1/M$, and by definition of E_2 we have $\delta(s, a, b) \notin U$. Thus, for $\alpha_a = \widetilde{p}(s, \xi^a, \xi^b)(\overline{U}) > 0$, we have $\widetilde{p}(s, \xi^a, \xi_2)(\overline{U}) > \alpha_a / M$. Hence, for a general distribution $\xi_1 \in \mathcal{D}(\Gamma_1(s))$, letting $\alpha = \min\{\alpha_a \mid a \notin \mathcal{A}_k\}$, we have

$$\widetilde{p}(s, \xi_1, \xi_2)(\overline{U}) \geq \frac{\alpha}{M} \sum_{a \notin \mathcal{A}_k} \xi_1(a) \,. \tag{21}$$

The result then follows from (20) and (21) by taking $\kappa = \alpha / M$. ∎

From Lemmas 3 and 4, we obtain as a corollary the proof of Lemma 1.

Proof of Lemma 1. If Algorithm 5 terminates at iteration k, then $\mathcal{A}_k = \mathcal{A}$ and $\mathcal{B}_k = \mathcal{B}$, where $\mathcal{A}_k$, $\mathcal{B}_k$ are the sets of moves computed by Algorithm 5, and $\mathcal{A}$, $\mathcal{B}$ are the fixed-points mentioned by Lemma 1. The lemma is then an immediate consequence of Lemmas 3 and 4. ∎

In the following, we consider a slightly modified version of Algorithm 5, which removes the limit-escape states one at a time. The modified algorithm is given below.

Algorithm 7
Input: Game structure G, two sets $W \subseteq U \subseteq S$ of states.
Output: *Lim-safe'*$(W, U) \subseteq S$.

> **Initialization:** Let $V_0 = W$.
> **Repeat** For $k \geq 0$, let $L_k = \{s \in V_k \mid s$ limit-escape w.r.t. V_k and $U\}$.
> If $L_k = \emptyset$, then let $V_{k+1} = V_k$.
> Otherwise, pick $t_k \in L_k$ and let $V_{k+1} = V_k \setminus \{t_k\}$.
> **Until** $V_{k+1} = V_k$.
> **Return:** V_k.

Clearly, *Lim-safe*(W, U) = *Lim-safe'*(W, U) for all $W \subseteq U \subseteq S$, since both algorithms compute the largest subset $C \subseteq W$ that does not contain any limit-escape state w.r.t. C and U. We use this modified algorithm to define winning and spoiling strategies for limit-sure reachability.

Theorem 15 *Assume that Algorithm 6 terminates with output U. Clearly, Lim-safe$'(U \setminus R, U) = \emptyset$: hence, we can write $U \setminus R = \{t_0, \ldots, t_k\}$, where t_0, ..., t_k are as selected by Algorithm 7 at iterations 0, ..., k. Given $0 \leq \varepsilon \leq 1/(2M)$, define the memoryless strategy $\pi_1^*[\varepsilon] \in \Pi_1$ for player 1 by taking, for for $0 \leq i \leq k$,*

$$\pi_1^*[\varepsilon](t_i) = evasion(t_i, \{t_i, t_{i+1}, \ldots, t_k\}, U) \left[\varepsilon^{[(M+2)^i]}\right] , \tag{22}$$

and define $\pi_1^*[\varepsilon]$ arbitrarily outside of $U \setminus R$. Then, $\{\pi_1^*[\varepsilon] \mid 0 < \varepsilon \leq 1/(2M)\}$ is a family of winning strategies for limit-sure reachability.

Theorem 16 *Assume that Algorithm 6 terminates at iteration m, and let $U_0, \ldots, U_m$ and $C_0, \ldots, C_m$ be the sets computed by the algorithm, with $C_m = \emptyset$. Let $\pi_2^* \in \Pi_2$ be the memoryless strategy for player 2 defined as follows:*

- *At $s \in C_i$, for $0 \leq i < m$, we have $\pi_2^*(s) = imprison(s, C_i, U_i)$.*
- *At $s \notin \bigcup_{i=0}^{m-1} C_i$, π_2^* selects a move from $\Gamma_2(s)$ uniformly at random.*

Then, π_2^* is a spoiling strategy for limit-sure reachability.

