THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

The Stochastic Arrival Problem

Citation for published version:
Webster, T 2022, The Stochastic Arrival Problem. in AW Lin, G Zetzsche & I Potapov (eds), *Proceedings of the 16th International Conference on Reachability Problems.* Lecture Notes in Computer Science, vol. 13608, Springer, Cham, pp. 93-107, The 16th International Conference on Reachability Problems, 2022, Kaiserslautern, Germany, 17/10/22. https://doi.org/10.1007/978-3-031-19135-0

Digital Object Identifier (DOI):
10.1007/978-3-031-19135-0

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 16th International Conference on Reachability Problems

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 18. May. 2024

The Stochastic Arrival Problem

Thomas Webster[1]

University of Edinburgh, Edinburgh, UK
thomas.webster@ed.ac.uk

Abstract. We study a new modification of the `Arrival` problem, which allows for nodes that exhibit random as well as controlled behaviour, in addition to switching nodes. We study the computational complexity of these extensions, building on existing work on Reachability Switching Games. In particular, we show for versions of the arrival problem involving just switching and random nodes it is PP-hard to decide if their value is greater than a half and we give a PSPACE decision algorithm.

Keywords: Arrival · Markov Chains · Reachability Switching Games · MDPs · Simple Stochastic Games

1 Introduction

`Arrival` is a simple to describe decision problem defined by Dohrau, Gärtner, Kohler, Matoušek and Welzl [4]. In simplistic terms, it asks whether a train moving along the vertices of a given directed graph, with n vertices, will eventually reach a given target vertex, starting at a given start vertex. At each vertex, v, the train moves deterministically, based on a given listing of outgoing edges of v, taking the first out-edge, then the second, and so on, as it revisits that vertex repeatedly, until the listing is exhausted after which it restarts cyclically at the beginning of the listing of outgoing edges again. This process is known as "switching" and can be viewed as a deterministic simulation of a random walk on the directed graph. It can also be viewed as a natural model of a state transition system where a local deterministic cyclic scheduler is provided for repeated transitions out of each state.

Dohrau et al. showed this `Arrival` decision problem lies in the complexity class $\mathsf{NP} \cap \mathsf{coNP}$, but it is not known to be in P. There has been a lot of recent work, showing that a search version of the `Arrival` problem lies in sub-classes of TFNP including PLS [12], CLS [8], and UniqueEOPL [7], as well as showing that `Arrival` is in $\mathsf{UP} \cap \mathsf{coUP}$ [8]. There has also been work on lower bounds, including PL-hardness and CC-hardness [13]. Further recent work by Gärtner et al [9] gives an algorithm for `Arrival` with running time $2^{\mathcal{O}(\sqrt{n}\log(n))}$, the first known sub-exponential algorithm. In addition, they give a polynomial-time algorithm for "almost acyclic" instances.

The complexity of `Arrival` is particularly interesting in the context of other games on graphs, such as Condon's simple stochastic games, mean-payoff games, and parity games [2, 16, 11], for which the two-player variants are known to be

in NP ∩ coNP, whereas the one-player variants have polynomial time algorithms. Arrival however is a zero-player game which has no known polynomial time algorithm and furthermore it was shown by Fearnley et al. [6] that a one-player generalisation of arrival is in fact NP-complete, in stark contrast to these two-player graph games.

Further generalisations of `Arrival` to Reachability Switching Games were considered, adding player controlled nodes to the game, by Fearnley, Gairing, Mnich and Savani [6]. We provide a further generalisation, by introducing probabilistic nodes, out of which we have random transitions according to a given probability distribution, thus combining the elements of Fearnley et al. [6] and those of Condon's [2], by allowing a mixture of randomisation, switching, and controlled or game behaviour.

Some of our main results consider a mixture of switching and randomisation. In this case we show there is an exponential upper bound on the expected termination time of such a switching run. We also show that deciding whether the value is greater than 0 (or equal to 1 resp.) is complete for NP (resp. coNP) and that the quantitative decision problem is both hard for PP, under many-one (Karp) reductions, and contained in PSPACE thus showing it is harder than the single player switching games of Fearnley et al. [6]. We also give hardness results for the natural generalisation with players, showing these are hard for PSPACE. Some simpler upper bounds follow from viewing these as succinctly presented instances of MDPs, or Condon's simple stochastic games. A full summary of our complexity results (and prior complexity results) can be found in Table 1.

Due to space limits, most proofs are relegated to the appendix.

2 Preliminaries

Our arrival instances represent a reachability problem in a given directed graph, $G = (V, E)$, with given start and target vertices $s, t \in V$, and where the nodes V are partitioned into different types according to a given partition $\mathcal{V}$, with nodes of each type having slightly different behaviour. We use $d_{\text{in}}(v)$ and $d_{\text{out}}(v)$ to represent the in-degree and out-degree of a vertex v in a directed graph. Four distinct types of nodes may be contained in $\mathcal{V}$:

- **Probabilistic nodes** - We denote the set of probabilistic nodes by $V_R \in \mathcal{V}$, and we require a probability distribution, P, to be given on their outgoing edges. These are sometimes also called as random, stochastic or nature nodes.
- **Switching nodes** - We call the set of switching nodes $V_S \in \mathcal{V}$, and require an ordering, *Ord*, to be given on their outgoing edges.
- **Max Player nodes** - We call the set of max player nodes $V_1 \in \mathcal{V}$ at which choices are controlled by a player aiming to reach t. These are also referred to as player 1 nodes.
- **Min Player nodes** - We call the set of min player nodes $V_2 \in \mathcal{V}$ at which choices are controlled by a player aiming to avoid t. These are also referred to as player 2 nodes.

Our instances then have the following structure.

Definition 1. *An instance of an arrival problem has the following signature* $(V, E, s, t, \mathcal{V}, P, \mathit{Ord})$ *where:*

- (V, E) *is a finite directed graph.*
- $s, t \in V$. s *is called the* start *and* t *the* target *node.*
- *For all* $v \in V$, *we require* $d_{out}(v) \geq 1$, *and we allow self loop edges of the form* (v, v).
- *For* t *we require* $(t, v) \in E \implies v = t$, *i.e. the only out-edge at the target is a self-loop.*
- $\mathcal{V} \subseteq \mathcal{P}(V)$ *is a partition of the vertices of* $V - \{t\}$ *into different node types. Often we will take* $\mathcal{V} = \{V_R, V_S, V_1, V_2\}$, *omitting empty sets, with each of these sets as described above.*
- *A function* $P : V_R \times V \to [0, 1]$ *with the properties that for any* $v \in V_R$ *we have* $\sum_{w \in V} P(v, w) = 1$ *and where* $P(v, w) > 0$ *if and only if* $(v, w) \in E$.
- *A function* $\mathit{Ord} : V_S \to V^+$ *from switching nodes to a finite sequence of vertices. We require that, for* $v \in V_S$, $(v, w) \in E$ *if and only if there exists an* i *such that* $w = \mathit{Ord}(v)_i$. *So, every outgoing edge from* v *is "used" in* $\mathit{Ord}(v)$, *but can be used more than once.*

Given such a model, we wish to define a play of the game. To do so we first need to define the current state. Due to how switching nodes work we will also include the current positions of those nodes into our game state.

Definition 2. *Given a set of switching nodes* V_S *the* current switching node position *is a function* $q : V_S \to \mathbb{N}_0$, *i.e., a function from vertices to natural numbers, where we require that* $\forall v \in V_S$, $q(v) < |\mathit{Ord}(v)|$. *We call the set of all such position functions* Q. *If there are no switching vertices then* Q *is a singleton containing only the empty function.*

Definition 3. *A* state of the game *consists of an ordered pair* $(v, q) \in V \times Q$ *with* $v \in V$ *denoting the current vertex, and* $q \in Q$, *denoting the current position of the switching nodes (Definition 2). Thus we call the set* $V \times Q$ *our state space.*

Now that we have a state space we can define valid transitions between states.

Definition 4. *We let* $\mathit{Valid} : V \times Q \to \mathcal{P}(V \times Q)$ *be the function defined as follows:*

- *For* $v \in V_S$ *and any* $q \in Q$, *where by definition* $q : V_S \to \mathbb{N}_0$, *we define* $\mathit{Valid}(v, q) := \{(u, q')\}$, *where* u *and* q' *are defined as follows:*
  - *Suppose* $\mathit{Ord}(v) = (u_0, \ldots, u_{k-1})$. *We let* $u := u_{q(v)}$. *Note that this is well defined, i.e.,* $0 \leq q(v) < |\mathit{Ord}(v)| = k$, *because* (v, q) *is a state.*
  - *For* $x \in V_S$ *with* $x \neq v$ *we let* $q'(x) := q(x)$.
  - *Furthermore, we let* $q'(v) := (q(v) + 1 \mod k)$.
- *For* $v \in V_1 \cup V_2$ *and any* $q \in Q$, *we let* $\mathit{Valid}(v, q) := \{(u, q) : (v, u) \in E\}$.
- *For* $v \in V_R$ *and any* $q \in Q$ *we let* $\mathit{Valid}(v, q) := \{(u, q) : P(v, u) > 0\}$

We call a transition from a state (v, q) *to a state* $(u, q') \in \mathit{Valid}(v, q)$ valid*, and otherwise we call it* invalid*.*

It follows directly from the definitions that for any state (v, q), $\mathit{Valid}(v, q) \neq \emptyset$.

We call an infinite sequence $\pi = (v_0, q_0)(v_1, q_1)(v_2, q_2) \cdots \in (V \times Q)^\omega$ over the state space $V \times Q$ a *play* if for every $i \in \mathsf{N}_0$ we have $(v_{i+1}, q_{i+1}) \in \mathit{Valid}(v_i, q_i)$. We use Ω to denote the set of all (infinite) plays. A *partial play* of the game is a finite initial prefix $w \in (V \times Q)^*$ of a play. For a partial play w, we define its *basic cylinder*, $\mathsf{C}(w) \subseteq w(V \times Q)^\omega$, as the set of all plays with w as an initial segment. We use $\mathit{\Pi} \subseteq (V \times Q)^*$ to denote the set of all finite partial plays. We say a play π is *winning* for player 1 if there exists some index i with $\pi_i = (t, q)$. Otherwise, it is a losing play (winning for player 2).

It follows from known results, namely, deterministic memoryless determinacy of simple stochastic games ([2]), that for all our generalised arrival games it suffices to consider deterministic "essentially memoryless" strategies for a player i given by $\mathit{Strat}_i : (V_i \times Q) \to V$, which ignore the history in a partial play π, and only considers the current state (v, q) in order to choose (deterministically) a move to the next vertex, v', such that $(v', q) \in \mathit{Valid}(v, q)$. (Note that switching positions only change during transitions out of switching nodes.) Indeed, we can view our instances of generalised arrival as defining exponentially larger simple stochastic games over the state space $V \times Q$, because of the deterministic way the switching position q updates with each transition.

Fixing a start state s, and strategies σ_1 and τ_2 for the two players, naturally determines a probability space $(\Omega, \mathcal{F}, \mathsf{P}_{\sigma_1,\tau_2})$ on the set Ω of (infinite) plays. Here $\mathcal{F}$ denotes the Borel σ-algebra of events generated by the set of basic cylinders $\{\mathsf{C}(w) \mid w \in \mathit{\Pi}\}$, and $\mathsf{P}_{\sigma_1,\tau_2}$ denotes the probability measure defined on events in $\mathcal{F}$ uniquely determined by probabilities of basic cylinders, which are defined inductively in the standard way, starting with the base case given by $\mathsf{P}(\mathsf{C}((s, q_0))) := 1$, where by definition $q_0(v) := 0$ for all $v \in V_S$. In other words, all plays begin, with probability 1, with state (s, q_0) as the initial state.

Definition 5. *Given an instance* $G = (V, E, s, t, \{V_R, V_S, V_1, V_2\}, P, \mathit{Ord})$ *we define the* value of the instance *as follows. Let* $\mathit{Reach} \in \mathcal{F}$ *be the event* $\mathit{Reach} := \{\pi = (s, q_0)(v_1, q_1)(v_2, q_2) \ldots \in \Omega : \exists i \in \mathsf{N}_0, v_i = t\}$ *and let* σ_1 *and* τ_2 *range over strategies for each player:*

$$\mathit{val}(G) := \max_{\sigma_1} \min_{\tau_2} \mathsf{P}_{\sigma_1,\tau_2}(\mathit{Reach})$$

We may sometimes refer to the value $\mathit{val}(G)$ *as the \winning probability" (for player 1).*

It follows from known results for simple stochastic games that these games are determined, meaning that $\mathit{val}(G) = \min_{\tau_2} \max_{\sigma_1} \mathsf{P}_{\sigma_1,\tau_2}(\mathit{Reach})$ and that these maxima and minima are obtained.

We generalise of the notion of a "hopeful edges" of [4, Definition 3]:

Definition 6. *Given an instance $G := (V, E, s, t, \mathcal{V}, P, \mathsf{Ord})$ we say a vertex $v \in G$ is* hopeful *if Player 1 can win the reachability game $(V, E, v, t, \{V_1', V_2\})$, where $V_1' := V_R \cup V_S \cup V_1$ and v is our start vertex. We call an edge $(v, w) \in E$ a hopeful edge if w is a hopeful vertex. A vertex or edge which isn't hopeful is called* dead.

