Verifying Algorithmic Versions of the Lovász Local Lemma

Abstract. Algorithmic versions of the Lovász Local Lemma (ALLLs), or rather, the Moser-Tardos algorithm and its variants, are impactful in both theory and practice. In this paper, we take the first step towards the goal of formally verifying ALLLs by applying programming language techniques. We propose two proof recipes, called loop truncation and resampling-table-based coupling, for bridging the gap between Hoare-style program logics and ALLLs' original informal proofs. We formally verify six existing important results related to ALLLs, and propose a new result which generalizes several existing results. Our proof recipes can also be used to verify general properties of other probabilistic programs in addition to ALLLs.

1 Introduction

The Lovász Local Lemma [21, 58] (LLL) is a powerful tool in combinatorics. It guarantees the existence of a combinatorial object with certain properties in a probability space. It has also been helpful for proving the existence of solutions to numerous significant problems in computer science, such as the Boolean Satisfiability Problem and the Graph Coloring Problem, since these problems can be viewed as instances of the problem of finding some combinatorial objects.

Besides proving the solution's existence, we also want to *efficiently construct* a solution. To this end, people have devised algorithmic versions of the Lovász Local Lemma (ALLLs). The most notable one is the Moser-Tardos (MT) algorithm proposed by Moser and Tardos in their Gödel Prize-winning paper [51]. The algorithm searches the probability space for the desired combinatorial object iteratively, bringing us a constructive proof for LLL. It is efficient in that the expected total number of iterations is bounded. Since then, a huge number of works have emerged, some explore the power of the MT algorithm [54, 44, 32, 43, 1, 38], some find variants of the MT algorithm [32, 18, 35, 31, 26, 37, 30, 15], and some utilize the MT algorithm to solve problems in various areas of computer science [32, 43, 34, 11, 27, 56, 17, 16, 28, 24], including applications in real-world systems [2, 40].

Therefore it is of great importance to formally verify the (total) correctness of ALLLs, in particular, that the MT algorithm and its variants almost surely terminate (i.e. terminate with probability 1) and their expected iteration times have certain upper bounds. Previous works (e.g. [51]) have given proofs for the correctness of ALLLs, though these proofs are rather informal. Therefore, a natural choice is to formally verify ALLLs by formalizing existing informal proofs.

However, we encounter a challenge when verifying ALLLs by following existing proofs. We propose *Proof Recipe 1* to circumvent this challenge, and propose *Proof Recipe 2* for completing the verification after applying *Proof Recipe 1*.

Challenge: Handling infinite execution traces. It is challenging to formulate some subgoals in ALLLs' existing informal proofs using distribution-based semantics, which is commonly used in the literature of probabilistic program verification. The reason is that, on the one hand, these subgoals are about complex properties of the algorithm's execution traces, and we have to take *infinite* traces into account until we prove their absence. On the other hand, distribution-based semantics can only describe certain simple properties of these infinite traces, e.g. their overall probability.

Proof Recipe 1. We propose a proof recipe called *loop truncation* to circumvent the above challenge. For a loop in an ALLL, we transform it to a set of arbitrarily truncated loops. Now we have a set of "truncated algorithms", which can only generate *finite* execution traces. Then, instead of directly verifying the original algorithm, we prove a common bound of the expected iteration times for all the truncated algorithms. The latter can be proved following existing proofs, and now we do not have to handle infinite traces when formulating the subgoals.

Proof Recipe 2. A crucial step commonly found in many proofs of ALLLs, is to prove *an inequality between probabilities involving two programs.* Specifically, for the original ALLL program C_1 and a property $\mathbf{p}$, one constructs a program C_2 and a property $\mathbf{q}$, and shows that the probability of $\mathbf{p}$ holding after C_1's execution is not greater than the probability of $\mathbf{q}$ holding after C_2's execution.

To prove this inequality, existing informal proofs introduce variants of C_1 and C_2, say C_1' and C_2', that use a new random source called *resampling table*. By assuming that C_1 and C_2 are respectively equivalent to C_1' and C_2', they reduce the original inequality to a similar inequality that involves C_1' and C_2', and prove the latter. We elaborate on these proofs in Sec. 2.1.

Following the above proof idea, we propose a proof recipe called *resampling-table-based coupling* to formally prove the aforementioned inequality. At the core of this proof recipe is a new measure-theoretic semantics for probabilistic programs, which we call a *resampling-table-based semantics*. This semantics formalizes the *resampling table* in existing proofs as a built-in structure. We formulate C_1' (C_2') by giving C_1 (C_2) this new semantics without changing its syntax, and express the equivalence between C_1 and C_1' (C_2 and C_2') as the equivalence between a classic probabilistic semantics and the new semantics. We prove the semantics equivalence once and for all, instead of repeatedly proving the equivalence between every pair of programs. Then it remains to prove the inequality involving C_1' and C_2', which is now an inequality on the new semantics.

Our proof recipe, resampling-table-based coupling, further reduces the problem to verifying the two programs C_1' and C_2' individually. The idea is to introduce an intermediate assertion specifying the resampling table as the common random source to bridge the two programs' unary verification. The unary verification can be done using a simple Hoare-style program logic.

Contributions. Using the above two proof recipes, we have successfully verified several ALLL-related results. In summary, we make the following contributions:

– We verify six important results from [51, 54, 44, 32] for the first time. They include all the three "probabilistic" results from Moser and Tardos's Gödel Prize-winning paper [51].
– We propose a proof recipe called *loop truncation*, which circumvents the challenge when verifying ALLLs with classic distribution-based semantics.
– We propose a proof recipe called *resampling-table-based coupling.* It expresses the informal proof idea of an important inequality in a formal and concise way, taking a perspective of semantics equivalence and Hoare-style reasoning.
– We propose a new result related to the Moser-Tardos algorithm, with results from [51, 54, 44] as its corollaries. The statement and the proof of this result are formal, and the proof is done by applying our proof recipes.

Our proof recipes can also be used to prove general properties (i.e. total correctness and inequalities between probabilities) of probabilistic programs *beyond* ALLLs (see Ex. 1 and Ex. 2). We also discuss the relationship between our proof recipes and existing formal proof methods for *positive almost sure termination* and *asynchronous coupling* in Sec. 7.

Outline. We review the original informal proof of the MT algorithm, and introduce the challenge and our main ideas in Sec. 2. We then give the mathematical preliminaries in Sec. 3, and define the programming language, including our new semantics, in Sec. 4. Then we introduce our two proof recipes in Sec. 5. By applying these recipes, we verify six existing important ALLL-related results and a new result in Sec. 6. We finally discuss related work in Sec. 7.

The appendix contains the full formal details of this work, including all the definitions and all the proofs for lemmas, theorems and examples.

2 Informal Development

To formally verify the ALLL-related results, a natural choice is to follow their original informal proofs. Below we first provide a brief overview of the original informal proof of Moser and Tardos's seminal result [51], which serves as an example for understanding the ideas behind the original proofs of many ALLL-related results. We then explain the verification challenge and our proof recipes.

2.1 Moser and Tardos's Proof

The Moser-Tardos (MT) algorithm efficiently constructs a solution for the following problem. Given N mutually independent random variables $X_1, \ldots, X_N$ and M events $\eta_1, \ldots, \eta_M$, where each variable is associated with some random

Algorithm 1 The MT algorithm

```
Independently sample $X_1, \dots, X_N$
while $\exists j \in [1, M]$. $\eta_j$ holds do
    Choose such an $\eta_j$
    for all $X_i$ that $\eta_j$ depends on do
        Resample $X_i$
Output the current values of $X_1, \dots, X_N$
```

Program 2 check(wt)

```
$succ := 1$
for all $\eta_j \in g_{\mathsf{WT}}(wt)$ do
    for all $X_i$ that $\eta_j$ depends on do
        Resample $X_i$
    if $\eta_j$ does not hold then
        $succ := 0$
Output $succ$
```

Fig. 1. The Moser-Tardos algorithm and the check(wt) program.

distribution and each event depends on some of $X_1, \dots, X_N$, we would like to construct an assignment of $X_1, \dots, X_N$ such that none of the M events occurs. The Lovász Local Lemma [21, 58] provides the Erdős-Lovász condition which specifies the probability space, and sufficiently ensures the existence of such assignments. The MT algorithm finds such an assignment as shown in Fig. 1.

Moser and Tardos prove that, under the Erdős-Lovász condition, the expectation of the total iteration number of the algorithm's outer loop is no more than a real number r_{EL}, and thus the algorithm almost surely terminates. (Here we do not expose the definitions of the Erdős-Lovász condition and r_{EL}, which can be found in Thm. 4.) In the remainder of this subsection, we sketch their proof.

Restatement of the proof goal. Moser and Tardos restate their proof goal using *execution logs*. For every execution of the algorithm, its execution log Λ is a sequence of events η_j, which are dynamically chosen at the beginning of the outer loop iterations. We write $\Lambda\langle i \rangle$ for the i-th element of Λ, which is the event chosen at the i-th iteration. We write $|\Lambda|$ for the length of Λ, so it specifies the total number of the outer loop iterations. If the loop does not terminate in an execution, then $|\Lambda| = \infty$. Now, Moser and Tardos restate their proof goal as

$$\mathbb{E}[|\Lambda|] \le r_{\mathsf{EL}}. \tag{1}$$

That is, the expected length of the execution log has an upper bound r_{EL}, where the randomness of Λ comes from the randomness of the MT algorithm. From (1), Moser and Tardos conclude that the program almost surely terminates. The proof of (1) can be divided into three stages, which will be discussed in turn.

Stage 1. In this stage, Moser and Tardos rewrite $\mathbb{E}[|\Lambda|]$ by defining a special mathematical structure called *witness trees*. A witness tree wt is a tree with some special properties, where each node is labeled with an event from $\eta_1, \dots, \eta_M$. One can construct a witness tree wt from an execution log Λ following some specific procedure, and we write $wt = f_{\mathsf{WT}}(\Lambda)$ for this. From the concrete definitions and properties of wt and f_{WT} (which we omit here), Moser and Tardos rewrite $\mathbb{E}[|\Lambda|]$ as the infinite series in (2). It enumerates all witness trees wt, and sums

Algorithm 3 The RT-MT algorithm

```
Randomly generate an $RT$
Assign the first col. of $RT$ to $X_1, \dots, X_N$
while $\exists j \in [1, M]$. $\eta_j$ holds do
    Choose such an $\eta_j$
    for all $X_i$ that $\eta_j$ depends on do
        Assign the next number of
        the $i$-th row of $RT$ to $X_i$
Output the current values of $X_1, \dots, X_N$
```

Program 4 RT-check(wt)

```
Randomly generate an $RT$
$succ := 1$
for all $\eta_j \in g_{\mathsf{WT}}(wt)$ do
    for all $X_i$ that $\eta_j$ depends on do
        Assign the next number of
        the $i$-th row of $RT$ to $X_i$
    if $\eta_j$ does not hold then
        $succ := 0$
Output $succ$
```

Fig. 2. The RT-MT algorithm and the RT-check(wt) program.

the probabilities that wt can be constructed from some prefix of Λ.

$$\mathbb{E}[|\Lambda|] = \sum_{wt} \Pr[wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda)]. \tag{2}$$

Stage 2. Next, Moser and Tardos give an upper bound of the probability in (2). That is, for all witness trees wt, they prove that

$$\Pr[wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda)] \leq p(wt), \tag{3}$$

where $p(wt)$ is a specific real number related to wt and we omit its definition. Instead of directly proving (3) (which is challenging), Moser and Tardos construct a program check(wt), which outputs either 0 or 1, and then prove the following:

(a) The check(wt) program outputs 1 with probability $p(wt)$.
(b) $\Pr[wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda)] \leq \Pr[\text{check}(wt) \text{ outputs } 1]$.

(3) then follows from the above two properties. The proof of (a) is not difficult. What is really interesting is the proof of (b). To see this, we present the check(wt) program in Fig. 1, where $g_{\mathsf{WT}}(wt)$ gives us an event sequence collecting the labels of wt's nodes in a certain order (in fact, a reversed BFS ordering of wt).

To prove (b), Moser and Tardos observe that whenever wt can be generated by the MT algorithm and check(wt) is run on the *same* random source, check(wt) outputs 1. They capture this observation by specifying the random sources using *resampling tables* (RT) and letting the algorithms explicitly use the tables.

Specifically, Moser and Tardos give the RT-MT algorithm[1] as shown in Fig. 2, and assume that it is "equivalent" to the MT algorithm, i.e., the two algorithms produce the same distribution of execution logs.

At the beginning, the RT-MT algorithm randomly generates a resampling table RT, which has N rows and infinite number of columns. For all $i \in [1, N]$, this step independently samples X_i for an infinite number of times, and fills the

[1] In [51], Moser and Tardos did *not* explicitly introduce new algorithms (RT-MT and RT-check). The code we show here is a possible interpretation of their English texts.

i-th row of RT with these samples. Subsequently, every sampling step of the MT algorithm is replaced by a table-query step in the RT-MT algorithm. For instance, resampling X_i is replaced by reading the leftmost unread element from the i-th row of RT, and assigning the result to X_i. The idea here is to transfer the lazy samplings in the MT algorithm to eager ones: the RT-MT algorithm performs all the samplings ahead of time and stores the results in RT so that it can interpret all subsequent samplings to *deterministic* table queries.

Similarly, Moser and Tardos give the RT-check(wt) program as shown in Fig. 2, and assume that it is "equivalent" to check(wt), i.e., the two programs have the same output distribution.

Since the MT algorithm and check(wt) are "equivalent" to their RT-based counterparts respectively, to prove (b), we only need to show that,

(b') $\Pr[wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda \text{ of the RT-MT algorithm})]$
$\leq \Pr[\text{RT-check}(wt) \text{ outputs } 1].$

Note that the first lines of the RT-MT algorithm and RT-check(wt) are the same, and all other parts of these two programs are non-probabilistic. Thus, we couple the random sources of the RT-MT algorithm and RT-check(wt), or rather, let the first lines of these two programs generate the same RT. Then it remains to prove that, for any RT, if wt can be generated from the RT-MT algorithm using this RT, then RT-check(wt) with the same RT must output 1.

The proof is based on the following observation. If wt can be generated from the RT-MT algorithm using RT, then *in retrospect* RT must have some crucial properties, and these properties will make RT-check(wt) output 1. More precisely, for all events η_j in wt, at the time η_j is chosen in the execution of the RT-MT algorithm, it must hold under the current assignment formed by some of RT's entries. Then, during the execution of RT-check(wt), when the program tests η_j, the test passes because the current assignment must be formed by (almost) the same entries of RT.

Stage 3. Finally, Moser and Tardos prove that,

$$\sum_{wt} p(wt) \leq r_{\mathsf{EL}}, \quad \text{if the Erdős-Lovász condition holds.} \tag{4}$$

It can be proved in a purely mathematical (i.e. program-independent) yet simple way, as pointed out by Srinivasan [59].

Combining all three stages above, Moser and Tardos obtain (1):

$$\begin{aligned}
\mathbb{E}[|\Lambda|] &= \sum_{wt} \Pr[wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda)] && \text{Stage 1, (2)}\\
&\leq \sum_{wt} p(wt) && \text{Stage 2, (3)}\\
&\leq r_{\mathsf{EL}}. && \text{Stage 3, (4)}
\end{aligned}$$

Holes in Moser and Tardos's reasoning. There are at least two issues in the above Moser and Tardos's proof.

First, Moser and Tardos restate their ultimate proof goal as (1) using $|\Lambda|$, the length of the execution log Λ. However, their restatement is ambiguous, since without defining the program semantics, it is unclear how programs are executed and generate execution logs. Similar ambiguity arises when stating those subgoals that also involve quantities related to Λ, e.g. (2) and (3).

Second, Moser and Tardos's original proof of *Stage 2* is far from rigorous. To prove (b), they assume that the MT algorithm and check(wt) are "equivalent" to their RT-based variants, but they did not strictly define and prove the "equivalences". Besides, they did not give a rigorous proof of (b') with these RT-based variants strictly defined.

In the next subsections, we show how we address these issues, and formally state and verify Moser and Tardos's result. We illustrate the proof path in Fig. 3, which is also explained below.

2.2 Stating Proof Goals Using Distribution-Based Semantics

To formally state Moser and Tardos's ultimate proof goal, we must formulate the program semantics and the expected total number of iterations (or equivalently, the expected length of the execution log Λ).

We use a classic distribution-based semantics as the formal program semantics. This semantics (and other equivalent semantics, e.g. the probabilistic wp-semantics [46, 49] and Kozen's "Semantics 2" [45]) is commonly used in the literature of probabilistic program verification (e.g. [46, 49, 5, 9, 3]). It interprets the execution result of a program C as a sub-distribution μ over states. For any state σ, this final state sub-distribution μ specifies the probability that the program C terminates at σ.

For specifying the expected total number of iterations, we introduce a fresh program variable cnt that records the number of iterations. Our code of the MT algorithm, $C_{\mathsf{MT}}(cnt)$, sets cnt to zero at the beginning, and increments it in each iteration of the outer loop. Consequently, when $C_{\mathsf{MT}}(cnt)$ terminates, the value of cnt is the total number of iterations.

Now, our proof goal can be stated as the following *total correctness* Hoare triple (assuming that the Erdős-Lovász condition holds on the probability space):

$$\models [\mathbf{true}]\, C_{\mathsf{MT}}(cnt)\, [\mathbb{E}[cnt] \leq r_{\mathsf{EL}}]\,. \tag{5}$$

Informally it says, the execution of $C_{\mathsf{MT}}(cnt)$ in the distribution-based semantics almost surely terminates (i.e., terminates with probability 1), and the expectation of the value of cnt (represented as $\mathbb{E}[cnt]$) at the final state sub-distribution is no greater than r_{EL}. The goal is shown on the top of Fig. 3.

For proving (5), we follow the original proof. That is, we formulate the subgoals in the three stages in Sec. 2.1 using distribution-based semantics, and then prove them. However, we encounter a challenge when formulating (2) and (3).

$$\begin{tikzcd}
\vDash [\mathbf{true}]\, C_{\mathsf{MT}}(cnt)\, [\mathbb{E}[cnt] \le r_{\mathsf{EL}}] & \\
\boxed{\forall K \in \mathbb{N}. \quad \vDash [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K)\, [\mathbb{E}[cnt] \le r_{\mathsf{EL}} \wedge \lceil cnt \ge 0 \rceil]} \arrow[u, Rightarrow, "①"] & \text{Recipe 1} \\
\boxed{\cdots \quad \wedge \quad \vDash \{\mathbf{true}\}\, C'_{\mathsf{MT}}(cnt, K) \le \text{check}(wt)\{\mathbf{q}_1, \mathbf{q}_2\}} \arrow[u, Rightarrow, "②"] & \\
\vDash_{\mathrm{RT}} \{\mathbf{true}\}\, C'_{\mathsf{MT}}(cnt, K) \le \text{check}(wt)\{\mathbf{q}_1, \mathbf{q}_2\} \arrow[u, squiggly, "\mathrm{RT}", "③"'] & \text{Recipe 2} \\
\boxed{\begin{array}{c} \vDash_{\mathrm{RT}} \{\cdots\}\, C'_{\mathsf{MT}}(cnt, K)\, \{\mathbf{q}_1 \Rightarrow \mathbf{R}\} \wedge \\ \vDash_{\mathrm{RT}} [\cdots \wedge \mathbf{R}]\, \text{check}(wt)\, [\mathbf{q}_2] \end{array}} \arrow[u, Rightarrow, "④"] &
\end{tikzcd}$$

Fig. 3. Our proof path of Moser and Tardos's result, where $\mathbf{q}_1 = \mathsf{Gen}(wt, cnt, K)$ and $\mathbf{q}_2 = \mathsf{Succ}$.

Challenge: Handling infinite execution traces. The problem arises when formulating the probability (6), which appears in both (2) and (3).

$$\Pr[wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda)] \tag{6}$$

Let μ be the final state sub-distribution of $C_{\mathsf{MT}}(cnt)$. Then, it is challenging to formulate (6) using μ. Note that (6) can be positive even when $C_{\mathsf{MT}}(cnt)$ never terminates. But if we simply define (6) as the probability of some event on μ, this probability must be 0 if $C_{\mathsf{MT}}(cnt)$ never terminates, since μ is now a null sub-distribution (which specifies that $C_{\mathsf{MT}}(cnt)$ terminates at σ with probability 0 for any σ). Other definition attempts using μ may also fail.

The difficulty in formulating (6) lies in the following facts. On the one hand, (6) is the total probability of $C_{\mathsf{MT}}(cnt)$'s possibly infinite execution traces on which $wt = f_{\mathsf{WT}}(\text{some prefix of } \Lambda)$ holds. This is a complex property that may involve only some of $C_{\mathsf{MT}}(cnt)$'s infinite traces. On the other hand, distribution-based semantics can only express certain simple properties of infinite traces, and thus cannot express (6). From μ, all we know about $C_{\mathsf{MT}}(cnt)$'s infinite traces is their overall probability $1 - |\mu|$, where $|\mu|$ is the weight of μ (see Sec. 3.1).

One should not simply rule out infinite traces by strengthening (2) and (3) to include almost sure termination of $C_{\mathsf{MT}}(cnt)$, since in Sec. 2.1 the termination has not been derived until the ultimate goal is fully proved (also, it is not easy to prove the termination alone, as discussed in Sec. 7).

2.3 Proof Recipe 1: Loop Truncation

We circumvent the aforementioned challenge by proposing *loop truncation*. Our idea is to do a code transformation on loops, so that the codes after transformation do not generate infinite traces. For the main loop in $C_{\mathsf{MT}}(cnt)$, our transformation introduces a loop bound K whose value is an arbitrary natural number, and turns the original loop **while** (b) **do** C into a set of truncated loops $\{$ **while** $(b \wedge cnt < K)$ **do** $C \mid K \in \mathbb{N}\}$. Since we increment cnt in the loop body C, each truncated loop **while** $(b \wedge cnt < K)$ **do** C terminates in at most K rounds, and thus can only generate finite execution traces.

Soundness of this transformation can be captured by Lem. 1 below (we will show the more general form in Thm. 2 in Sec. 5.1). It says, the original loop guarantees almost sure termination and its expected total iteration number is bounded by r, as long as all the truncated loops terminate and their expected total iteration numbers have the same upper bound r. Here $\lceil cnt \geq 0 \rceil$ says, cnt, the number of iterations, is always non-negative after **while** $(b \wedge cnt < K)$ **do** C's execution. Without this condition the transformation is unsound.

Lemma 1. *For all P, b, C, r, if*

$$\forall K \in \mathbb{N}. \quad \models [P]\, \textbf{while}\ (b \wedge cnt < K)\ \textbf{do}\ C\, [\mathbb{E}[cnt] \leq r \wedge \lceil cnt \geq 0 \rceil]\,,$$

then $\models [P]\, \textbf{while}\ (b)\ \textbf{do}\ C\, [\mathbb{E}[cnt] \leq r]$.

Using this transformation, we can reduce (5) to proving the total correctness of $C'_{\mathsf{MT}}(cnt, K)$ for all K, where $C'_{\mathsf{MT}}(cnt, K)$ is the resulting code after transforming the main loop of $C_{\mathsf{MT}}(cnt)$ to a truncated one. That is, we prove (7) for all K.

$$\models [\textbf{true}]\, C'_{\mathsf{MT}}(cnt, K)\, [\mathbb{E}[cnt] \leq r_{\mathsf{EL}} \wedge \lceil cnt \geq 0 \rceil] \tag{7}$$

We show this as Step ① in Fig. 3. The double arrow represents logical implication. Then we can prove (7) following Moser and Tardos's proof ideas explained in Sec. 2.1. We formulate subgoals (2) and (3) for $C'_{\mathsf{MT}}(cnt, K)$; however, we will not encounter the aforementioned challenge, since $C'_{\mathsf{MT}}(cnt, K)$ does not have infinite execution traces.

Serving as a proof method for PAST. Lem. 1 is itself a general proof method for positive almost sure termination (PAST) [13], whenever we use cnt to record the number of program steps. The PAST property says, the program terminates not only almost surely, but also within finite number of steps in expectation. We give an example in Ex. 1 in Sec. 5.1.

2.4 Proof Recipe 2: Resampling-Table-Based Coupling

Following the ideas in Sec. 2.1, we prove (7) in three stages. The most challenging part is proving (b) in *Stage 2*, which is an inequality between probabilities involving two programs.

We first formally specify the inequality. To this end, we introduce the tuple $\models \{P\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}$. Here P is a predicate specifying state distributions μ, while $\mathbf{q}_1$ and $\mathbf{q}_2$ are predicates over states σ. The tuple says that, the probability of $\mathbf{q}_1$ holding at the terminating states of C_1 is not greater than the probability of $\mathbf{q}_2$ holding at the terminating states of C_2, where C_1 and C_2's executions start from the same μ satisfying P and use the distribution-based semantics. Then, we can formulate (b) for $C'_{\mathsf{MT}}(cnt, K)$ and check(wt) as follows.

$$\models \{\mathbf{true}\} C'_{\mathsf{MT}}(cnt, K) \leq \text{check}(wt)\{\mathsf{Gen}(wt, cnt, K),\ \mathsf{Succ}\}. \tag{8}$$

Here $\mathsf{Gen}(wt, cnt, K)$ roughly says that wt can be generated and is well-formed with respect to cnt and K. The predicate Succ says that the output $succ$ is 1. See Step 2 in Fig. 3.

Following Moser and Tardos's proof in Sec. 2.1, we introduce the RT-MT algorithm (now with a truncated loop) and the RT-check(wt) program. We need to give strict definitions of these variants, and to prove that they are indeed equivalent to the original $C'_{\mathsf{MT}}(cnt, K)$ and check(wt) respectively.

Resampling-table-based semantics. Instead of introducing the RT-MT algorithm and the RT-check(wt) program with explicit statements for generating the RT and accessing it, our approach is to *keep the program syntax unchanged but re-interpret the code using a new semantics*. Our RT is a built-in structure of the new semantics, and it is randomly generated before programs start execution.

More specifically, we re-interpret (8) using the novel *RT-based semantics*. In this semantics, we let a program execute with a resampling table RT, which stores all sampling results of the program in advance, and serves as an oracle for the sampling statements in the program. Each sampling statement is interpreted as a query to RT. So this semantics is deterministic given a specific RT.