Proof of Theorems 10, 11(2,3), 15, 16. Assume that Algorithm 6 terminates at iteration m, and let $U_0, \ldots, U_m$ and $C_0, \ldots, C_m$ be the sets computed by the algorithm, with $C_m = \emptyset$. Clearly, *Lim-safe$'(U_m \setminus R, U_m) = \emptyset$*: hence, we can write $U_m \setminus R = \{t_0, \ldots, t_k\}$, where t_0, ..., t_k are as selected by Algorithm 7.

First, we show $U_m \subseteq Limit(R)$. For $0 < \varepsilon < 1/(2M)$, let $\pi_1^*[\varepsilon]$ be the strategy described by Theorem 15. Our goal is to show that for all $s \in U_m$,

$$\lim_{\varepsilon \to 0} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1^*[\varepsilon], \pi_2}(\Diamond R) = 1 . \tag{23}$$

Since strategy $\pi_1^*[\varepsilon]$ is memoryless, the game from the point of view of player 2 is equivalent to a Markov decision process. The results on Markov decision processes mentioned in Section 2.4.1 ensure that there is a memoryless strategy $\pi_2[\varepsilon]$ realizing the inf in (23). Hence, we can replace $\inf_{\pi_2 \in \Pi_2}$ with $\inf_{\pi_2 \in \Pi_2^M}$ in (23), where Π_2^M is the set of memoryless strategies for player 2. For all $0 \leq i \leq k$, define

$$P_i^{exit}[\varepsilon] = \inf_{\pi_2 \in \Pi_2^M} \min_{0 \leq j \leq k} \Pr_{t_j}^{\pi_1^*[\varepsilon], \pi_2}\Big(\Diamond(U_m \setminus \{t_i, t_{i+1}, \ldots, t_k\})\Big) . \tag{24}$$

The quantity $P_i^{exit}[\varepsilon]$ represents a lower bound on the probability of eventually leaving $\{t_i, t_{i+1}, \ldots, t_k\}$ and proceeding to $U_m \setminus \{t_i, t_{i+1}, \ldots, t_k\}$. Since $R = U \setminus \{t_0, t_1, \ldots, t_k\}$, we can prove (23) by proving that $\lim_{\varepsilon \to 0} P_0^{exit}[\varepsilon] = 1$. To prove the latter result, we show that for every $0 \leq i \leq k$ there is $\kappa_i > 0$ such that, for ε in a right neighborhood of 0,

$$P_i^{exit}[\varepsilon] \geq 1 - \kappa_i \varepsilon^{[(M+2)^i]} . \tag{25}$$

We prove (25) by induction on i, from $i = k$ down to 0. As the base case is a simplified version of the induction step, we concentrate on the latter. To simplify the notation, we let $V_i = \{t_i, t_{i+1}, \ldots, t_k\}$, for $0 \leq i \leq k$.

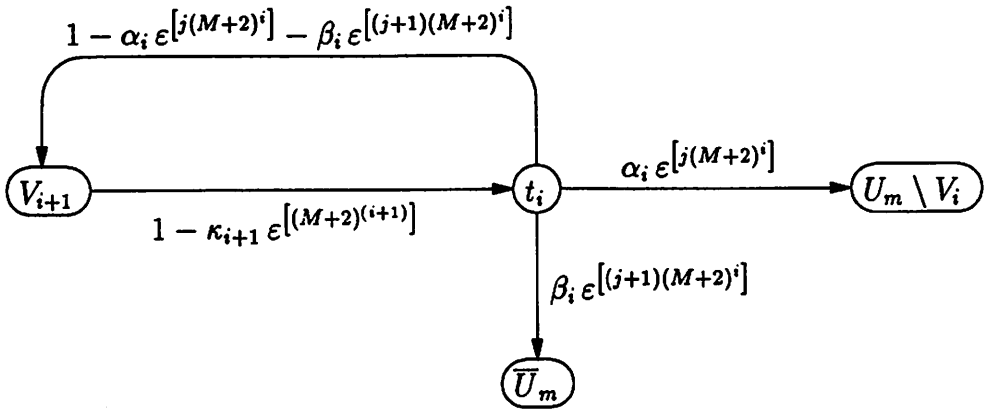

Figure 4: Worst-case transition probabilities for the escape from $V_i = \{t_i\} \cup V_{i+1}$ to $U_m \setminus V_i$.