We note that we can decide whether $v \in G$ is hopeful in NL if we have no player 2 nodes in G, and otherwise in P by solving the 2-player reachability game. We now define different versions of the computational problems we wish to study, using a common notation. We use a subset $B \subseteq \{R, S, 1, 2\}$ to denote the different kinds of nodes that are present in the instances for the problem in question.

Definition 7. *For a subset $B \subseteq \{R, S, 1, 2\}$, given an instance structure $G = (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$, we define the following associated decision problems. Let $val(G)$ be the value of the underlying arrival/reachability game associated with G, and let $p \in (0, 1)$ be a (rational) probability given as part of the input. We define three variants of quantitative and qualitative B-Arrival decision problems that we wish to study:*

- B-`Arrival-Quant`*: Decide whether $val(G) > p$.*
- B-`Arrival-Qual-0`*: Decide whether $val(G) > 0$.*
- B-`Arrival-Qual-1`*: Decide whether $val(G) = 1$.*

The original arrival problem studied in [4] corresponds to the above definition with $B = \{S\}$. Reachability Switching Games defined in [6] correspond to $B = \{S, 1\}$ and $B = \{S, 1, 2\}$. Taking $B \subseteq \{R, 1, 2\}$ corresponds to Markov Chains, Markov Decision Processes and Simple Stochastic Games.

We note that when $R \notin B$ these problems all coincide, since in that case $val(G) \in \{0, 1\}$ and such instances constitute an (exponentially large) deterministic reachability game. In such a case we use B-`Arrival` to refer to the problem of deciding if $val(G) = 1$. Several of these problems have previously known complexity. Throughout this work we aim to show complexity results for the cases when $R \in B$. When referring to an instance of some variant of the above arrival problems, with node types B, we use the expression "instance of a generalised B-arrival problem". It is not hard to show:

Proposition 1. *Given an instance of a generalised B-arrival problem $G = (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$, with $R \in B$, and given any rational $p \in (0, 1)$, the decision problem B-`Arrival-Quant` is polynomial-time equivalent to B-`Arrival-Quant` where $p = 1/2$.*

Hence we will use B-`Arrival-Quant` to refer to the quantitative arrival problem when $p = \frac{1}{2}$, and it suffices to only consider this quantitative decision problem. The complexity status of the various different arrival problems, including the results established in this paper, is summarized in Table 1, with references to the original works, or to specific results in this paper that establish it.

While Fearnley et al. do not explicitly consider the $\{S, 2\}$-`Arrival` problem in [6] we are able to deduce NP-completeness using their results and our generalised notion of hopefulness.

Table 1: Complexity of `Arrival` variants with different node types.

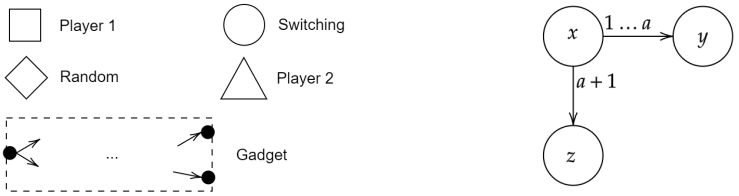
Problem Name	Known Complexity	Cite
$\{S\}$-`Arrival`	PL-hard, CC-hard (explicit input) P-hard (succinct input) in UEOPL, CLS,PLS, UP ∖ coUP	[13] [6] [8, 4]
$\{S; 1\}$-`Arrival`	NP-complete	[6]
$\{S; 2\}$-`Arrival`	NP-complete	Proposition 2
$\{S; 1; 2\}$-`Arrival`	PSPACE-hard in EXPTIME	[6] [6]
$\{R; S\}$-`Arrival-Qual-0`	NP-complete	Theorem 2
$\{R; S\}$-`Arrival-Qual-1`	coNP-complete	Theorem 4
$\{R; S\}$-`Arrival-Quant`	PP-hard, in PSPACE	Theorem 6, Theorem 5
$\{R; S; 1\}$-`Arrival-Qual-0`	NP-complete	Theorem 2
$\{R; S; 1\}$-`Arrival-Qual-1`	coNP-hard, in EXPTIME	Theorem 4
$\{R; S; 1\}$-`Arrival-Quant`	PSPACE-hard, in EXPTIME	Theorem 1
$\{R; S; 2\}$-`Arrival-Qual-0`	equiv $\{S; 1; 2\}$-`Arrival`	Theorem 3
$\{R; S; 2\}$-`Arrival-Qual-1`	in EXPTIME	Proposition 5
$\{R; S; 2\}$-`Arrival-Quant`	PSPACE-hard, in EXPTIME	Corollary 2
$\{R; S; 1; 2\}$-`Arrival-Qual-0`	equiv $\{S; 1; 2\}$-`Arrival`	Theorem 3
$\{R; S; 1; 2\}$-`Arrival-Qual-1`	in NEXPTIME ∖ coNEXPTIME	Proposition 5
$\{R; S; 1; 2\}$-`Arrival-Quant`	PSPACE-hard, in NEXPTIME ∖ coNEXPTIME	Theorem 1, Proposition 5

Proposition 2. *The* $\{S, 2\}$-`Arrival` *problem is* NP*-complete.*

We need a convenient notation for drawing instances of arrival diagrammatically. To do so we use the shapes shown in Figure 1a to distinguish the different node types. We also make use of gadgets, shown in dashed lines, which are repeated pieces of smaller graphs performing a specific function. Gadgets are shown with entry and exit ports and are permitted to contain other gadgets in a non-recursive way. At probabilistic nodes we assume there is a uniform distribution over outgoing edges, otherwise we label each edge with the probability assigned to it. At switching nodes we label each outgoing edge with numbers such that if $\mathsf{Ord}(x) = u_0 \dots u_k$ we label an edge (x, y) with all the indices, i, such that $u_i = y$. For instance in Figure 1b we have $k = a + 1$ and $\mathsf{Ord}(x) = y \dots yz$, with a consecutive y's.

2.1 Preliminary Results

We may assume Arrival instances have simplified forms, any instance may be transformed in polynomial-time to an equivalent form by Lemma 2. In our simplified form we have two distinguished vertices t and d, with a single self-loop edge. Every other $v \in V \setminus \{t, d\}$ has $d_{\text{out}}(v) = 2$ and $(v, v) \notin E$. For $v \in V$ and $(v, w) \in E$ we have $P(v, w) = \frac{1}{2}$ and for $v \in V_S$ we have $|\mathsf{Ord}(v)| = 2$



(a) Different node types. (b) Switching Orders.

Fig. 1: Pictorial representations of B-Arrival instances.

and there exists functions $s_0, s_1 : V_S \to V$ with $(v, s_0(v)), (v, s_1(v)) \in E$, $\mathsf{Ord}(v) = s_0(v)s_1(v)$ and $s_0(v) \neq s_1(v)$.

We may also view a generalised Arrival instance, G, as concise ways of specifying a expanded (exponentially larger) game, $Exp(G)$, without switching. These results are shown in the appendix Lemmas 2 and 3. Using this fact we can establish lower bounds on how close the value of such an instance can be to zero, without being equal to zero. Namely, if $val(G)$ is not 0, then, $val(G) = \Omega(2^{2^{n}})$ where n is our instance bit encoding size.

Corollary 1. *The value of an instance G of a generalised B-arrival problem is a rational number $val(G) := \frac{p}{q}$ which in lowest terms has $0 \leq p, q \leq 4^k$ with $k = 2n(|V| \times M^{|V_S|})$ with $M = \max_{v \in V_S} |\mathsf{Ord}(v)|$.*

However we can show that we can actually obtain a value of this small magnitude, even in the case where we only have $B = \{R, S\}$.

Proposition 3. *For any positive integer n, we can construct an instance G of the generalised B-arrival problem containing node types $B = \{R, S\}$, such that G has encoding size $O(n)$, and such that $val(G)$ is a positive value that is at most $\frac{1}{2^{2^n}}$.*

We note that, just as in the case of simple stochastic games, we could force these games to terminate, i.e., reach either the target t or dead end d, by modifying them by applying a small discount, ending the game with a small probability after each step. However, unlike the situation with simple stochastic games, even applying a very small discount of the form $\frac{1}{2^{poly(n)}}$ can change the value of the game drastically (taking a value close to 1 down to a value close to zero). We can however use Proposition 3 to reduce a version of the quantitative B-arrival problem with greater than or equals to the strict inequality decision problem:

Proposition 4. *Given an instance of a generalised B-arrival problem $G = (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$, with $R \in B$, and given any rational $p \in (0, 1)$, deciding whether $val(G) \geq p$ is polynomial-time equivalent to B-`Arrival-Quant` where $p = 1/2$, i.e., to deciding whether $val(G) > 1/2$.*

We can also see from interpreting these models as succinct representations of exponentially large Markov chains, MDPs, and simple stochastic games, respectively, that we have the following simple upper bounds on these problems.

Proposition 5. *The* $\{R, S, 1\}$-`Arrival-Quant` *and* $\{R, S, 2\}$-`Arrival-Quant` *problems are contained in* EXPTIME *and the* $\{R, S, 1, 2\}$-`Arrival-Quant` *is contained in* NEXPTIME $\cap$ coNEXPTIME.

3 PSPACE-hardness with three or more node types

Here we show that $\{R, S, 1\}$-`Arrival-Quant` and $\{R, S, 2\}$-`Arrival-Quant` are both hard for PSPACE. Our proof takes inspiration from Fearnley et al.'s proof of PSPACE-hardness for $\{S, 1, 2\}$-`Arrival` ([6, Theorem 4.3]), but requires some new tricks. From these results it trivially follows that $\{R, S, 1, 2\}$-`Arrival-Quant` is also PSPACE-hard.

To show the $\{R, S, 1\}$-`Arrival-Quant` is PSPACE-hard we reduce from the SSAT problem as defined by Papadimitriou ([14]):

Definition 8 (SSAT). *Given a 3CNF Boolean formula* $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ *with three literals per clause, involving variables* $x_1, \ldots, x_n$*, where* n *is even, we are asked whether there is a choice of Boolean value for* x_1 *such that, for a random choice (with probability of true and false each equal to* $\frac{1}{2}$*) of truth value for* x_2*, there is a choice for* x_3*, etc., so that the probability that* φ *comes out true under these choices is greater than* $1/2$*. We denote this as follows (read* R *as \for uniformly random"):*

$$\exists x_1 \mathsf{R} x_2 \exists x_3 \ldots \mathsf{R} x_n [\mathsf{P}(\varphi(x_1, \ldots, x_n) = \top) > \frac{1}{2}] \tag{1}$$

By [14, Theorem 2] this problem is PSPACE-complete. Our aim is to take an instance of SSAT and construct an instance $G(\varphi)$ of generalised $\{R, S, 1\}$-`Arrival` with the following property:

$$val(G(\varphi)) = \max_{x_1}[\mathsf{E}_{x_2}[\max_{x_3}[\ldots \mathsf{E}_{x_n}[\chi[\varphi(x_1, \ldots, x_n) = \top] \ldots] \tag{2}$$

Where χ represents the indicator function for an event. With this we can see that $val(G(\varphi)) > 1/2$ if and only if (1) holds. We now outline this construction and show it can be performed efficiently, and that the value is as required.

Given an instance of SSAT with 3CNF φ, n variables and m clauses, we construct the instance $G(\varphi)$ of generalised $\{R, S, 1\}$-`arrival` shown in Figure 2 where each of the boxes represents the gadgets shown in Figures 3 and 4, respectively and the values a_i, b_i and D are computable from the formula φ.