Our RT-based semantics is equivalent to the classic distribution-based semantics explained in Sec. 2.2. By specifying and proving the semantics equivalence, we essentially show that all programs (including the MT algorithm and check(wt) in Sec. 2.1) are "equivalent" to their RT-based variants.

Based on the semantics equivalence, we can show the equivalence between $\models \{P\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}$ and $\models_{\text{RT}} \{P\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}$. The latter specifies the same relational property as the former but uses the RT-based semantics for execution. See Step 3 in Fig. 3.

Resampling-table-based coupling. Our proof recipe reduces the relational verification for $\models_{\text{RT}} \{P\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}$ to unary verification of each of C_1 and C_2 in the RT-based semantics.

Specifically, we couple the random sources of C_1 and C_2, i.e. let them use the same RT in their executions. We prove: for all RT, if C_1 using RT terminates on a state satisfying $\mathbf{q}_1$, then C_2 using the same RT must also terminate on a state satisfying $\mathbf{q}_2$.

To prove this, we introduce an intermediate assertion $\mathbf{R}$ to describe what kind of RT can make $\mathbf{q}_1$ hold after the execution of C_1. Usually $\mathbf{R}$ specifies that "some entries in RT have some properties". With $\mathbf{R}$, we can split the goal into the following two subgoals:

– For all RT, if C_1 using RT terminates at a state satisfying $\mathbf{q}_1$, then *in retrospect* RT must satisfy $\mathbf{R}$. This is formulated as the Hoare-triple

$$\models_{\text{RT}} \{\cdots\}\, C_1\, \{\mathbf{q}_1 \Rightarrow \mathbf{R}\}\,. \tag{9}$$

The post-condition reflects this retrospective reasoning. We omit the pre-condition, which usually degenerates to a regular state assertion. Then we only need classical (non-probabilistic) Hoare-style proofs for the Hoare triple.
– Starting with any RT satisfying $\mathbf{R}$, the execution of C_2 must terminate at a final state satisfying $\mathbf{q}_2$, that is,

$$\models_{\mathrm{RT}} [\cdots \wedge \mathbf{R}]\, C_2\, [\mathbf{q}_2]\,. \tag{10}$$

Here $\mathbf{R}$ is in the precondition. We omit the rest parts of the precondition.

Note that the first subgoal (9) only needs to be *partial correctness*. It says, for any execution of C_1, if it terminates and the final state satisfies $\mathbf{q}_1$, RT must satisfy $\mathbf{R}$. Then the *total correctness* of C_2 (the second subgoal (10)) says, starting from the same RT, C_2 terminates at a final state satisfying $\mathbf{q}_2$. This way we can prove that the probability of $\mathbf{q}_1$ at the end of C_1 is not greater than the probability of $\mathbf{q}_2$ at the end of C_2. Step ④ in Fig. 3 shows this reduction of the relational reasoning to unary proofs of the two programs separately.

Our reasoning above benefits from a key novelty of our RT-based semantics with respect to existing random-source-based semantics (e.g. Kozen's "Semantics 1" [45] and those in [12, 19]). That is, our RT is an immutable structure that never changes during program execution. In particular, used samples are not popped out of RT. Therefore the assertion $\mathbf{R}$ derived from the post-condition of (9) must also hold over the RT at the beginning of the execution. So we can use it in the precondition in (10).

Finding such an $\mathbf{R}$ is not difficult in many cases, especially when verifying ALLLs. We give another example in Sec. 5.2.

3 Preliminaries

In this section, we review some fundamentals of probability theory in two stages. We first introduce some basics of discrete probability theory without mentioning their measure-theoretic extensions, serving as the foundation of our distribution-based semantics in Sec. 4.1. Then we turn to the measure-theoretic probability theory, which forms the basis of our RT-based semantics in Sec. 4.2.

3.1 Discrete Probability Theory

We use notations from [3, 5]. A (discrete) *sub-distribution* over a set A is defined as a function $\mu : A \to [0, 1]$ that satisfies the following conditions:

– The support of μ, denoted by $supp(\mu) = \{a \in A : \mu(a) > 0\}$, is countable;
– $|\mu| \leq 1$, where $|\mu| = \sum_{a \in A} \mu(a)$ is μ's weight. Since $supp(\mu)$ is countable, the above possibly infinite series is well-defined.

Intuitively, for $a \in A$, $\mu(a)$ is the probability of drawing a from μ, and $supp(\mu)$ is the set of all elements that can be drawn from μ with positive probability. A sub-distribution μ is called a *distribution* if $|\mu| = 1$. We denote by $\mathbb{SD}_A$ all of the sub-distributions over A, and by $\mathbb{D}_A$ all of the distributions over A.

An event[2] is a function $E : A \to \text{Prop}$, and a random variable is a function $V : A \to \mathbb{R}$. We write $\Pr_{a\sim\mu}[E(a)]$ for the probability of E on the sub-distribution μ, and $\mathbb{E}_{a\sim\mu}[V(a)]$ for the expected value of V on μ. They are defined as follows.

$$\Pr_{a\sim\mu}[E(a)] \;\triangleq\; \textstyle\sum_{a\in A:E(a)} \mu(a) \qquad \mathbb{E}_{a\sim\mu}[V(a)] \;\triangleq\; \textstyle\sum_{a\in A} \mu(a)\cdot V(a)$$

For an infinite sequence $\vec{\mu}$, we define the limit of $\vec{\mu}$ as follows:

$$\lim \vec{\mu} \;\triangleq\; \mu, \;\text{ if } \lim_{n\to\infty} \textstyle\sum_{a\in A} |\vec{\mu}[n](a) - \mu(a)| = 0.$$

One can prove that such a μ is unique if it exists, otherwise we say $\vec{\mu}$ diverges and leave $\lim \vec{\mu}$ undefined.

For $\mu \in \mathbb{SD}_A$ and function $f \in A \to \mathbb{SD}_B$, we define the *expected sub-distribution* $\mathbb{E}_{a\sim\mu}\{f(a)\} \in \mathbb{SD}_B$ as follows:

$$\mathbb{E}_{a\sim\mu}\{f(a)\} \;\triangleq\; \lambda b.\; \textstyle\sum_{a\in A} \mu(a)\cdot f(a)(b).$$

3.2 Measure-Theoretic Probability Theory

To define the RT-based semantics (Sec. 4.2), we need tools from measure theory.

A set of subsets of a set Ω, say $\mathcal{F}$, is a σ-algebra on Ω if it contains Ω and is closed under complement and countable union. A measurable space is defined as a pair $(\Omega, \mathcal{F})$, where $\mathcal{F}$ is a σ-algebra on Ω. We call Ω the *sample space*.

A function $\mathcal{M} : \mathcal{F} \to [0, \infty)$ is called a (finite) *measure* on measurable space $(\Omega, \mathcal{F})$ if it satisfies $\mathcal{M}(\varnothing) = 0$ and is countably additive. A *measure space* is defined as a triple $(\Omega, \mathcal{F}, \mathcal{M})$, where $\mathcal{M}$ is a measure on measurable space $(\Omega, \mathcal{F})$. $(\Omega, \mathcal{F}, \mathcal{M})$ is called a *probability space* if $\mathcal{M}(\Omega) = 1$.

A discrete distribution μ can be lifted to a measure-theoretic probability space $(\Omega, \mathcal{F}, \mathcal{M})$, where $\Omega = \mathit{supp}(\mu)$, $\mathcal{F} = \mathcal{P}(\mathit{supp}(\mu))$, and $\mathcal{M}(A) = \sum_{a\in A} \mu(a)$ for all $A \subseteq \mathit{supp}(\mu)$. Thus, for $a \in \mathit{supp}(\mu)$, $\mathcal{M}(\{a\})$ is exactly $\mu(a)$, the probability of drawing a from the distribution μ.

Let $\{(\Omega_i, \mathcal{F}_i, \mathcal{M}_i) : i \in I\}$ be a collection of probability spaces for some possibly infinite set I. We denote by $\prod_{i\in I}(\Omega_i, \mathcal{F}_i, \mathcal{M}_i)$ the *product probability space* of $\{(\Omega_i, \mathcal{F}_i, \mathcal{M}_i) : i \in I\}$, defined as $(\Omega, \mathcal{F}, \mathcal{M})$, where

- $\Omega = \prod_{i\in I} \Omega_i$;
- $\mathcal{F}$ is the smallest σ-algebra containing all $\prod_{i\in I} A_i$, where $A_i \in \mathcal{F}_i$ for each $i \in I$ and $\{j : A_j \subsetneq \Omega_j\}$ is finite;
- $\mathcal{M}(\prod_{i\in I} A_i) = \prod_{j\in J} \mathcal{M}_j(A_j)$, where $A_i \in \mathcal{F}_i$ for each $i \in I$ and $J = \{j : A_j \subsetneq \Omega_j\}$ is finite.

The above $(\Omega, \mathcal{F}, \mathcal{M})$ exists and is unique (see [55]).

$$(\mathit{Dsts})\ \mathcal{D} ::= (\kappa_1, \ldots, \kappa_N) \qquad (\mathit{Evts})\ \mathcal{E} ::= (\eta_1, \ldots, \eta_M)$$
$$(\mathit{Dst})\ \kappa \in \mathbb{D}_{\mathit{Real}} \qquad (\mathit{Evt})\ \eta \in \underbrace{\mathit{Real} \times \cdots \times \mathit{Real}}_{N\ \mathit{Real}\text{'s}} \to \{\mathrm{true}, \mathrm{false}\}$$

$$\mathrm{vbl}(\eta, j) \ \text{ iff } \ \exists r_1, \ldots, r_N, r'.\ \eta(r_1, \ldots, r_N) \neq \eta(r_1, \ldots, r_{j-1}, r', r_{j+1}, \ldots, r_N)$$
$$\mathrm{P}(\eta) \triangleq \sum_{\substack{r_1 \in \mathit{supp}(\mathcal{D}[1]), \ldots, r_N \in \mathit{supp}(\mathcal{D}[N]) \\ \eta(r_1, \ldots, r_N) = \mathrm{true}}} \ \prod_{i \in [1,N]} \mathcal{D}[i](r_i)$$
$$\Gamma(j) \triangleq \{k : \exists i.\ \mathrm{vbl}(\mathcal{E}[j], i) \wedge \mathrm{vbl}(\mathcal{E}[k], i)\} \setminus \{j\}$$

$$(\mathit{Expr})\ e ::= v \mid x \mid e_1 + e_2 \mid a[e] \mid e_1\langle e_2 \rangle \mid \mathsf{len}(e) \mid \mathsf{app}(e_1, e_2) \mid \ldots$$
$$(\mathit{Bexp})\ b ::= \mathrm{true} \mid \mathrm{false} \mid e_1 = e_2 \mid b_1 \wedge b_1 \mid \mathsf{hold}(e, e_1, \ldots, e_N) \mid \mathsf{vbl}(e_1, e_2) \mid \ldots$$
$$(\mathit{Stmt})\ C ::= \mathbf{skip} \mid x := e \mid x := \mathsf{Sample}(e) \mid a[e_1] := e_2$$
$$\mid C_1; C_2 \mid \mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \mid \mathbf{while}\ (b)\ \mathbf{do}\ C \mid \ldots$$

Fig. 4. Syntax of the programming language.

4 Two Semantics of the Language

In this section we define the programming language. We first define the language syntax, and then give two equivalent semantics in Sec. 4.1 and Sec. 4.2. We give a detailed definition of the language in App. A.

Global parameters. Throughout the paper, we assume four global parameters for programs: N, M, $\mathcal{D}$ and $\mathcal{E}$. They are viewed as meta-variables, and can be configured differently for different programs.

As defined at the top of Fig. 4, $\mathcal{D}$ and $\mathcal{E}$ represent the "N distributions" and "M events" in ALLL's setting (see Sec. 2.1) respectively. Each event η_j in $\mathcal{E}$ takes N reals as input, and outputs a boolean value. Each κ_i in $\mathcal{D}$ is a distribution over reals, and is associated with the i-th argument of every η_j in $\mathcal{E}$.

Fig. 4 also gives important notations related to $\mathcal{D}$ and $\mathcal{E}$, which are used in the statements and the formal proofs of ALLL-related results. $\mathrm{vbl}(\eta, j)$ holds iff the event η depends on its j-th argument.[3] $\mathrm{P}(\eta)$ is the probability of the event η occurring, given that its N arguments are independently distributed according to $\mathcal{D}[1], \ldots, \mathcal{D}[N]$ respectively. $\Gamma(j)$ is the index set of events that depend on some argument that $\mathcal{E}[j]$ also depends on, except $\mathcal{E}[j]$ itself.

Syntax of the programming language. As shown at the bottom of Fig. 4, we use customized program statements, expressions and boolean expressions to formulate ALLLs' code. We write $x := \mathsf{Sample}(e)$ to sample from the distribution $\mathcal{D}[e]$ and store the result in the program variable x. The boolean expression $\mathsf{hold}(e, e_1, \ldots, e_N)$ tests if the event $\mathcal{E}[e]$ holds with arguments $e_1, \ldots, e_N$. Moreover, $\mathsf{vbl}(e_1, e_2)$ tests if the event $\mathcal{E}[e_1]$ depends on its e_2-th argument.

[2] Note that the "event" here is different from the "event" in the "M events" in the Moser-Tardos algorithm (see Sec. 2.1), though they have the same name.

[3] The name "vbl" is short for "variables". Moser and Tardos [51] used $\mathrm{vbl}(\eta)$ as the minimal set of variables (i.e. arguments of the event) that determine η.

We use arrays to formulate the N variables $X_1, \ldots, X_N$ in ALLLs. We use $a[e]$ to represent the element of array a with index e, and use $a[e_1] := e_2$ for the in-place update.

We use lists to formulate the execution logs in ALLLs. To access and manipulate the execution log, we introduce list-related expressions. We use $e_1\langle e_2\rangle$ for the e_2-th element of list e_1, use $\mathsf{len}(e)$ for the length of list e, and use $\mathsf{app}(e_1, e_2)$ for appending an element e_2 to list e_1.

Using the syntax in Fig. 4, we can formulate the code of the MT algorithm, $C_{\mathsf{MT}}(cnt)$, in Fig. 10 in Sec. 6.

States and state distributions. As defined below, a state σ maps each program variable in $PVar$ to some value v. For simplicity, we view each array element as a program variable. A value v is either a real r or a list Λ of natural numbers.

$$(State) \quad \sigma \quad \in \quad PVar \to Val \qquad\qquad (DState) \quad \mu \quad \in \quad \mathbb{D}_{State}$$

State distributions μ are used to specify that, with probability $\mu(\sigma)$, the program state before or after the execution of a program is exactly σ. We write $[\![e]\!]_\sigma$ and $[\![b]\!]_\sigma$ for the evaluation of e and b in a state σ.

Below we give two equivalent probabilistic semantics of our language, a classic distribution-based semantics and an RT-based semantics. We use n for natural numbers and p, r for reals. Throughout this paper, we assume that the program's execution does not get stuck, and the evaluation of expressions does not abort.

4.1 Distribution-Based Semantics

Following [3, 5], we first define the semantic function $[\![C]\!](\sigma) \in \mathbb{SD}_{State}$. Here $[\![C]\!](\sigma)(\sigma')$ represents the probability of C's execution from σ finally reaching σ'. For example, for the sampling operation $x := \mathsf{Sample}(e)$ that samples from the distribution $\mathcal{D}[i]$ and gets r as the result, the probability is $\mathcal{D}[i](r)$. That is,

$$[\![x := \mathsf{Sample}(e)]\!](\sigma)(\sigma') = \begin{cases} \mathcal{D}[i](r) & \text{if } [\![e]\!]_\sigma = i \in [1, N] \text{ and } \sigma' = \sigma\{x \rightsquigarrow r\} \\ 0 & \text{otherwise} \end{cases} .$$

We give the full definition of $[\![C]\!](\sigma)$ in App. A. We further define $[\![C]\!](\mu) \in \mathbb{SD}_{State}$ (where $\mu \in DState$) by lifting $[\![C]\!](\sigma)$, using the expected sub-distribution in Sec. 3.1:

$$[\![C]\!](\mu) \quad \triangleq \quad \mathbb{E}_{\sigma\sim\mu}\{[\![C]\!](\sigma)\}$$

4.2 Resampling-Table-Based Semantics

Informally, in our new RT-based semantics, a program first randomly generates a resampling table (RT); with this table, the program then starts its deterministic execution. Below we first give the definition of an RT, and specify how the semantics "generates" an RT. Then we define an RT-based operational semantics, which describes the deterministic execution of the program with a certain

r_{10}	r_{11}	r_{12}	r_{13}	$\cdots$
r_{20}	r_{21}	r_{22}	r_{23}	$\cdots$

Fig. 5. A resampling table RT with $N = 2$.

RT. Finally, we combine all the above definitions into the RT-based semantic functions $[\![C]\!]_{\mathrm{RT}}(\sigma)$ and $[\![C]\!]_{\mathrm{RT}}(\mu)$.

The resampling table is defined as follows.

$$\begin{array}{llll} (\mathit{RTable}) & RT \in & [1,N] \times \mathit{Nat} \to \mathit{Real} & \text{where generable}(RT) \\ \text{generable}(RT) & \text{iff} & \forall i,j.\ RT[i][j] \in \mathit{supp}(\mathcal{D}[i]) & \end{array}$$

A resampling table RT is a matrix with size $N \times \infty$. An example of such table is shown in Fig. 5, where $N = 2$ and $RT[i][j] = r_{ij}$ for $i \in [1,2]$ and $j \in \mathit{Nat}$. Intuitively, as described in Sec. 2.1, the i-th row of RT stores the ahead-of-time samples from the distribution $\mathcal{D}[i]$. Additionally, we require that generable(RT) holds. That is, every entry in the i-th row of RT must be able to be sampled from the distribution $\mathcal{D}[i]$. This accords with the intuition of the RT.

We specify how the semantics "generates" an RT. To this end, we define the probability space of all (generable) RTs as $(\Omega, \mathcal{F}, \mathcal{M})$, and thus $\mathcal{M}(\{RT \mid \cdots\})$ represents the probability of some RT from set $\{RT \mid \cdots\}$ being generated. The definition is shown below:

$$(\Omega, \mathcal{F}, \mathcal{M}) \triangleq \prod_{(i,j)\in[1,N]\times \mathit{Nat}} (\Omega_{i,j}, \mathcal{F}_{i,j}, \mathcal{M}_{i,j}),$$

where the collection of probability spaces $\{(\Omega_{i,j}, \mathcal{F}_{i,j}, \mathcal{M}_{i,j}) : (i,j) \in [1,N] \times \mathit{Nat}\}$ extends $\mathcal{D}$ such that

- $\Omega_{i,j} = \mathit{supp}(\mathcal{D}[i])$;
- $\mathcal{F}_{i,j} = \mathcal{P}(\mathit{supp}(\mathcal{D}[i]))$;
- $\mathcal{M}_{i,j}(A) = \sum_{r\in A} \mathcal{D}[i](r)$ for $A \subseteq \mathit{supp}(\mathcal{D}[i])$.

Note that $\Omega = \mathit{RTable}$, that is, the sample space of the probability space is indeed the set of all RTs, since by definition we have

$$\Omega = \prod_{(i,j)\in[1,N]\times \mathit{Nat}} \Omega_{i,j} = \prod_{(i,j)\in[1,N]\times \mathit{Nat}} \mathit{supp}(\mathcal{D}[i]) = \mathit{RTable}.$$

As we describe below, the sample space $\Omega_{i,j}$ is the set of all possible values of the table entry with row number i and column number j. Thus, the infinite product of these sets, Ω, is exactly the set of all RTs.

Below we explain our construction of $(\Omega, \mathcal{F}, \mathcal{M})$. Recall that an RT is generated by filling its entries by infinite number of independent samples from $\mathcal{D}[1], \ldots, \mathcal{D}[N]$. Hence, the probability space of all RTs is the *infinite product* of probability spaces of all entries. For the probability space of the entry in row i and (arbitrary) column j, say $(\Omega_{i,j}, \mathcal{F}_{i,j}, \mathcal{M}_{i,j})$, we lift the discrete distribution $\mathcal{D}[i]$ to a measure-theoretic probability space. Since that entry is generated by

$$\frac{[\![e]\!]_\sigma = v}{RT \vdash (x := e, \sigma, \iota) \to (\mathbf{skip}, \sigma\{x \rightsquigarrow v\}, \iota)}$$

$$\frac{[\![e]\!]_\sigma = i \in [1, N] \qquad \iota' = (\iota[1], \ldots, \iota[i-1], \iota[i]+1, \iota[i+1], \ldots, \iota[N])}{RT \vdash (x := \mathsf{Sample}(e), \sigma, \iota) \to (\mathbf{skip}, \sigma\{x \rightsquigarrow RT[i][\iota[i]]\}, \iota')}$$

Fig. 6. RT-based operational semantics.

sampling from $\mathcal{D}[i]$, we have $\Omega_{i,j} = \mathit{supp}(\mathcal{D}[i])$. $\mathcal{F}_{i,j}$ is the set of all subsets of $\Omega_{i,j}$, and the measure of a set $A \in \mathcal{F}_{i,j}$ is given by $\mathcal{M}_{i,j}(A) = \sum_{r \in A} \mathcal{D}[i](r)$, which represents the probability of some element from A being generated as the value of the entry by sampling from $\mathcal{D}[i]$.

We then define the RT-based operational semantics, with selected semantics rules shown in Fig. 6. The definition is almost standard, except that it interprets sampling operations to table queries. Recall that, when the program performs a sampling from the distribution $\mathcal{D}[i]$, it reads the leftmost unread entry in the i-th row of RT as the result. To keep track of these entries, we maintain the heads ι in the program configuration to record their column numbers.

$$(\mathit{Heads}) \quad \iota \quad ::= \quad (n_1, \ldots, n_N)$$

ι is an N-tuple. Its i-th component, $\iota[i]$, represents the column number of the leftmost unread entry in the i-th row of RT. Now, $RT \vdash (C, \sigma, \iota) \to^* (C', \sigma', \iota')$ says that, starting from the program state σ, with the leftmost unread entries of RT initially specified by ι, C deterministically executes to C' using RT, where the result state is σ' and finally the leftmost unread entries in RT are specified by ι'. When the program performs a sampling from $\mathcal{D}[i]$, it takes $RT[i][\iota[i]]$ as the result and increments $\iota[i]$. In other program steps, ι remains unchanged.

Now the RT-based semantic functions are defined below, where $\iota_{\mathsf{init}} = (0, \ldots, 0)$ represents the initial positions of heads.

$$\begin{aligned} [\![C]\!]_{\mathrm{RT}}(\sigma) &\triangleq \lambda\sigma'.\ \mathcal{M}(\{RT \mid RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\}) \\ [\![C]\!]_{\mathrm{RT}}(\mu) &\triangleq \mathbb{E}_{\sigma \sim \mu}\{[\![C]\!]_{\mathrm{RT}}(\sigma)\} \end{aligned}$$

Informally, the probability of C's execution from σ finally reaching σ', say $[\![C]\!]_{\mathrm{RT}}(\sigma)(\sigma')$, is the probability of some RT, which satisfies the following property, being generated: starting from σ, C's execution using RT finally reaches σ'. This property is formally stated as $RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)$, with the help of the operational semantics.

Lem. 2 shows that the RT-based semantics is indeed well-defined. We give the proof in App. A.1.

Lemma 2. *For all $C, \sigma, \sigma', \iota$, $\{RT \mid RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \_)\} \in \mathcal{F}$.*

To conclude this subsection, we give the following theorem, which states the equivalence between the distribution-based semantics defined in Sec. 4.1 and the RT-based semantics. We give the proof in App. A.2.

Theorem 1 (Semantics Equivalence). *For all C and μ, $[\![C]\!](\mu) = [\![C]\!]_{\mathrm{RT}}(\mu)$.*

$$\begin{array}{lrcl}
(\mathit{Assn}) & \mathbf{p},\mathbf{q},\mathbf{r} & ::= & b \mid \neg\mathbf{q} \mid \mathbf{q}_1 \wedge \mathbf{q}_2 \mid \mathbf{q}_1 \vee \mathbf{q}_2 \mid \forall X.\, \mathbf{q} \mid \exists X.\, \mathbf{q} \mid \ \ldots \\
(\mathit{PExp}) & \xi & ::= & r \mid \mathbb{E}[e] \mid \Pr[\mathbf{q}] \mid \xi_1 + \xi_2 \mid \xi_1 - \xi_2 \mid \ \ldots \\
(\mathit{PAssn}) & P, Q, R & ::= & \lceil \mathbf{q} \rceil \mid \xi_1 = \xi_2 \mid \neg Q \mid Q_1 \wedge Q_2 \mid \forall X.\, Q \mid \exists X.\, Q \mid \ \ldots
\end{array}$$

$$\begin{array}{ll}
[\![r]\!]_\mu \triangleq r & \mu \vDash \lceil \mathbf{q} \rceil \ \text{ iff } \forall \sigma.\ \sigma \in \mathit{supp}(\mu) \Longrightarrow \sigma \vDash \mathbf{q} \\
[\![\mathbb{E}[e]]\!]_\mu \triangleq \mathbb{E}_{\sigma\sim\mu}[[\![e]\!]_\sigma] & \mu \vDash \xi_1 = \xi_2 \ \text{ iff } \ [\![\xi_1]\!]_\mu = [\![\xi_2]\!]_\mu \\
[\![\Pr[\mathbf{q}]]\!]_\mu \triangleq \Pr_{\sigma\sim\mu}[\sigma \vDash \mathbf{q}] & \mu\{X \rightsquigarrow v\} \triangleq \mathbb{E}_{\sigma\sim\mu}\{\delta(\sigma\{X \rightsquigarrow v\})\} \\
[\![\xi_1 + \xi_2]\!]_\mu \triangleq [\![\xi_1]\!]_\mu + [\![\xi_2]\!]_\mu & \mu \vDash \exists X.\, Q \ \text{ iff } \ \exists v.\ \mu\{X \rightsquigarrow v\} \vDash Q
\end{array}$$

Fig. 7. Assertions over states and state distributions.

5 Proof Recipes

Our ultimate proof goals are formulated as total correctness Hoare triples $\vDash [P]C[Q]$ using the distribution-based semantics of Sec. 4.1.