We now consider the worst-case escape scenario from V_i, for $0 \leq i < k$. By induction hypothesis, the probability of eventual escape from V_{i+1} to $U_m \setminus V_{i+1}$ is at least

$$1 - \kappa_{i+1}\,\varepsilon^{[(M+2)^{(i+1)}]}\,.$$

In the worst case, these escapes lead to t_i, rather than to $U_m \setminus V_i$. Denote by

$$\xi_1^i = \pi_1^*(t_i), \qquad \xi_2^i = \pi_2(t_i)$$

the distributions used by players 1 and 2 at t_i, respectively. By Lemmas 3 and 4 there are $\alpha_i > 0$, $\beta_i < \infty$, and $0 \leq j \leq M$ such that

$$\widetilde{p}(t_i, \xi_1^i, \xi_2^i)(U_m \setminus V_i) \geq \alpha_i\,\varepsilon^{[j(M+2)^i]} \tag{26}$$

$$\widetilde{p}(t_i, \xi_1^i, \xi_2^i)(S \setminus U_m) \leq \beta_i\,\varepsilon^{[(j+1)(M+2)^i]}\,. \tag{27}$$

The worst case is the one in which (26) and (27) hold with equality. Moreover, in the worst case the remaining probability

$$1 - \alpha_i\,\varepsilon^{[j(M+2)^i]} - \beta_i\,\varepsilon^{[(j+1)(M+2)^i]}$$

corresponds to transitions to V_{i+1}, rather than also back to t_i. The worst-case transition probabilities out of V_{i+1} and t_i are summarized in Figure 4. Thus, as $\varepsilon \to 0$ we can write

$$\begin{aligned} P_i^{exit}[\varepsilon] &\geq \frac{\left(1-\kappa_{i+1}\,\varepsilon^{[(M+2)^{(i+1)}]}\right)\alpha_i\,\varepsilon^{[j(M+2)^i]}}{1-\left(1-\kappa_{i+1}\,\varepsilon^{[(M+2)^{(i+1)}]}\right)\left(1-\alpha_i\,\varepsilon^{[j(M+2)^i]}-\beta_i\,\varepsilon^{[(j+1)(M+2)^i]}\right)} \\ &= \frac{\alpha_i\,\varepsilon^{[j(M+2)^i]} + \mathcal{O}\left(\varepsilon^{[(M+2)^{(i+1)}+j(M+2)^i]}\right)}{\alpha_i\,\varepsilon^{[j(M+2)^i]} + \beta_i\,\varepsilon^{[(j+1)(M+2)^i]} + \mathcal{O}\left(\varepsilon^{[(M+2)^{(i+1)}]}\right)} \\ &= \frac{\alpha_i + \mathcal{O}\left(\varepsilon^{[(M+2)^{(i+1)}]}\right)}{\alpha_i + \beta_i\,\varepsilon^{[(M+2)^i]} + \mathcal{O}\left(\varepsilon^{[2(M+2)^i]}\right)}\,. \end{aligned}$$

Finally, from the Taylor series expansion of the last fraction, for ε in a right neighborhood of 0 we have

$$P_i^{exit}[\varepsilon] \geq 1 - 2\beta_i \varepsilon^{[(M+2)^i]} ,$$

which proves (25). This completes the proof of $U_m \subseteq \mathit{Limit}(R)$.

The proof of $\mathit{Limit}(R) \subseteq U_m$ follows the general lines of the proof of Theorem 7. For all $s \in S$, let

$$P_{sup}(s) = \sup_{\pi_1 \in \Pi_1} \Pr_s^{\pi_1, \pi_2^*}(\Diamond R)$$

where π_2^* is the strategy described in Theorem 16. We prove that, for $0 \leq i < m$, if $s \in U_i \setminus U_{i+1}$ then $P_{sup}(s) < 1$. The proof proceeds by complete induction on i, for $0 \leq i < m$. Again, for each $0 \leq i < m$ there are two cases, depending whether $s \in C_i$ or $s \in U_i \setminus (C_i \cup U_{i+1})$.