We now explain this construction in more detail. Given $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, to begin with, in polynomial time we enumerate our n variables as $x_1, \ldots, x_n$ and for each we compute constants $a_i = |\{l \in \{1, \ldots, m\} \mid x_i \in C_l\}|$ and $b_i = |\{l \in \{1, \ldots, m\} \mid \neg x_i \in C_l\}|$. Here a_i is the number of clauses in which

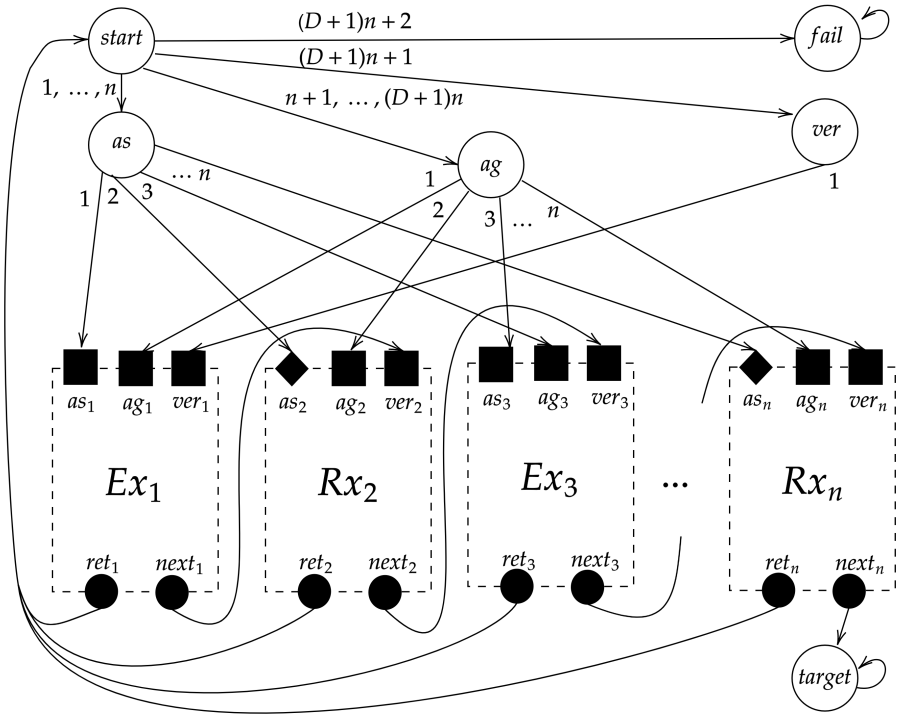
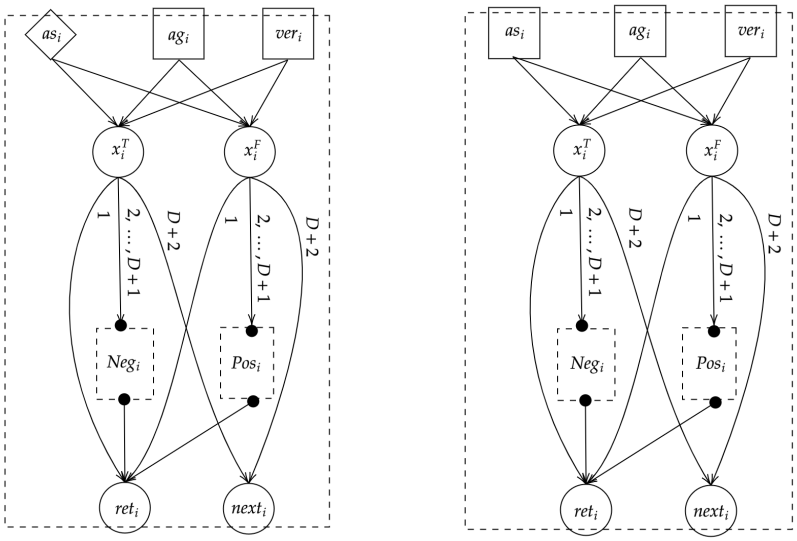


Fig. 2: Control gadget

the literal x_i appears, and b_i is the number of clauses in which the literal $\neg x_i$ appears. We let $D = \max_i \{a_i, b_i\}$ be the maximum number of occurrences of any literal. We divide the game into three phases which correspond to the different nodes in $\mathsf{Ord}(\mathsf{start})$: the "assignment" phase, consisting of the time strictly before the $n+1$'th visit to the vertex start where the switching node takes us to the node as, the "agreement" phase, consisting of the time strictly before the $Dn+1$'th visit to the vertex start where the switching node takes us to ag, and the "verification" phase consisting of the time afterwards where the switching takes us to either ver or fail. Each phases has the following objectives:

- **Assignment Phase** - In this phase the player and nature alternate in choosing values of $x_1, \ldots, x_n$ in sequence.
- **Agreement Phase** - In this phase the player must continue to agree with the choices in the "assignment" phase. Each time we visit we go through a list of clauses which our choice of assignment to that variable doesn't satisfy.
- **Verification Phase** - In this phase we verify that the player acted honestly and did agree with the choices in the "assignment" phase by moving through each variable gadget.

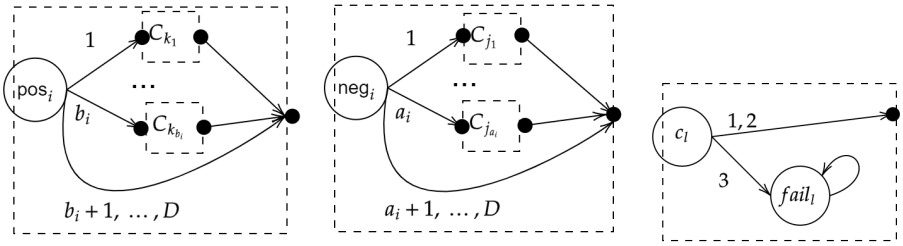


(a) Randomly Quantified (b) Existentially Quantified

Fig. 3: Gadgets for quantified variables.

These phases correspond to the three distinct entries to each of our quantified variable gadgets and we only use the entrance matching the phase we are in. We use "pass" to refer to a path from an entry to the exit, the "initial pass" is the one made in the "assignment" phase. Our gadgets function like:

- **The Control Gadget.** In this structure shown in Figure 2 we enforce the phases using the switching behaviour at *start*. The nodes *as* and *ag* cycle through the n quantified variable gadgets, visiting each once in the "assignment" phase and D times in the "agreement" phase. The node *ver* finally starts the verification process by moving to ver_1. We note any more visits to *start* send us to *fail*. We note our quantified gadgets are connected with edges between *as* and all as_i and between *ag* and all ag_i, return edges from ret_i to *start* and a chain of edges going from *ver* to ver_1, $next_1$ to ver_2,..., and finally $next_n$ to *target*.
- **Quanti ed Variable Gadget.** We have two variations of this gadget shown in Figures 3a and 3b which depend on whether x_i is existentially or randomly quantified in φ, differing only in the node type of as_i. On the initial pass the assignment is chosen by the player or uniformly at random respectively. The three entries correspond to the different phases of the game and we have two exits, ret_i returns back to the *start* and $next_i$ moves us on to the next variable's verification entry ver_{i+1}, or to *target* if $i = n$. The nodes x_i^T and x_i^F represent choosing an assignment of the variable x_i on this pass, and the "initial assignment" is the one from the initial pass. The switching behaviour of x_i^T and x_i^F prevents $next_i$ being reached without $D + 2$ visits to one of



(a) Pos_i consequence gadget (b) Neg_i consequence gadget (c) Clause gadget

Fig. 4: Gadgets for Consequences of variable x_i and Clauses C_l

the two nodes, which forces D visits to the respective Consequence gadget Neg_i or Pos_i.

- **Consequences Gadget.** We have two consequences gadgets for each variable, Neg_i and Pos_i, shown in Figures 4a and 4b. Neg_i (resp. Pos_i) enumerates the gadgets for clauses, $C_{j_1}, \ldots, C_{j_{a_i}}$ (resp. $C_{k_1}, \ldots, C_{k_{b_i}}$), where the literal $\neg x_i$ (resp. x_i) appears. When we choose an assignment of true (resp. false) these clauses aren't immediately satisfied by our assignment. As any literal appears in at most D clauses by visiting this gadget D times we are guaranteed to go through each of the contained clause gadgets. If we have $a_i < D$ (resp. $b_i < D$) then any further edges proceed straight to the exit to ensure if we make exactly D passes we visit each clause gadget exactly once.
- **The Clause Gadget.** This is shown in Figure 4c. Here we check if it is possible to still satisfy a clause. Note we pass through the clause gadget for C_l only in the following situations:
  - From a Neg_i gadget where we have assigned x_i true on this pass and $\neg x_i$ appears in C_l,
  - From a Pos_i gadget where we have assigned x_i false on this pass and x_i appears in C_l,

  Thus as a consequence of our truth assignment to x_i it doesn't witness the truth of C_l. Our clause C_l has width 3 and if our assignment is satisfying then we must have at least one of the 3 literals as a witness to the truth of C_l. Thus our gadget acts as a simple counter of the number of literals in the clause which evaluate to false, after 3 passes our switch sends the play to the fail state, because the assignment we have chosen does not satisfy C_l. On the first and second passes the counter is just incremented and we use this gadget to ensure the clause is satisfied.

We can prove that instance $G(\varphi)$ has value $val(G(\varphi))$ satisfying Equation (2).

We note that this construction remains polynomial in the size of the formula, with the control gadget (Figure 2) only containing instances of the randomly and existentially quantified variable gadgets, the quantified variable gadgets (Figure 3) only containing the Consequence gadgets Pos_i and Neg_i and the Conse-

quence gadgets (Figures 4a and 4b) only containing Clause Gadgets (Figure 4c). Further the ret_i exits and all exits of the consequence and clause gadgets may be treated as the node ***start***, independent of the index i or l of the gadget, as each has an onward path containing only nodes of out-degree one leading to ***start***.

Theorem 1. $\{R, S, 1\}$-`Arrival-Quant` *is* PSPACE-*hard.*

Proof (sketch). We prove this by showing the above construction, which can easily be carried out in polynomial time, given a SSAT instance, φ, constructs an instance $G(\varphi)$ whose value $val(G(\varphi))$ satisfies Equation (2). To do so we note any play must reach the "agreement" phase, as there is no way to reach a consequence gadget (containing ***fail*** nodes) or the $\mathbf{next}_i$ nodes with a single pass of each variable. Thus every play makes an initial assignment $V : [n] \to \{T, F\}$ where we visit $x_i^{V(i)}$ from $\mathbf{as}_i$.

We can then show that in any play we can only make at most $D + 2$ passes of the Ex_1 gadget, once through entrance $\mathbf{as}_1$, D times through $\mathbf{ag}_1$ and once through $\mathbf{ver}_1$ and thus use the edge from $\mathbf{next}_1$ at most once. We may extend this inductively to show in any play we can make at most $D + 2$ passes of any quantified variable gadget and use the $\mathbf{next}_i$ exit at most once. We can also show by induction if we reach ***target*** we must make exactly $D+2$ passes of each gadget and use the $\mathbf{next}_i$ exit exactly once. To use the $\mathbf{next}_i$ exit we must visit one of x_i^T or x_i^F exactly $D + 2$ times.

Firstly we can use this to show in any play that reaches ***target*** that the initial valuation V was satisfying. As we make $D + 2$ visits to x_i^T (resp. x_i^F) in the "agreement" phase we must visit exactly one of Neg_i (resp. Pos_i) exactly D times, which means we visit every clause gadget they contain exactly once. If we reach the end of the "agreement" phase then there is at least one edge incoming to each clause gadget that was unused, as there are three incoming edges which can be used at most once each and we can not make three passes of the clause gadget as it has an internal ***fail*** state. This lets us show valuation V satisfies φ.

Secondly we can show that under the "agreement strategy", where the player agrees with the initial assignment in the "agreement" and "verification" phases, the play reaches ***target*** when V satisfies φ, and by the above we can never reach ***target*** otherwise. Thus this strategy is optimal for the player in the "agreement" and "verification" phases.

We then show our value is the maximum over strategies for the "assignment" phase. In this phase we can consider the player and nature playing a game on a binary tree, where the leaves are possible valuations $V : [n] \to \{T, F\}$ and we call a leaf accepting if it's a valuation satisfying φ. At the root the player makes the choice between $V(1) = T$ and $V(1) = F$. On the next level nature randomises between $V(2) = T$ or $V(2) = F$. The player then chooses between $V(3) = T$ or $V(3) = F$, etc... At each stage the player knows the past decisions and maximises their choice with the aim that they reach an accepting leaf, which gives exactly Equation (2). □

As an immediate corollary we can deduce hardness for $\{R, S, 2\}$-`Arrival-Quant`.

Corollary 2. $\{R, S, 2\}$-`Arrival-Quant` *is* PSPACE-*hard.*

4 The {R,S}-Arrival Problems

Firstly we give some bounds on the qualitative problems, then we give an interesting bound on the expected number of times we use edges in each play. Finally, for $\{R, S\}$-Arrival-Quant both a PSPACE algorithm and PP-hardness.

We are able to give two easy reductions by creating new instances where we give control of random nodes to player 1 or randomise over player 1 choices, these allow us to deduce NP-completeness for two of our problems.

Theorem 2. *$\{R, S\}$-Arrival-Qual-0, $\{S, 1\}$-Arrival and $\{R, S, 1\}$-Arrival-Qual-0 are all poly-time equivalent and NP-complete.*

Theorem 3. *$\{R, S, 1, 2\}$-Arrival-Qual-0, $\{S, 1, 2\}$-Arrival and $\{R, S, 2\}$-Arrival-Qual-0 are all poly-time equivalent.*

While the above arguments exploit exchanging player 1 and random nodes, we note that a similar exchange for player 2 is not immediately possible. Consider the case of a cycle of random nodes. Any play must almost surely escape this cycle, however under player 2 control it is optimal to always stay in the cycle.

We now show coNP-hardness of $\{R, S\}$-Arrival-Qual-1, by exploiting a construction in [4]. They showed that the $\{S\}$-Arrival problem lies in the class NP ∩ coNP by constructing succinct witnesses for the fact that the play does *not* reach the target t, by modifying the graph (such that reachability of t is preserved) introducing a new dead end state d, and showing that exactly one of t or d is reached in any play in the modified graph. Here we show we can use a similar construction to reduce the complement of $\{R, S\}$-Arrival-Qual-0 to $\{R, S\}$-Arrival-Qual-1. We assume (w.l.o.g., by Lemma 2) that we are working with the instances of Arrival in our simplified form.