Before showing the definition of $\vDash [P]C[Q]$, we first define assertions in Fig. 7, following the assertion language in [3]. We write $\mathbf{p}, \mathbf{q}, \mathbf{r}$ for non-probabilistic assertions on program states, and P, Q, R for probabilistic assertions on state distributions. The assertion $\lceil \mathbf{q} \rceil$ holds on the distribution μ iff $\mathbf{q}$ holds on all states in the support of μ. We write **true** as a shorthand for $\lceil \text{true} \rceil$. The expression $\Pr[\mathbf{q}]$ represents the probability that $\mathbf{q}$ holds, and $\mathbb{E}[e]$ represents the expected value of e. The assertion $\exists X.\, Q$ holds on μ, if Q holds on μ' obtained by assigning some constant v to X in all states in μ (here δ gives the Dirac distribution). We give a detailed definition of this assertion language in App. B.1.

Then, $\vDash [P]C[Q]$ says that, starting from a state distribution satisfying P, C's execution terminates with probability 1, and thus the sub-distribution of the result states is actually a state distribution, which satisfies Q. We show the definition in Def. 1.

Definition 1 (Total Correctness). *For all P, C, Q, $\vDash [P]C[Q]$ holds iff*

$$\forall \mu.\ \ \mu \vDash P \implies |[\![C]\!](\mu)| = 1 \wedge [\![C]\!](\mu) \vDash Q.$$

In the following subsections, we formalize our two proof recipes, loop truncation and RT-based coupling.

5.1 Loop Truncation

We have explained a specialized form of loop truncation in Lem. 1 in Sec. 2.3. Below we show the more general theorem (Thm. 2). We give the proof in App. C.

Theorem 2 (Loop Truncation). *For all $P, b, C, \mathbf{E}, Q, e$ and r, if*

$$\forall K \in \mathbb{N}.\ \ \vDash [P]\ \mathbf{E}[\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]\ [Q \wedge \mathbb{E}[e] \le r \wedge \lceil e \ge 0 \rceil]\,,$$

$\mathbf{modbf}(\mathbf{E}, e)$ *and* t-$\mathbf{closed}(Q)$*, then* $\vDash [P]\, \mathbf{E}[\mathbf{while}\ (b)\ \mathbf{do}\ C]\, [Q]$.

Here $\mathbf{E}$ is a program context, and $\mathbf{E}[\mathbf{while}\ (b)\ \mathbf{do}\ C]$ fills the hole in $\mathbf{E}$ with the loop $\mathbf{while}\ (b)\ \mathbf{do}\ C$.

$$\begin{array}{rl}(\mathit{Ctx})\ \ \mathbf{E}\ ::= & [\,]\ \mid\ C;\mathbf{E}\ \mid\ \mathbf{E};C\ \mid\ \mathbf{while}\ (b)\ \mathbf{do}\ \mathbf{E}\\ & \mid\ \mathbf{if}\ (b)\ \mathbf{then}\ C\ \mathbf{else}\ \mathbf{E}\ \mid\ \mathbf{if}\ (b)\ \mathbf{then}\ \mathbf{E}\ \mathbf{else}\ C\end{array}$$

Thm. 2 says that, to prove total correctness of $\mathbf{E}[\mathbf{while}\ (b)\ \mathbf{do}\ C]$, we transform the code to $\mathbf{E}[\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]$ with a specific e. How to choose e is application-dependent. Usually we choose as e the loop counter incremented in the loop body, such as cnt in $C_{\mathsf{MT}}(\mathit{cnt})$ (see Sec. 2.2 and Fig. 10). With an inappropriate e, the first premise of the theorem may be invalid or still hard to prove, though how e is chosen does not affect the validity of the theorem.

In addition to e, the first premise also asks users to find a common bound r (a real number) that can bound $\mathbb{E}[e]$ at the end of $\mathbf{E}[\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]$ for all K. Usually the postcondition Q can help us find such an r. Besides the upper bound r, we require that evaluating e at the end of $\mathbf{E}[\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]$ must result in a *non-negative* real number. These two bounds are crucial for ensuring almost sure termination of $\mathbf{E}[\mathbf{while}\ (b)\ \mathbf{do}\ C]$.

The second premise, $\mathbf{modbf}(\mathbf{E}, e)$, rules out those contexts $\mathbf{E}$ that make $\mathbb{E}[e] \leq r$ hold at the end of $\mathbf{E}[\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]$ vacuously, e.g. those that modify the program variables in e at the end of the context and make $e = r$ hold. $\mathbf{modbf}(\mathbf{E}, e)$ syntactically restricts $\mathbf{E}$ such that the variables in e can be modified in $\mathbf{E}$ only before the code in the hole of $\mathbf{E}$ is executed. For example, $\mathbf{modbf}(C'; [\,], e)$ holds for any C' and e, since only C', which is executed before the hole, can modify the variables in e in the context. Similarly, $\mathbf{modbf}([\,], e)$ holds. We give the definition of $\mathbf{modbf}(\mathbf{E}, e)$ in App. C.

The third premise, t-$\mathbf{closed}(Q)$, is for deriving the postcondition Q of $\mathbf{E}[\mathbf{while}\ (b)\ \mathbf{do}\ C]$ from the same Q of $\mathbf{E}[\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]$. We say an assertion Q is t-closed [5], denoted by t-$\mathbf{closed}(Q)$, if for all infinite state distribution sequences $\vec{\mu}$, if Q holds on $\vec{\mu}[i]$ for each i and $\lim \vec{\mu} = \mu$, then Q holds on μ. Many assertions are t-closed. For example, we can prove that t-$\mathbf{closed}(\mathbb{E}[e] \leq r \wedge \lceil e \geq 0 \rceil)$ always holds for any e and any r.

Since $\mathbf{modbf}([\,], e)$ and t-$\mathbf{closed}(\mathbb{E}[e] \leq r \wedge \lceil e \geq 0 \rceil)$ both hold, Lem. 1 can be derived from Thm. 2.

Proof Sketch of Thm. 2. Due to the space limit, below we only show the case of $\mathbf{E} = [\,]$. We prove 1) almost sure termination and 2) the establishment of the postcondition Q, respectively.

For 1), assuming that $\mathbf{while}\ (b)\ \mathbf{do}\ C$ terminates with probability $p < 1$, we derive a contradiction. From the premise we know $\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C$ almost surely terminates, so it terminates in a state where $e \geq K$ with probability at least $1 - p$. Thus, by the semantics of $\mathbb{E}[e]$ (and since the value of e is non-negative), we know $\mathbb{E}[e] \geq (1-p)K$ holds at the end of $\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C$. Therefore, we can find a sufficiently large K such that $\mathbb{E}[e] \geq (1-p)K > r$, which contradicts the premise.

For 2), the key is proving that, for all $\mu \models P$,

$$[\![\mathbf{while}\ (b)\ \mathbf{do}\ C]\!](\mu) = \lim_{K \to \infty} [\![\mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C]\!](\mu).$$

Then, we can establish Q for **while** (b) **do** C, from t-**closed**(Q) and that Q is the postcondition for each **while** $(b \wedge e < K)$ **do** C. $\square$

We apply Thm. 2 for the verification of the MT algorithm and its variants in Sec. 6. Here we show another example beyond ALLLs, which is taken from [42] (with slight modifications).

Example 1. Let $N = 1$ and $\mathcal{D}[1] = \{(0, \frac{1}{2}), (1, \frac{1}{2})\}$. The code C_{flip} is defined as **while** $(y = 1)$ **do** $\{\ y := \mathsf{Sample}(1);\ cnt := cnt + 1; \}$. We prove:

$$\vDash [\lceil cnt = 0 \wedge y = 1 \rceil]\ C_{\text{flip}}\ [\mathbb{E}[cnt] \le 2] \tag{11}$$

Here C_{flip} repeatedly flips a fair coin by sampling from $\mathcal{D}[1]$, until it gets heads $(y = 0)$. We use cnt to record the number of coin flips. Then our proof goal (11) says that C_{flip} almost surely terminates, and it flips at most twice in expectation.

To prove (11), by Thm. 2 (or Lem. 1), we only need to prove that, for all $K \in \mathbb{N}$, $\vDash [\lceil cnt = 0 \wedge y = 1 \rceil]\ C'_{\text{flip}}(K)\ [\mathbb{E}[cnt] \le 2 \wedge \lceil cnt \ge 0 \rceil]$, where $C'_{\text{flip}}(K)$ is defined as **while** $(y = 1 \wedge cnt < K)$ **do** $\{y := \mathsf{Sample}(1);\ cnt := cnt + 1; \}$. We adapt the program logic Ellora [5] to complete the proof. We give the proof in App. H.1.

5.2 Resampling-Table-Based Coupling

As informally explained in Sec. 2.4, our RT-based coupling is for proving the relational tuple $\vDash \{P\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\}$, an intermediate proof goal that appears in ALLLs' verification. We show the formal definition of $\vDash \{P\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\}$ in Def. 2. Note that in this definition we neither require nor assume the termination of C_1 and C_2's executions.

Definition 2 (Inequality between Probabilities). *For all* $P, C_1, C_2, \mathbf{q}_1, \mathbf{q}_2$, $\vDash \{P\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\}$ *holds iff*

$$\forall \mu.\ \ \mu \vDash P \implies \Pr_{\sigma \sim [\![C_1]\!](\mu)}[\sigma \vDash \mathbf{q}_1] \le \Pr_{\sigma \sim [\![C_2]\!](\mu)}[\sigma \vDash \mathbf{q}_2].$$

Our RT-based coupling reduces the verification of the relational tuple to proving unary properties of C_1 and C_2's executions in the RT-based semantics respectively (i.e. the subgoals (9) and (10) in Sec. 2.4). We show the formal theorem in Thm. 3. We give the proof of Thm. 3 in App. D.

Theorem 3 (RT-Based Coupling). *For all* $\mathbf{p}, C_1, C_2, \mathbf{q}_1, \mathbf{R}, \mathbf{q}_2$, *if*

- $\mathbf{RTonly}(\mathbf{R})$;
- $\vDash_{\text{RT}} \{\mathbf{p} \wedge \mathsf{hdinit}\}C_1\{\mathbf{q}_1 \Rightarrow \mathbf{R}\}$;
- $\vDash_{\text{RT}} [\mathbf{p} \wedge \mathbf{R} \wedge \mathsf{hdinit}]C_2[\mathbf{q}_2]$;

then $\vDash \{\lceil \mathbf{p} \rceil\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\}$.

We apply Thm. 3 for verifying ALLLs, which we will explain in Sec. 6. Below we explain Thm. 3 in four aspects: (1) requiring $\lceil \mathbf{p} \rceil$ as the precondition in the relational tuple; (2) the assertions $\mathbf{R}$, hdinit and the requirement $\mathbf{RTonly}(\mathbf{R})$; (3) the RT-based unary triples $\vDash_{\text{RT}}$; and (4) its proof ideas. We also show another example beyond ALLLs, and briefly discuss an extension of Thm. 3 at the end.

$$
\begin{array}{llll}
(\mathit{RTExpr}) & E & ::= & e \mid \mathsf{RT}[E_1][E_2] \mid \mathsf{hd}_1 \mid \ldots \mid \mathsf{hd}_N \mid E_1 + E_2 \mid \ldots \\
(\mathit{RTBexp}) & B & ::= & b \mid E_1 = E_2 \mid E_1 < E_2 \mid \ldots \\
(\mathit{RTAssn}) & \mathbf{P}, \mathbf{Q}, \mathbf{R} & ::= & \mathbf{q} \mid B \mid \neg \mathbf{Q} \mid \mathbf{Q}_1 \wedge \mathbf{Q}_2 \mid \mathbf{Q}_1 \vee \mathbf{Q}_2 \mid \forall X.\, \mathbf{Q} \mid \exists X.\, \mathbf{Q} \mid \ldots
\end{array}
$$

$$
(\sigma, RT, \iota) \models \mathbf{q} \ \text{ iff } \ \sigma \models \mathbf{q} \qquad\qquad [\![\mathsf{hd}_n]\!]_{(\sigma,RT,\iota)} \triangleq \iota[n]
$$

$$
[\![\mathsf{RT}[E_1][E_2]]\!]_{(\sigma,RT,\iota)} \triangleq RT[i][j], \ \text{if } [\![E_1]\!]_{(\sigma,RT,\iota)} = i, [\![E_2]\!]_{(\sigma,RT,\iota)} = j
$$

$$
\mathsf{hdinit} \ \triangleq \ \textstyle\bigwedge_{i \in [1,N]} . \ \mathsf{hd}_i = 0
$$

$$
\mathbf{RTonly}(\mathbf{R}) \ \text{ iff } \ \forall \sigma, RT, \iota.\ (\sigma, RT, \iota) \models \mathbf{R} \implies \forall \sigma', \iota'.\ (\sigma', RT, \iota') \models \mathbf{R}
$$

Fig. 8. Non-probabilistic assertions on RT-extended states.

Lifting state assertions as preconditions. The relational tuples we prove are in a restricted form, namely that the precondition P is in the form of $\lceil \mathbf{p} \rceil$, where $\mathbf{p}$ is an assertion over states. Recall that $\lceil \mathbf{p} \rceil$ holds over μ iff $\mathbf{p}$ holds over any σ such that $\sigma \in \mathit{supp}(\mu)$ (see Fig. 7). Therefore the precondition $\lceil \mathbf{p} \rceil$ says we are only interested in the executions of C_1 and C_2 with the initial states satisfying $\mathbf{p}$. So we can fill the omitted part of the two subgoals (9) and (10) with $\mathbf{p}$, and turn them into classical (*deterministic*) Hoare triples $\models_{\mathrm{RT}} \{\mathbf{p}\}\, C_1\, \{\mathbf{q}_1 \Rightarrow \mathbf{R}\}$ and $\models_{\mathrm{RT}} [\mathbf{p} \wedge \mathbf{R}]\, C_2\, [\mathbf{q}_2]$.

Assertions over RT-extended states. Thm. 3 requires us to find an "intermediate assertion" $\mathbf{R}$ that describes (and *only* describes) the (non-probabilistic) properties of the resampling table RT. Since we need explicit reasoning about RT, the assertions used in the classical reasoning of $\models_{\mathrm{RT}}$ actually specify RT and the heads ι as well as the states σ.

In Fig. 8, we define non-probabilistic assertions $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ over the extended states (σ, RT, ι). Besides using $\mathbf{q}$ to describe σ in the extended states, we introduce RT-expressions to specify RT and ι. We use $\mathsf{RT}[E_1][E_2]$ to represent the entry at row E_1 and column E_2 of RT, and use hd_n to represent the n-th head $\iota[n]$, where $n \in [1, N]$.

The assertion hdinit (defined as a shorthand in Fig. 8) says that all of the heads ι point to the first column of RT. It specifies the initial heads before program execution, so it appears in the preconditions of the two $\models_{\mathrm{RT}}$ triples in Thm. 3.

The requirement $\mathbf{RTonly}(\mathbf{R})$ (defined in Fig. 8) says that changing σ and/or ι in the extended state does not affect whether $\mathbf{R}$ holds. That is, $\mathbf{R}$ describes RT only. One can check that $\mathbf{RTonly}(\mathbf{R})$ holds if $\mathbf{R}$ does not syntactically contain any free variables and hd_n's.

We give a detailed definition of this assertion language in App. B.2.

RT-based unary triples. Now we can define the RT-based unary triples, $\models_{\mathrm{RT}}$ $[\mathbf{P}]C[\mathbf{Q}]$ and $\models_{\mathrm{RT}}$ $\{\mathbf{P}\}C\{\mathbf{Q}\}$. They are standard Hoare triples for total correctness and partial correctness respectively, using the RT-based operational semantics (in Fig. 6 of Sec. 4.2) for program execution.

Definition 3 (Total Correctness in RT-Based Operational Semantics). *For all* $\mathbf{P}, C, \mathbf{Q}$*,* $\models_{\mathrm{RT}} [\mathbf{P}]C[\mathbf{Q}]$ *holds iff*

$$\begin{aligned}\forall \sigma, RT, \iota.\ \ &(\sigma, RT, \iota) \models \mathbf{P} \implies \\ &\exists \sigma', \iota'.\ RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota') \wedge (\sigma', RT, \iota') \models \mathbf{Q}.\end{aligned}$$

Definition 4 (Partial Correctness in RT-Based Operational Semantics). *For all* $\mathbf{P}, C, \mathbf{Q}$*,* $\models_{\mathrm{RT}} \{\mathbf{P}\}C\{\mathbf{Q}\}$ *holds iff*

$$\begin{aligned}\forall \sigma, RT, \iota, \sigma', \iota'.\ \ &(\sigma, RT, \iota) \models \mathbf{P} \wedge RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota') \\ &\implies (\sigma', RT, \iota') \models \mathbf{Q}.\end{aligned}$$

For total correctness, Def. 3 says there exists a terminating execution of (C, σ, ι) under RT. This essentially ensures the absence of non-terminating executions, because the RT-based operational semantics is *deterministic*.

We can use a classical Hoare-style program logic to prove the $\models_{\mathrm{RT}}$ triples. We give such a logic in App. F.

Proof ideas of the theorem. To prove Thm. 3, we need to bridge two gaps between the $\models_{\mathrm{RT}}$ triples in the premises and the $\models$ tuple in the conclusion. First, the $\models_{\mathrm{RT}}$ triples use the RT-based semantics, while the $\models$ tuple uses the distribution-based semantics. Second, the $\models_{\mathrm{RT}}$ triples are unary, while the $\models$ tuple is relational.

The key to bridging the gaps is reduction through the following RT-based tuple as an intermediate form, which is the counterpart of Def. 2 in the RT-based semantics.

Definition 5 (Inequality between Pr. in RT-Based Semantics). *For all* $P, C_1, C_2, \mathbf{q}_1, \mathbf{q}_2$*,* $\models_{\mathrm{RT}} \{P\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\}$ *holds iff*

$$\forall \mu.\ \ \mu \models P \implies \Pr_{\sigma \sim [\![C_1]\!]_{\mathrm{RT}}(\mu)}[\sigma \models \mathbf{q}_1] \le \Pr_{\sigma \sim [\![C_2]\!]_{\mathrm{RT}}(\mu)}[\sigma \models \mathbf{q}_2].$$

Lemma 3 shows the equivalence between the two relational tuples, which follows from the semantics equivalence (Thm. 1). This lemma bridges the first gap, and is interesting in its own right.

Lemma 3. *For all* $P, C_1, C_2, \mathbf{q}_1, \mathbf{q}_2$*,*

$$\models \{P\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\} \iff \models_{\mathrm{RT}} \{P\}C_1 \le C_2\{\mathbf{q}_1, \mathbf{q}_2\}.$$

Our "intermediate assertion" $\mathbf{R}$ allows us to split the $\models_{\mathrm{RT}}$ relational tuple into two unary $\models_{\mathrm{RT}}$ triples, bridging the second gap.

Example 2. This example is adapted from an intermediate goal in [8]'s proof of the PRP/PRF switching lemma[4]. Let $k \ge 1$. For any $n_1, \ldots, n_k$, we prove that

$$\begin{aligned}&\models \{\lceil \mathsf{inp} \rceil\}\, C_{\mathrm{PRF}}^{\mathsf{bad}} \le C_{\mathrm{PRF}} \\ &\quad \{bad = 1,\ \ \exists X_1, X_2, Y.\ X_1 \ne X_2 \wedge \mathsf{find}(L, (X_1, Y)) \wedge \mathsf{find}(L, (X_2, Y))\}.\end{aligned} \tag{12}$$

[4] In [8], $C_{\mathrm{PRF}}^{\mathsf{bad}}$ and C_{PRF} are defined using procedure calls. We adapt the code here.

```
1   $L := []$; $d := 1$;
2   $\mathit{bad} := 0$;
3   while $(d \leq k)$ do
4       if $(\neg\mathsf{findkey}(L, x[d]))$ then
5           $y := \mathsf{Sample}(1)$;
6           if $(\mathsf{findval}(L, y))$ then $\mathit{bad} := 1$;
7           $L := \mathsf{app}(L, (x[d], y))$;
8       $d := d + 1$
```

Fig. 9. The code $C^{\mathsf{bad}}_{\mathrm{PRF}}$ in Ex. 2.

We show the code of $C^{\mathsf{bad}}_{\mathrm{PRF}}$ in Fig. 9, and the code of C_{PRF} results from removing lines 2 and 6 from the figure. The assertion inp says that $n_1, \ldots, n_k$ are the inputs stored in $x[1], \ldots, x[k]$, which is defined as $\bigwedge_{i\in[1,k]} . \, x[i] = n_i$.

By extending the programming language, we implement a map in the program variable L, which stores some key-value pairs. One can insert a pair into the map by writing $\mathsf{app}(L, (e_1, e_2))$, and query for the existence of a key, a value or a pair by writing $\mathsf{findkey}(L, e)$, $\mathsf{findval}(L, e)$ or $\mathsf{find}(L, (e_1, e_2))$.

$C^{\mathsf{bad}}_{\mathrm{PRF}}$ and C_{PRF} do the following: for $n = x[1], \ldots, x[k]$, the programs check if n has been inserted in L as a key; if not, they sample a value y from $\mathcal{D}[1]$, and then insert the key-value pair (n, y) into L; if y has been inserted in L as a value, $C^{\mathsf{bad}}_{\mathrm{PRF}}$ marks bad.

(12) then says that, the probability of $C^{\mathsf{bad}}_{\mathrm{PRF}}$ terminating with $\mathit{bad} = 1$ is no more than the probability of C_{PRF} terminating with two key-value pairs with the same value left in L.

To prove (12), we apply Thm. 3. We take $\mathbf{R} = \mathsf{coll}$, where

$$\mathsf{coll} \triangleq \bigvee_{0\leq i<j<|\{n_1,\ldots,n_k\}|} . \, \mathsf{RT}[1][i] = \mathsf{RT}[1][j].$$

coll says that, there exist two identical entries in the first row of RT, which are picked as samples in the executions of both $C^{\mathsf{bad}}_{\mathrm{PRF}}$ and C_{PRF}. Therefore coll specifies the kind of RT that can make $\mathit{bad} = 1$ hold after the execution of $C^{\mathsf{bad}}_{\mathrm{PRF}}$.

We can check that $\mathbf{RTonly}(\mathsf{coll})$ holds. Then, by applying Thm. 3, it remains to prove the following two unary $\vDash_{\mathrm{RT}}$ triples:

$$\vDash_{\mathrm{RT}} \{\mathsf{inp} \wedge \mathsf{hdinit}\}\, C^{\mathsf{bad}}_{\mathrm{PRF}}\, \{\mathit{bad} = 1 \Rightarrow \mathbf{R}\}$$

$$\vDash_{\mathrm{RT}} [\mathsf{inp} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathrm{PRF}}\, [\exists X_1, X_2, Y. \; X_1 \neq X_2 \wedge \mathsf{find}(L, (X_1, Y)) \wedge \mathsf{find}(L, (X_2, Y))]$$

We prove them using a simple Hoare-style program logic. We give the full proof of this example in App. H.2.

An extension of RT-based coupling. In Thm. 5 in App. D, we give another relational proof recipe that extends Thm. 3. It asks users to provide two intermediate assertions $\mathbf{R}_1$ and $\mathbf{R}_2$ for splitting the $\vDash_{\mathrm{RT}}$ relational tuple, and provides more flexibility for reasoning about inequalities between probabilities.

6 Case Studies

We show the usefulness of our proof recipes (Thm. 2 and Thm. 3) by verifying several representative existing results about ALLLs and a new result about the

MT algorithm. Below we first give a brief survey of several important research lines on ALLLs. Then we summarize the existing ALLL-related results that we have verified, and show how we verify Theorem 1.2 of [51] as an example. Finally, we explain our new result about the MT algorithm.

Research lines of ALLLs. The MT algorithm is first proposed in [51], where the expected iteration number of the algorithm is bounded under the Erdős-Lovász condition [21, 58] and the Erdős-Spencer condition [22]. Following [51], some works [54, 44, 32, 1, 43, 38] further analyze the termination property and the iteration times of the MT algorithm under other conditions. Besides analyzing the iteration times of the MT algorithm, a number of works (including [51]) also analyze other sequential ALLLs [32, 35, 37, 30], explore properties of output distributions of ALLLs [32, 36, 30, 33], or design parallel and distributed ALLLs [51, 18, 31, 26, 15]. However, the proofs in all these works are relatively informal.

Existing results we verify. As listed below, we verify *six* representative results that cover the aforementioned research lines.

First, we verify the termination and the expected iteration times of the MT algorithm, under the Erdős-Lovász condition [21, 58], the cluster expansion condition [10], the Shearer's condition [57], and the Erdős-Spencer condition [22]. These four results are proposed and informally proved in Theorem 1.2 of [51], Theorem 1.4 of [54], Theorem 4 of [44] and Theorem 6.1 of [51].

Second, we verify (the second part of) Theorem 2.2 of [32] that estimates the output distribution of the MT algorithm under the Erdős-Lovász condition. This result can also be viewed as estimating the output distribution of a sequential ALLL that only executes on core events (see Theorem 3.3 of [32]).

Finally, we verify the termination and a tail bound of the iteration times of a parallelizable version of the MT algorithm, under the Erdős-Lovász condition with ϵ-slack. This variant and the tail bound are given in Theorem 1.3 of [51].

It is worth noting that we verify all the three "probabilistic" results from Moser and Tardos's Gödel Prize-winning paper [51][5].

We give the statements and formal proofs of these six results in App. J.

Verifying Theorem 1.2 of [51]. As an example, we explain in more detail how we verify Theorem 1.2 of [51], which we informally described in Sec. 2.