- $s \in C_i$. At all $t \in C_i$, strategy π_2^* plays with distribution $\mathit{imprison}(t, C_i, U_i)$. For each $t \in C_i$, let $\kappa_t > 0$ be the constant given for t by Lemma 4, and let $\kappa = \min\{\kappa_t \mid t \in C_i\}$. As a consequence of Lemma 4,

  $$\Pr_s^{\pi_1, \pi_2^*}(\Diamond \overline{U}_i) \geq \kappa \Pr_s^{\pi_1, \pi_2^*}(\Diamond \overline{C}_i) .$$

  Let $q = \max\{P_{sup}(t) \mid t \in \overline{U}_i\}$. Denoting by $r = \Pr_s^{\pi_1, \pi_2^*}(\Diamond \overline{C}_i)$, we have

  $$\Pr_s^{\pi_1, \pi_2^*}(\Diamond R) \leq r(1 - \kappa(1-q)) \leq 1 - \kappa(1-q) .$$

  Since $\kappa > 0$ and $q < 1$, we have $1 - \kappa(1-q) < 1$ and $P_{sup}(s) \leq 1 - \kappa(1-q)$, which gives us the desired bound.

- $s \in U_i \setminus (C_i \cup U_{i+1})$. Note that in $U_i \setminus (C_i \cup U_{i+1})$ strategy π_2^* plays uniformly at random moves from Γ_2. Hence, the result follows easily by Lemma 2, and by the induction hypothesis.

The above results prove immediately Theorems 15 and 16 on the winning and spoiling strategies, and hence also Theorem 11(2,3). Theorem 10 follows easily from an analysis of Algorithm 6. ∎

Proof of Theorem 11(1). The correctness of Algorithm 6 is a consequence of the previous arguments. To prove the result about the running time of Algorithm 6, it suffices to show that for each $k \geq 0$, the set $C_k = \mathit{Lim\text{-}safe}(U_k \setminus R, U_k)$ can be computed in linear time. In fact, for each $k \geq 0$ the set $U_{k+1} = \mathit{Safe}_1(U_k \setminus C_k, \Gamma_1, \Gamma_2)$ can also be computed in linear time [AHK97], and the algorithm terminates after a number of iterations bounded by the size of the state space. From the remarks of Section 3.4.3, with small modifications, the argument also shows that the sets C_k of Algorithm 3 can be computed in linear time, for $k \geq 0$.

To show that the set $C_k = \mathit{Lim\text{-}safe}(U_k \setminus R, U_k)$ can be computed in linear time, we examine in more detail the process of eliminating limit-escape states from $U_k \setminus R$. Algorithm 5 is not a very efficient way of computing C_k, since it may invoke the limit-escape test more than once for each state. To obtain a more efficient algorithm, we rely on the following observations.

- To compute C_k, we initially set $V := U_k \setminus R$, and we progressively remove from V the states that are limit-escape w.r.t. V and U_k. Consider a state $s \in V$, with its related bipartite graph $(\Gamma_1(s), \Gamma_2(s), E_1(s), E_2(s))$, where

$$E_1(s) = \Big\{(a,b) \in \Gamma_1(s) \times \Gamma_2(s) \;\Big|\; \delta(s,a,b) \not\subseteq V\Big\}, \tag{28}$$

$$E_2(s) = \Big\{(b,a) \in \Gamma_2(s) \times \Gamma_1(s) \;\Big|\; \delta(s,a,b) \not\subseteq U_k\Big\}. \tag{29}$$

  Suppose that state $t \in V \setminus \{s\}$ is eliminated from V, and let $V' = V \setminus \{t\}$. If $E_1'(s)$ is defined similarly to (28) but with respect to V' instead of V, we have $E_1(s) \subseteq E_1'(s)$. Hence, as we remove limit-escape states from V, the sets of edges $E_1(\cdot)$ for the remaining states in V increase monotonically.