Definition 9 (cf. [4, Definition 3]). *Let $(V, E, s, t, \{V_S, V_R\}, P, Ord)$ be an instance of generalised $\{R, S\}$-arrival. If $(v, w) \in E$ is hopeful (Definition 6) we call its* desperation *the length of the shortest directed path from w to t.*

We proceed to give our generalised versions of a Lemma in [4], generalised to the randomised setting. We note that it is simple to process our inputs and replace dead edges of the form (v, w) by an edge (v, d) immediately to the dead end.

Definition 10. *Let G be an instance of the generalised B-arrival problem and $e \in E$ an edge. Define the random variable T_e to be the number of traversals of e in a run of the instance starting from s.*

Lemma 1. *Let G be an instance of the generalised $\{R, S\}$-arrival problem in simple form, and let $e \in E$ be a hopeful edge of desperation k in G. Then $\mathsf{E}[T_e] \leq 2^{k+1} - 1$.*

Lemma 1 (which is closely related to [4, Lemma 2]) enables us to bound the expected length of a play by a single exponential in our input $\{R, S\}$-arrival instance size. Note this is despite the fact the $\{R, S\}$-arrival instance succinctly represents an exponentially larger Markov chain, and in general for an exponentially large Markov chain the worst case expected termination (hitting) time can be double-exponential. Note also that by contrast, by Proposition 3, the probability of reaching the target can be double-exponentially small. Using Lemma 1 we construct instances that almost surely terminate and given an instance G construct a new instance G' with $val(G') = 1 - val(G)$. These allow us to show:

Theorem 4. *The* $\{R, S\}$-`Arrival-Qual-1` *problem is* coNP*-complete.*

Theorem 2 and Theorem 4 together imply that $\{R, S\}$-`Arrival-Quant` is both NP-hard & coNP-hard. In Theorem 6 we will show a stronger PP-hardness result for $\{R, S\}$-`Arrival-Quant`. As an upper bound, we can show the following:

Theorem 5. *The* $\{R, S\}$-`Arrival-Quant` *problem is in* PSPACE*.*

Proof (sketch). We can view our instance G as an exponentially larger Markov Chain (MC) with a succinct respresented transition probability matrix P. Using suitable preprocessing, we can simplify the model so that the matrix $(I - P)$ is invertible, without altering the probability of reaching the target. We can compute individual bits of the hitting probabilities on such a MC by computing entries of $(I - P)^{-1}$, which can be done in PSPACE, using the fact that an (explicitly given) linear system of equations can be solved in NC2 ([3]). Using these bits we can decide $\{R, S\}$-`Arrival-Quant`. □

We can finally use a construction similar to Theorem 1 to construct a hard instance.

Theorem 6. $\{R, S\}$-`Arrival-Quant` *is* PP*-hard.*

Proof (sketch). We show this by a reduction from the MAJSAT problem, namely deciding whether, for a given CNF formula $\varphi(x)$ over n variables, the probability, p_φ, that a uniformally random assignment of truth values to the variables x satisfy φ. MAJSAT is PP-complete ([10, 15]). We use similar gadgets to those in the proof of Theorem 1, however for our PP-hardness proof for $\{R, S\}$-`Arrival-Quant`, we make a new random assignment on each pass of the variable gadget and use switching nodes to ensure this is the same as past choices. Where we make different assignments to a variable on different passes we move to a state which moves us randomly to *target* or *fail*, thus we only reach the verification phase when we make the same assignment on every pass. Our "verification" phase then checks if all clauses are satisfied. This allows us to distinguish three distinct cases, "invalid random assignment", "valid, unsatisfying assignment" and "valid, satisfying assignment", which we can use to determine if $p_\varphi > \frac{1}{2}$. □

Acknowledgements. Thanks to a prior anonymous reviewer who sketched a proof of Theorem 5, improving on our prior result which only showed that approximation of the $\{R, S\}$-`Arrival` value to within any given desired accuracy $\epsilon > 0$ is in PSPACE.

A Omitted Proofs

A.1 Proofs of Section 2

Proposition 2. *The* $\{S,2\}$-`Arrival` *problem is* NP*-complete.*

Proof. We adapt the proof of NP-hardness of $\{S,1\}$-`Arrival` given by Fearnley et al. ([6, Theorem 3.8]). Given a 3SAT formula φ we construct their instance $G = (V, E, \textit{start}, \textit{target}, \{V_S, V_1\}, \textit{Ord})$ of $\{S,1\}$-`Arrival` with max player nodes V_1 and nodes *target* and *fail*. We construct a new instance $G' = G = (V, E, \textit{start}, \textit{fail}, \{V_S, V_2\}, \textit{Ord})$ where $V_2 := V_1$ and we have interchanged *target* and *fail* so that *fail* becomes our target and the node *target* is now a dead-end state. Thus player 2 aims to avoid reaching *fail*.

It can be observed that in G' all cycles in (V, E) contain the node *start*, and after $3n + 2$ visits to *start* we take the edge to *fail*, thus player 2 can only win by reaching *target*. However this is now identical to the proof given by Fearnley et al. [6] that $\{S,1\}$-`Arrival` is NP-hard.

To show containment in NP, we modify G as follows. We introduce a new dead end state d. If v is not a hopeful vertex, we remove it from G and replace any edges $(u, v) \in E$ by edges (u, d). If player 2 has a strategy to reach d then there exists a "controlled switching flow", as defined by [6, Section 3.1], from s to d which witnesses the existence of a strategy by player 2 to reach d (and hence not to reach t) in this new instance. A "controlled switching flow" has a polynomial size encoding, and can be verified in polynomial time. Thus, all that remains is to show show that in this modified instance any infinite play reaches either t or d. Assume, to the contrary, that there is some play which cycles $(v_1, q_1), \ldots, (v_n, q_n), (v_1, q_1), \ldots$ forever, never reaching t or d. We must have that v_1 is hopeful, as we have removed all other vertices. However, since starting at (v_1, q_1) we return to the same switching state (v_1, q_1), we must have used along this cycle all outgoing edges from any nodes $v_i \in V_S$ that appears on this cycle. But this can be interpreted as providing a strategy for player 2 to win the 2-player reachability game (i.e., avoiding t) starting from v_1, where the nodes in V_S are controlled by player 1 (and the nodes in V_2 controlled by player 2). But this implies that v_1 is not hopeful. Thus such a cycle cannot exist. □

Lemma 2. *Given an instance* G *of a generalised* B*-arrival problem,* $G := (V, E, s, t, \mathcal{V}, P, \textit{Ord})$ *satisfying De nition 1, any of the following restrictions may be placed upon* G *without altering the complexity of the* B-`Arrival-Qual-0/1` *and* B-`Arrival-Quant` *problems.*

1. *There is a distinguished vertex* $d \neq t$*, the dead end node, where for any* $v \in V$ *if* $d_{out}(v) = 1$ *then* $v = d$ *or* $v = t$*, and where* $(d, d) \in E$ *and if* $(v, v) \in E$ *then* $v = t$ *or* $v = d$*.* $\mathcal{V}$ *partitions* $V - \{t, d\}$*.*
2. *For any* $v \in V$ *we have* $d_{out}(v) \leq 2$ *and if* $d_{out}(v) = 2$ *we have:*
   - *If* $v \in V_R$ *then for* $(v, u), (v, w) \in E$*,* $u \neq w$*, we have* $P(v, u) = P(v, w) = 1/2$*.*

- *If* $v \in V_S$ *then* $|\mathsf{Ord}(v)| = 2$ *and there exists functions* $s_0, s_1 : V_S \to V$ *with* $(v, s_0(v)), (v, s_1(v)) \in E$, $\mathsf{Ord}(v) = s_0(v)s_1(v)$ *and* $s_0(v) \neq s_1(v)$.

Proof. We prove that given any instance of the form $(V, E, s, t, \{V_R, V_S, V_1, V_2\}, \mathsf{Ord}, P)$ we may create a new instance in polynomial time which has the desired properties and doesn't alter the probability any play is winning.

1. To begin we consider the instance $(V + d, E + (d, d), s, t, d, \mathcal{V}, \mathsf{Ord}, P)$. We then construct a new instance, with fewer nodes of out degree 1, $(V + d - x, E' + (d, d), s, t, d, \mathcal{V}, \mathsf{Ord}', P')$ as follows, take a vertex $x \in V - t$ with $d_{\text{out}}(x) = 1$. We can find the unique $w \in V$ with $(x, w) \in E$, if $w = x$ then we replace all xs with d. Otherwise for any edges $(v, x) \in E$ we add an edge $(v, w) \in E'$, if v is a switching node we replace all occurrences of x in $\mathsf{Ord}(v)$ by w and if it's a probabilistic node then we set $P'(v, w) = P(v, w) + P(v, x)$. We also remove x from the relevant element of $\mathcal{V}$. This new instance has one fewer node of out degree 1 and can be computed in polynomial time, thus after at most $|V|$ iterations we are done.
2. We construct a new instance, which removes a vertex x with $d_{\text{out}}(x) > 2$, $(V', E', s, t, \mathcal{V}, \mathsf{Ord}', P')$ as follows:
   - If $x \in V_1 \cup V_2$, we let k be such $2^k \geq d_{\text{out}}(x)$ and replace x by a binary tree of depth $k - 1$ and connect the 2^k outgoing edges according to the figure, as shown in Figure 5. This adds at most $2d_{\text{out}}(x)$ new vertices. We know there are at least 2^{k-1} outgoing edges from x, we connect each of the 2^{k-1} leaves to one such vertex, then for any remaining vertices we connect those using a second edge from those leaves. This adds at most
   - If $x \in V_S$ we let k be such $2^k \geq d_{\text{out}}(x)$ and replace x by a binary tree of depth $k - 1$ and connect the 2^k outgoing edges according to the figure, as shown in Figure 5. This adds at most $2d_{\text{out}}(x)$ new vertices. As shown in Figure 5 we can assign a binary word to each outgoing edge of our 2^{k-1} leaves, we take the lexicographic order over such words and if $\mathsf{Ord}(x) = (u_0, \ldots, u_{m-1})$ we connect the edge labeled with the i'th word to u_i, then connect edges with the largest $2^k - m$ words to x. We let s_0, s_1 be the lexicographically first and second successors respectively.
   - If $x \in V_R$ then we can apply the classical construction from Markov chains, for instance as in as demonstrated in [5, Proposition 2.1]. We assume we have probability $\frac{a}{b}$ of moving from x to v, and thus probability $1 - \frac{a}{b}$ of moving to any other vertex from x, we will call this a new vertex w. We let k be the number of bits required to write a and b. We now use the random transitions to flip a sequence of coins, generating a k-bit binary number. We check if this number is $\leq a$, in which case we move to v, $\leq b$, in which case we move to w, or $> b$, in which case we return to the start and generate a new number. Naively this would require 2^k states, however we may only use $2k$ if we see these flips as generating the number starting from the most significant bit and after each flip checking if we can determine early which case we are in. We note the new vertex

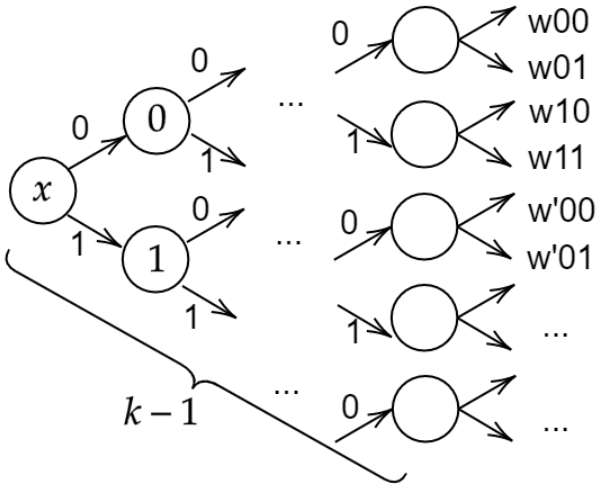


Fig. 5: Binary Tree Construction

w has a strictly smaller outdegree than x, thus we can apply the same argument again to it's transitions.

Finally we have to eliminate vertices $v \in V_R$ where $d_{\text{out}}(v) = 2$ but where have some $(v,u),(v,w) \in E$ with $u \neq w$, but $P(v,u) \neq P(v,w) \neq 1/2$. To do so we let $P(v,u) = \frac{a_u}{b_u}$ and $P(v,w) = \frac{a_w}{b_w}$ and let $B = b_u \times b_w$ and choose some k with $2^k \geq B$. We note we can express B in a polynomial number of bits as it is a product of 2 numbers given as part of our input. We then construct a binary tree of depth 2^{k-1} with 2^k outgoing edges. We connect exactly $a_u \times b_w$ edges to u and exactly $a_w \times b_u$ edges to v. We then connect the remaining $2^k - B$ edges to x.