Fig. 10 shows $C_{\mathsf{MT}}(cnt)$, the code of the MT algorithm that we verify. It first does independent samplings and stores the results in $x[1], \ldots, x[N]$ (line 1), where d and a are temporal variables. For the main loop (lines 3-13), we introduce *flag* to indicate whether a required assignment is found, *cnt* to record the number of iteration times, and *lst* to collect the indexes of the events in the execution log. They are initialized at line 2. In the main loop (lines 3-13), we use z to represent the index of the chosen event, which is an event that holds under the current $x[1], \ldots, x[N]$ (lines 4-7). If no such event exists, the code marks *flag*

[5] In [51], Moser and Tardos propose four results, three related to the MT algorithm and its probabilistic variants, and one related to a deterministic variant.

```
1   $d := 1;$ while $(d \le N)$ do $\{a := \mathsf{Sample}(d); x[d] := a; d := d+1;\}$
2   $\mathit{flag} := 0;\ \mathit{cnt} := 0;\ \mathit{lst} := [];$
3   while $(\mathit{flag} = 0)$ do
4       $z := 0;\ h := 1;$
5       while $(h \le M)$ do
6           if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then $z := h;$
7           $h := h + 1;$
8       if $(z = 0)$ then $\mathit{flag} := 1;$
9       else
10          $\mathit{cnt} := \mathit{cnt} + 1;\ \mathit{lst} := \mathsf{app}(\mathit{lst}, z);\ d := 1;$
11          while $(d \le N)$ do
12              if $(\mathsf{vbl}(z, d))$ then $\{a := \mathsf{Sample}(d);\ x[d] := a;\}$
13              $d := d + 1;$
```

Fig. 10. The code of the MT algorithm, $C_{\mathsf{MT}}(\mathit{cnt})$.

(line 8) and exits the loop (line 3). Otherwise, it resamples from $\mathcal{D}[d]$ for every d such that $\mathsf{vbl}(z, d)$ holds, and updates the corresponding $x[d]$ (lines 10-13).

Having defined the code of the MT algorithm, Moser and Tardos's result (Theorem 1.2 of [51]) is formally stated in Thm. 4. Note that $N, M, \mathcal{D}$ and $\mathcal{E}$ are global parameters and thus not fixed in Thm. 4, and r_{EL} is parametrized by M.

Theorem 4. *For all reals* $\alpha_1, \ldots, \alpha_M \in (0, 1)$*, if the Erdős-Lovász condition [21, 58] holds, i.e.* $\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \le \alpha_i \prod_{j \in \Gamma(i)} (1 - \alpha_j)$*, and let* $r_{\mathsf{EL}} = \sum_{i \in [1,M]} \alpha_i (1 - \alpha_i)^{-1}$*, then* $\models [\mathbf{true}]\, C_{\mathsf{MT}}(\mathit{cnt})\, [\mathbb{E}[\mathit{cnt}] \le r_{\mathsf{EL}}]$.

Proof Sketch. Our proof follows the path in Fig. 3. Due to the space limit, here we only explain our construction of $\mathbf{R}$, used in the two RT-triples at the bottom of Fig. 3. Let $\Lambda = g_{\mathsf{WT}}(wt)$. Then,

$$\mathbf{R} \triangleq \forall l \in [1, |\Lambda|].\ \forall V_1, \ldots V_N.\ \mathsf{RTAssign}(V_1, \ldots V_N, l, \Lambda) \Rightarrow \mathsf{hold}(\Lambda\langle l \rangle, V_1, \ldots, V_N)$$

$$\text{where } \mathsf{RTAssign}(V_1, \ldots, V_N, l, \Lambda) \triangleq \forall i \in [1, N].\ \mathsf{vbl}(\Lambda\langle l \rangle, i) \Rightarrow V_i = \mathsf{RT}[i][\mathsf{ve}(i, \Lambda, l-1)]$$

$$\mathsf{ve}(i, \Lambda, l) \triangleq \textstyle\sum_{l' \in [1,l]} [\mathsf{vbl}(\Lambda\langle l' \rangle, i)]$$

Informally $\mathbf{R}$ says that, every event in wt (denoted by $\Lambda\langle l \rangle$) must hold under any assignment of $V_1, \ldots, V_N$ satisfying $\mathsf{RTAssign}$. $\mathsf{RTAssign}$ says, the assignment contains the "relevant" entries of RT which make the event $\Lambda\langle l \rangle$ hold when it is chosen in the execution of $C'_{\mathsf{MT}}(\mathit{cnt}, K)$. For each such entry, its row number i corresponds to a variable that the event depends on (i.e. $\mathsf{vbl}(\Lambda\langle l \rangle, i)$ holds), and its column number is computed by $\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l - 1)$. Note our $\mathbf{R}$ only talks about the RT (and the wt), not about the actual execution of $C'_{\mathsf{MT}}(\mathit{cnt}, K)$.

We prove the remaining intermediate proof goals in Fig. 3 by adapting the program logic Ellora [5] (for proving $\models$ triples) and using a classical Hoare-style logic (for proving $\models_{\mathrm{RT}}$ triples). $\square$

Our new result. Thm. 4 shows the MT algorithm's total correctness with r_{EL} as the upper bound of expected iteration times, under the Erdős-Lovász condition. There are many works [54, 44, 1, 43, 38] that informally study similar properties of the MT algorithm under other conditions. Most of these results use similar ideas with Moser and Tardos to analyze the algorithm, except that they introduce other witness-tree-like structures for analysis and derive various bounds. Like [51], they generate their witness-tree-like structures ds from prefixes of the execution log, enumerate the events in ds in some specific order, and bound a sum over all such structures to get their final upper bounds.

We unify these results to a general one. Our new result enables that, when proving the expected iteration number of the MT algorithm, without doing the complete proof following Moser and Tardos's idea, one only needs to instantiate the required witness-tree-like structures and prove some relevant mathematical side conditions.

Specifically, we ask users to provide: (a) the set DS of all instances of such witness-tree-like structures, (b) a function f for generating a structure $ds \in DS$ from a *prefix of the execution log* Λ, (c) a function g for specifying the enumeration order of ds, and (d) a function $DSMap$ mapping each natural number K to a subset of DS, such that each $ds \in DSMap(K)$ has "size" no more than K.

The function f, like f_{WT} used in Sec. 2.1, maps prefixes of the execution log (an event sequence) Λ of the MT algorithm to the witness-tree-like structures ds, and feeds them to the "check" program (see Sec. 2.1). In check(ds), the function g, like g_{WT} used in check(wt), decides the enumeration order of ds and maps it back to another event sequence Λ'. As an important side condition over f and g, we require that, for any $\Lambda' = g(f(\Lambda))$, the event ordering in Λ and Λ' satisfy certain crucial constraints. The enumeration generated by g determines the order of samplings in check(ds), which would further affect the coupling reasoning that relates the samplings in the MT algorithm and check(ds).

We show that Theorem 1.2 of [51], Theorem 1.4 of [54] and Theorem 4 of [44] are corollaries of our new result. When proving Theorem 1.2 of [51] as a corollary of our new result, we simply take DS as the set of all witness trees, $f = f_{\mathsf{WT}}$, and $g = g_{\mathsf{WT}}$. We give details of our new result and proofs in App. J.2.

7 Related Work

(Positive) almost sure termination. Existing proof methods for almost sure termination (AST) can be roughly classified into the following two categories: "direct" methods [49, 13, 14, 25, 50, 39, 48], which prove termination by constructing probabilistic ranking functions, and "indirect" methods [42, 53, 52, 41], which infer finite bounds on the expected runtime and then imply the termination.

However, these methods may not apply to ALLLs' termination. To construct the structures (e.g. ranking supermatingales [14, 25] and upper ω-invariants [42]) required by these methods, we need to understand what occurs during *each iteration* of the algorithm's outer loop, which is, however, not yet well understood. For example, [51] only analyzes the properties of the *entire* MT algorithm (e.g. (2)), not of each individual iteration.

In Sec. 2.3, we emphasize Lem. 1 as a general proof method for positive almost sure termination (PAST) [13]. Lem. 1 also serves as a *fallback plan* for proving

(P-)AST. Informally, a part of existing methods [14, 25, 50, 42] provide stronger premises than Lem. 1's. These premises are easier to prove in most scenarios, except for ALLLs. For most programs, one can still apply these existing methods; for programs like ALLLs, one should take a step back and apply Lem. 1.

Asynchronous coupling. In Sec. 2.4, we apply the RT-based coupling proof recipe to (8), which involves $C'_{\mathsf{MT}}(cnt, K)$ and check(wt). Existing probabilistic relational program logics [6, 7, 8] support couplings, but none of them can prove (8). Specifically, these works only provide proof rules for *synchronous* couplings. Their rules say that, when the two programs sample from the same distribution synchronously, we can reason as if the two sampling statements return the same value. But, it may *not* be possible to synchronize the sampling statements in $C'_{\mathsf{MT}}(cnt, K)$ and check(wt) for the following reason. Given an execution log's prefix Λ and the corresponding witness tree $wt = f_{\mathsf{WT}}(\Lambda)$, $C'_{\mathsf{MT}}(cnt, K)$ resamples the variables that η_j depends on for every event η_j in Λ, and check(wt) does similar resamplings but its events are taken from the sequence $g_{\mathsf{WT}}(wt)$. However, $g_{\mathsf{WT}}(wt)$ can be different from Λ, since the construction of wt (i.e. $f_{\mathsf{WT}}(\Lambda)$) may drop some events in Λ and lose some ordering information of Λ, which $g_{\mathsf{WT}}(wt)$ cannot recover.

Recently [29] proposes a probabilistic relational program logic that supports *asynchronous* coupling. They introduce *presampling tapes*, a new kind of ghost state, which store the sampling results ahead of time. Our work is developed independently, with a more focused goal of verifying ALLLs. Technically, our RTs look similar to their tapes, but there are several key differences as follows.

First, we give an RT-based operational semantics, where all the samples (which could be infinitely many) are generated at once and stored in the *RT before programs start execution*, and the *RT* is immutable during the program execution. By contrast, sample values are added into their tapes *one at a time* and *on demand* by ghost operations in the logical reasoning, and are popped out at sampling statements. We think their approach is more flexible, but ours is more suitable for complicated examples like ALLLs. In particular, as we explain at the end of Sec. 2.4, we can use an intermediate assertion $\mathbf{R}$ to specify *the whole sampling history.* $\mathbf{R}$ can be derived as the post-condition of the unary reasoning of one program, and then used as the pre-condition of the other, thanks to the immutability of *RT*. With dynamically changing tapes, they would need ghost variables to track the popped samples, and write complicated assertions to describe the correspondence between the tapes used by the two programs. We give a more detailed comparison in App. K.

Second, the two works have different focuses. We mainly focus on verifying ALLLs, so we verify almost sure termination as well as a restricted form of relational properties (like (8)). Their work verifies contextual refinement, but does not verify termination.

Finally, in addition to the above differences in the formal verification techniques, verifying ALLLs itself is a challenging task, and is one of the major contributions of this work.

Other related works. [23] proposes the *guard strengthening* proof rule for verifying lower bounds of expected values at the end of while loops. This rule introduces a loop with strengthened loop guard, which is similar to the truncated one in the premise of our loop truncation (Lem. 1 and Thm. 2). However, these two methods have different focuses. Their rule focuses on proving *lower bounds*, while our loop truncation focuses on proving general total correctness and PAST. The PAST is about an *upper bound* of the expected runtime.

We have discussed other related works in Sec. 2.2 and Sec. 2.4, including: the semantics that are equivalent to the distribution-based semantics [46, 49, 45], and the semantics with explicit random sources [45, 12, 19]. In the future, we would like to test our proof recipes with more applications, such as the other ALLL-related results mentioned in Sec. 6. We also plan to mechanize our work in a proof assistant, as [20] have mechanized the classical (i.e. non-constructive) proof of the Lovász Local Lemma in Isabelle/HOL.

References

1. Achlioptas, D., Gouleakis, T.: Algorithmic Improvements of the Lovász Local Lemma via Cluster Expansion. In: D'Souza, D., Radhakrishnan, J., Telikepalli, K. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012). Leibniz International Proceedings in Informatics (LIPIcs), vol. 18, pp. 16–23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2012). https://doi.org/10.4230/LIPIcs.FSTTCS.2012.16, https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.FSTTCS.2012.16
2. Anderson, E., Phillips, C., Sicker, D., Grunwald, D.: Optimization decomposition for scheduling and system configuration in wireless networks. IEEE/ACM Transactions on Networking **22**(1), 271–284 (2014). https://doi.org/10.1109/TNET.2013.2289980
3. Anonymous Author(s): A program logic for concurrent randomized programs in the oblivious adversary model. Submitted to ESOP 2025 (under review)
4. Bansal, N., Sviridenko, M.: The santa claus problem. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing. p. 31–40. STOC '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1132516.1132522, https://doi.org/10.1145/1132516.1132522
5. Barthe, G., Espitau, T., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.Y.: An assertion-based program logic for probabilistic programs. In: Ahmed, A. (ed.) Programming Languages and Systems. pp. 117–144. Springer International Publishing, Cham (2018)
6. Barthe, G., Espitau, T., Grégoire, B., Hsu, J., Stefanesco, L., Strub, P.Y.: Relational reasoning via probabilistic coupling. In: Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning - Volume 9450. p. 387–401. LPAR-20 2015, Springer-Verlag, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48899-7_27, https://doi.org/10.1007/978-3-662-48899-7_27
7. Barthe, G., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.Y.: Proving differential privacy via probabilistic couplings. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. p. 749–758. LICS '16, Association

for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2933575.2934554, https://doi.org/10.1145/2933575.2934554

8. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 90–101. POPL '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1480881.1480894, https://doi.org/10.1145/1480881.1480894
9. Batz, K., Kaminski, B.L., Katoen, J.P., Matheja, C.: Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. Proc. ACM Program. Lang. **5**(POPL) (jan 2021). https://doi.org/10.1145/3434320, https://doi.org/10.1145/3434320
10. Bissacot, R., Fernández, R., Procacci, A., Scoppola, B.: An improvement of the lovász local lemma via cluster expansion. Combinatorics, Probability and Computing **20**(5), 709–719 (2011)
11. Boissonnat, J.D., Dyer, R., Ghosh, A.: A probabilistic approach to reducing algebraic complexity of delaunay triangulations. In: Bansal, N., Finocchi, I. (eds.) Algorithms - ESA 2015. pp. 595–606. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
12. Borgström, J., Dal Lago, U., Gordon, A.D., Szymczak, M.: A lambda-calculus foundation for universal probabilistic programming. SIGPLAN Not. **51**(9), 33–46 (sep 2016). https://doi.org/10.1145/3022670.2951942, https://doi.org/10.1145/3022670.2951942
13. Bournez, O., Garnier, F.: Proving positive almost-sure termination. In: RTA. Lecture Notes in Computer Science, vol. 3467, pp. 323–337. Springer (2005). https://doi.org/10.1007/978-3-540-32033-3_24, https://doi.org/10.1007/978-3-540-32033-3_24
14. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification. pp. 511–526. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
15. Chang, Y.J., He, Q., Li, W., Pettie, S., Uitto, J.: Distributed edge coloring and a special case of the constructive lovász local lemma. ACM Trans. Algorithms **16**(1) (nov 2019). https://doi.org/10.1145/3365004, https://doi.org/10.1145/3365004
16. Chen, A., Harris, D.G., Srinivasan, A.: Partial resampling to approximate covering integer programs. Random Structures & Algorithms **58**(1), 68–93 (2021). https://doi.org/https://doi.org/10.1002/rsa.20964, https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.20964
17. Cheng, K., Haeupler, B., Li, X., Shahrasbi, A., Wu, K.: Synchronization strings: highly efficient deterministic constructions over small alphabets. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. p. 2185–2204. SODA '19, Society for Industrial and Applied Mathematics, USA (2019)
18. Chung, K.M., Pettie, S., Su, H.H.: Distributed algorithms for the lovász local lemma and graph coloring. In: Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing. p. 134–143. PODC '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2611462.2611465, https://doi.org/10.1145/2611462.2611465
19. Culpepper, R., Cobb, A.: Contextual equivalence for probabilistic programs with continuous random variables and scoring. In: Yang, H. (ed.) Programming Languages and Systems. pp. 368–392. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)

20. Edmonds, C., Paulson, L.C.: Formal probabilistic methods for combinatorial structures using the lovász local lemma. In: Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs. p. 132–146. CPP 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3636501.3636946, https://doi.org/10.1145/3636501.3636946
21. Erdős, P., Lovász, L.: Problems and results on 3-chromatic hypergraphs and some related questions. Infinite and finite sets **10**(2), 609–627 (1975)
22. Erdős, P., Spencer, J.: Lopsided lovsz local lemma and latin transversals. Discrete Applied Mathematics **30**(151-154), 10–1016 (1991)
23. Feng, S., Chen, M., Su, H., Kaminski, B.L., Katoen, J.P., Zhan, N.: Lower bounds for possibly divergent probabilistic programs. Proc. ACM Program. Lang. **7**(OOPSLA1) (apr 2023). https://doi.org/10.1145/3586051, https://doi.org/10.1145/3586051
24. Fernández, M., Livieratos, J., Martín, S.: Bounds and constructions of parent identifying schemes via the algorithmic version of the lovász local lemma. IEEE Transactions on Information Theory **69**(11), 7049–7069 (2023). https://doi.org/10.1109/TIT.2023.3282452
25. Ferrer Fioriti, L.M., Hermanns, H.: Probabilistic termination: Soundness, completeness, and compositionality. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 489–501. POPL '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2676726.2677001, https://doi.org/10.1145/2676726.2677001
26. Fischer, M., Ghaffari, M.: Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy. In: Richa, A. (ed.) 31st International Symposium on Distributed Computing (DISC 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 91, pp. 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2017). https://doi.org/10.4230/LIPIcs.DISC.2017.18, https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2017.18
27. Gebauer, H., Szabó, T., Tardos, G.: The local lemma is asymptotically tight for sat. J. ACM **63**(5) (dec 2016). https://doi.org/10.1145/2975386, https://doi.org/10.1145/2975386
28. Graf, A., Harris, D.G., Haxell, P.: Algorithms for weighted independent transversals and strong colouring. ACM Trans. Algorithms **18**(1) (dec 2021). https://doi.org/10.1145/3474057, https://doi.org/10.1145/3474057
29. Gregersen, S.O., Aguirre, A., Haselwarter, P.G., Tassarotti, J., Birkedal, L.: Asynchronous probabilistic couplings in higher-order separation logic. Proc. ACM Program. Lang. **8**(POPL) (jan 2024). https://doi.org/10.1145/3632868, https://doi.org/10.1145/3632868
30. Guo, H., Jerrum, M., Liu, J.: Uniform sampling through the lovász local lemma. J. ACM **66**(3) (apr 2019). https://doi.org/10.1145/3310131, https://doi.org/10.1145/3310131
31. Haeupler, B., Harris, D.G.: Parallel algorithms and concentration bounds for the lovász local lemma via witness dags. ACM Trans. Algorithms **13**(4) (dec 2017). https://doi.org/10.1145/3147211, https://doi.org/10.1145/3147211
32. Haeupler, B., Saha, B., Srinivasan, A.: New constructive aspects of the lovász local lemma. J. ACM **58**(6), 28:1–28:28 (2011). https://doi.org/10.1145/2049697.2049702, https://doi.org/10.1145/2049697.2049702

33. Harris, D.G.: New bounds for the moser-tardos distribution. Random Structures & Algorithms **57**(1), 97–131 (2020). https://doi.org/https://doi.org/10.1002/rsa.20914, https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.20914
34. Harris, D.G., Srinivasan, A.: Constraint satisfaction, packet routing, and the lovasz local lemma. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing. p. 685–694. STOC '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2488608.2488696, https://doi.org/10.1145/2488608.2488696
35. Harris, D.G., Srinivasan, A.: A constructive algorithm for the lovász local lemma on permutations. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. p. 907–925. SODA '14, Society for Industrial and Applied Mathematics, USA (2014)
36. Harris, D.G., Srinivasan, A.: Algorithmic and enumerative aspects of the moser-tardos distribution. ACM Trans. Algorithms **13**(3) (mar 2017). https://doi.org/10.1145/3039869, https://doi.org/10.1145/3039869
37. Harris, D.G., Srinivasan, A.: The moser–tardos framework with partial resampling. J. ACM **66**(5) (aug 2019). https://doi.org/10.1145/3342222, https://doi.org/10.1145/3342222
38. He, K., Li, Q., Sun, X.: Moser-tardos algorithm: Beyond shearer's bound (2021)
39. Huang, M., Fu, H., Chatterjee, K., Goharshady, A.K.: Modular verification for almost-sure termination of probabilistic programs. Proc. ACM Program. Lang. **3**(OOPSLA), 129:1–129:29 (2019). https://doi.org/10.1145/3360555, https://doi.org/10.1145/3360555
40. Jiang, N., Gu, Y., Xue, Y.: Learning markov random fields for combinatorial structures via sampling through lovász local lemma. Proceedings of the AAAI Conference on Artificial Intelligence **37**(4), 4016–4024 (Jun 2023). https://doi.org/10.1609/aaai.v37i4.25516, https://ojs.aaai.org/index.php/AAAI/article/view/25516
41. Kaminski, B.L.: Advanced weakest precondition calculi for probabilistic programs. Ph.D. thesis, RWTH Aachen University, Germany (2019), http://publications.rwth-aachen.de/record/755408
42. Kaminski, B.L., Katoen, J.P., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected run–times of probabilistic programs. In: Thiemann, P. (ed.) Programming Languages and Systems. pp. 364–389. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
43. Kolipaka, K., Szegedy, M., Xu, Y.: A sharper local lemma with improved applications. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. pp. 603–614. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
44. Kolipaka, K.B.R., Szegedy, M.: Moser and tardos meet lovász. In: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing. p. 235–244. STOC '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/1993636.1993669, https://doi.org/10.1145/1993636.1993669
45. Kozen, D.: Semantics of probabilistic programs. Journal of Computer and System Sciences **22**(3), 328–350 (1981). https://doi.org/https://doi.org/10.1016/0022-0000(81)90036-2, https://www.sciencedirect.com/science/article/pii/0022000081900362
46. Kozen, D.: A probabilistic pdl. Journal of Computer and System Sciences **30**(2), 162–178 (1985). https://doi.org/https://doi.org/10.1016/0022-0000(85)90012-1, https://www.sciencedirect.com/science/article/pii/0022000085900121

47. Luby, M.: A simple parallel algorithm for the maximal independent set problem. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing. p. 1–10. STOC '85, Association for Computing Machinery, New York, NY, USA (1985). https://doi.org/10.1145/22145.22146, https://doi.org/10.1145/22145.22146
48. Majumdar, R., Sathiyanarayana, V.R.: Positive almost-sure termination: Complexity and proof rules. Proc. ACM Program. Lang. **8**(POPL) (jan 2024). https://doi.org/10.1145/3632879, https://doi.org/10.1145/3632879
49. McIver, A., Morgan, C.: Abstraction, refinement and proof for probabilistic systems. Springer Science & Business Media (2005)
50. McIver, A., Morgan, C., Kaminski, B.L., Katoen, J.P.: A new proof rule for almost-sure termination. Proc. ACM Program. Lang. **2**(POPL) (dec 2017). https://doi.org/10.1145/3158121, https://doi.org/10.1145/3158121
51. Moser, R.A., Tardos, G.: A constructive proof of the general lovász local lemma. J. ACM **57**(2), 11:1–11:15 (2010). https://doi.org/10.1145/1667053.1667060, https://doi.org/10.1145/1667053.1667060
52. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.: Bounded expectations: Resource analysis for probabilistic programs. In: Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation. p. 496–512. PLDI 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3192366.3192394, https://doi.org/10.1145/3192366.3192394
53. Olmedo, F., Kaminski, B.L., Katoen, J.P., Matheja, C.: Reasoning about recursive probabilistic programs. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. p. 672–681. LICS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2933575.2935317, https://doi.org/10.1145/2933575.2935317
54. Pegden, W.: An extension of the moser–tardos algorithmic local lemma. SIAM Journal on Discrete Mathematics **28**(2), 911–917 (2014). https://doi.org/10.1137/110828290, https://doi.org/10.1137/110828290
55. Saeki, S.: A proof of the existence of infinite product probability measures. The American Mathematical Monthly **103**(8), 682–683 (1996), http://www.jstor.org/stable/2974880
56. Sarkar, K., Colbourn, C.J., Bonis, A.D., Vaccaro, U.: Partial covering arrays: Algorithms and asymptotics. Theory Comput. Syst. **62**(6), 1470–1489 (2018). https://doi.org/10.1007/S00224-017-9782-9, https://doi.org/10.1007/s00224-017-9782-9
57. Shearer, J.B.: On a problem of spencer. Combinatorica **5**, 241–245 (1985)
58. Spencer, J.: Asymptotic lower bounds for ramsey functions. Discrete Mathematics **20**, 69–76 (1977)
59. Srinivasan, A.: Progress on algorithmic versions of the lovász local lemma. https://www.ias.edu/sites/default/files/video/Aravind.pdf (April 2013), accessed: 2024–01-01
60. Vondrák, J.: Stanford University Math 233A: Non-constructive methods in combinatorics. https://theory.stanford.edu/~jvondrak/MATH233A-2018/Math233-lec04.pdf (2018), accessed: 2023–12-09

$$
\begin{array}{llll llll}
(\mathit{PVar}) & x, a_n & ::= & \ldots & (\mathit{Nat}) & n, N, M \in \mathbb{N} & (\mathit{Real})\ p, q, r \in \mathbb{R} & \\
(\mathit{Val}) & v & ::= & r \mid \Lambda & (\mathit{Seq}) & \Lambda & \in [\,] \mid n :: \Lambda & \\
(\mathit{Dsts}) & \mathcal{D} & ::= & (\kappa_1, \ldots, \kappa_N) & (\mathit{Dst}) & \kappa & \in \mathbb{D}_{\mathit{Real}} & \\
(\mathit{Evts}) & \mathcal{E} & ::= & (\eta_1, \ldots, \eta_M) & (\mathit{Evt}) & \eta & \in \underbrace{\mathit{Real} \times \cdots \times \mathit{Real}}_{N\ \mathit{Real}\text{'s}} \to \{\text{true}, \text{false}\} &
\end{array}
$$

$$
\begin{array}{llll}
(\mathit{Expr}) & e & ::= & v \mid x \mid e_1 + e_2 \mid e_1 - e_2 \mid a[e] \mid e_1\langle e_2\rangle \\
 & & \mid & \mathsf{len}(e) \mid \mathsf{app}(e_1, e_2) \mid \mathsf{concat}(e_1, e_2) \mid \mathsf{pf}(e_1, e_2) \mid \ldots \\
(\mathit{Bexp}) & b & ::= & \text{true} \mid \text{false} \mid e_1 = e_2 \mid e_1 < e_2 \mid \neg b \mid b \wedge b \mid b \vee b \\
 & & \mid & \mathsf{hold}(e, e_1, \ldots, e_N) \mid \mathsf{vbl}(e_1, e_2) \mid \ldots \\
(\mathit{Stmt}) & C & ::= & \mathbf{skip} \mid x := e \mid x := \mathsf{Sample}(e) \mid a[e_1] := e_2 \\
 & & \mid & C_1; C_2 \mid \mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \mid \mathbf{while}\ (b)\ \mathbf{do}\ C \mid \ldots
\end{array}
$$

Fig. 11. Syntax of the programming language.