- Given $\Gamma_1(s)$, $\Gamma_2(s)$ and the two sets of edges $E_1(s)$ and $E_2(s)$, let $\mathcal{A}(s)$ and $\mathcal{B}(s)$ be the sets of moves satisfying the fixed-point characterization given by Lemma 1. Suppose that new edges are added to $E_1(s)$, yielding $E_1'(s)$. The new sets $\mathcal{A}'(s)$ and $\mathcal{B}'(s)$ computed with respect to $E_1'(s)$ and $E_2(s)$ are such that $\mathcal{A}(s) \subseteq \mathcal{A}'(s)$ and $\mathcal{B}(s) \subseteq \mathcal{B}'(s)$. Jointly with the previous observation, this indicates that as we remove limit-escape states from V, the sets of labeled moves at the other states in V increase monotonically.

These observations lead to the following algorithm for the computation of $C_k = \mathit{Lim\text{-}safe}(U_k \setminus R, U_k)$.

Algorithm 8
Input: Game structure G, and two sets $W \subseteq U \subseteq S$ of states.
Output: $\mathit{Lim\text{-}safe}''(W,U) \subseteq S$.

Initialization: Set $V := W$. For each $s \in V$, construct the sets of edges

$$E_1(s) := \Big\{(a,b) \in \Gamma_1(s) \times \Gamma_2(s) \;\Big|\; \delta(s,a,b) \not\subseteq V\Big\},$$

$$E_2(s) := \Big\{(b,a) \in \Gamma_2(s) \times \Gamma_1(s) \;\Big|\; \delta(s,a,b) \not\subseteq U\Big\},$$

and let $\mathcal{A}(s)$ and $\mathcal{B}(s)$ be the least subsets of $\Gamma_1(s)$, $\Gamma_2(s)$ respectively that satisfy:

1. for all $a \in \Gamma_1(s)$, if $\{b \mid (b,a) \in E_2(s)\} \subseteq \mathcal{B}(s)$, then $a \in \mathcal{A}(s)$;
2. for all $b \in \Gamma_2(s)$, if there is $a \in \mathcal{A}(s)$ with $(a,b) \in E_1(s)$, then $b \in \mathcal{B}(s)$.

While there is $t \in V$ such that $\mathcal{B}(t) = \Gamma_2(t)$ do:

1. Let $V' := V \setminus \{t\}$.
2. For each $s \in V'$, let

$$E_1'(s) := E_1(s) \cup \Big\{(a,b) \in \Gamma_1(s) \times \Gamma_2(s) \;\Big|\; \delta(s,a,b) \subseteq V \wedge \delta(s,a,b) \not\subseteq V'\Big\}.$$

3. For each $s \in V'$, update the sets $\mathcal{A}(s)$ and $\mathcal{B}(s)$ by labeling additional moves, until the resulting sets $\mathcal{A}'(s)$ and $\mathcal{B}'(s)$ are the least sets satisfying Properties 1 and 2 above with respect to the sets of edges $E_1'(s)$ and $E_2(s)$.
4. Rename $V := V'$, and for all $s \in V$ rename $E_1(s) := E_1'(s)$, $\mathcal{A}(s) := \mathcal{A}'(s)$, and $\mathcal{B}(s) := \mathcal{B}'(s)$.

Return: V.

From the above considerations, it is not difficult to see that $\mathit{Lim\text{-}safe}''(U_k \setminus R, U_k) = \mathit{Lim\text{-}safe}(U_k \setminus R, U_k)$. By introducing appropriate bookkeeping in Algorithm 8, we can ensure that the changes in the sets of edges and labeled moves are propagated gradually. Specifically, whenever a state t is removed from V, the removal can be propagated (by tracking backwards the combinations of moves that can lead to t), yielding the additional edges described in Step 2 of the algorithm. In turn, the introduction of the new edges can be used to trigger the propagation of move labelings in Step 3. Finally, once a state t has $\mathcal{B}(t) = \Gamma_2(t)$, the state becomes a candidate for removal from V at some following iteration. We can implement this propagation process so that no move, edge, or state has to be considered more than once, leading to an algorithm with linear running time in the size of the game. ∎