Lemma 3. *Given an instance G of a generalised B-arrival problem with $S \in B$, we can construct an (exponentially larger) instance, $Exp(G)$, of $(B \backslash \{S\})$-arrival which has the same value, and there is a one-to-one correspondence between both winning plays and strategies, in the two games.*

Proof. We let our original instance be $G := (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$ and our new instance to be $G' := (V', E', s', t', \{V'_\sigma : \sigma \in B \setminus \{S\}\}, P')$. We begin by letting t' be a new vertex and define $V' := (V \times Q) + t'$ we can then let $q_0 : V_S \to \mathsf{N}_0$ by $q_0(v) = 0$ and take $s' = (s, q_0)$. We know define our P' and edges E' in different cases.

- For $v \in (V_1 \cup V_2)$ and any $(v,u) \in E$ we let $\{((v,q),(u,q)) : q \in Q\} \subseteq E'$.
- For $v \in V_R t$ and any $(v,u) \in E$ we let $\{((v,q),(u,q)) : q \in Q\} \subseteq E'$ and define $P'((v,q),(u,q)) = P(v,q)$.
- For any $v \in V_S$ and $q \in Q$ we let $\{((v,q),(u,q')) : (u,q') \in \mathsf{Valid}(v,q)\} \subseteq E'$.

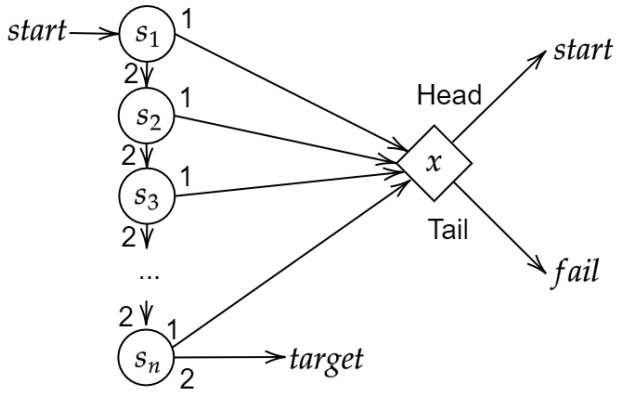


Fig. 6: Instance with double exponential probability of reaching *target*.

{ We let $\{((t,q),t') : q \in Q\} \subseteq E'$.

We note that vertices in this game correspond exactly to states in the original and the edges to valid state transitions. □

Corollary 1. *The value of an instance G of a generalised B-arrival problem is a rational number $val(G) := \frac{p}{q}$ which in lowest terms has $0 \leq p, q \leq 4^k$ with $k = 2n(|V| \times M^{|V_S|})$ with $M = \max_{v \in V_S} |\mathsf{Ord}(v)|$.*

Proof. We apply the conversion from Lemma 3 to create a new exponentially larger simple stochastic game, G', on $|V| \times |Q|$ vertices, we then let $M = \max_{v \in V_S} |\mathsf{Ord}(v)|$ and then apply the construction from Lemma 2 to the resulting exponentially larger simple stochastic game, to modify G' so that all out degrees are 2 and that $P(u,v) = \frac{1}{2}$ for all $u \in V_R$ and $(u,v) \in E$, which we can do by replacing each vertex in $v \in V_1 \cup V_2 \cup V_R$ by at most $2d_{\text{out}}(v) \leq 2n$ new vertices. We note that for vertices in V_S these have out-degree 1 in G'.

Hence by applying [2, Lemma 2], we can bound p and q by 4^{N-1}, where N is the number of vertices in the simple stochastic game G', as our games has out-degree 2 and uniformally random choices, as required by Condon's definitions. Thus we have $N \leq 2n(|V| \times |Q|)$ and further we know $|Q| \leq M^{|V_S|}$, thus we can take $k = 2n(|V| \times |Q|)$ and have $1 \leq p, q \leq 4^k$. □

Proposition 3. *For any positive integer n, we can construct an instance G of the generalised B-arrival problem containing node types $B = \{R, S\}$, such that G has encoding size $O(n)$, and such that $val(G)$ is a positive value that is at most $\frac{1}{2^{2^n}}$.*

Proof. Consider the instance shown in Figure 6. This has a sequence of switching nodes $s_1, \ldots, s_n$ and a single random node x with uniform distribution on Heads and Tails. The instance in Figure 6 indeed has bit encoding size $\mathsf{poly}(n)$. This

instance is just a modification of the example given by Dohrau et. al. [4, Figure 1] showing that a pure switching arrival instance can require exponentially many steps to reach the target. We now compute the probability that a random play starting at ***start*** reaches ***target***.

It is easy to see the only way to reach ***target*** is by passing through the node s_n twice, and and inductively we can see that this requires visiting the node s_{n-i}, 2^{i+1} times, for all $i \in \{1, \ldots, n\}$. Hence this requires 2^n visits to s_1. Thus we must make $2^n - 1$ visits to the vertex x, and at each of these visits the random choice between "$Heads$" and "$Tails$" must choose "$Heads$" because otherwise if we ever chose "$Tails$" our play reaches the node ***fail***. Thus the probability of reaching the target is:

$$val(G) = \left(\frac{1}{2}\right)^{2^n - 1} = 2^{-(2^n - 1)}$$

□

Proposition 4. *Given an instance of a generalised B-arrival problem $G = (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$, with $R \in B$, and given any rational $p \in (0, 1)$, deciding whether $val(G) \geq p$ is polynomial-time equivalent to B-`Arrival-Quant` where $p = 1/2$, i.e., to deciding whether $val(G) > 1/2$.*

Proof. Given some instance G we reduce the case of deciding $val(G) \geq \frac{1}{2}$ to deciding $val(G) > \frac{1}{2}$. By Corollary 1 we know that $val(G) = \frac{p}{q}$ where in lowest form we have $1 \leq p, q \leq 4^k$ and $k = 2n(|V| \times M^{|V_S|})$, where $M = max_{v \in V_S} |\mathsf{Ord}(v)|$. Note that M is upper bounded by the input's bit encoding size. We can thus say $p, q \leq 2^{2^{2Mn+2n+2}}$, because we have $k \leq 2n \cdot n \cdot M^n = 2^{1+2\log(n)+n\log(M)} \leq 2^{2Mn+2n+1}$.

We construct a new instance G' as shown in Figure 7. We will show that $val(G') > 1/2$ if and only if $val(G) \geq 1/2$. G' has a new start vertex s' and in it we begin by running a game analogous to Figure 6 which with large probability moves to the start of our original instance G and with tiny probability moves to the original target immediately. By Proposition 3 we know the value of this instance is $\epsilon = 2^{-(2^l - 1)}$, we take $l = 3Mn + 3n + 3$, which is polynomial in the input size. Thus we have that $val(G') = (1-\epsilon)val(G) + \epsilon = val(G) + \epsilon(1 - val(G))$.

Assuming that $val(G) = \frac{p}{q} < \frac{1}{2}$ we have that $\frac{1}{2} - \frac{p}{q} = \frac{q-2p}{2q} > \frac{1}{2q} \geq 2^{-(2k+1)}$. Then $\frac{1}{2} - val(G') \geq 2^{-(2k+1)} - \epsilon > 0$, with the final inequality following by our choice of ϵ, where we can see $2k + 1 \leq 2^{2Mn+2n+2} + 1 < 2^{3Mn+3n+3} - 1$, and hence have $val(G') < \frac{1}{2}$. By construction we can also see that $val(G') \geq val(G)$ and this is a strict increase when $val(G) \neq 1$, hence if $val(G) \geq \frac{1}{2}$ we know we have $val(G') > \frac{1}{2}$.

We may also perform a similar reduction from the case of deciding $val(G) > \frac{1}{2}$ to deciding $val(G) \geq \frac{1}{2}$ by performing the same construction, instead with a small probability of moving to a dead-end, d. This strictly decreases the value by ϵ. This gives the equivalence. □

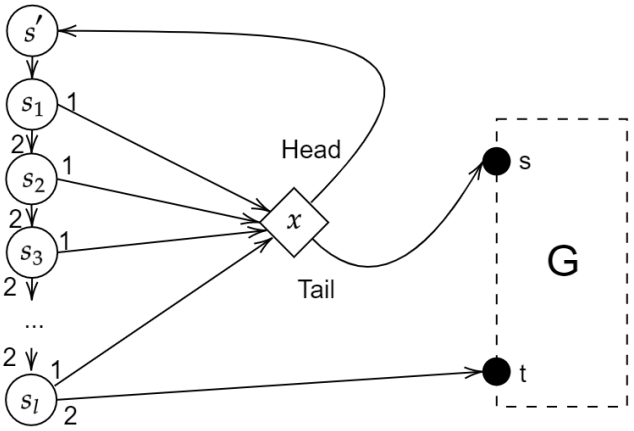


Fig. 7: Proof of 4: reducing deciding $\geq 1/2$ to deciding $> 1/2$: construction of G' for a given instance G.

A.2 Proof of Section 3

Theorem 1. $\{R, S, 1\}$-`Arrival-Quant` *is* PSPACE-*hard.*

Proof. We note that given a formula φ we can easily compute the values $a_i = |\{l \in \{1, \ldots, m\} \mid x_i \in C_l\}|$ and $b_i = |\{l \in \{1, \ldots, m\} \mid \neg x_i \in C_l\}|$, by a single loop over the m clauses, and we can compute $D = \max_i \{a_i, b_i\}$. We trivially have that $D \leq m$, as without loss of generality we may assume each variable appears at most once in each clause of the 3CNF formula. We can bound the size of the created instance by polynomials in m and n as follows:

$$
\begin{aligned}
|V| &= 6 + 5n + 2m + \frac{n}{2} + 2n + \frac{n}{2} = 6 + 8n + 2m \\
|E| &= 8 + 2n + (n-1) + 13n + 2n + \sum_i (a_i + b_i) + 3m \\
&= 7 + 18n + 3m + \sum_i (a_i + b_i) \\
&\leq 7 + 18n + 3m + 2Dn \leq 7 + 18n + 3m + 2mn \\
|Ord| &= (D+1)n + 2 + 2n + 3 + 2n(D+2) + (n-1) + 1 + 3m + m \\
&= 5 + 8n + 4m + 3Dn \\
&\leq 5 + 8n + 4m + 3mn
\end{aligned}
$$

Hence the instance constructed from a given SSAT instance is contained within an amount of space bounded by a polynomial in n, the number of variables, and m, the number of clauses, of that instance.

We first show that any play, π, must reach the "agreement" phase, under any player 1 strategy. Assume otherwise, as we have not hit *ag* in our play π we

made at most n visits to start, thus we made at most one pass of any quantified variable gadget. With only a single pass it is impossible for a variable gadget to reach a fail state, because on the initial visit to x_i^T or x_i^F our switching order requires us to move to ret_i, thus start. Hence we can not reach a fail_l state internally.

As we reach the "agreement" phase we can define the "initial assignment" as a function $V_\pi : [n] \to \{T, F\}$ with the property that $x_i^{V(i)}$ was visited on the initial pass of the i'th quantified variable gadget. As the "agreement" phase must be reached this function is entire and well-defined.

Given a play π that reaches target, then we show for each i we must make exactly $D + 2$ passes of the i'th variable gadget, using the next_i exactly once and can only visit one of the nodes x_i^T or x_i^F. Considering any play it is evident we can only visit the $(i + 1)$'th gadget at most as often as we have visited the i'th gadget, as our switching orders and next_i edges always increase. Assume we visit the Ex_1 gadget $D + 3$ times. Because of the switching order at start we can see we only visit once using the edge (as, as_i), D times by (ag, ag_i) and once via $(\mathsf{ver}, \mathsf{ver}_i)$, however we can not use any of these again without making more than $(D + 1)n + 2$ visits to start, which would use the final edge to fail, contradicting us reaching target. Thus we can visit Ex_1 at most $D+2$ times, and thus can visit each at most $D + 2$ times. If π reaches target then we must have used the edge $(\mathsf{next}_n, \mathsf{target})$. To reach next_n we need to make at least $D + 2$ passes of Rx_n, so we must then visit all gadgets at least $D + 2$ times. Thus any play reaching target must make exactly $D + 2$ passes. It is then trivial that we must visit next_i and exactly one of x_i^T or x_i^F, otherwise we must make more than $D + 2$ passes or can not reach target.

Thus for our player in the "agreement" and "verification" phases it is optimal for our player to play such that we only visit one of x_i^T and x_i^F, because we know one of these was visited during the "assignment" phase and if they choose to visit both they will be unable to reach the target. Thus any optimal strategy must pick the node that was visited in the "assignment" phase and we can assume the player uses such an "agreement strategy" once it reaches these stages.

Given a play π reaching target we now show that V_π satisfies the given formula $\varphi = C_1 \wedge \cdots \wedge C_m$. Assume not, then we can find some clause C_l in φ which is not satisfied by V_π. We consider the clause gadget for C_l, this has exactly 3 incoming edges corresponding to the three atoms in the clause. As π reaches target we can visit the node C_l at most twice, thus there is an edge into C_l which is not used. We call this unused edge (neg_i, C_l), if it was in fact of the form (pos_i, C_l) we can exchange true and neg_i for false and pos_i respectively in this argument. We now consider the value $V_\pi(i)$. If we have $V_\pi(i) = F$ then as (neg_i, C_l) is an edge by the construction we have that $\neg x_i$ appears in C_l, however our valuation makes x_i false, thus C_l is satisfied, contradicting our choice of C_l. If $V_\pi(i) = T$ we must use the edge (x_i^T, next_i), requiring us to make D visits to neg_i. However neg_i has at most D edges, so we use each at least once, including the edge (neg_i, C_l), contradicting our assumption we didn't use this edge.