A The Programming Language

We give the definitions of the syntax, the semantics rules and some important definitions of our programming language in Fig. 11, Fig. 12, Fig. 13, Fig. 14, Fig. 15 and Fig. 16.

We show the semantics rules of the distribution-based operational semantics in Fig. 13. Informally, $(C, \sigma) \xrightarrow{p} (C', \sigma')$ says that (C, σ) steps to (C', σ') with probability p. For the step which samples from the distribution $\mathcal{D}[i]$ and gets r as the result, the probability is $\mathcal{D}[i](r)$. The probabilities of other execution steps are simply 1. Moreover, $(C, \sigma) \xrightarrow{p}{}^{n} (C', \sigma')$ holds if p is the sum of the probabilities of all n-step paths from (C, σ) to (C', σ'), where each path's probability is the product of all steps' probabilities in the path. Then, for $(C, \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$, since we allow **skip** to stutter (see the first rule in Fig. 13), the probability p is actually the sum of the probabilities of all execution paths which start from (C, σ) and terminate on σ' in no more than n steps.

Now we define the semantic functions, $[\![C]\!](\sigma) \in \mathbb{SD}_{\mathit{State}}$ and $[\![C]\!](\mu) \in \mathbb{SD}_{\mathit{State}}$ (where $\mu \in \mathit{DState}$), as follows:

$$
\begin{array}{lcl}
[\![C]\!](\sigma) & \triangleq & \lambda\sigma'.\ \lim \vec{p},\ \text{ where } \forall n.\ (C, \sigma) \xrightarrow{\vec{p}[n]}{}^{n} (\mathbf{skip}, \sigma') \\
[\![C]\!](\mu) & \triangleq & \mathbb{E}_{\sigma \sim \mu}\{[\![C]\!](\sigma)\}
\end{array}
$$

The probability of C's execution from σ finally reaching σ', say $[\![C]\!](\sigma)(\sigma')$, is the sum of probabilities of all execution paths from (C, σ) to $(\mathbf{skip}, \sigma')$. This is defined by letting n approach infinity in $(C, \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$. We further define $[\![C]\!](\mu)$ by lifting $[\![C]\!](\sigma)$, using the expected sub-distribution in Sec. 3.1.

A.1 Measurability

Below we prove that our resampling-table-based semantics is indeed well-defined (Lem. 2).

$$\begin{aligned}
\mathrm{vbl}(\eta, j) &\text{ iff } \exists r_1, \ldots, r_N, r'.\ \eta(r_1, \ldots, r_N) \neq \eta(r_1, \ldots, r_{j-1}, r', r_{j+1}, \ldots, r_N) \\
\mathrm{P}(\eta) &\triangleq \sum_{r_1 \in supp(\mathcal{D}[1]), \ldots, r_N \in supp(\mathcal{D}[N]) : \eta(r_1, \ldots, r_N) = \mathrm{true}} \prod_{i \in [1,N]} \mathcal{D}[i](r_i) \\
\Gamma(j) &\triangleq \{k \in [1, M] : \exists i \in [1, N].\ \mathrm{vbl}(\mathcal{E}[j], i) \wedge \mathrm{vbl}(\mathcal{E}[k], i))\} \setminus \{j\} \\
\Gamma^+(j) &\triangleq \Gamma(j) \uplus \{j\} \\
\mathrm{Indep}(J) &\text{ iff } \forall j \in J.\ j \in [1, M] \wedge (\forall k \in J.\ k \notin \Gamma(j)) \\
\Gamma'(j) &\triangleq \{k \in [1, M] : \exists r_1, \ldots, r_N, r'_1, \ldots, r'_N. \\
&\qquad \wedge (\forall i \in [1, N].\ (\mathrm{vbl}(\mathcal{E}[j], i) \wedge \mathrm{vbl}(\mathcal{E}[k], i)) \vee r_i = r'_i) \\
&\qquad \wedge \mathsf{hold}(j, r_1, \ldots, r_N) = \mathrm{true} \wedge \mathsf{hold}(k, r'_1, \ldots, r'_N) = \mathrm{true} \\
&\qquad \wedge (\mathsf{hold}(j, r'_1, \ldots, r'_N) = \mathrm{false} \vee \mathsf{hold}(k, r_1, \ldots, r_N) = \mathrm{false})\} \setminus \{j\} \\
\Gamma'^+(j) &\triangleq \Gamma'(j) \uplus \{j\}
\end{aligned}$$

Fig. 12. Definitions related to $\mathcal{D}$ and $\mathcal{E}$.

$$\frac{}{(\mathbf{skip}, \sigma) \xrightarrow{1} (\mathbf{skip}, \sigma)} \qquad \frac{[\![e]\!]_\sigma = v}{(x := e, \sigma) \xrightarrow{1} (\mathbf{skip}, \sigma\{x \rightsquigarrow v\})}$$

$$\frac{[\![e]\!]_\sigma = i \in [1, N] \qquad \mathcal{D}[i](r) = p}{(x := \mathsf{Sample}(e), \sigma) \xrightarrow{p} (\mathbf{skip}, \sigma\{x \rightsquigarrow r\})} \qquad \frac{[\![e_1]\!]_\sigma = n \qquad [\![e_2]\!]_\sigma = v}{(a[e_1] := e_2, \sigma) \xrightarrow{1} (\mathbf{skip}, \sigma\{a_n \rightsquigarrow v\})}$$

$$\frac{}{(\mathbf{skip}; C, \sigma) \xrightarrow{1} (C, \sigma)} \qquad \frac{(C_1, \sigma) \xrightarrow{p} (C'_1, \sigma') \qquad C_1 \neq \mathbf{skip}}{(C_1; C_2, \sigma) \xrightarrow{p} (C'_1; C_2, \sigma')}$$

$$\frac{[\![b]\!]_\sigma = \mathrm{true}}{(\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, \sigma) \xrightarrow{1} (C_1, \sigma)} \qquad \frac{[\![b]\!]_\sigma = \mathrm{false}}{(\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, \sigma) \xrightarrow{1} (C_2, \sigma)}$$

$$\frac{}{(\mathbf{while}\ (b)\ \mathbf{do}\ C, \sigma) \xrightarrow{1} (\mathbf{if}\ (b)\ \mathbf{then}\ (C; \mathbf{while}\ (b)\ \mathbf{do}\ C)\ \mathbf{else}\ \mathbf{skip}, \sigma)}$$

$$\frac{}{(C, \sigma) \xrightarrow{1}{}^0 (C, \sigma)} \qquad \frac{p = \sum_{C', \sigma'} \{p_1 \cdot p_2 \mid (C, \sigma) \xrightarrow{p_1} (C', \sigma') \wedge (C', \sigma') \xrightarrow{p_2}{}^n (C'', \sigma'')\}}{(C, \sigma) \xrightarrow{p}{}^{n+1} (C'', \sigma'')}$$

Fig. 13. Distribution-based operational semantics.

Proof of Lem. 2. By definition, we only need to show that

$$\bigcup_n \{RT \mid RT \vdash (C, \sigma, \iota) \to^n (\mathbf{skip}, \sigma', \_)\} \in \mathcal{F}.$$

Since $\mathcal{F}$ is closed under countable union, this follows from Lem. 4. □

Lemma 4. *For all $C, \sigma, \sigma', \iota$ and n,*

$$\{RT \mid RT \vdash (C, \sigma, \iota) \to^n (\mathbf{skip}, \sigma', \_)\} \in \mathcal{F}.$$

$$
\begin{aligned}
[\![v]\!]_\sigma &\triangleq v\\
[\![x]\!]_\sigma &\triangleq \sigma(x)\\
[\![e_1 + e_2]\!]_\sigma &\triangleq [\![e_1]\!]_\sigma + [\![e_2]\!]_\sigma\\
[\![e_1 - e_2]\!]_\sigma &\triangleq [\![e_1]\!]_\sigma - [\![e_2]\!]_\sigma\\
[\![a[e]]\!]_\sigma &\triangleq \sigma(a_n)\\
&\ \text{where}\ \ [\![e]\!]_\sigma = n\\
[\![e_1\langle e_2\rangle]\!]_\sigma &\triangleq \Lambda\langle n\rangle\\
&\ \text{where}\ \ [\![e_1]\!] = \Lambda, [\![e_2]\!]_\sigma = n \in [1, |\Lambda|]\\
[\![\mathsf{len}(e)]\!]_\sigma &\triangleq |\Lambda|\\
&\ \text{where}\ \ [\![e]\!]_\sigma = \Lambda\\
[\![\mathsf{app}(e_1, e_2)]\!]_\sigma &\triangleq n :: \Lambda\\
&\ \text{where}\ \ [\![e_1]\!]_\sigma = \Lambda, [\![e_2]\!]_\sigma = n\\
[\![\mathsf{concat}(e_1, e_2)]\!]_\sigma &\triangleq \Lambda_1 \parallel \Lambda_2\\
&\ \text{where}\ \ [\![e_1]\!]_\sigma = \Lambda_1, [\![e_2]\!]_\sigma = \Lambda_2\\
[\![\mathsf{pf}(e_1, e_2)]\!]_\sigma &\triangleq \Lambda\langle 1 \dots n\rangle\\
&\ \text{where}\ \ [\![e_1]\!]_\sigma = \Lambda, [\![e_2]\!]_\sigma = n
\end{aligned}
$$

$$
|\Lambda| \triangleq \begin{cases} 0 & \text{if } \Lambda = [] \\ 1 + |\Lambda'| & \text{if } \Lambda = n :: \Lambda' \end{cases}
$$

$$
\Lambda\langle i\rangle \triangleq \begin{cases} n & \text{if } i = |\Lambda| \wedge \Lambda = n :: \Lambda' \\ \Lambda'\langle i\rangle & \text{if } i < |\Lambda| \wedge \Lambda = n :: \Lambda' \end{cases}
$$

$$
\Lambda_1 \parallel \Lambda_2 \triangleq \begin{cases} \Lambda_1 & \text{if } \Lambda_2 = [] \\ n :: (\Lambda_1 \parallel \Lambda_2') & \text{if } \Lambda_2 = n :: \Lambda_2' \end{cases}
$$

$$
\Lambda\langle 1 \dots n\rangle \triangleq \begin{cases} \epsilon & \text{if } n = 0 \\ \Lambda\langle n\rangle :: \Lambda\langle 1 \dots n-1\rangle & \text{if } 1 \le n \le \mathsf{len}(\Lambda) \end{cases}
$$

$$
\Lambda_1 \prec \Lambda_2 \text{ iff } \exists n.\ n < |\Lambda_2| \wedge \Lambda_1 = \Lambda_2\langle 1 \dots n\rangle
$$

Fig. 14. Auxiliary definitions of the programming language (part I).

Proof. Denoting

$$
S_{C,\sigma,\iota,n} \;=\; \{RT \mid RT \vdash (C, \sigma, \iota) \to^n (\mathbf{skip}, \sigma', \_)\},
$$

it suffices to show that $S_{C,\sigma,\iota,n} \in \mathcal{F}$. We prove by induction on n.

- $n = 0$. Note that $S_{C,\sigma,\iota,0} = ((C = \mathbf{skip} \wedge \sigma = \sigma')\ ?\ \mathit{RTable} : \varnothing) \in \mathcal{F}$.
- $n = k + 1$. We have the following two cases.
  - C does not perform a sampling. That is, there exist unique C'' and σ'' such that, for all RT, $RT \vdash (C, \sigma, \iota) \to (C'', \sigma'', \iota)$. Thus $S_{C,\sigma,\iota,n} = S_{C'',\sigma'',\iota,k}$, and from the induction hypothesis we have $S_{C,\sigma,\iota,n} \in \mathcal{F}$.
  - C performs a sampling. That is, there exist unique $x, C'', \iota'' \neq \iota$ and $i \in [1, N]$ such that, for all RT, $RT \vdash (C, \sigma, \iota) \to (C'', \sigma\{x \rightsquigarrow RT[i][\iota_i]\}, \iota'')$.

$$
\begin{aligned}
[\![\mathrm{true}]\!]_\sigma &\triangleq \mathrm{true} \\
[\![\mathrm{false}]\!]_\sigma &\triangleq \mathrm{false} \\
[\![e_1 = e_2]\!]_\sigma &\triangleq \begin{cases} \mathrm{true} & \text{if } [\![e_1]\!]_\sigma = [\![e_2]\!]_\sigma \\ \mathrm{false} & \text{otherwise} \end{cases} \\
[\![e_1 < e_2]\!]_\sigma &\triangleq \begin{cases} \mathrm{true} & \text{if } [\![e_1]\!]_\sigma < [\![e_2]\!]_\sigma \\ \mathrm{false} & \text{otherwise} \end{cases} \\
[\![\neg b]\!]_\sigma &\triangleq \begin{cases} \mathrm{true} & \text{if } [\![b]\!]_\sigma = \mathrm{false} \\ \mathrm{false} & \text{otherwise} \end{cases} \\
[\![b_1 \wedge b_2]\!]_\sigma &\triangleq \begin{cases} \mathrm{true} & \text{if } [\![b_1]\!]_\sigma = \mathrm{true} \text{ and } [\![b_1]\!]_\sigma = \mathrm{true} \\ \mathrm{false} & \text{otherwise} \end{cases} \\
[\![b_1 \vee b_2]\!]_\sigma &\triangleq \begin{cases} \mathrm{true} & \text{if } [\![b_1]\!]_\sigma = \mathrm{true} \text{ or } [\![b_1]\!]_\sigma = \mathrm{true} \\ \mathrm{false} & \text{otherwise} \end{cases} \\
[\![\mathsf{hold}(e, e_1, \ldots, e_N)]\!]_\sigma &\triangleq \mathcal{E}[k](r_1, \ldots, r_N) \\
&\quad \text{where } \; [\![e]\!]_\sigma = k \in [1, M], \\
&\qquad\qquad\; [\![e_i]\!]_\sigma = r_i \text{ for each } i \in [1, N] \\
[\![\mathsf{vbl}(e_1, e_2)]\!]_\sigma &\triangleq \begin{cases} \mathrm{true} & \text{if } \mathrm{vbl}(\mathcal{E}[k], j) \\ \mathrm{false} & \text{otherwise} \end{cases} \\
&\quad \text{where } \; [\![e_1]\!]_\sigma = k \in [1, M], \\
&\qquad\qquad\; [\![e_2]\!]_\sigma = j \in [1, N]
\end{aligned}
$$

Fig. 15. Auxiliary definitions of the programming language (part II).

Thus

$$S_{C,\sigma,\iota,n} = \bigcup_{r \in supp(\mathcal{D}[i])} \begin{pmatrix} S_{C'',\sigma\{x \rightsquigarrow r\},\iota'',k} \\ \cap \{RT \mid RT[i][\iota_i] \;\; = r\} \end{pmatrix}.$$

Note that $\{RT \mid RT[i][\iota_i] \;\; = r\} \in \mathcal{F}$ for all $r \in supp(\mathcal{D}[i])$. Since $\mathcal{F}$ is closed under countable union and intersection, from the induction hypothesis we have $S_{C,\sigma,\iota,n} \in \mathcal{F}$.

□

A.2 Semantics Equivalence

To prove the equivalence between the distribution-based semantics and the RT-based semantics (Thm. 1), we define an auxiliary semantics, called *partial-table-based semantics*, as presented in Fig. 17. Informally, a partial table PT is a "prefix" of some resampling table. We first prove that the distribution-based semantics is equivalent to the PT-based semantics (Lem. 10), and then prove that the PT-based semantics is equivalent to the RT-based semantics (Lem. 11).

We first give some important properties related to the PT-based semantics.

$$\frac{}{RT \vdash (\mathbf{skip}, \sigma, \iota) \to (\mathbf{skip}, \sigma, \iota)}$$

$$\frac{[\![e]\!]_\sigma = v}{RT \vdash (x := e, \sigma, \iota) \to (\mathbf{skip}, \sigma\{x \rightsquigarrow v\}, \iota)}$$

$$\frac{[\![e_1]\!]_\sigma = n \qquad [\![e_2]\!]_\sigma = v}{\vdash (a[e_1] := e_2, \sigma, \iota) \to (\mathbf{skip}, \sigma\{a_n \rightsquigarrow v\}, \iota)}$$

$$\frac{[\![e]\!]_\sigma = i \in [1, N] \qquad \iota' = (\iota[1], \ldots, \iota[i-1], \iota[i]+1, \iota[i+1], \ldots, \iota[N])}{RT \vdash (x := \mathsf{Sample}(e), \sigma, \iota) \to (\mathbf{skip}, \sigma\{x \rightsquigarrow RT[i][\iota[i]]\}, \iota')}$$

$$\frac{RT \vdash (C_1, \sigma, \iota) \to (C_1', \sigma', \iota') \qquad C_1 \neq \mathbf{skip}}{RT \vdash (C_1; C_2, \sigma, \iota) \to (C_1'; C_2, \sigma', \iota')} \qquad \frac{}{RT \vdash (\mathbf{skip}; C, \sigma, \iota) \to (C, \sigma, \iota)}$$

$$\frac{[\![b]\!]_\sigma = \text{true}}{RT \vdash (\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, \sigma, \iota) \to (C_1, \sigma, \iota)}$$

$$\frac{[\![b]\!]_\sigma = \text{false}}{RT \vdash (\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, \sigma, \iota) \to (C_2, \sigma, \iota)}$$

$$\frac{}{RT \vdash (\mathbf{while}\ (b)\ \mathbf{do}\ C, \sigma, \iota) \to (\mathbf{if}\ (b)\ \mathbf{then}\ (C; \mathbf{while}\ (b)\ \mathbf{do}\ C)\ \mathbf{else\ skip}, \sigma, \iota)}$$

Fig. 16. Resampling-table-based operational semantics.

Lemma 5. *For all $n, RT, C, C', \sigma, \sigma', \iota$ and ι', if*

$$RT \vdash (C, \sigma, \iota) \to^n (C', \sigma', \iota'),$$

then $\max(\iota) \leq \max(\iota') \leq \max(\iota) + n$*.*

Proof. By induction on n. $\square$

Lemma 6. *For all $k, n, RT, RT', C, C', \sigma, \sigma', \iota$ and ι', if*

– $RT \vdash (C, \sigma, \iota) \to^k (C', \sigma', \iota')$*;*
– *For all $i \in [1, N]$ and $j \in [0, n)$, $RT[i][j] = RT'[i][j]$;*
– $\max(\iota') \leq n$*;*

then $RT' \vdash (C, \sigma, \iota) \to^k (C', \sigma', \iota')$.

Proof. By induction on k. $\square$

Lemma 7. *For all $k, n, RT, C, C', \sigma, \sigma', \iota'$ and $PT \in \mathcal{PT}_n$, if $k \leq n$ and $PT \sqsubseteq RT$, then*

$$PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota') \iff RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota').$$

Proof. If $PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota')$, then there exists RT' such that $RT' \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota')$, $PT \sqsubseteq RT'$, and $\max(\iota') \leq n$. Since $PT \sqsubseteq RT$, we know that $RT[i][j] = RT'[i][j]$ for all $i \in [1, N]$ and $j \in [0, n)$, and thus from Lem. 6 we have $RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota')$.

If $RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota')$, then from Lem. 5 we have $\max(\iota') \leq k \leq n$. Thus by definition $PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^k (C', \sigma', \iota')$. $\square$

$$(\mathit{PTable})\ PT \in \biguplus_{n\geq 0}([1,N]\times[0,n)\to \mathit{Real})$$
$$\text{where } \forall i,j.\ PT[i][j] \in \mathit{supp}(\mathcal{D}[i])$$

$$\begin{aligned}
\mathsf{ncol}(PT) &\triangleq n \qquad \text{where } \mathrm{dom}(PT) = [1,N]\times[0,n)\\
\mathcal{PT}_n &\triangleq \{PT \mid \mathsf{ncol}(PT) = n\}\\
\omega(PT) &\triangleq \textstyle\prod_{i\in[1,N],j\in[0,\mathsf{ncol}(PT))} \mathcal{D}[i](PT[i][j])\\
\max(\iota) &\triangleq \max(\iota_1,\ldots,\iota_N)
\end{aligned}$$

$$PT \sqsubseteq RT \text{ iff } \forall i \in [1,N], j\in[0,\mathsf{ncol}(PT)).\ PT[i][j] = RT[i][j]$$
$$PT_1 \sqsubseteq PT_2 \text{ iff } \forall i \in [1,N], j\in[0,\mathsf{ncol}(PT_1)).\ PT_1[i][j] = PT_2[i][j]$$

$$\frac{PT \sqsubseteq RT \qquad RT \vdash (C,\sigma,\iota) \to^n (C',\sigma',\iota') \qquad \max(\iota') \leq \mathsf{ncol}(PT)}{PT \vdash (C,\sigma,\iota) \to^n (C',\sigma',\iota')}$$

$$[\![C]\!]_{\mathrm{PT}}(\sigma) \triangleq \lambda\sigma'.\ \lim_{n\to\infty} \sum_{PT\in\mathcal{PT}_n} \omega(PT)\cdot\left[PT \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^n (\mathbf{skip},\sigma',\_)\right]$$
$$[\![C]\!]_{\mathrm{PT}}(\mu) \triangleq \mathbb{E}_{\sigma\sim\mu}\{[\![C\,]\!]_{\mathrm{PT}}(\sigma)\}$$

Fig. 17. Partial-table-based semantics and auxiliary definitions.

Lemma 8. *For all $k, n_1, n_2, C, C', \sigma, \sigma', \iota', PT_1 \in \mathcal{PT}_{n_1}$ and $PT_2 \in \mathcal{PT}_{n_2}$, if $k \leq n_1 \leq n_2$ and $PT_1 \sqsubseteq PT_2$, then*

$$PT_1 \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota') \iff PT_2 \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota').$$

Proof. If $PT_1 \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$, then there exists RT such that $RT \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$, $PT_1 \sqsubseteq RT$ and $\max(\iota') \leq n_1 \leq n_2$. Now we define a new resampling table RT' by replacing the first n_2 columns of RT with PT_2, then $PT_1 \sqsubseteq PT_2 \sqsubseteq RT'$, and thus $RT[i][j] = RT'[i][j]$ for all $i \in [1,N]$ and $j \in [0,n_1)$. From Lem. 6, we have $RT' \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$, and thus by definition $PT_2 \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$.

If $PT_2 \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$, then there exists RT such that $RT \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$ and $PT_1 \sqsubseteq PT_2 \sqsubseteq RT$. From Lem. 5, we have $\max(\iota') \leq k \leq n_1$, and thus by definition $PT_1 \vdash (C,\sigma,\iota_{\mathsf{init}}) \to^k (C',\sigma',\iota')$. □

Lemma 9. *For all $n \geq 1$, $PT_1, PT_2 \in \mathcal{PT}_n$, $k, C, C', \sigma, \sigma', \iota$ and ι', if*

- *There exists $i' \in [1,N]$ such that:*
  - *For all $i \in [1,N]\setminus\{i'\}$ and $j \in [0,n)$, $PT_1[i][j] = PT_2[i][j]$;*
  - *$PT_2[i'][0] = PT_1[i'][n-1]$;*
  - *For all $j \in [0,n-1)$, $PT_2[i'][j+1] = PT_1[i'][j]$;*
- *$\max(\iota') < n$;*

then

$$\begin{aligned}
&PT_1 \vdash (C,\sigma,\iota) \to^k (C',\sigma',\iota') \iff\\
&\quad PT_2 \vdash (C,\sigma,\iota\{i' \rightsquigarrow \iota[i']+1\}) \to^k (C',\sigma',\iota'\{i' \rightsquigarrow \iota'[i']+1\}).
\end{aligned}$$

Proof. Let $PT_1 \vdash (C, \sigma, \iota) \to^k (C', \sigma', \iota')$, then there exists RT such that $PT_1 \sqsubseteq RT$, $\max(\iota') < n$ and $RT \vdash (C, \sigma, \iota) \to^k (C', \sigma', \iota')$. Now we define a new resampling table RT' by replacing the first n columns of RT with PT_2, then $PT_2 \sqsubseteq RT'$. By definition, to prove $PT_2 \vdash (C, \sigma, \iota\{i' \rightsquigarrow \iota[i'] + 1\}) \to^k (C', \sigma', \iota'\{i' \rightsquigarrow \iota'[i'] + 1\})$, since $\max(\iota'\{i' \rightsquigarrow \iota'[i'] + 1\}) \leq \max(\iota') + 1 \leq n$, it remains to show that $RT' \vdash (C, \sigma, \iota\{i' \rightsquigarrow \iota[i'] + 1\}) \to^k (C', \sigma', \iota'\{i' \rightsquigarrow \iota'[i'] + 1\})$. This can be proved by induction on k.

Let $PT_2 \vdash (C, \sigma, \iota\{i' \rightsquigarrow \iota[i'] + 1\}) \to^k (C', \sigma', \iota'\{i' \rightsquigarrow \iota'[i'] + 1\})$, then there exists RT such that $PT_2 \sqsubseteq RT$ and $RT \vdash (C, \sigma, \iota\{i' \rightsquigarrow \iota[i'] + 1\}) \to^k (C', \sigma', \iota'\{i' \rightsquigarrow \iota'[i'] + 1\})$. Now we define a new resampling table RT' by replacing the first n columns of RT with PT_1, then $PT_1 \sqsubseteq RT'$. By definition, since $\max(\iota') < n$, to prove $PT_1 \vdash (C, \sigma, \iota) \to^k (C', \sigma', \iota')$, we only need to show that $RT' \vdash (C, \sigma, \iota) \to^k (C', \sigma', \iota')$. This can again be proved by induction on k. □

The following lemma states the equivalence between the distribution-based semantics and the PT-based semantics.