5.4 Randomized ATL

Proof of Theorem 12. Without loss of generality, consider a two-player game structure, and assume that the set of agents is $A = \{1\}$; i.e., it consists of the first player. The case for $\psi \equiv \bigcirc\varphi$ can be proved by inspection. If $\psi \equiv \Box\varphi$, let $U = \{s \in S \mid s \models \varphi\}$ be the set of states where φ holds. Clearly,

$$s \models \langle\!\langle 1 \rangle\!\rangle_{\mathit{sure}} \Box\varphi \;\; \mathit{iff} \;\; s \in \mathit{Safe}_1(U, \Gamma_1, \Gamma_2)\,. \tag{30}$$

By Lemma 2, the probability of staying forever in U (which corresponds to the probability of satisfying $\Box U$) is bounded away from 1 for all states of $U \setminus \mathit{Safe}_1(U, \Gamma_1, \Gamma_2)$. Hence, we have also

$$s \models \langle\!\langle 1 \rangle\!\rangle_{\mathit{almost}} \Box\varphi \;\; \mathit{iff} \;\; s \in \mathit{Safe}_1(U, \Gamma_1, \Gamma_2)\,,$$

$$s \models \langle\!\langle 1 \rangle\!\rangle_{\mathit{limit}} \Box\varphi \;\; \mathit{iff} \;\; s \in \mathit{Safe}_1(U, \Gamma_1, \Gamma_2)$$

which, by comparison with (30), completes the argument. ∎

Proof of Theorem 13. Again without loss of generality, consider a two-player game structure, and a formula $\langle\!\langle 1 \rangle\!\rangle_{\mathit{almost}} \varphi_1 \mathcal{U} \varphi_2$ (the case for $\langle\!\langle 1 \rangle\!\rangle_{\mathit{limit}} \varphi_1 \mathcal{U} \varphi_2$ is analogous). To justify modification (17), note that the set C_0 of (17) can be written as $C_0 = C_0' \cup C_0''$, where

$$C_0' = \mathit{Safe}_2\big(\{s \in S \mid s \not\models \varphi_2\}, \Gamma_1, \Gamma_2\big)$$

is the set that would have been computed as C_0 by Algorithms 3 and 6 prior to their modification, and

$$C_0'' = \{s \in S \mid s \not\models \varphi_1 \vee \varphi_2\}$$

is the set of states that do not satisfy neither φ_1 nor the eventuality φ_2. Note also that, while $C_0' \cap R = \emptyset$, it can be $C_0'' \cap R \neq \emptyset$.

First, consider a winning state $s \in \mathit{Almost}(R)$, and let π_1 be a winning strategy π_1 for player 1. Under strategy π_1, we have for all $\pi_2 \in \Pi_2$:

$$\Pr_s^{\pi_1,\pi_2}(\Diamond R) = 1\,, \qquad \Pr_s^{\pi_1,\pi_2}(\Diamond C_0) = 0$$

where the second equality is a consequence of the structure of Algorithm 3 and of Theorem 9. In turn, these two equalities imply that the eventuality φ_2 will be satisfied with probability 1, and that φ_1 will hold at all times prior to the satisfaction of φ_2, regardless of the strategy used by player 2. Hence, we conclude $s \models \langle\langle 1 \rangle\rangle_{almost} \varphi_1 \mathcal{U} \varphi_2$.

Consider then a state $s \notin \mathit{Almost}(R)$. By repeating the arguments used in the proof of Theorem 7(1), we can prove that player 2 has a spoiling strategy π_2 such that, for all $\pi_1 \in \Pi_1$,

$$\left[1 - \Pr_s^{\pi_1,\pi_2}(\Diamond R)\right] + \Pr_s^{\pi_1,\pi_2}(\Diamond C_0) > 0\,.$$

If $\Pr_s^{\pi_1,\pi_2}(\Diamond R) < 1$, then $\Pr_s^{\pi_1,\pi_2}([\![\varphi_1 \mathcal{U} \varphi_2]\!]) < 1$. If $\Pr_s^{\pi_1,\pi_2}(\Diamond R) = 1$, then it must be

$$\Pr_s^{\pi_1,\pi_2}(\Diamond C_0) = \Pr_s^{\pi_1,\pi_2}(\Diamond C_0'') > 0\,,$$

implying that with positive probability φ_1 becomes false before φ_2 becomes true. Also in this case, we have $\Pr_s^{\pi_1,\pi_2}([\![\varphi_1 \mathcal{U} \varphi_2]\!]) < 1$. Hence, we conclude $s \not\models \langle\langle 1 \rangle\rangle_{almost} \varphi_1 \mathcal{U} \varphi_2$. ∎