If V_π is satisfying after the "assignment" phase then we are able to reach ***target*** by following the "agreement strategy", for contradiction assume there is some satisfying V_π which does not reach ***target*** under the "agreement strategy". Then our play must reach either ***fail*** or some $\textit{fail}_l$ node. If we reach $\textit{fail}_l$ for some clause C_l then as this gadget has exactly 3 incoming edges we must either use some edge twice or use all three edges once. We show each of these cases leads to a contradiction:

- If we reach $\textit{fail}_l$ and use all three incoming edges to C_l once we note by construction we have assigned each of the literals in C_l a false value, however then V_π can't be satisfying as C_l is false which is a contradiction.
- If we reach $\textit{fail}_l$ and we've used some edge (neg_i, C_l) twice, it follows we've made at least $D+1$ visits to neg_i, which would require at least $D+3$ passes of the i'th variable gadget, but we know we can't make $D+3$ passes without using the edge $(\textit{start}, \textit{fail})$, contradicting that we reach $\textit{fail}_l$.
- If we reach ***fail*** by the switching order at ***start*** we must visit ***ver*** and enter the "verification" phase. As we enter the "verification" phase we must have already made $D+1$ passes of each variable gadget and by the "agreement strategy" visited only one of x_i^T or x_i^F for each i. Thus from ***ver*** we proceed to $\textit{ver}_1$ where we can make a $D+2$'th visit to $x_i^{V\ (1)}$ and proceed to $\textit{next}_1$ and $\textit{ver}_2$. We can continue this and show we reach ***target***, contradicting that we reached ***fail***.

We now compute the value of the game, which, by the above, will only depend on the edge used out of each $\textit{as}_i$ in the "assignment" phase. As we have shown the player reaches ***target*** if and only if V_π is satisfying, hence the player's goal will to be to maximise the probability V_π is satisfying and we will have $val(G)$ equal to the probability V_π is satisfying under an optimal strategy in the "assignment" phase. Consider a tree of partial valuations $V : [n] \rightharpoonup \{T, F\}$ where we have so far assigned an initial sequence of $[n]$. It is easy to see the "assignment" phase is equivalent to a game on this tree where we start from the root on level 1 and at odd levels allow the player to choose to move to some child and at even levels play moves randomly to one of the children. The game wins if the total valuation reached satisfies F. From this game we can see that we must have:

$$val(G(\varphi)) = \max_{x_1}[\mathsf{E}_{x_2}[\max_{x_3}[\dots \mathsf{E}_{x_n}[\chi[\varphi(x_1, \dots, x_n) = \top] \dots]$$

Hence as SSAT is a PSPACE-complete problem and SSAT is poly-time reducible to
$\{R, S, 1\}$-`Arrival-Quant`, thus problem is PSPACE-hard. □

Corollary 2. $\{R, S, 2\}$-`Arrival-Quant` *is* PSPACE*-hard.*

Proof. We can modify the construction of Theorem 1 by making the following changes to also derive a hardness result for $\{R, S, 2\}$-`Arrival-Quant`, we replace player 1 with player 2 and exchange the nodes ***target*** and ***fail***, including in the

clause gadgets. By the same argument above we will construct an instance $G'(\varphi)$ where:

$$val(G'(\varphi)) = \min_{x_1}[\mathsf{E}_{x_2}[\min_{x_3}[\ldots(1 - \mathsf{E}_{x_n}[\chi[\varphi(x_1, \ldots, x_n) = \top]])\ldots] \quad (3)$$

We can see that $val(G'(\varphi)) = 1 - val(G(\varphi))$, where $G(\varphi)$ is the instance constructed above. As we know that coPSPACE $\equiv$ PSPACE we have shown this problem is also hard for PSPACE by reducing from the complement of the $\{R, S, 1\}$-`Arrival-Quant-Eq` problem. □

A.3 Proofs of Section 4

Lemma 4. *Suppose $R \in B$ and let $B' = (B - \{R\}) \cup \{1\}$, then the problem B-`Arrival-Qual-0` is poly-time reducible to B'-`Arrival`.*

Proof. Given an instance $G := (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$ we define an instance
$G' := (V, E', s, t, \{V'_\sigma : \sigma \in B'\}, \mathsf{Ord})$ where we take the following:

- $V'_1 := V_1 \cup V_R$, i.e. we give the max player control of all random nodes.
- $E' := \{(v, u) \in E : (v \notin V_R) \vee (v \in V_R \wedge P(v, u) > 0)\}$. i.e. we removed edges $(v, u) \in E$ if $P(v, u) = 0$, thus they couldn't be chosen in a valid random transition.
- If S or $2 \in B$ we take $V_2 = V'_2$ and $V_S = V'_S$.

This can easily be computed in polynomial time. We then claim that any winning play of the new instance corresponds to a winning play in the original instance. Consider a winning play $(v_0, q_0), \ldots, (v_n, q_n)$ in the new instance, we then consider the conditions for the play to be a valid and winning play in the original:

- $v_0 = s$ and for all $v \in V_S$ we have $q_0(v) = 0$. This follows from it being a valid play in the new instance, making it valid in the new instance.
- For all indices i with $v_i \notin V_S$, $(v_{i+1}, q_{i+1}) \in \mathsf{Valid}(v_i, q_i)$, as there are no changes to edges outside V_S anything valid in the new instance is valid in the original.
- For indices i with $v_i \in V'_1 - V_1 = v_R$ we know that $(v_i, v_{i+1}) \in E'$, by our definition we must have $P(v_i, v_{i+1}) > 0$, thus this edge also forms a valid transition from a probabilistic node in state (v_i, q_i) in the original instance.
- If it was a winning play it is of finite length n and $v_n = t$, which makes it winning in the original instance.

Hence this play is also valid and winning in the original instance.

We also claim that if a play was winning in the original instance then it is still winning in the new instance. Consider a winning play $(v_0, q_0), \ldots, (v_n, q_n)$ in the original instance, we then consider the conditions for the play to be a valid and winning play in the new instance:

- $v_0 = s$ and for all $v \in V_S$ we have $q_0(v) = 0$. This follows from it being a valid play in the original instance, making it valid in the new instance.
- For all indices i with $v_i \notin V_S$, $(v_{i+1}, q_{i+1}) \in \mathsf{Valid}(v_i, q_i)$, as there are no changes to edges outside V_S anything valid in the original instance is valid in the new instance.
- For indices i with $v_i \in V_1' - V_1 = v_R$ we know that $(v_i, v_{i+1}) \in E$, thus we must have $P(v_i, v_{i+1}) > 0$, hence this edge also forms a valid transition for the player in state (v_i, q_i) in the new instance.
- If it was a winning play it is of finite length n and $v_n = t$, which makes it winning in the new instance.

Hence this play is also valid and winning in the new instance.

Hence if $v_G > 0$ then we have a winning play in G then there is a winning play in G' then $v_{G^0} = 1$. Hence B-`Arrival` is poly-time reducible to B'-`Arrival`. □

Lemma 5. *Suppose $1 \in B$ and let $B' = (B - \{1\}) \cup \{R\}$, then the problem B-`Arrival` is poly-time reducible to B'-`Arrival-Qual-0`.*

Proof. Given an instance $G := (V, E, s, t, \{V_\sigma : \sigma \in B\}, P, \mathsf{Ord})$ of the original we define a new instance $(V, E, s, t, \{V'_\sigma : \sigma \in B'\}, P', \mathsf{Ord})$ as follows:

- $V'_R := V_R \cup V_1$, i.e. we replace the player with a random choice.
- We then define $P' : V'_R \times V \to [0, 1]$.
- For a $v \in V_1$ we let $k := d_{\text{out}}(v)$ and then for $(v, u) \in E$ we take $P'(v, u) := 1/k$ and for $(v, u) \notin E$ we take $P'(v, u) := 0$, this satisfies that $\sum_{u \in V} P'(v, u) = 1$ by the choice of k and as $k \geq 1$ we have $P'(v, u) \in [0, 1]$.
- For $v \in v_R$ and $u \in V$ we define $P'(v, u) := P(v, u)$. This satisfies the constraints as P does.
- If S or $2 \in B$ we let $V'_S = V_S$ and $V'_2 = V_2$.

This can easily be computed in polynomial time. Given an arbitrary strategy for player 2, we can find a winning play of the original instance. We then claim any winning play of the new instance corresponds to a winning strategy for player 1 in the original instance. Consider a play $(v_0, q_0), \ldots, (v_n, q_n), \ldots$ in this new instance with $v_n = t$. We are able to "cut out" loops in our play and assume that if $i \neq j$ then either $v_i \neq v_j$ or $q_i \neq q_j$ or we have reached t. We then construct the strategy for the original instance as follows:

- For (v, q) with $v \in V_1$ appearing in our play there exists (a unique) i with $(v, q) = (v_i, q_i)$, thus we define $\mathsf{Strat}(v, q) := v_{i+1}$
- For any other (v, q) we may define $\mathsf{Strat}(v, q)$ arbitrarily.

We then claim that the "cut out" play constitutes a valid, winning play in the new instance under the given Strat for the max player. This is as follows:

- $v_0 = s$ and for all $v \in V_S$ we have $q_0(v) = 0$. This follows from it being a valid play in the new instance, making it valid in the original.

- For all indices i with $v_i \notin V_1$, $(v_{i+1}, q_{i+1}) \in \mathit{Valid}(v_i, q_i)$, as there are no changes to edges or node types outside of V_1 anything valid in the new instance is valid in the original.
- For indices i with $v_i \in V_1$ we require that $\mathit{Strat}(v_i, q_i) = v_{i+1}$ and $q_i = q_{i+1}$, however this is how we defined Strat and as $q_i = q_{i+1}$ in a probabilistic transition this a valid player transition under Strat.
- If it was a winning play it is still winning after "cutting out" loops, and thus this play is of finite length n and has $v_n = t$. Thus it is winning in the original instance.

Hence this play is also valid and winning in the original instance.

Given an arbitrary strategy for players 1 and 2 and a corresponding winning play in the original instance we show this play is also winning in the new instance as follows:

- $v_0 = s$ and for all $v \in V_S$ we have $q_0(v) = 0$. This follows from it being a valid play in the original instance, making it valid in the new instance.
- For all indices i with $v_i \notin V_1$, $(v_{i+1}, q_{i+1}) \in \mathit{Valid}(v_i, q_i)$, as there are no changes to edges or node types outside of V_1 anything valid in the original instance is valid in the new instance.
- For indices i with $v_i \in V_1$ we know that $\mathit{Strat}(v_i, q_i) = v_{i+1}$ and $q_i = q_{i+1}$. However by our choice of random probabilities we know $P(v_i, v_{i+1}) > 0$, thus this is a valid probabilistic transition.
- If it was a winning play then it is of finite length n and has $v_n = t$. Thus it is winning in the new instance.

Thus deciding if there is a winning strategy for player 1 in the original instance with $1 \in B$ has been reduced to determining if there is a winning play in the new instance with $1 \notin B$ but $R \in B$. □

Theorem 2. $\{R, S\}$-`Arrival-Qual-0`*,* $\{S, 1\}$-`Arrival` *and* $\{R, S, 1\}$-`Arrival-Qual-0` *are all poly-time equivalent and* NP*-complete.*

Proof. By the results in [6] we know that $\{S, 1\}$-`Arrival` is NP-complete. We can use Lemma 4 to see that $\{R, S\}$-`Arrival-Qual-0` is poly-time reducible to $\{S, 1\}$-`Arrival` and then Lemma 5 to see $\{S, 1\}$-`Arrival` is poly-time reducible to $\{R, S\}$-`Arrival-Qual-0`. Similarly, we can use Lemma 4 to see that $\{R, S, 1\}$-`Arrival-Qual-0` is poly-time reducible to $\{S, 1\}$-`Arrival` and the reverse reduction is obvious. Thus both are NP-complete as they are reducible to $\{S, 1\}$-`Arrival`. □

Theorem 3. $\{R, S, 1, 2\}$-`Arrival-Qual-0`*,* $\{S, 1, 2\}$-`Arrival` *and* $\{R, S, 2\}$-`Arrival-Qual-0` *are all poly-time equivalent.*

Proof. We can use Lemma 4 to see that $\{R, S, 2\}$-`Arrival-Qual-0` is poly-time reducible to $\{S, 1, 2\}$-`Arrival` and then Lemma 5 to see $\{S, 1, 2\}$-`Arrival` is poly-time reducible to $\{R, S, 2\}$-`Arrival-Qual-0`. Similarly, we can use Lemma 4 to see that $\{R, S, 1, 2\}$-`Arrival-Qual-0` is poly-time reducible to $\{S, 1, 2\}$-`Arrival` and the reverse reduction is obvious. Thus all are polynomial-time equivalent. □

Lemma 1. *Let G be an instance of the generalised $\{R,S\}$-arrival problem in simple form, and let $e \in E$ be a hopeful edge of desperation k in G. Then $\mathsf{E}[T_e] \leq 2^{k+1}-1$.*

Proof. We prove by induction on the desperation k of $e=(v,w)$. Consider a hopeful edge of desperation 0, then we must have $w=t$ and thus any run traversing e reaches the destination, thus $T_e \in \{0,1\}$. From this $\mathsf{E}[T_e] \leq 1 = 2^{0+1}-1$. Hence we have shown the base case of our induction.