Lemma 10. *For all C and μ,*

$$[\![C]\!](\mu) = [\![C]\!]_{\mathrm{PT}}(\mu).$$

Proof. Below we prove that, for all σ, σ',

$$[\![C]\!](\sigma)(\sigma') = [\![C]\!]_{\mathrm{PT}}(\sigma)(\sigma').$$

By definition, we only need to prove that

$$\vec{p}[n] = \sum_{PT \in \mathcal{PT}_n} \omega(PT) \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_)]$$

holds for all n, where $(C, \sigma) \xrightarrow{\vec{p}[n]}{}^n (\mathbf{skip}, \sigma')$. We prove by induction on n. For $n = 0$, we have LHS $= [C = \mathbf{skip} \wedge \sigma = \sigma'] =$ RHS. Below we assume that $n = k + 1$. We have the following two cases.

– C does not perform a sampling. That is, there exist unique C'', σ'' such that $(C, \sigma) \xrightarrow{1} (C'', \sigma'')$. From the induction hypothesis and Lem. 8, we have

$$\begin{aligned}
\vec{p}[n] &= p \qquad (\text{where } (C'', \sigma'') \xrightarrow{p}{}^k (\mathbf{skip}, \sigma')) \\
&= \sum_{PT \in \mathcal{PT}_k} \omega(PT) \cdot \left[PT \vdash (C'', \sigma'', \iota_{\mathsf{init}}) \to^k (\mathbf{skip}, \sigma', \_)\right] \\
&= \sum_{PT \in \mathcal{PT}_n} \omega(PT) \cdot \left[PT \vdash (C'', \sigma'', \iota_{\mathsf{init}}) \to^k (\mathbf{skip}, \sigma', \_)\right] \\
&= \sum_{PT \in \mathcal{PT}_n} \omega(PT) \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_)] .
\end{aligned}$$

– C performs a sampling. That is, there exist unique C'', x and $i \in [1, N]$ such that: for all $r \in supp(\mathcal{D}[i])$, $(C, \sigma) \xrightarrow{\mathcal{D}[i](r)} (C'', \sigma\{x \rightsquigarrow r\})$ holds. From Lem. 5, Lem. 8, Lem. 9 and the induction hypothesis, we have

$$\begin{aligned}
\vec{p}[n] &= \sum_{r \in supp(\mathcal{D}[i])} \{\mathcal{D}[i](r) \cdot p_2 \mid (C'', \sigma\{x \rightsquigarrow r\}) \xrightarrow{p_2}{}^{k} (\mathbf{skip}, \sigma')\} \\
&= \sum_{r \in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \sum_{PT \in \mathcal{PT}_k} \omega(PT) \\
&\qquad \cdot \left[PT \vdash (C'', \sigma\{x \rightsquigarrow r\}, \iota_{\mathsf{init}}) \rightarrow^{k} (\mathbf{skip}, \sigma', \_)\right] \\
&= \sum_{r \in supp(\mathcal{D}[i])} \sum_{PT \in \mathcal{PT}_n : PT[i][k] = r} \omega(PT) \\
&\qquad \cdot \left[PT \vdash (C'', \sigma\{x \rightsquigarrow r\}, \iota_{\mathsf{init}}) \rightarrow^{k} (\mathbf{skip}, \sigma', \_)\right] \\
&= \sum_{r \in supp(\mathcal{D}[i])} \sum_{PT \in \mathcal{PT}_n : PT[i][0] = r} \omega(PT) \\
&\qquad \cdot \left[PT \vdash (C'', \sigma\{x \rightsquigarrow r\}, \iota_{\mathsf{init}}\{i \rightsquigarrow 1\}) \rightarrow^{k} (\mathbf{skip}, \sigma', \_)\right] \\
&= \sum_{r \in supp(\mathcal{D}[i])} \sum_{PT \in \mathcal{PT}_n : PT[i][0] = r} \omega(PT) \\
&\qquad \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \rightarrow^{n} (\mathbf{skip}, \sigma', \_)] \\
&= \sum_{PT \in \mathcal{PT}_n} \omega(PT) \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \rightarrow^{n} (\mathbf{skip}, \sigma', \_)] \,.
\end{aligned}$$

□

The following lemma states the equivalence between the PT-based semantics and the RT-based semantics.

Lemma 11. *For all C and μ,*

$$[\![C]\!]_{\mathrm{RT}}(\mu) = [\![C]\!]_{\mathrm{PT}}(\mu).$$

Proof. We prove that, for all σ, σ',

$$[\![C]\!]_{\mathrm{RT}}(\sigma)(\sigma') = [\![C]\!]_{\mathrm{PT}}(\sigma)(\sigma').$$

By definition, we only need to prove the following:

$$\begin{aligned}
&\mathcal{M}(\{RT \mid RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \rightarrow^{*} (\mathbf{skip}, \sigma', \_)\}) \\
&= \lim_{n \to \infty} \sum_{PT \in \mathcal{PT}_n} \omega(PT) \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \rightarrow^{n} (\mathbf{skip}, \sigma', \_)] \,.
\end{aligned}$$

Since $\{RT : PT \sqsubseteq RT\} \in \mathcal{F}$ for all PT, from Lem. 7 we have

$$\begin{aligned}
\mathrm{LHS} &= \mathcal{M}\left(\lim_{n \to \infty} \{RT \mid RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \rightarrow^{n} (\mathbf{skip}, \sigma', \_)\}\right) \\
&= \lim_{n \to \infty} \mathcal{M}(\{RT \mid RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \rightarrow^{n} (\mathbf{skip}, \sigma', \_)\})
\end{aligned}$$

$$
\begin{aligned}
&= \lim_{n\to\infty} \mathcal{M}\left(\biguplus_{PT \in \mathcal{PT}_n} \left\{ \begin{array}{l} RT \mid PT \sqsubseteq RT \wedge \\ \qquad RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_) \end{array} \right\} \right) \\
&= \lim_{n\to\infty} \sum_{PT \in \mathcal{PT}_n} \mathcal{M}\left(\left\{ \begin{array}{l} RT \mid PT \sqsubseteq RT \wedge \\ \qquad RT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_) \end{array} \right\} \right) \\
&= \lim_{n\to\infty} \sum_{PT \in \mathcal{PT}_n} \mathcal{M}\left(\left\{ \begin{array}{l} RT \mid PT \sqsubseteq RT \wedge \\ \qquad PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_) \end{array} \right\} \right) \\
&= \lim_{n\to\infty} \sum_{PT \in \mathcal{PT}_n} \mathcal{M}(\{RT \mid PT \sqsubseteq RT\}) \\
&\qquad \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_)] \\
&= \lim_{n\to\infty} \sum_{PT \in \mathcal{PT}_n} \omega(PT) \cdot [PT \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^n (\mathbf{skip}, \sigma', \_)] \,.
\end{aligned}
$$

□

Proof of Thm. 1. Directly from Lem. 10 and Lem. 11. □

B Assertion Languages

This section gives detailed definitions of our assertion languages.

B.1 Assertions over States and State Distributions

We give the definition of assertions over states and state distributions in Fig. 18. We use $\#\{e_1, \ldots, e_n\}$ [5] as a shorthand of the assertion

$$\forall X_1, \ldots, X_n.\ \Pr\left[\bigwedge_{i\in[1,n]} . e_i = X_i\right] = \prod_{i\in[1,n]} \Pr[e_i = X_i].$$

Below we define the partial correctness in the distribution-based semantics.

Definition 6 (Partial Correctness). *For all P, C, Q, $\models \{P\}C\{Q\}$ holds iff*

$$\forall \mu.\ \ \mu \models P \wedge |[\![C]\!](\mu)| = 1 \implies [\![C]\!](\mu) \models Q.$$

B.2 Assertions over RT-Extended States

We give the definition of assertions over RT-extended states in Fig. 19, Fig. 20 and Fig. 21.

C Soundness of Loop Truncation

Below we prove the soundness of loop truncation (Thm. 2).

$$\begin{array}{llcl}
(\mathit{Assn}) & \mathbf{p},\mathbf{q},\mathbf{r} & ::= & b \mid \neg\mathbf{q} \mid \mathbf{q}_1 \wedge \mathbf{q}_2 \mid \mathbf{q}_1 \vee \mathbf{q}_2 \mid \forall X.\, \mathbf{q} \mid \exists X.\, \mathbf{q} \mid \ldots \\
(\mathit{PExp}) & \xi & ::= & r \mid \mathbb{E}[e] \mid \Pr[\mathbf{q}] \mid \xi_1 + \xi_2 \mid \xi_1 - \xi_2 \mid \cdots \\
(\mathit{PAssn}) & P,Q,R & ::= & \lceil \mathbf{q} \rceil \mid e_1 \sim e_2 \mid \xi_1 = \xi_2 \mid \xi_1 < \xi_2 \\
 & & & \mid \neg Q \mid Q_1 \wedge Q_2 \mid Q_1 \vee Q_2 \mid Q_1 \oplus_p Q_2 \mid \forall X.\, Q \mid \exists X.\, Q \mid \ldots
\end{array}$$

$$\begin{array}{ll}
\mu_1 \oplus_p \mu_2 & \triangleq \lambda\sigma.\, p \cdot \mu_1(\sigma) \\
 & \quad + (1-p) \cdot \mu_2(\sigma) \\
\mu\{X \rightsquigarrow v\} & \triangleq \mathbb{E}_{\sigma\sim\mu}\{\delta(\sigma\{X \rightsquigarrow v\})\} \\
 & \\
[\![r]\!]_\mu & \triangleq r \\
[\![\mathbb{E}[e]]\!]_\mu & \triangleq \mathbb{E}_{\sigma\sim\mu}[[\![e]\!]_\sigma] \\
[\![\Pr[\mathbf{q}]]\!]_\mu & \triangleq \Pr_{\sigma\sim\mu}[\sigma \vDash \mathbf{q}] \\
[\![\xi_1 + \xi_2]\!]_\mu & \triangleq [\![\xi_1]\!]_\mu + [\![\xi_2]\!]_\mu \\
[\![\xi_1 - \xi_2]\!]_\mu & \triangleq [\![\xi_1]\!]_\mu - [\![\xi_2]\!]_\mu \\
 & \\
\sigma \vDash b & \text{iff } [\![b]\!]_\sigma = \text{true} \\
\sigma \vDash \neg\mathbf{q} & \text{iff } \neg(\sigma \vDash \mathbf{q}) \\
\sigma \vDash \mathbf{q}_1 \wedge \mathbf{q}_2 & \text{iff } (\sigma \vDash \mathbf{q}_1) \wedge (\sigma \vDash \mathbf{q}_2) \\
\sigma \vDash \mathbf{q}_1 \vee \mathbf{q}_2 & \text{iff } (\sigma \vDash \mathbf{q}_1) \vee (\sigma \vDash \mathbf{q}_2) \\
\sigma \vDash \forall X.\, \mathbf{q} & \text{iff } \forall v.\, \sigma\{X \rightsquigarrow v\} \vDash \mathbf{q} \\
\sigma \vDash \exists X.\, \mathbf{q} & \text{iff } \exists v.\, \sigma\{X \rightsquigarrow v\} \vDash \mathbf{q}
\end{array}$$

$$\begin{array}{rl}
\mu \vDash \lceil \mathbf{q} \rceil & \text{iff } \forall \sigma.\, \sigma \in \mathit{supp}(\mu) \implies \sigma \vDash \mathbf{q} \\
\mu \vDash e_1 \sim e_2 & \text{iff } \exists n.\, (\mu \vDash \lceil e_2 = n \rceil) \wedge \\
 & \quad (\forall r.\, [\![\Pr[e_1 = r]]\!]_\mu = \mathcal{D}[n](r)) \\
\mu \vDash \xi_1 = \xi_2 & \text{iff } [\![\xi_1]\!]_\mu = [\![\xi_2]\!]_\mu \\
\mu \vDash \xi_1 < \xi_2 & \text{iff } [\![\xi_1]\!]_\mu < [\![\xi_2]\!]_\mu \\
\mu \vDash Q_1 \oplus_p Q_2 & \text{iff } (p = 1 \wedge \mu \vDash Q_1) \vee \\
 & \quad (p = 0 \wedge \mu \vDash Q_2) \vee \\
 & \quad (\exists \mu_1, \mu_2.\, \mu = \mu_1 \oplus_p \mu_2 \wedge \\
 & \qquad \mu_1 \vDash Q_1 \wedge \mu_2 \vDash Q_2) \\
\mu \vDash \neg Q & \text{iff } \neg(\mu \vDash Q) \\
\mu \vDash Q_1 \wedge Q_2 & \text{iff } (\mu \vDash Q_1) \wedge (\mu \vDash Q_2) \\
\mu \vDash Q_1 \vee Q_2 & \text{iff } (\mu \vDash Q_1) \vee (\mu \vDash Q_2) \\
\mu \vDash \forall X.\, Q & \text{iff } \forall v.\, \mu\{X \rightsquigarrow v\} \vDash Q \\
\mu \vDash \exists X.\, Q & \text{iff } \exists v.\, \mu\{X \rightsquigarrow v\} \vDash Q \\
\vDash Q & \text{iff } \forall \mu.\, \mu \vDash Q
\end{array}$$

Fig. 18. Assertions over states and state distributions.

$$\begin{array}{llcl}
(\mathit{RTState}) & \Sigma & ::= & (\sigma, \mathit{RT}, \iota) \\
(\mathit{RTExpr}) & E & ::= & e \mid \mathsf{RT}[E_1][E_2] \mid \mathsf{hd}_1 \mid \ldots \mid \mathsf{hd}_N \mid E_1 + E_2 \mid [B] \mid \ldots \\
(\mathit{RTBexp}) & B & ::= & b \mid E_1 = E_2 \mid E_1 < E_2 \mid \neg B \mid B_1 \wedge B_2 \mid B_1 \vee B_2 \mid \ldots \\
(\mathit{RTAssn}) & \mathbf{P},\mathbf{Q},\mathbf{R} & ::= & \mathbf{q} \mid B \mid \neg\mathbf{Q} \mid \mathbf{Q}_1 \wedge \mathbf{Q}_2 \mid \mathbf{Q}_1 \vee \mathbf{Q}_2 \mid \forall X.\, \mathbf{Q} \mid \exists X.\, \mathbf{Q} \mid \ldots
\end{array}$$

Fig. 19. Assertions over RT-extended states.

Lemma 12. *For all $E, \vec{\mu}$ and μ, if $\lim \vec{\mu} = \mu$, then*

$$\Pr_{a\sim\mu}[E(a)] = \lim_{n\to\infty} \Pr_{a\sim\vec{\mu}[n]}[E(a)].$$

Proof. For all n,

$$\begin{aligned}
0 &\leq \left| \Pr_{a\sim\mu}[E(a)] - \Pr_{a\sim\vec{\mu}[n]}[E(a)] \right| \\
&= \left| \sum_{a\in A : E(a)} (\mu(a) - \vec{\mu}[n](a)) \right| \\
&\leq \sum_{a\in A} |\mu(a) - \vec{\mu}[n](a)|.
\end{aligned}$$

$$\begin{array}{ll}
[\![e]\!]_{(\sigma,RT,\iota)} & \triangleq [\![e]\!]_\sigma \\
[\![\mathsf{RT}[E_1][E_2]]\!]_\Sigma & \triangleq RT[i][j] \\
 & \quad \text{where } [\![E_1]\!]_\Sigma = i, [\![E_2]\!]_\Sigma = j \\
[\![\mathsf{hd}_n]\!]_{(\sigma,RT,\iota)} & \triangleq \iota[n] \\
[\![E_1 + E_2]\!]_\Sigma & \triangleq [\![E_1]\!]_\Sigma + [\![E_2]\!]_\Sigma \\
[\![[B]]\!]_\Sigma & \triangleq [\![B]\!]_\Sigma = \text{true} \,?\, 1 : 0
\end{array}$$

$$\begin{array}{ll}
(\sigma, RT, \iota) \models \mathbf{q} & \text{iff } \sigma \models \mathbf{q} \\
\Sigma \models B & \text{iff } [\![B]\!]_\Sigma = \text{true} \\
\Sigma \models \neg\mathbf{Q} & \text{iff } \neg(\Sigma \models \mathbf{Q}) \\
\Sigma \models \mathbf{Q}_1 \wedge \mathbf{Q}_2 & \text{iff } (\Sigma \models \mathbf{Q}_1) \wedge (\Sigma \models \mathbf{Q}_2) \\
\Sigma \models \mathbf{Q}_1 \vee \mathbf{Q}_2 & \text{iff } (\Sigma \models \mathbf{Q}_1) \vee (\Sigma \models \mathbf{Q}_2) \\
(\sigma, RT, \iota) \models \forall X.\, \mathbf{Q} & \text{iff } \forall v.\ (\sigma\{X \rightsquigarrow v\}, RT, \iota) \models \mathbf{Q} \\
(\sigma, RT, \iota) \models \exists X.\, \mathbf{Q} & \text{iff } \exists v.\ (\sigma\{X \rightsquigarrow v\}, RT, \iota) \models \mathbf{Q} \\
RT \models \mathbf{Q} & \text{iff } \forall \sigma, \iota.\ (\sigma, RT, \iota) \models \mathbf{Q} \\
\models_{\mathrm{RT}} \mathbf{Q} & \text{iff } \forall \Sigma.\ \Sigma \models \mathbf{Q}
\end{array}$$

Fig. 20. Auxiliary definitions of assertions over RT-extended states (part I).

$$\begin{array}{ll}
[\![b]\!]_{(\sigma,RT,\iota)} & \triangleq [\![b]\!]_\sigma \\
[\![E_1 = E_2]\!]_\Sigma & \triangleq \begin{cases} \text{true} & \text{if } [\![E_1]\!]_\Sigma = [\![E_2]\!]_\Sigma \\ \text{false} & \text{otherwise} \end{cases} \\
[\![E_1 < E_2]\!]_\Sigma & \triangleq \begin{cases} \text{true} & \text{if } [\![E_1]\!]_\Sigma < [\![E_2]\!]_\Sigma \\ \text{false} & \text{otherwise} \end{cases} \\
[\![\neg B]\!]_\Sigma & \triangleq \begin{cases} \text{true} & \text{if } [\![B]\!]_\sigma = \text{false} \\ \text{false} & \text{otherwise} \end{cases} \\
[\![B_1 \wedge B_2]\!]_\Sigma & \triangleq \begin{cases} \text{true} & \text{if } [\![B_1]\!]_\sigma = \text{true and } [\![B_2]\!]_\sigma = \text{true} \\ \text{false} & \text{otherwise} \end{cases} \\
[\![B_1 \vee B_2]\!]_\Sigma & \triangleq \begin{cases} \text{true} & \text{if } [\![B_1]\!]_\sigma = \text{true or } [\![B_2]\!]_\sigma = \text{true} \\ \text{false} & \text{otherwise} \end{cases}
\end{array}$$

Fig. 21. Auxiliary definitions of assertions over RT-extended states (part II).

Then by the squeeze theorem we have

$$\lim_{n\to\infty} \left| \Pr_{a\sim\mu}[E(a)] - \Pr_{a\sim\vec{\mu}[n]}[E(a)] \right| = 0,$$

and the lemma follows. □

We define **E** and **modbf** in Fig. 22. The set $\mathsf{fv}(e)$ contains all the program variables in e. The set $\mathsf{wv}(C)$ contains all the variables in e modified by C, and

$$\begin{array}{rl} (\mathit{Ctx})\ \mathbf{E} ::= & [\,] \mid C;\mathbf{E} \mid \mathbf{E};C \mid \textbf{while } (b) \textbf{ do } \mathbf{E} \\ & \mid \textbf{if } (b) \textbf{ then } C \textbf{ else } \mathbf{E} \mid \textbf{if } (b) \textbf{ then } \mathbf{E} \textbf{ else } C \end{array}$$

$$\begin{array}{ll}
\mathbf{modbf}([\,], e) & \text{always holds} \\
\mathbf{modbf}(C;\mathbf{E}, e) & \text{iff } \mathbf{modbf}(\mathbf{E}, e) \\
\mathbf{modbf}(\mathbf{E};C, e) & \text{iff } \mathbf{modbf}(\mathbf{E}, e) \wedge \mathsf{fv}(e) \cap \mathsf{wv}(C) = \varnothing \\
\mathbf{modbf}(\textbf{if } (b) \textbf{ then } C \textbf{ else } \mathbf{E}, e) & \text{iff } \mathbf{modbf}(\mathbf{E}, e) \\
\mathbf{modbf}(\textbf{if } (b) \textbf{ then } \mathbf{E} \textbf{ else } C, e) & \text{iff } \mathbf{modbf}(\mathbf{E}, e) \\
\mathbf{modbf}(\textbf{while } (b) \textbf{ do } \mathbf{E}, e) & \text{iff } \mathsf{fv}(e) \cap \mathsf{wv}(\mathbf{E}) = \varnothing
\end{array}$$

Fig. 22. Definitions of **E** and **modbf**

$$\begin{array}{ll}
\mathsf{fv}(v) & \triangleq \varnothing \\
\mathsf{fv}(x) & \triangleq \{x\} \\
\mathsf{fv}(e_1 + e_2) & \triangleq \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\
\mathsf{fv}(e_1 - e_2) & \triangleq \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\
\mathsf{fv}(a[e]) & \triangleq \{a_n : n \in \mathit{Nat}\} \\
\mathsf{fv}(e_1\langle e_2\rangle) & \triangleq \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\
\mathsf{fv}(\mathsf{len}(e)) & \triangleq \mathsf{fv}(e) \\
\mathsf{fv}(\mathsf{app}(e_1, e_2)) & \triangleq \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\
\mathsf{fv}(\mathsf{concat}(e_1, e_2)) & \triangleq \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\
\mathsf{fv}(\mathsf{pf}(e_1, e_2)) & \triangleq \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2)
\end{array}$$

$$\begin{array}{ll}
\mathsf{wv}(\mathbf{skip}) & \triangleq \varnothing \\
\mathsf{wv}(x := e) & \triangleq \{x\} \\
\mathsf{wv}(x := \mathsf{Sample}(e)) & \triangleq \{x\} \\
\mathsf{wv}(a[e_1] := e_2) & \triangleq \{a_n : n \in \mathit{Nat}\} \\
\mathsf{wv}(C_1; C_2) & \triangleq \mathsf{wv}(C_1) \cup \mathsf{wv}(C_2) \\
\mathsf{wv}(\textbf{if } (b) \textbf{ then } C_1 \textbf{ else } C_2) & \triangleq \mathsf{wv}(C_1) \cup \mathsf{wv}(C_2) \\
\mathsf{wv}(\textbf{while } (b) \textbf{ do } C) & \triangleq \mathsf{wv}(C)
\end{array}$$

$$\begin{array}{ll}
\mathsf{wv}([\,]) & \triangleq \varnothing \\
\mathsf{wv}(C;\mathbf{E}) & \triangleq \mathsf{wv}(C) \cup \mathsf{wv}(\mathbf{E}) \\
\mathsf{wv}(\mathbf{E};C) & \triangleq \mathsf{wv}(\mathbf{E}) \cup \mathsf{wv}(C) \\
\mathsf{wv}(\textbf{if } (b) \textbf{ then } C \textbf{ else } \mathbf{E}) & \triangleq \mathsf{wv}(C) \cup \mathsf{wv}(\mathbf{E}) \\
\mathsf{wv}(\textbf{if } (b) \textbf{ then } \mathbf{E} \textbf{ else } C) & \triangleq \mathsf{wv}(\mathbf{E}) \cup \mathsf{wv}(C) \\
\mathsf{wv}(\textbf{while } (b) \textbf{ do } \mathbf{E}) & \triangleq \mathsf{wv}(\mathbf{E})
\end{array}$$

Fig. 23. Definitions of fv and wv.

$\mathsf{wv}(\mathbf{E})$ contains all the variables modified by $\mathbf{E}$. We give the definitions of fv and wv in Fig. 23. One can prove that $\mathsf{fv}(e) \cap \mathsf{wv}(\mathbf{E}) = \varnothing \implies \mathbf{modbf}(\mathbf{E}, e)$.

Proof of Thm. 2. We use the following notations:

$$C_{\mathsf{W}} \triangleq \textbf{while}\ (b)\ \textbf{do}\ C$$
$$C'_{\mathsf{W}}(K) \triangleq \textbf{while}\ (b \wedge e < K)\ \textbf{do}\ C$$

Let $\mu \vDash P$. From the first premise, we know that

$$\begin{aligned}\forall K \in \mathbb{N}.\ & |[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)| = 1 \wedge \\ & ([\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu) \vDash Q) \wedge \\ & ([\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu) \vDash \lceil e \geq 0 \rceil \wedge \mathbb{E}[e] \leq r).\end{aligned}$$

Then from Lem. 15 and $\mathbf{modbf}(\mathbf{E}, e)$ we have $|[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| = 1$. Moreover, from Lem. 19 and $\mathbf{modbf}(\mathbf{E}, e)$ we have

$$[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu) = \lim_{K \to \infty} [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu).$$

Since t-$\mathbf{closed}(Q)$, we then have $[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu) \vDash Q$. □

Following [5], we define restricted state sub-distributions. For $\mu \in \mathbb{SD}_{State}$, the restricted sub-distribution on set $S \subseteq PVar$ is defined as $\mu_{|S} \in \mathbb{SD}_{State_{|S}}$, where

$$\forall \sigma \in State_{|S}.\ \ \mu_{|S}(\sigma) = \Pr_{\sigma' \sim \mu}[\sigma = \sigma'_{|S}].$$

Here $State_{|S}$ and $\sigma'_{|S}$ both restrict their domains to S.