Proof of Corollary 2. The corollary is an immediate consequence of Theorem 13, along with Theorems 7(1) and 11(1) on the complexity of computing the sets of almost-sure and limit-sure reachability states. ∎

Acknowledgments. We thank Rajeev Alur, Jerzy Filar, Christos Papadimitriou, T.E.S. Raghavan, Valter Sorana, and Mihalis Yannakakis for helpful discussions and pointers to the literature.

References

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pages 100-109, 1997.

[BCM+92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Information and Computation*, 95(2):142-170, 1992.

[Bee80] C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. on Database Systems*, 5:241–259, 1980.

[Ber95] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.

[BO82] T. Başar and G.J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, 1982.

[BT91] D.P. Bertsekas and J.N. Tsitsiklis. An analysis of stochastic shortest path problems. *Math. of Op. Res.*, 16(3):580–595, 1991.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lect. Notes in Comp. Sci.*, pages 52–71. Springer-Verlag, 1981.

[CS91] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal μ-calculus. In *Computer Aided Verification*, volume 575 of *Lect. Notes in Comp. Sci.*, pages 48–58. Springer-Verlag, 1991.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proc. 29th IEEE Symp. Found. of Comp. Sci.*, pages 338–354, 1988.

[dA97] L. de Alfaro. *Formal verification of probabilistic systems.* PhD thesis, Stanford University, 1997. Technical Report STAN-CS-TR-98-1601.

[Der70] C. Derman. *Finite State Markovian Decision Processes.* Acedemic Press, 1970.

[Eme90] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[Eve57] H. Everett. Recursive games. In *Contributions to the Theory of Games III*, volume 39 of *Annals of Mathematical Studies*, pages 47–78, 1957.

[Fil81] J.A. Filar. Ordered field property for stochastic games when the player who controls transitions changes from state to state. *J. Optimization Theory and Applications*, 34(4):503–515, 1981.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes.* Springer-Verlag, 1997.

[Gil57] D. Gillette. Stochastic games with zero stop probabilities. In *Contributions to the Theory of Games III*, volume 39 of *Annals of Mathematical Studies*, 1957.

[HSP83] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Trans. Prog. Lang. Sys.*, 5(3):356–380, July 1983.

[Imm81] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.

[Isa65] R. Isaacs. *Differential Games.* John Wiley, 1965.

[Jon75] N.D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11:68–75, 1975.

[KS81] P.R. Kumar and T.H. Shiau. Existence of value and randomized strategies in zero-sum discrete-time stochastic dynamic games. *SIAM J. Control and Optimization*, 19(5):617–634, 1981.

[KSK66] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains.* D. Van Nostrand Company, 1966.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1991.

[vN28] J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Math. Annal*, 100:295–320, 1928.

[QS81] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symp. on Programming*, Lecture Notes in Computer Science, volume 137, pages 337–351. Springer-Verlag, 1981.

[RF91] T.E.S. Raghavan and J.A. Filar. Algorithms for stochastic games — a survey. *ZOR — Methods and Models of Op. Res.*, 35:437–472, 1991.

[Sec97] P. Secchi. Stationary strategies for recursive games. *Math. of Op. Res.*, 22(2):494–512, 1997.

[TV87] F. Thuijsman and O.J. Vrieze. The bad match, a total reward stochastic game. *Operations Research Spektrum*, 9:93–99, 1987.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. Proc. 26th IEEE Symp. on Foundations of Computer Science, pages 327–338, 1985.

[VTRF83a] O.J. Vrieze, S.H. Tijs, T.E.S. Raghavan, and J.A. Filar. A finite algorithm for the switching controller stochastic game. *OR Spectrum*, 5:15–24, 1983.

[Yan98] M. Yannakakis. Personal communication, 1998.

[ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comp. Sci.*, 158:343–359, 1996.