Now consider a hopeful edge of desperation $k>0$ and assume the result holds for all hopeful edges of desperation $k-1$. There are two successor edges from w, $(w,s_0(w))$ and $(w,s_1(w))$ and we must have that one of these is a hopeful edge of desperation $k-1$. Without loss of generality assume it is $f:=(w,s_0(w))$ and thus we know that $\mathsf{E}[T_f] \leq 2^k-1$.

We let $f':=(w,s_1(w))$ be the other edge. Then we know by flow conservation that $T_e \leq T_f + T_{f^0}$ thus by linearity of expectation we have $\mathsf{E}[T_e] \leq \mathsf{E}[T_f] + \mathsf{E}[T_{f^0}] \leq (2^k-1)+\mathsf{E}[T_{f^0}]$.

We can then consider the value of $\mathsf{E}[T_{f^0}]$ in the two cases of $w \in V_S$ and $w \in V_R$. If $w \in V_R$ as we make a uniformly random choice between edges f and f' thus the expected number of times we use each edge must be the same, $\mathsf{E}[T_{f^0}] = \mathsf{E}[T_f] \leq 2^k-1$. If $w \in V_S$ then by the switching behaviour we must have $|T_{f^0}-T_f| \leq 1$ due to our alternating choices, hence $\mathsf{E}[T_{f^0}] \leq \mathsf{E}[T_f]+1 \leq 2^k$. Thus in either case we have $\mathsf{E}[T_{f^0}] \leq 2^k$ thus $\mathsf{E}[T_e] \leq (2^k-1)+2^k = 2^{k+1}-1$ as required. □

Proposition 6. *Let G be an instance of the generalised $\{R,S\}$-arrival problem, then the probability any run terminates, at either a dead end or target is 1.*

Proof. Let L be a random variable defined as the number of steps until a run terminates, $L \in [0,\infty]$. If a path uses a dead edge (i.e., an edge to the dead end node) then it must terminate. We note that no hopeful edge can have desperation, k, greater than n, as any shortest path from that edge can't use an edge more than once, hence $k \leq n$. We then let $l := m \cdot (2^{n+1}-1)$ and consider the events $A_i := (L > iml+1)$, by the choice of l and the pigeon hole principle the event A_i implies we use some hopeful edge e at least $i \cdot m \cdot (2^{n+1}-1)$ times, hence $P(A_i) \leq P(\bigvee_e [T_e > i \cdot m \cdot (2^{n+1}-1)]) \leq \sum_e P(T_e > i \cdot m \cdot (2^{n+1}-1))$. By Lemma 1 we have that $\mathsf{E}[T_e] \leq m \cdot (2^{n+1}-1)$ for any edge e, and thus by Markov's inequality:

$$P(T_e > i \cdot m \cdot (2^{n+1}-1)) \leq P(T_e > i \cdot m \cdot \mathsf{E}[T_e]) \leq \frac{1}{im}$$

Thus $P(A_i) \leq \frac{1}{i}$ and since $\neg Term \subseteq A_i$ for any i thus $P(\neg Term) \leq \frac{1}{i}$ for any i and thus $P(\neg Term) = 0$. □

Corollary 3. *Given G an instance of the generalised $\{R,S\}$-arrival, in polynomial time we can construct another instance of $\{R,S\}$-*`Arrival` *with $val(G') = 1 - val(G)$.*

Proof. We apply Lemma 2 to create a dead end state d in polynomial time, and this state and t are the only absorbing states in this new instance. By Proposition 6 we have that any play almost surely reaches one of d or t in finite time. If $Reach$ is the event of reaching t and $Dead$ that of reaching d we have $1 = \mathsf{P}(Term) = \mathsf{P}(Dead) + \mathsf{P}(Reach) \implies \mathsf{P}(Dead) = 1 - \mathsf{P}(Reach) = 1 - val(G)$. Thus we may exchange t and d to construct such an instance. □

Theorem 4. *The* $\{R, S\}$-`Arrival-Qual-1` *problem is* coNP*-complete.*

Proof. Given an instance of generalised $\{R, S\}$-arrival G we use Corollary 3 to construct the instance G' with $val(G') = 1 - val(G)$. We note $val(G) > 0$ if and only if $val(G') < 1$, thus $val(G') = 1$ if and only if $val(G) = 0$. Hence this question is poly-time equivalent to the complement of $\{R, S\}$-`Arrival-Qual-0`, which is NP-complete by Theorem 2. □

Corollary 4. *The* $\{R, S\}$-`Arrival-Quant` *problem is* NP*-hard &* coNP*-hard, under many-one (Karp) reductions.*

Proof. We begin by showing NP-hardness. We know by Theorem 2 that $\{R, S\}$-`Arrival-Qual-0` is NP-complete. Considering an instance of generalised $\{R, S\}$-arrival, $G := (V, E, s, t, \{V_R, V_S\}, P, \mathsf{Ord})$ we construct a new instance $G' := (V + s', E + (s', t) + (s', s), s', t, \{(V_R + +s'), v_S\}, P', \mathsf{Ord})$ where we add a new start vertex s' which transitions to either the original start s or the target t. Then it is easy to see that $val(G') = \frac{1}{2}(1 + val(G))$, thus is strictly greater than a half iff we had $val(G) > 0$. Thus we have a many-one reduction from a NP-complete problem.

For coNP-hardness we know by Theorem 4 that $\{R, S\}$-`Arrival-Qual-1` is coNP-complete. Considering an instance of generalised $\{R, S\}$-arrival, $G := (V, E, s, t, \{V_R, V_S\}, P, \mathsf{Ord})$ we construct a new instance $G' := (V + s' + d, E + (s', d) + (s', s) + (d, d), s', t, \{(V_R + s' + d), v_S\}, P', \mathsf{Ord})$ where we add a new start state s' which transitions to either the original start s or a dead-end d. Then it is easy to see that $val(G') = \frac{1}{2} val(G)$, thus is greater than or equal to a half iff we had $val(G) = 1$. Hence we have a many-one reduction from a coNP-complete problem. □

Proof of Theorem 5.

Given an instance, G, of generalised $\{R, S\}$-Arrival we let $Exp(G)$ be the expanded, exponentially larger, instance given by Lemma 3 corresponding to a Markov Chain on $V \times Q$ and let $Reach(Exp(G), (v', q'), (v, q))$ be the problem of deciding whether the vertex $(v, q) \in Exp(G)$ can be reached from the start state (v', q') of $Exp(G)$. We define the decision problem $Potential(G, (v, q))$ for each pair $(v, q) \in V \times Q$ as the problem of deciding $\{R, S\}$-`Arrival-Qual-0` where we start in state (v, q) instead of (s, q_0).

We let $t \in V$ be our unique target and now define the index set $\mathcal{J} := ((V \setminus \{t, d\}) \times Q) \cup \{(t, \star)\}$, where $(t, \star)$ represents all states of the form (t, q)

together, because all correspond to reaching the target. Where the probabilities in a row sum to a positive value less than 1 this represents the fact that there may be some transitions out of that state that go directly to a state that can never reach t (i.e., a dead end). The matrix $P \in [0,1]^{\mathcal{J}\times\mathcal{J}}$ which is our modified transition probability matrix in $Exp(G)$, is defined as follows. For all $v, w \in V$ and $q, q' \in Q$ we define :

$$
\begin{aligned}
P((t,\star),(w,q')) &:= 0,\\
P((t,\star),(t,\star)) &:= 0\\
P((v,q),(w,q')) &:= \begin{cases} 0, & \text{if } \neg Reach(Exp(G),(s,q_0),(v,q))\\ 0, & \text{if } \neg Reach(Exp(G),(s,q_0),(w,q'))\\ 0, & \text{if } \neg Potential(G,(v,q))\\ 0, & \text{if } \neg Potential(G,(w,q'))\\ P_{Exp(G)}((v,q),(w,q')), & \text{otherwise}\end{cases}\\
P((v,q),(t,\star)) &:= \begin{cases} 0, & \text{if } \neg Reach(Exp(G),(s,q_0),(v,q))\\ 0, & \text{if } \neg Potential(G,(v,q))\\ \sum_{q^?\in Q} P_{Exp(G)}((v,q),(t,q^\star)), & \text{otherwise}\end{cases}
\end{aligned}
\tag{4}
$$

Lemma 6. *Given as input an instance of a generalised B-Arrival problem G and pairs $(v,q),(w,q') \in \mathcal{J}$ we can compute in* PSPACE *the entry $P((v,q),(w,q'))$ of the matrix P, given by the equations (4).*

Proof. To show this is in PSPACE we note that to compute $P((v,q),(w,q'))$ we need to compute the following:

- $Reach(Exp(G),(s,q_0),(v,q))$ and $Reach(Exp(G),(s,q_0),(w,q'))$ - We note this corresponds to an reachability problem on a succinctly represented exponentially large directed graph. We can solve an explicit reachability problem in NL and we can thus solve our succinctly represented version in PSPACE.
- $Potential(G,(w,q'))$ and $Potential(G,(w,q'))$ - We note this corresponds to an instance of $\{R,S\}$-`Arrival-Qual-0` which by Theorem 2 is NP-complete. Hence it can be solved in PSPACE.
- $P_{Exp(G)}((v,q),(w,q'))$ - To compute this we check if $v \in V_R$ or $v \in V_S$. If $v \in V_R$ then we return $P_G(v,w)$. If $v \in V_S$ then we check if $(w,q') \in \mathit{Valid}_G(v,q)$ and return 1 if it is or 0 otherwise.
- $\sum_{q^?\in Q} P_{Exp(G)}((v,q),(t,q^\star))$ - We note that there is at most one $q^\star \in Q$ where the term $P_{Exp(G)}((v,q),(t,q^\star))$ can be non-zero and we can determine $q^\star$ from (v,q). If $v \in V_R$ then we know transitions where $q^\star \neq q$ are impossible, thus $P_{Exp(G)}((v,q),(t,q))$ is the only term which may be non-zero. If $v \in V_S$ we can determine the next switching state $q^\star$ and know $P_{Exp(G)}((v,q),(t,q^\star))$ is the only term which may be non-zero. Thus to compute the sum we only have to evaluate a single transition probability, which we can do as in the case when $q' \neq \star$. □

Lemma 7. *The matrix P, given by equations (4), is substochastic, can be written as $P = \begin{matrix} A & 0 \\ 0 & 0 \end{matrix}$ where A is a square matrix with some row summing to less than 1. Finally we have $\lim_{n\to\infty} P^n = 0$.*

Proof. First note that P is substochastic. P has row sums bounded by the row sums of $P_{Exp(G)}$, which is the transition probability matrix of a Markov Chain, thus substochastic.

We let $H \subseteq \mathcal{J}$ be defined as:

$$H := \{(v,q) \in (V \backslash \{t\}) \times Q : Reach(Exp(G), (s,q_0), (v,q)) \wedge Potential(G, (v,q))\}$$

Then let $H^\star := H \cup \{(t,\star)\}$ and let A be the sub-matrix corresponding to rows and columns in $H^\star$. We note the row or column corresponding to any $(v,q) \notin H^\star$ is all zeros, because one of $Reach(Exp(G), (s,q_0), (v,q))$ or $Potential(G, v, q)$ is false. The row corresponding to $(t,\star)$ is also all zeros, however the column is not. Thus $P = \begin{matrix} A & 0 \\ 0 & 0 \end{matrix}$.

We let $r^n_{(v,q)}$ for $(v,q) \in H$ and $n \in \mathsf{N}_0$ correspond to the (v,q)th row of A^n and let $R^n_{(v,q)}$ be the sum of entries in $r^n_{(v,q)}$. We know that for any $(v,q) \in H$ we have $Potential(G, v, q)$, thus there is some strictly positive probability that starting from (v,q) we reach t. Thus we can find some $N_{(v,q)} \in \mathsf{N}_0$ such that there is a positive probability, $p_{(v,q)} > 0$, that the $\{R,S\}$-Arrival instance starting from (v,q) terminates in exactly $N_{(v,q)}$ steps. We know that that the entries of the matrix $A^{N_{(v;q)}}_{(v,q),(w,q^0)}$ correspond to the probability that after $N_{(v,q)}$ steps, starting at (v,q) we will be in state $(w,q') \in H^\star$. Thus we must have $A^{N_{v;q)}}_{(v,q),(t,\star)} = p_{(v,q)} > 0$ and thus we have $R^{N_{(v;q)}+1}_{(v,q)} < 1$. We also trivially have that $r^n_{t,\star} = 0$ for any n.