Lemma 13. *For all C, σ, σ', n, p and $S \subseteq PVar$, if $S \cap \mathsf{wv}(C) = \varnothing$, $p > 0$ and $(C, \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$, then $\sigma_{|S} = \sigma'_{|S}$.*

Proof. By induction on n. □

Lemma 14. *For all σ, σ', e and $S \subseteq PVar$, if $\sigma_{|S} = \sigma'_{|S}$ and $\mathsf{fv}(e) \subseteq S$, then $[\![e]\!]_{\sigma} = [\![e]\!]_{\sigma'}$.*

Proof. By induction on the structure of e. □

Lemma 15. *For all $\mu, b, C, \mathbf{E}, e$ and r, if*

$$\begin{aligned}\forall K \in \mathbb{N}.\ & |[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)| = 1 \wedge \\ & ([\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu) \vDash \lceil e \geq 0 \rceil \wedge \mathbb{E}[e] \leq r)\end{aligned}$$

and $\mathbf{modbf}(\mathbf{E}, e)$, then $|[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| = 1$, where

$$C_{\mathsf{W}} = \textbf{while}\ (b)\ \textbf{do}\ C,$$
$$C'_{\mathsf{W}}(K) = \textbf{while}\ (b \wedge e < K)\ \textbf{do}\ C.$$

Proof. We prove $|[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| = 1$ by contradiction. Assume that there exists some p_0 such that

$$|[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| = p_0 < 1.$$

Take

$$K_0 = \left\lceil \frac{1 + \max(r, 0)}{1 - p_0} \right\rceil,$$

then from Lem. 18 and $|[\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)| = 1$ we have

$$\begin{aligned} p_0 &= |[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| \\ &= \sum_{\sigma} \mu(\sigma) \sum_{\sigma'} [\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') \\ &\geq \sum_{\sigma} \mu(\sigma) \sum_{\sigma' : [\![e]\!]_{\sigma'} < K_0} [\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') \\ &\geq \sum_{\sigma} \mu(\sigma) \sum_{\sigma' : [\![e]\!]_{\sigma'} < K_0} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\sigma)(\sigma') \\ &= 1 - \sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_0} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)(\sigma'), \end{aligned}$$

and then from $[\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu) \vDash \lceil e \geq 0 \rceil$ we have

$$\begin{aligned} [\![\mathbb{E}[e]]\!]_{[\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)} &= \mathbb{E}_{\sigma' \sim [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)} [[\![e]\!]_{\sigma'}] \\ &= \sum_{\sigma'} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\ &= \sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_0} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\ &\quad + \sum_{\sigma' : [\![e]\!]_{\sigma'} < K_0} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\ &\geq \sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_0} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\ &\geq \left(\sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_0} [\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu)(\sigma') \right) \cdot K_0 \\ &\geq (1 - p_0) \cdot K_0 > r. \end{aligned}$$

This implies that

$$[\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu) \vDash \mathbb{E}[e] > r,$$

which contradicts $[\![\mathbf{E}[C'_{\mathsf{W}}(K_0)]]\!](\mu) \vDash \mathbb{E}[e] \leq r$. Thus we have $|[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| = 1$. □

Lemma 16. *For all $b, C, \mathbf{E}, K, e, \sigma, \sigma', n$ and $p > 0$, if $\mathsf{fv}(e) \cap \mathsf{wv}(\mathbf{E}) = \varnothing$, $[\![e]\!]_{\sigma'} < K$ and $(\mathbf{E}[C'_{\mathsf{W}}(K)], \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$, then $[\![e]\!]_{\sigma} < K$, where*

$$\begin{aligned} C_{\mathsf{W}} &= \mathbf{while}\ (b)\ \mathbf{do}\ C, \\ C'_{\mathsf{W}}(K) &= \mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C. \end{aligned}$$

Proof. We prove by induction on n and case analysis on $\mathbf{E}$.

$\mathbf{E} = [\,]$. If $[\![e]\!]_\sigma \in \Lambda$, then $(\mathbf{E}[C'_{\mathsf{W}}(K)], \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$ does not hold. If $[\![e]\!]_\sigma \geq K$, then $[\![e]\!]_{\sigma'} = [\![e]\!]_\sigma \geq K$, a contradiction. Thus $[\![e]\!]_\sigma < K$.

$\mathbf{E} = C'; \mathbf{E}'$. Know that there exist σ'', p_1, n_1, p_2 and $n_2 < n$ such that $p_1, p_2 > 0$, $(C', \sigma) \xrightarrow{p_1}{}^{n_1} (\mathbf{skip}, \sigma'')$ and $(\mathbf{E}'[C'_{\mathsf{W}}(K)], \sigma'') \xrightarrow{p_2}{}^{n_2} (\mathbf{skip}, \sigma')$. From the induction hypothesis, we have $[\![e]\!]_{\sigma''} < K$. Then, since $\mathsf{fv}(e) \cap \mathsf{wv}(C') = \varnothing$, by Lem. 13 and Lem. 14 we have $[\![e]\!]_\sigma = [\![e]\!]_{\sigma''} < K$.

$\mathbf{E} = \mathbf{E}'; C'$. Know that there exist σ'', p_1, p_2, n_2 and $n_1 < n$ such that $p_1, p_2 > 0$, $(\mathbf{E}'[C'_{\mathsf{W}}(K)], \sigma) \xrightarrow{p_1}{}^{n_1} (\mathbf{skip}, \sigma'')$ and $(C', \sigma'') \xrightarrow{p_2}{}^{n_2} (\mathbf{skip}, \sigma')$. Since $\mathsf{fv}(e) \cap \mathsf{wv}(C') = \varnothing$, by Lem. 13 and Lem. 14 we have $[\![e]\!]_{\sigma''} = [\![e]\!]_{\sigma'} < K$. Then from the induction hypothesis, we have $[\![e]\!]_\sigma < K$.

$\mathbf{E} = \mathbf{if}\ (b')\ \mathbf{then}\ C'\ \mathbf{else}\ \mathbf{E}'$. From $(\mathbf{E}[C'_{\mathsf{W}}(K)], \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$, we know that either $(C', \sigma) \xrightarrow{p}{}^{n-1} (\mathbf{skip}, \sigma')$ or $(\mathbf{E}'[C'_{\mathsf{W}}(K)], \sigma) \xrightarrow{p}{}^{n-1} (\mathbf{skip}, \sigma')$. For the former case, since $\mathsf{fv}(e) \cap \mathsf{wv}(C') = \varnothing$, by Lem. 13 and Lem. 14 we have $[\![e]\!]_\sigma = [\![e]\!]_{\sigma'} < K$. For the latter case, from the induction hypothesis we know that $[\![e]\!]_\sigma < K$.

$\mathbf{E} = \mathbf{if}\ (b')\ \mathbf{then}\ \mathbf{E}'\ \mathbf{else}\ C'$. This is similar to the previous case.

$\mathbf{E} = \mathbf{while}\ (b')\ \mathbf{do}\ \mathbf{E}'$. If $[\![b']\!]_\sigma = \text{false}$, then $[\![e]\!]_\sigma = [\![e]\!]_{\sigma'} < K$. If $[\![b']\!]_\sigma = \text{true}$, then there exist σ'', $p_1, p_2 > 0$ and $n_1, n_2 < n$ such that $(\mathbf{E}'[C'_{\mathsf{W}}(K)], \sigma) \xrightarrow{p_1}{}^{n_1} (\mathbf{skip}, \sigma'')$ and $(\mathbf{E}[C'_{\mathsf{W}}(K)], \sigma'') \xrightarrow{p_2}{}^{n_2} (\mathbf{skip}, \sigma')$. From the induction hypothesis, we know that $[\![e]\!]_{\sigma''} < K$, and then $[\![e]\!]_\sigma < K$.

$\square$

Lemma 17. *For all b, C, K, e, σ and σ', if $[\![e]\!]_{\sigma'} < K$, then*

$$[\![C_{\mathsf{W}}]\!](\sigma)(\sigma') \geq [\![C'_{\mathsf{W}}(K)]\!](\sigma)(\sigma'),$$

where

$$C_{\mathsf{W}} = \mathbf{while}\ (b)\ \mathbf{do}\ C,$$
$$C'_{\mathsf{W}}(K) = \mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C.$$

Proof. Let $[\![e]\!]_{\sigma'} < K$. From Lem. 40 (take $\mu = \delta(\sigma)$) and Lem. 12, for all σ, we have

$$[\![C_{\mathsf{W}}]\!](\sigma)(\sigma') = \lim_{n\to\infty} [\![C^n_{\mathsf{C}}; C_{\mathsf{CW}}]\!](\sigma)(\sigma'),$$
$$[\![C'_{\mathsf{W}}(K)]\!](\sigma)(\sigma') = \lim_{n\to\infty} [\![C^n_{\mathsf{CL}}; C_{\mathsf{CW}}(K)]\!](\sigma)(\sigma'),$$

where

$$\begin{aligned}
C_{\mathsf{C}} &= \mathbf{if}\ (b)\ \mathbf{then}\ C,\\
C^0_{\mathsf{C}} &= \mathbf{skip},\\
C^{n+1}_{\mathsf{C}} &= C^n_{\mathsf{C}}; C_{\mathsf{C}},\\
C_{\mathsf{CL}} &= \mathbf{if}\ (b \wedge e < K)\ \mathbf{then}\ C,\\
C^0_{\mathsf{CL}} &= \mathbf{skip},\\
C^{n+1}_{\mathsf{CL}} &= C^n_{\mathsf{CL}}; C_{\mathsf{CL}},\\
C_{\mathsf{CW}} &= \mathbf{if}\ (b)\ \mathbf{then}\ (\mathbf{while}\ (\text{true})\ \mathbf{do}\ \mathbf{skip}),\\
C_{\mathsf{CW}}(K) &= \mathbf{if}\ (b \wedge e < K)\ \mathbf{then}\ (\mathbf{while}\ (\text{true})\ \mathbf{do}\ \mathbf{skip}).
\end{aligned}$$

From Lem. 29, we know that, for all σ,

$$\begin{aligned}
[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') &= \sum_{\sigma''} [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{CW}}]\!](\sigma'')(\sigma') \\
&= [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}], \\
[\![C_{\mathsf{CL}}^n; C_{\mathsf{CW}}(K)]\!](\sigma)(\sigma') &= \sum_{\sigma''} [\![C_{\mathsf{CL}}^n]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{CW}}(K)]\!](\sigma'')(\sigma') \\
&= [\![C_{\mathsf{CL}}^n]\!](\sigma)(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}],
\end{aligned}$$

thus we only need to prove that, for all n and σ,

$$[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') \geq [\![C_{\mathsf{CL}}^n]\!](\sigma)(\sigma').$$

We prove by induction on n. The case of $n = 0$ is trivial. For $n = k + 1$, from Lem. 29 and the induction hypothesis we know that

$$\begin{aligned}
[\![C_{\mathsf{C}}^{k+1}]\!](\sigma)(\sigma') &= \sum_{\sigma''} [\![C_{\mathsf{C}}^k]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}]\!](\sigma'')(\sigma') \\
&\geq \sum_{\sigma'' : [\![e]\!]_{\sigma''} < K} [\![C_{\mathsf{C}}^k]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}]\!](\sigma'')(\sigma') \\
&\geq \sum_{\sigma'' : [\![e]\!]_{\sigma''} < K} [\![C_{\mathsf{CL}}^k]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{CL}}]\!](\sigma'')(\sigma') \\
&= \sum_{\sigma'' : [\![e]\!]_{\sigma''} < K} [\![C_{\mathsf{CL}}^k]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{CL}}]\!](\sigma'')(\sigma') \\
&\quad + \sum_{\sigma'' : [\![e]\!]_{\sigma''} \geq K} [\![C_{\mathsf{CL}}^k]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{CL}}]\!](\sigma'')(\sigma') \\
&= [\![C_{\mathsf{CL}}^{k+1}]\!](\sigma)(\sigma').
\end{aligned}$$

□

Lemma 18. *For all $b, C, \mathbf{E}, K, e, \sigma$ and σ', if $\mathbf{modbf}(\mathbf{E}, e)$ and $[\![e]\!]_{\sigma'} < K$, then*

$$[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') \geq [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma'),$$

where

$$\begin{aligned}
C_{\mathsf{W}} &= \textbf{while } (b) \textbf{ do } C, \\
C'_{\mathsf{W}}(K) &= \textbf{while } (b \wedge e < K) \textbf{ do } C.
\end{aligned}$$

Proof. Let $[\![e]\!]_{\sigma'} < K$. We prove by induction on the structure of $\mathbf{E}$.

$\mathbf{E} = [\,]$. This follows from Lem. 17.

$\mathbf{E} = C'; \mathbf{E}'$. From Lem. 29 and the induction hypothesis,

$$\begin{aligned}
[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') &= \sum_{\sigma''} [\![C']\!](\sigma)(\sigma'') \cdot [\![\mathbf{E}'[C_{\mathsf{W}}]]\!](\sigma'')(\sigma') \\
&\geq \sum_{\sigma''} [\![C']\!](\sigma)(\sigma'') \cdot [\![\mathbf{E}'[C'_{\mathsf{W}}(K)]]\!](\sigma'')(\sigma')
\end{aligned}$$

$$= [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma').$$

$\mathbf{E} = \mathbf{E}'; C'$. We have $\mathsf{fv}(e) \cap \mathsf{wv}(C') = \varnothing$. From Lem. 13 and Lem. 14, for all σ'' such that $[\![C']\!](\sigma'')(\sigma') > 0$, we have $[\![e]\!]_{\sigma''} = [\![e]\!]_{\sigma'} < K$. Then, from Lem. 29 and the induction hypothesis,

$$\begin{aligned}
&[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') \\
&= \sum_{\sigma''} [\![\mathbf{E}'[C_{\mathsf{W}}]]\!](\sigma)(\sigma'') \cdot [\![C']\!](\sigma'')(\sigma') \\
&= \sum_{\sigma'' : [\![e]\!]_{\sigma''} < K} [\![\mathbf{E}'[C_{\mathsf{W}}]]\!](\sigma)(\sigma'') \cdot [\![C']\!](\sigma'')(\sigma') \\
&\geq \sum_{\sigma'' : [\![e]\!]_{\sigma''} < K} [\![\mathbf{E}'[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma'') \cdot [\![C']\!](\sigma'')(\sigma') \\
&= \sum_{\sigma''} [\![\mathbf{E}'[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma'') \cdot [\![C']\!](\sigma'')(\sigma') \\
&= [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma').
\end{aligned}$$

$\mathbf{E} = \mathbf{if}\ (b')\ \mathbf{then}\ C'\ \mathbf{else}\ \mathbf{E}'$. If $[\![b']\!]_\sigma = \text{true}$, then from Lem. 26 we have

$$[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') = [\![C']\!](\sigma)(\sigma') = [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma').$$

If $[\![b']\!]_\sigma = \text{false}$, then from Lem. 26 and the induction hypothesis we have

$$\begin{aligned}
[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') &= [\![\mathbf{E}'[C_{\mathsf{W}}]]\!](\sigma)(\sigma') \\
&\geq [\![\mathbf{E}'[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma') \\
&= [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma').
\end{aligned}$$

$\mathbf{E} = \mathbf{if}\ (b')\ \mathbf{then}\ \mathbf{E}'\ \mathbf{else}\ C'$. This is similar to the previous case.

$\mathbf{E} = \mathbf{while}\ (b')\ \mathbf{do}\ \mathbf{E}'$. We use the following notations:

$$\begin{aligned}
C_{\mathsf{C}} &\triangleq \mathbf{if}\ (b')\ \mathbf{then}\ \mathbf{E}'[C_{\mathsf{W}}] \\
C_{\mathsf{C}}^0 &\triangleq \mathbf{skip} \\
C_{\mathsf{C}}^{n+1} &\triangleq C_{\mathsf{C}}^n; C_{\mathsf{C}} \\
C_{\mathsf{CL}} &\triangleq \mathbf{if}\ (b')\ \mathbf{then}\ \mathbf{E}'[C'_{\mathsf{W}}(K)] \\
C_{\mathsf{CL}}^0 &\triangleq \mathbf{skip} \\
C_{\mathsf{CL}}^{n+1} &\triangleq C_{\mathsf{CL}}^n; C_{\mathsf{CL}} \\
C_{\mathsf{CW}} &\triangleq \mathbf{if}\ (b')\ \mathbf{then}\ (\mathbf{while}\ (\text{true})\ \mathbf{do\ skip})
\end{aligned}$$

From Lem. 40 (take $\mu = \delta(\sigma)$) and Lem. 12, for all σ, we have the following:

$$\begin{aligned}
[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') &= \lim_{n\to\infty} [\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \\
[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma') &= \lim_{n\to\infty} [\![C_{\mathsf{CL}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma')
\end{aligned}$$

Thus we only need to prove that, for all n and σ,

$$[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \geq [\![C_{\mathsf{CL}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma').$$

From $\mathbf{modbf}(\mathbf{E}, e)$, we have $\mathsf{fv}(e) \cap \mathsf{wv}(\mathbf{E}') = \varnothing$, which implies that $\mathbf{modbf}(\mathbf{if}\ (b')\ \mathbf{then}\ \mathbf{E}', e)$. Note that for all σ_1 and σ_2 such that $[\![e]\!]_{\sigma_2} < K$, similar to the previous case, from the induction hypothesis we can prove that $[\![C_{\mathsf{CL}}]\!](\sigma_1)(\sigma_2) \leq [\![C_{\mathsf{C}}]\!](\sigma_1)(\sigma_2)$. Moreover, if $[\![C_{\mathsf{CL}}]\!](\sigma_1)(\sigma_2) > 0$, from $\mathsf{fv}(e) \cap \mathsf{wv}(\mathbf{if}\ (b')\ \mathbf{then}\ \mathbf{E}') = \varnothing$ and Lem. 16 we have $[\![e]\!]_{\sigma_1} < K$. Hence, by repeatedly using the above two properties, we have

$$\begin{aligned}
&[\![C_{\mathsf{CL}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \\
&= \sum_{\sigma_1,\ldots,\sigma_n} [\![C_{\mathsf{CL}}]\!](\sigma)(\sigma_1)\cdots[\![C_{\mathsf{CL}}]\!](\sigma_{n-1})(\sigma_n)\cdot[\![C_{\mathsf{CW}}]\!](\sigma_n)(\sigma') \\
&= \sum_{\substack{\sigma_1,\ldots,\sigma_n \\ [\![e]\!]_{\sigma_n} < K}} [\![C_{\mathsf{CL}}]\!](\sigma)(\sigma_1)\cdots[\![C_{\mathsf{CL}}]\!](\sigma_{n-1})(\sigma_n)\cdot[\![C_{\mathsf{CW}}]\!](\sigma_n)(\sigma') \\
&\leq \sum_{\substack{\sigma_1,\ldots,\sigma_n \\ [\![e]\!]_{\sigma_{n-1}} < K}} [\![C_{\mathsf{CL}}]\!](\sigma)(\sigma_1)\cdots[\![C_{\mathsf{C}}]\!](\sigma_{n-1})(\sigma_n)\cdot[\![C_{\mathsf{CW}}]\!](\sigma_n)(\sigma') \\
&\leq \cdots \\
&\leq \sum_{\substack{\sigma_1,\ldots,\sigma_n \\ [\![e]\!]_{\sigma_1} < K}} [\![C_{\mathsf{CL}}]\!](\sigma)(\sigma_1)\cdots[\![C_{\mathsf{C}}]\!](\sigma_{n-1})(\sigma_n)\cdot[\![C_{\mathsf{CW}}]\!](\sigma_n)(\sigma') \\
&\leq \sum_{\sigma_1,\ldots,\sigma_n} [\![C_{\mathsf{C}}]\!](\sigma)(\sigma_1)\cdots[\![C_{\mathsf{C}}]\!](\sigma_{n-1})(\sigma_n)\cdot[\![C_{\mathsf{CW}}]\!](\sigma_n)(\sigma') \\
&= [\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma').
\end{aligned}$$

□

Lemma 19. *For all $\mu, b, C, \mathbf{E}, e$ and r, if*

$$\begin{aligned}\forall K \in \mathbb{N}.\ \ &|[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)| = 1 \wedge \\ &([\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu) \vDash \lceil e \geq 0\rceil \wedge \mathbb{E}[e] \leq r)\end{aligned}$$

and $\mathbf{modbf}(\mathbf{E}, e)$*, then*

$$[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu) = \lim_{K\to\infty}[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu), \tag{13}$$

where

$$\begin{aligned} C_{\mathsf{W}} &= \mathbf{while}\ (b)\ \mathbf{do}\ C, \\ C'_{\mathsf{W}}(K) &= \mathbf{while}\ (b \wedge e < K)\ \mathbf{do}\ C. \end{aligned}$$

Proof. By applying Lem. 15, we have

$$|[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)| = 1. \tag{14}$$

For all σ, σ' and K such that $[\![e]\!]_{\sigma'} < K$, from Lem. 18 we have

$$[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma') \leq [\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma'). \tag{15}$$

For all K, from $|[\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)| = 1$, (14) and (15) we have

$$\begin{aligned}
&\sum_{\sigma'} |[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\mu)(\sigma') - [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)(\sigma')| \\
&= \sum_{\sigma'} \left| \sum_{\sigma} \mu(\sigma) \left([\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') - [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma')\right) \right| \\
&\le \sum_{\sigma'} \sum_{\sigma} \mu(\sigma) \, |[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') - [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma')| \\
&= \sum_{\sigma} \mu(\sigma) \left(\sum_{\sigma' : [\![e]\!]_{\sigma'} \not< K} |[\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') - [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma')| + \right. \\
&\qquad \left. \sum_{\sigma' : [\![e]\!]_{\sigma'} < K} \left([\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') - [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma')\right) \right) \\
&\le \sum_{\sigma} \mu(\sigma) \left(\sum_{\sigma' : [\![e]\!]_{\sigma'} \not< K} \left([\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') + [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma')\right) + \right. \\
&\qquad \left. 1 - \sum_{\sigma' : [\![e]\!]_{\sigma'} < K} [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\sigma)(\sigma') \right) \\
&= \sum_{\sigma} \mu(\sigma) \left(1 - \sum_{\sigma' : [\![e]\!]_{\sigma'} < K} [\![\mathbf{E}[C_{\mathsf{W}}]]\!](\sigma)(\sigma') \right) \\
&\quad + 2 \left(1 - \sum_{\sigma' : [\![e]\!]_{\sigma'} < K} [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)(\sigma') \right) \\
&\le 3 \left(1 - \sum_{\sigma' : [\![e]\!]_{\sigma'} < K} [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)(\sigma') \right).
\end{aligned}$$

Let

$$f_K = 1 - \sum_{\sigma' : [\![e]\!]_{\sigma'} < K} [\![\mathbf{E}[C'_{\mathsf{W}}(K)]]\!](\mu)(\sigma').$$

Then we only need to prove that

$$\lim_{K \to \infty} f_K = 0. \tag{16}$$

We prove (16) by contradiction. Assume that (16) does not hold, then there exists some $r_1 > 0$ such that, for all K, there exists $K' > K$ such that $f_{K'} \ge r_1$. Take

$$K_1 = \left\lceil \frac{1 + \max(r, 0)}{r_1} \right\rceil$$

and $K_1' > K_1$ such that $f_{K_1'} \geq r_1$. Then, from $|[\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)| = 1$ and $[\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu) \vDash \lceil e \geq 0 \rceil$, we have

$$\begin{aligned}
[\![\mathbb{E}[e]]\!]_{[\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)} &= \sum_{\sigma'} [\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\
&= \sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_1'} [\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\
&\quad + \sum_{\sigma' : [\![e]\!]_{\sigma'} < K_1'} [\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\
&\geq \sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_1'} [\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)(\sigma') \cdot [\![e]\!]_{\sigma'} \\
&\geq \left(\sum_{\sigma' : [\![e]\!]_{\sigma'} \geq K_1'} [\![\mathbf{E}[C'_{\mathsf{W}}(K_1')]]\!](\mu)(\sigma') \right) \cdot K_1' \\
&= f_{K_1'} \cdot K_1' \\
&> f_{K_1'} \cdot K_1 \\
&\geq r_1 \cdot K_1 > r,
\end{aligned}$$

which implies

$$[\![\mathbf{E}[C'_{\mathsf{W}}(K_1)]]\!](\mu) \vDash \mathbb{E}[e] > r,$$

but this contradicts the premise that $[\![\mathbf{E}[C'_{\mathsf{W}}(K_1)]]\!] \vDash \mathbb{E}[e] \leq r$. Thus (16) holds. □

D Soundness of RT-Based Coupling

Below we prove the soundness of our relational proof recipe, the resampling-table-based coupling (Thm. 3). We also give an extension of Thm. 3 at the end of this section.

Proof of Lem. 3. Directly from the definitions and Thm. 1. □

Lemma 20. *For all* $\mathbf{p}, C_1, C_2, \mathbf{q}_1, \mathbf{R}, \mathbf{q}_2$, *if*

- $\mathbf{RTonly}(\mathbf{R})$;
- $\vDash_{\mathrm{RT}} \{\mathbf{p} \wedge \mathsf{hdinit}\} C_1 \{\mathbf{q}_1 \Rightarrow \mathbf{R}\}$;
- $\vDash_{\mathrm{RT}} [\mathbf{p} \wedge \mathbf{R} \wedge \mathsf{hdinit}] C_2 [\mathbf{q}_2]$;

then $\vDash_{\mathrm{RT}} \{\lceil \mathbf{p} \rceil\} C_1 \leq C_2 \{\mathbf{q}_1, \mathbf{q}_2\}$.