Taking $N := \max_{(v,q)\in H}(N_{(v,q)} + 1)$ we note that thus $R^N_{(v,q)} < 1$ for any $(v,q) \in H^\star$. Thus each row of A^N sums to strictly less than 1. Consider A^{jN}, for integers $j > 0$. We must have $A^{jN} \to 0$ as $j \to \infty$. Therefore $P^n \to 0$ as $n \to \infty$, because $P^n = \begin{matrix} A^n & 0 \\ 0 & 0 \end{matrix}$. □

Lemma 8. *The matrix $(I - P)$, where I is the identity matrix, is invertible and for any $(v,q),(w,q') \in \mathcal{J}$ the value of $(I-P)^{-1}_{((v,q),(w,q^0))}$ can be computed in* PSPACE*, meaning (despite the fact that the rational number itself can be exponentially large in terms of bit encoding size), we can query the bits of $(I - P)^{-1}_{((v,q),(w,q^0))}$ in* PSPACE.

Proof. By Lemma 7, the matrix $(I-P)$ is invertible because $P^n \to 0$ as $n \to \infty$, and in fact $(I-P)^{-1} = \sum_{i=0}^{\infty} P^i$.

We can compute the matrix inverse for an explicit matrix in NC2 ([3]) and hence in polylogarithmic space. Thus we can compute bits of the inverse of the succinctly presented matrix $(I-P)^{-1}$ in PSPACE. □

Theorem 5. *The* $\{R, S\}$-`Arrival-Quant` *problem is in* PSPACE.

Proof. We know by Proposition 4 that the $\{R, S\}$-`Arrival-Quant` problem is polynomial-time equivalent to $\{R, S\}$-`Arrival-Quant-Eq`, the problem of deciding whether $val(G) \geq \frac{1}{2}$. Thus we can show $\{R, S\}$-`Arrival-Quant` is in PSPACE by showing that $\{R, S\}$-`Arrival-Quant-Eq` is in PSPACE. We let G be an instance of $\{R, S\}$-`Arrival-Quant-Eq`. We observe that $(I - P)^{-1} = \sum_{n=1}^{\infty} P^n$. Thus $(I - P)^{-1}_{(s,q_0),(t,\star)}$ represents the hitting probability of reaching the state $(t, \star)$ starting from (s, q_0), which is $val(G)$. We know by Lemma 8 that we are able to compute arbitary bits of $(I - P)^{-1}_{(s,q_0),(t,\star)}$ in PSPACE. Thus we compute the leading bit of $(I - P)^{-1}_{(s,q_0),(t,\star)}$, and we know that this is 1 if and only if $val(G) \geq \frac{1}{2}$, which decides $\{R, S\}$-`Arrival-Quant-Eq`. □

Theorem 6. $\{R, S\}$-`Arrival-Quant` *is* PP*-hard.*

Proof. We begin by formally defining the problem MAJSAT that we are reducing from, this problem is complete for PP by the results of Gill and Simon [10, 15]. Unlike our definition of SSAT (Definition 8) we can not assume that φ is a 3CNF ([1]), we let w_l be the clause width of C_l.

Definition 11 (MAJSAT). *Given a CNF formula* φ *with* n *variables,* $x_1, \ldots, x_n$ *and* m *clauses,* $C_1, \ldots, C_m$*. We let* p_φ *be the probability that a valuation,* $V : [n] \to \{\top, \bot\}$*, chosen uniformly at random over all valuations satisfies* φ*. Decide if* $p_\varphi > \frac{1}{2}$*.*

To perform the reduction we will create an instance of $\{R, S\}$-`Arrival-Quant` where we have for some constant D computable from φ:

$$v = \frac{1}{2} + (p_\varphi - \frac{1}{2}) \cdot 2^{(D+1)n} \tag{5}$$

We note that we have from this that $v > \frac{1}{2}$ if and only if $p_\varphi > \frac{1}{2}$.

We now explain this construction in more detail. Given $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, to begin with, in polynomial time we enumerate our n variables as $x_1, \ldots, x_n$ and for each we compute constants $a_i = |\{l \in \{1, \ldots, m\} \mid x_i \in C_l\}|$ and $b_i = |\{l \in \{1, \ldots, m\} \mid \neg x_i \in C_l\}|$. Here a_i is the number of clauses in which the literal x_i appears, and b_i is the number of clauses in which the literal $\neg x_i$ appears. We let $D = \max_i \{a_i, b_i\}$ be the maximum number of occurrences of any literal.

We divide the game into two phases which correspond to the different nodes in $Ord(start)$: the "assignment" phase, consisting of the time strictly before the $(D + 1)n + 1$'th visit to the vertex *start* where the switching node takes us to the node as and the "verification" phase consisting of the time afterwards where the switching takes us to either *ver* or *target*. These phases correspond to the following key objectives:

- **Assignment Phase** - In this phase we make $D + 1$ random choices of assignment at each variable x_i. If we ever make an inconsistent choice at

some x_i the vertex cons_i will force us to visit bad, which brings the game to an early end. Every time we make a choice we also visit the consequence gadgets to initialise these. Assuming we make consistent choices we make D visits to the consequences gadget and can only make at most w_l visits to each C_l gadget which means we can't reach their internal fail state, thus we either enter the "verification" phase or reach the vertex *bad*.

- **Verification Phase** - In this phase we know we made consistent choices, then we check how many times we have visited each clause gadget by looping through each. Any clause C_l which was visited w_l times in "assignment" phase will take us to fail and otherwise our clauses will return us to start, thus in this phase we either reach some fail_l or visit all the ver_l vertices, return to start for a final time then reach target.

We use "pass" to refer to a path from an entry to the exit of a gadget. We now explain each of the gadgets and their purpose.

- **The Control Gadget.** In this structure shown in Figure 8 we enforce the phases using the switching behaviour at start. The node as cycles through the n variable gadgets, visiting each $D+1$ times in the "assignment" phase. The node ver finally starts the verification process by moving through ver_1 to ver_m, visiting each once. We note any more visits to start send us to target. We note our variable gadgets are have one exit back to start and another to the vertex *bad*, which randomly moves to either target or fail.
- **Variable Gadget.** In this gadget shown in Figure 9 we make random assignment choices for x_i and enforce consistency and initialise our clause gadgets. The nodes x_i^T and x_i^F represent choosing an assignment of the variable x_i on this pass. The first time we visit these we go to cons_i, this provides a check we have only visited one of x_i^T and x_i^F, if during our play we ever make an inconsistent choice we move to *bad*, preventing us from ever reaching both Neg_i and Pos_i. After our first visit we make successive visits to the respective Consequence gadget Neg_i or Pos_i. As we make up to $D+1$ passes we either reach bad or make exactly D passes of the respective consequence gadget.
- **Consequences Gadget.** We have two consequences gadgets for each variable, Neg_i and Pos_i, shown in Figures 10a and 10b. Neg_i (resp. Pos_i) enumerates the gadgets for clauses, $C_{j_1}, \ldots, C_{j_{a_i}}$ (resp. $C_{k_1}, \ldots, C_{k_{b_i}}$), where the literal $\neg x_i$ (resp. x_i) appears. As a consequence of choosing assignment of true (resp. false) these clauses aren't immediately satisfied by our assignment. As any literal appears in at most D clauses by visiting this gadget D times we are guaranteed to go through each of the contained clause gadgets. If we have $a_i < D$ (resp. $b_i < D$) then any further edges proceed straight to the exit to ensure if we make exactly D passes we visit each clause gadget exactly once. These respectively enumerate the clauses in which the literals $\neg x_i$ and x_i appear.
- **The Clause Gadget.** This is shown in Figure 4c for a clause C_l of width w_l. We note in the "assignment" phase we only ever use the c_l entrance and

in the "verification" phase we use the entrance ver_l. In the "assignment" phase we pass through the clause gadget only in the following situations:

- From a Neg_i gadget where we have assigned x_i true on this pass and $\neg x_i$ appears in C_l,
- From a Pos_i gadget where we have assigned x_i false on this pass and x_i appears in C_l,

Thus as a consequence of our truth assignment to x_i it doesn't witness the truth of C_l. Our clause C_l has width w_l and if our assignment is satisfying then we must have at least one of the w_l literals as a witness to the truth of C_l. Thus our gadget acts as a simple counter of the number of literals in the clause which evaluate to false, after w_l from c_l passes our switch sends the play to the sat_l state, because the assignment we have chosen does not satisfy C_l. In the "assignment" phase as we make at most w_l passes we can't reach fail_l. Finally in the "verification" phase we visit sat_l, if it was visited in the "assignment" phase we know that C_l wasn't satisfied and we move to the fail_l state, otherwise as it is our first visit we move to start and note that C_l was satisfied.

To compute the value of the instance $G(\varphi)$ we note there are three distinct cases which lead us to one of the dead end states $\mathit{target}, \mathit{fail}$ and each of the fail_l states:

- A - We reach one of target or fail from the outgoing edges from bad.
- B - We reach target using the edge from start.
- C - We reach fail_l using the edge from sat_l inside one of our C_l clause gadgets.

We note that we are in case (A) in any play where we reach bad, this occurs when we make two visits to cons_i inside some variable gadget Rx_i and in the other cases we don't reach bad and make at most one visit to each cons_i node. To be in case (B) or (C) we must reach the "verification" phase, requiring us to pass through each variable gadget exactly $D+1$ times. We consider the probability that our random choices at ag_i doesn't take us to cons_i twice with exactly $D+1$ passes, this means it must only visit exactly one of x_i^T or x_i^F, which it does with probability $2^{-(D+1)}$. Thus we reach the "verification" phase with probability $(2^{-(D+1)})^n$, as we independently progress through each of the n variable gadgets, thus the probability of case (A) is $1 - 2^{-n(D+1)}$.

We now assume we are not in case (A) and reach the "verification" phase. Thus we must have made $D+1$ passes of each variable gadget Rx_i and must have only visited exactly one of x_i^T or x_i^F, we let $V : [n] \to T, F$ be a function which chooses this vertex, so that for each i we visited $x_i^{V(i)}$. Each such V corresponds one-to-one with a play reaching the "verification" phase and this play has measure $2^{-n(D+1)}$ and from reaching the verification phase is deterministic as we can not revisit the nodes as or ag without taking the edge from start to target and this prevents us visiting any further random nodes. Thus each V corresponds to single play in either case (B) or case (C), we now show that V corresponds to a case (B) play if and only if V is a satisfying valuation of φ.

Assume V is a satisfying valuation of φ, then for each clause C_l in φ we can find some variable x_i which witnesses the truth of that clause, either by $V(i) = T$ and x_i appearing in C_l or by $V(i) = F$ and $\neg x_i$ appearing in C_l. Consider the "assignemnet" phase where we have $V(i) = T$ (resp. $V(i) = F$) then we note in the gadget Rx_i we only visit the Neg_i (resp. Pos_i) gadget. As we have that x_i (resp. $\neg x_i$) appears in C_l we know that there is an edge from Pos_i (resp. Neg_i) to C_l, and as we only visit the Neg_i (resp. Pos_i) gadget then we can not traverse this edge. Thus we can make at most $w_l - 1$ traversals of C_l via c_l as we can use each incoming edge at most once and we have shown there is one of the w_l incoming edges we can not use ever. Thus we must not visit sat_l in the "assignment" phase, thus if we visit ver_l in the "verification" phase we return to start. As this argument holds for each l we see we visit each ver_l and proceed to target. Thus V satisfying gives us a play in case (B).

Now assume V is not a satisfying valuation of φ, then there is some clause C_l in φ which evaluates to false. Let x_i be some variable where x_i (resp. $\neq x_i$) appears in C_l, then we must have $V(i) = F$ (resp. $V(i) = T$). we can find some variable x_i which witnesses the truth of that clause, either by $V(i) = T$ and x_i appearing in C_l or by $V(i) = F$ and $\neg x_i$ appearing in C_l. Consider the "assignemnet" phase where we have $V(i) = F$ (resp. $V(i) = T$) then as we make $D + 1$ visits to $x_i^{V(i)}$ we make D visits to Pos_i (resp. Neg_i), as we have that x_i (resp. $\neg x_i$) appears in C_l we must take the edge from pos_i (resp. neg_i) to the C_l gadget. As this applies for each literal appearing in C_l we make w_l visits to the C_l gadget in the "assignemnet" phase. Thus if we visit ver_l then we will make a second visit to sat_l and thus reach fail_l. Thus we must reach some fail_l state and thus V not satisfying corresponds to a play in case (C).

We note that each valuation V is obtained under some random choices with each possible valuation having probability $2^{-n(D+1)}$. We also have that a valuation chosen uniformly at random has probability p_φ of being satisfying, thus we have a probability of $p_\varphi \cdot 2^n \cdot 2^{-n(D+1)}$ of being in case (B) and of $(1 - p_\varphi) \cdot 2^n \cdot 2^{-n(D+1)}$. We note in case (A) we reach bad with probability $1 - 2^{-n(D+1)}$, thus in case (A) we have probability $\frac{1}{2}(1 - 2^{-n(D+1)})$ of reaching both target and fail. Combining the half of plays in case (A) and all case (B) we have $v = \frac{1}{2}(1 - 2^{-n(D+1)}) + p_\varphi \cdot 2^n \cdot 2^{-n(D+1)}$ which is as required in Equation (5). □