Proof. Let $\mu \vDash \lceil \mathbf{p} \rceil$. Below we prove that

$$\sum_{\sigma' : \sigma' \vDash \mathbf{q}_1} \sum_{\sigma} \mu(\sigma) [\![C_1]\!]_{\mathrm{RT}}(\sigma)(\sigma') \leq \sum_{\sigma' : \sigma' \vDash \mathbf{q}_2} \sum_{\sigma} \mu(\sigma) [\![C_2]\!]_{\mathrm{RT}}(\sigma)(\sigma').$$

Let σ be a state with $\mu(\sigma) > 0$ below. From $\mu \models \lceil \mathbf{p} \rceil$, we know that $\sigma \models \mathbf{p}$. Then we only need to prove that

$$\sum_{\sigma' : \sigma' \models \mathbf{q}_1} [\![C_1]\!]_{\mathrm{RT}}(\sigma)(\sigma') \leq \sum_{\sigma' : \sigma' \models \mathbf{q}_2} [\![C_2]\!]_{\mathrm{RT}}(\sigma)(\sigma').$$

For σ', RT and ι' satisfying $\sigma' \models \mathbf{q}_1$ and

$$RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \iota'),$$

since $(\sigma, RT, \iota_{\mathsf{init}}) \models \mathbf{p} \wedge \mathsf{hdinit}$, we know that $(\sigma', RT, \iota') \models \mathbf{q}_1 \Rightarrow \mathbf{R}$ from the premise, and thus $(\sigma', RT, \iota') \models \mathbf{R}$. From $\mathbf{RTonly}(\mathbf{R})$, we have $RT \models \mathbf{R}$, thus $(\sigma, RT, \iota_{\mathsf{init}}) \models \mathbf{R}$ and $(\sigma, RT, \iota_{\mathsf{init}}) \models \mathbf{p} \wedge \mathbf{R} \wedge \mathsf{hdinit}$. Then, from the premise, there exists σ'' such that $\sigma'' \models \mathbf{q}_2$ and

$$RT \vdash (C_2, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma'', \_).$$

Now, since $\{\sigma' \mid [\![C]\!]_{\mathrm{RT}}(\sigma)(\sigma') > 0\}$ is countable for $C = C_1, C_2$, we have

$$\begin{aligned}
&\sum_{\sigma' : \sigma' \models \mathbf{q}_1} [\![C_1]\!]_{\mathrm{RT}}(\sigma)(\sigma') \\
&= \sum_{\sigma' : \sigma' \models \mathbf{q}_1} \mathcal{M}(\{RT \mid RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\}) \\
&= \mathcal{M}\left(\biguplus_{\sigma' : \sigma' \models \mathbf{q}_1} \{RT \mid RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\} \right) \\
&= \mathcal{M}(\{RT \mid \exists \sigma'.\ \sigma' \models \mathbf{q}_1 \wedge RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\}) \\
&\leq \mathcal{M}(\{RT \mid \exists \sigma'.\ \sigma' \models \mathbf{q}_2 \wedge RT \vdash (C_2, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\}) \\
&= \sum_{\sigma' : \sigma' \models \mathbf{q}_2} \mathcal{M}(\{RT \mid RT \vdash (C_2, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\}) \\
&= \sum_{\sigma' : \sigma' \models \mathbf{q}_2} [\![C_2]\!]_{\mathrm{RT}}(\sigma)(\sigma').
\end{aligned}$$

□

Proof of Thm. 3. Directly from Lem. 3 and Lem. 20. □

We further give Thm. 5, an extension of Thm. 3, though it is not used in our verification of ALLLs. The probability space of all resampling tables used in Thm. 5, $(\Omega, \mathcal{F}, \mathcal{M})$, is defined in Sec. 4.2.

To apply Thm. 5, it is required to find two intermediate assertions $\mathbf{R}_1$ and $\mathbf{R}_2$, and to prove a inequality between measures of two sets of resampling tables that satisfy $\mathbf{R}_1$ and $\mathbf{R}_2$ respectively. This theorem can roughly be regarded as an extension of Thm. 3, since it degenerates to Thm. 3 when $\mathbf{R}_1 = \mathbf{R}_2 = \mathbf{R}$ and $\{RT \mid RT \models \mathbf{R}\} \in \mathcal{F}$ is provided.

Theorem 5. *For all* $\mathbf{p}, C_1, C_2, \mathbf{q}_1, \mathbf{R}_1, \mathbf{R}_2$ *and* $\mathbf{q}_2$*, if*

– $\mathbf{RTonly}(\mathbf{R}_1)$;
– $\{RT \mid RT \vDash \mathbf{R}_1\}, \{RT \mid RT \vDash \mathbf{R}_2\} \in \mathcal{F}$;
– $\mathcal{M}(\{RT \mid RT \vDash \mathbf{R}_1\}) \leq \mathcal{M}(\{RT \mid RT \vDash \mathbf{R}_2\})$;
– $\vDash_{\mathrm{RT}} \{\mathbf{p} \wedge \mathsf{hdinit}\}C_1\{\mathbf{q}_1 \Rightarrow \mathbf{R}_1\}$;
– $\vDash_{\mathrm{RT}} [\mathbf{p} \wedge \mathbf{R}_2 \wedge \mathsf{hdinit}]C_2[\mathbf{q}_2]$;

then

$$\vDash \{\lceil \mathbf{p} \rceil\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}.$$

Proof. From Lem. 3, we only need to prove that

$$\vDash_{\mathrm{RT}} \{\lceil \mathbf{p} \rceil\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}.$$

Let $\mu \vDash \lceil \mathbf{p} \rceil$. Below we prove that

$$\sum_{\sigma' : \sigma' \vDash \mathbf{q}_1} \sum_{\sigma} \mu(\sigma) [\![C_1]\!]_{\mathrm{RT}}(\sigma)(\sigma') \leq \sum_{\sigma' : \sigma' \vDash \mathbf{q}_2} \sum_{\sigma} \mu(\sigma) [\![C_2]\!]_{\mathrm{RT}}(\sigma)(\sigma').$$

Let σ be a state with $\mu(\sigma) > 0$ below. From $\mu \vDash \lceil \mathbf{p} \rceil$, we know that $\sigma \vDash \mathbf{p}$. Then we only need to prove that

$$\sum_{\sigma' : \sigma' \vDash \mathbf{q}_1} [\![C_1]\!]_{\mathrm{RT}}(\sigma)(\sigma') \leq \sum_{\sigma' : \sigma' \vDash \mathbf{q}_2} [\![C_2]\!]_{\mathrm{RT}}(\sigma)(\sigma').$$

For σ', RT and ι' such that $\sigma' \vDash \mathbf{q}_1$ and

$$RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma', \iota'),$$

since $(\sigma, RT, \iota_{\mathsf{init}}) \vDash \mathbf{p} \wedge \mathsf{hdinit}$, we know that $(\sigma', RT, \iota') \vDash \mathbf{q}_1 \Rightarrow \mathbf{R}_1$ from the premise, and thus $(\sigma', RT, \iota') \vDash \mathbf{R}_1$. From $\mathbf{RTonly}(\mathbf{R}_1)$, we have $RT \vDash \mathbf{R}_1$. From another perspective, suppose that $RT \vDash \mathbf{R}_2$, we know that $(\sigma, RT, \iota_{\mathsf{init}}) \vDash \mathbf{R}_2$ and $(\sigma, RT, \iota_{\mathsf{init}}) \vDash \mathbf{p} \wedge \mathbf{R}_2 \wedge \mathsf{hdinit}$, and thus from the premise there exists σ'' such that $\sigma'' \vDash \mathbf{q}_2$ and

$$RT \vdash (C_2, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma'', \_).$$

Now, since $\{\sigma' \mid [\![C]\!]_{\mathrm{RT}}(\sigma)(\sigma') > 0\}$ is countable for $C = C_1, C_2$, we have

$$\begin{aligned}
&\sum_{\sigma' : \sigma' \vDash \mathbf{q}_1} [\![C_1]\!]_{\mathrm{RT}}(\sigma)(\sigma') \\
&= \sum_{\sigma' : \sigma' \vDash \mathbf{q}_1} \mathcal{M}(\{RT \mid RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma', \_)\}) \\
&= \mathcal{M}\left(\biguplus_{\sigma' : \sigma' \vDash \mathbf{q}_1} \{RT \mid RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma', \_)\} \right) \\
&= \mathcal{M}(\{RT \mid \exists \sigma'.\ \sigma' \vDash \mathbf{q}_1 \wedge RT \vdash (C_1, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma', \_)\}) \\
&\leq \mathcal{M}(\{RT \mid RT \vDash \mathbf{R}_1\}) \\
&\leq \mathcal{M}(\{RT \mid RT \vDash \mathbf{R}_2\})
\end{aligned}$$

$$\frac{}{\vdash [Q[e/x]]x := e[Q]} \quad (\textsc{var-t})$$

$$\frac{x \notin \mathsf{fv}(S) \cup \mathsf{fv}(e) \cup \mathsf{fv}(Q) \qquad X \notin \mathsf{fv}(e) \qquad \vDash Q \Rightarrow (\exists X.\ \lceil e = X \rceil)}{\vdash [Q \wedge \#S]x := \mathsf{Sample}(e)[Q \wedge \#(S \cup \{x\}) \wedge x \sim e]} \quad (\textsc{smp-t})$$

$$\frac{\vDash P_1 \Rightarrow P_2 \qquad \vdash [P_2]C[Q_2] \qquad \vDash Q_2 \Rightarrow Q_1}{\vdash [P_1]C[Q_1]} \quad (\textsc{csq-t})$$

$$\frac{\vdash [P]C_1[Q] \qquad \vdash [Q]C_2[R]}{\vdash [P]C_1; C_2[R]} \quad (\textsc{seq-t}) \qquad \frac{}{\vdash [Q]\mathbf{skip}[Q]} \quad (\textsc{skip-t})$$

$$\frac{\vdash [P_1 \wedge \lceil b \rceil]C_1[Q_1] \qquad \vdash [P_2 \wedge \lceil \neg b \rceil]C_2[Q_2]}{\vdash [(P_1 \wedge \lceil b \rceil) \oplus_p (P_2 \wedge \lceil \neg b \rceil)]\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2[Q_1 \oplus_p Q_2]} \quad (\textsc{cond-t})$$

$$\frac{\begin{array}{c}\vdash [P \wedge \lceil b \wedge e = X \rceil]C[(P \wedge \lceil b \wedge e + 1 \leq X \rceil) \vee (Q \wedge \lceil \neg b \rceil)] \\ \vDash P \wedge \lceil b \rceil \Rightarrow (\exists X'.\ \lceil 0 \leq e \leq X' \rceil) \qquad X' \notin \mathsf{fv}(e) \\ X \notin \mathsf{fv}(P) \cup \mathsf{fv}(Q) \cup \mathsf{fv}(b) \cup \mathsf{fv}(e) \cup \mathsf{fv}(C)\end{array}}{\vdash [(P \wedge \lceil b \rceil) \vee (Q \wedge \lceil \neg b \rceil)]\mathbf{while}\ (b)\ \mathbf{do}\ C[Q \wedge \lceil \neg b \rceil]} \quad (\textsc{while-t})$$

$$\frac{\forall i \in [0, n).\ \vdash [Q_i]\ \mathbf{if}\ (b)\ \mathbf{then}\ C[Q_{i+1}] \qquad \vDash Q_n \Rightarrow \lceil \neg b \rceil}{\vdash [Q_0]\ \mathbf{while}\ (b)\ \mathbf{do}\ C[Q_n]} \quad (\textsc{while-tb})$$

$$\frac{\vdash [P_1]C[Q_1] \qquad \vdash [P_2]C[Q_2]}{\vdash [P_1 \wedge P_2]C[Q_1 \wedge Q_2]} \quad (\textsc{conj-t}) \qquad \frac{\vdash [P_1]C[Q_1] \qquad \vdash [P_2]C[Q_2]}{\vdash [P_1 \vee P_2]C[Q_1 \vee Q_2]} \quad (\textsc{disj-t})$$

$$\frac{\vdash [P]C[Q] \qquad X \notin \mathsf{fv}(C)}{\vdash [\exists X.\, P]C[\exists X.\, Q]} \quad (\textsc{exists-t}) \qquad \frac{\vdash [P]C[Q] \qquad X \notin \mathsf{fv}(C)}{\vdash [\forall X.\, P]C[\forall X.\, Q]} \quad (\textsc{forall-t})$$

Fig. 24. Selected rules of the probabilistic program logic.

$$\begin{aligned}
&\leq \mathcal{M}(\{RT \mid \exists \sigma'.\ \sigma' \vDash \mathbf{q}_2 \wedge RT \vdash (C_2, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma', \_)\}) \\
&= \sum_{\sigma' : \sigma' \vDash \mathbf{q}_2} \mathcal{M}(\{RT \mid RT \vdash (C_2, \sigma, \iota_{\mathsf{init}}) \rightarrow^* (\mathbf{skip}, \sigma', \_)\}) \\
&= \sum_{\sigma' : \sigma' \vDash \mathbf{q}_2} [\![C_2]\!]_{\mathrm{RT}}(\sigma)(\sigma').
\end{aligned}$$

□

E A Probabilistic Program Logic

We adapt the assertion-based program logic from [5] to prove some intermediate proof goals occurring in the verification of ALLLs and other examples. We prove the soundness of this program logic. Logic rules of this program logic are presented in Fig. 24.

For set A and $a \in A$, the Dirac distribution $\delta(a) \in \mathbb{D}_A$ is defined as follows:

$$\delta(a) \triangleq \lambda b. \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}.$$

Theorem 6. *For all P, C and Q,*

$$\vdash [P]C[Q] \implies \vDash [P]C[Q].$$

Proof. By Lem. 30, Lem. 31, Lem. 32, Lem. 33, Lem. 34, Lem. 37, Lem. 42, Lem. 43, Lem. 44, Lem. 45, Lem. 46 and Lem. 47. □

Lemma 21. *For all μ, Q, e and x, if $\mu \vDash Q[e/x]$, then $\mu' \vDash Q$, where $\mu' = \mathbb{E}_{\sigma\sim\mu}\{\delta(\sigma\{x \rightsquigarrow [\![e]\!]_\sigma\})\}$.*

Proof. By induction on the structure of Q. □

Lemma 22. *For all $\sigma, \mathbf{q}$ and x, if $x \notin \mathsf{fv}(\mathbf{q})$, then for all v we have*

$$\sigma \vDash \mathbf{q} \iff \sigma\{x \rightsquigarrow v\} \vDash \mathbf{q}.$$

Proof. By induction on the structure of $\mathbf{q}$. □

Lemma 23. *For all μ, Q and x, if $x \notin \mathsf{fv}(Q)$, then for all v we have*

$$\mu \vDash Q \iff \mu\{x \rightsquigarrow v\} \vDash Q.$$

Proof. By induction on the structure of Q. □

Lemma 24. *For all $C, \sigma, \sigma', n, n'$ and p, if*

- *$n' \geq n$;*
- *$(C, \sigma) \xrightarrow{p}{}^{n} (\mathbf{skip}, \sigma')$;*

then there exists a unique p' such that

- *$p' \geq p$;*
- *$(C, \sigma) \xrightarrow{p'}{}^{n'} (\mathbf{skip}, \sigma')$.*

Proof. Prove by induction on n. □

Lemma 25. *For all σ,*

$$[\![\mathbf{skip}]\!](\sigma) = \delta(\sigma).$$

Proof. Directly from the definition. □

Lemma 26. *For all C, σ and σ',*

$$[\![C]\!](\sigma)(\sigma') = \sum_{C'',\sigma''} \{p \cdot [\![C'']\!](\sigma'')(\sigma') \mid (C, \sigma) \xrightarrow{p} (C'', \sigma'')\}.$$

Proof. Define $p_{n,C,\sigma}$ such that $(C, \sigma) \xrightarrow{p_{n,C,\sigma}}{}^{n} (\mathbf{skip}, \sigma')$. By definition, for all n we have

$$p_{n+1,C,\sigma} = \sum_{C'',\sigma''} \{p \cdot p_{n,C'',\sigma''} \mid (C, \sigma) \xrightarrow{p} (C'', \sigma'')\},$$

and thus by Lem. 24 and the monotone convergence theorem we have

$$\begin{aligned}
[\![C]\!](\sigma)(\sigma') &= \lim_{n\to\infty} p_{n+1,C,\sigma} \\
&= \lim_{n\to\infty} \sum_{C'',\sigma''} \{p \cdot p_{n,C'',\sigma''} \mid (C,\sigma) \xrightarrow{p} (C'',\sigma'')\} \\
&= \sum_{C'',\sigma''} \{p \cdot \lim_{n\to\infty} p_{n,C'',\sigma''} \mid (C,\sigma) \xrightarrow{p} (C'',\sigma'')\} \\
&= \sum_{C'',\sigma''} \{p \cdot [\![C'']\!](\sigma'')(\sigma') \mid (C,\sigma) \xrightarrow{p} (C'',\sigma'')\}.
\end{aligned}$$

□

Lemma 27. *For all x, e, σ and σ',*

$$[\![x := \mathsf{Sample}(e)]\!](\sigma)(\sigma') = \begin{cases} \mathcal{D}[i](r) & \textit{if } [\![e]\!]_\sigma = i \in [1,N] \textit{ and } \sigma' = \sigma\{x \rightsquigarrow r\} \\ 0 & \text{otherwise} \end{cases}.$$

Proof. Directly from Lem. 25 and Lem. 26. □

Lemma 28. *For all n, C_1, C_2, σ and σ',*

$$p_{n,C_1;C_2,\sigma,\sigma'} \le \sum_{\sigma''} p_{n,C_1,\sigma,\sigma''} \cdot p_{n,C_2,\sigma'',\sigma'}.$$

where $p_{n,C,\sigma,\sigma'}$ satisfies $(C,\sigma) \xrightarrow{p_{n,C,\sigma,\sigma'}}{}^{n} (\mathbf{skip},\sigma')$.

Proof. We prove by induction on n. The case of $n = 0$ is trivial. For $n = k+1$, if $C_1 = \mathbf{skip}$, then by Lem. 24 we have

$$\begin{aligned}
p_{n+1,\mathbf{skip};C_2,\sigma,\sigma'} &= p_{n,C_2,\sigma,\sigma'} \\
&\le p_{n+1,C_2,\sigma,\sigma'} \\
&= p_{n+1,\mathbf{skip},\sigma,\sigma} \cdot p_{n+1,C_2,\sigma,\sigma'} \\
&= \sum_{\sigma''} p_{n+1,\mathbf{skip},\sigma,\sigma''} \cdot p_{n+1,C_2,\sigma'',\sigma'}.
\end{aligned}$$

If $C_1 \ne \mathbf{skip}$, then from the induction hypothesis and Lem. 24, we have

$$\begin{aligned}
& p_{n+1,C_1;C_2,\sigma,\sigma'} \\
&= \sum_{C'',\sigma''} \{p \cdot p_{n,C'',\sigma'',\sigma'} \mid (C_1;C_2,\sigma) \xrightarrow{p} (C'',\sigma'')\} \\
&= \sum_{C_1'',\sigma''} \{p \cdot p_{n,C_1'';C_2,\sigma'',\sigma'} \mid (C_1,\sigma) \xrightarrow{p} (C_1'',\sigma'')\} \\
&\le \sum_{C_1'',\sigma'',\sigma'''} \{p \cdot p_{n,C_1'',\sigma'',\sigma'''} \cdot p_{n,C_2,\sigma''',\sigma'} \mid (C_1,\sigma) \xrightarrow{p} (C_1'',\sigma'')\}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\sigma'''} p_{n+1,C_1,\sigma,\sigma'''} \cdot p_{n,C_2,\sigma''',\sigma'} \\
&\leq \sum_{\sigma'''} p_{n+1,C_1,\sigma,\sigma'''} \cdot p_{n+1,C_2,\sigma''',\sigma'}.
\end{aligned}
$$

□

Lemma 29. *For all C_1, C_2, σ and σ',*

$$
[\![C_1; C_2]\!](\sigma)(\sigma') = \sum_{\sigma''} [\![C_1]\!](\sigma)(\sigma'') \cdot [\![C_2]\!](\sigma'')(\sigma').
$$

Proof. The case of $C_1 = \textbf{skip}$ is straightforward from Lem. 25 and Lem. 26. Define $p_{n,C,\sigma,\sigma'}$ such that $(C, \sigma) \xrightarrow{p_{n,C,\sigma,\sigma'}}{}^{n} (\textbf{skip}, \sigma')$.

First, we prove the following: for all $C_1 \neq \textbf{skip}, \sigma$ and σ',

$$
[\![C_1; C_2]\!](\sigma)(\sigma') \leq \sum_{\sigma''} [\![C_1]\!](\sigma)(\sigma'') \cdot [\![C_2]\!](\sigma'')(\sigma'). \tag{17}
$$

From Lem. 28, for all n, we have

$$
p_{n,C_1;C_2,\sigma,\sigma'} \leq \sum_{\sigma''} p_{n,C_1,\sigma,\sigma''} \cdot p_{n,C_2,\sigma'',\sigma'}, \tag{18}
$$

and thus from Lem. 24 we know that

$$
\begin{aligned}
&\lim_{n\to\infty} \sum_{\sigma''} p_{n,C_1,\sigma,\sigma''} \cdot p_{n,C_2,\sigma'',\sigma'} \\
&= \sum_{\sigma''} \lim_{n\to\infty} p_{n,C_1,\sigma,\sigma''} \cdot p_{n,C_2,\sigma'',\sigma'} \\
&= \sum_{\sigma''} \left(\lim_{n\to\infty} p_{n,C_1,\sigma,\sigma''}\right) \cdot \left(\lim_{n\to\infty} p_{n,C_2,\sigma'',\sigma'}\right) \\
&= \sum_{\sigma''} [\![C_1]\!](\sigma)(\sigma'') \cdot [\![C_2]\!](\sigma'')(\sigma').
\end{aligned}
$$

Then we get (17) by taking $n \to \infty$ in (18).

We then prove the following: for all $n, C_1 \neq \textbf{skip}, \sigma$ and σ',

$$
\sum_{\sigma''} p_{n,C_1,\sigma,\sigma''} \cdot [\![C_2]\!](\sigma'')(\sigma') \leq [\![C_1; C_2]\!](\sigma)(\sigma'). \tag{19}
$$

We prove by induction on n. The case of $n = 0$ is trivial. For $n = k + 1$, from Lem. 26 and the induction hypothesis, we have

$$
\begin{aligned}
&\sum_{\sigma''} p_{n,C_1,\sigma,\sigma''} \cdot [\![C_2]\!](\sigma'')(\sigma') \\
&= \sum_{\sigma''} \left(\sum_{C_1''',\sigma'''} \left\{ p \cdot p_{k,C_1''',\sigma''',\sigma''} \;\middle|\; (C_1, \sigma) \xrightarrow{p} (C_1''', \sigma''') \right\} \right) \cdot [\![C_2]\!](\sigma'')(\sigma')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{C_1''',\sigma'''} \left\{ p \cdot \left(\sum_{\sigma''} p_{k,C_1''',\sigma''',\sigma''} \cdot [\![C_2]\!](\sigma'')(\sigma') \right) \middle| (C_1,\sigma) \xrightarrow{p} (C_1''',\sigma''') \right\} \\
&\leq \sum_{C_1''',\sigma'''} \left\{ p \cdot [\![C_1''';C_2]\!](\sigma''')(\sigma') \middle| (C_1,\sigma) \xrightarrow{p} (C_1''',\sigma''') \right\} \\
&= \sum_{C''',\sigma'''} \left\{ p \cdot [\![C''']\!](\sigma''')(\sigma') \middle| (C_1;C_2,\sigma) \xrightarrow{p} (C''',\sigma''') \right\} \\
&= [\![C_1;C_2]\!](\sigma)(\sigma').
\end{aligned}
$$

Thus (19) holds. Taking $n \to \infty$ in (19), by Lem. 24 we have

$$[\![C_1;C_2]\!](\sigma)(\sigma') \geq \sum_{\sigma''} [\![C_1]\!](\sigma)(\sigma'') \cdot [\![C_2]\!](\sigma'')(\sigma').$$

With (17) we complete the proof. □

Lemma 30 (Var-T-Sound). *For all Q, e and x,*

$$\models [Q[e/x]]x := e[Q].$$

Proof. Let $\mu \models Q[e/x]$ and $\mu' = \mathbb{E}_{\sigma\sim\mu}\{\delta(\sigma\{x \rightsquigarrow [\![e]\!]_\sigma\})\}$. By Lem. 21 we know that $\mu' \models Q$. For all σ', by Lem. 25 and Lem. 26 we know that

$$
\begin{aligned}
[\![x := e]\!](\mu)(\sigma') &= \sum_\sigma \mu(\sigma) \cdot [\![x := e]\!](\sigma)(\sigma') \\
&= \sum_\sigma \mu(\sigma) \cdot [\sigma\{x \rightsquigarrow [\![e]\!]_\sigma\} = \sigma'] = \mu'(\sigma'),
\end{aligned}
$$

and thus $[\![x := e]\!](\mu) = \mu'$. Therefore we have $|[\![x := e]\!](\mu)| = 1$ and $[\![x := e]\!](\mu) \models Q$. □

Lemma 31 (Smp-T-Sound). *For all Q, S, x, e and X, if*

- *$x \notin \mathsf{fv}(S) \cup \mathsf{fv}(e) \cup \mathsf{fv}(Q)$;*
- *$X \notin \mathsf{fv}(e)$;*
- *$\models Q \Rightarrow (\exists X.\ \lceil e = X \rceil)$;*

then

$$\models [Q \wedge \#S]x := \mathsf{Sample}(e)[Q \wedge \#(S \cup \{x\}) \wedge x \sim e].$$

Proof. Let $\mu \models Q \wedge \#S$. From the premise we know that $\mu \models \exists X.\ \lceil e = X \rceil$, and thus by $X \notin \mathsf{fv}(e)$ we know that there exists some i such that $[\![e]\!]_\sigma = i \in [1, N]$ for all $\sigma \in \mathit{supp}(\mu)$. Let

$$\mu' = \mathbb{E}_{\sigma\sim\mu, r\sim\mathcal{D}[i]}\{\delta(\sigma\{x \rightsquigarrow r\})\}.$$

For all σ', by Lem. 25 and Lem. 26 we have

$$[\![x := \mathsf{Sample}(e)]\!](\mu)(\sigma')$$

$$
\begin{aligned}
&= \sum_{\sigma} \mu(\sigma) \cdot [\![x := \mathsf{Sample}(e)]\!](\sigma)(\sigma') \\
&= \sum_{\sigma} \mu(\sigma) \sum_{r \in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \cdot [\sigma\{x \rightsquigarrow r\} = \sigma'] = \mu'(\sigma'),
\end{aligned}
$$

and thus $[\![x := \mathsf{Sample}(e)]\!](\mu) = \mu'$. Moreover,

$$
\begin{aligned}
|\mu'| &= \sum_{\sigma'} [\![x := \mathsf{Sample}(e)]\!](\mu)(\sigma') \\
&= \sum_{\sigma'} \sum_{\sigma} \mu(\sigma) \sum_{r \in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \cdot [\sigma\{x \rightsquigarrow r\} = \sigma'] \\
&= \sum_{\sigma} \mu(\sigma) \sum_{r \in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \\
&= \sum_{\sigma} \mu(\sigma) = 1.
\end{aligned}
$$

Below we only need to prove the following: $\mu' \vDash Q$, $\mu' \vDash \#(S \cup \{x\})$ and $\mu' \vDash x \sim e$.

First, we prove that $\mu' \vDash Q$. Since $\mu \vDash Q$ and $x \notin \mathsf{fv}(Q)$, by Lem. 23 we know that $\mu\{x \rightsquigarrow 0\} \vDash Q$. By $\mu\{x \rightsquigarrow 0\} = \mu'\{x \rightsquigarrow 0\}$, we have $\mu'\{x \rightsquigarrow 0\} \vDash Q$, and again by Lem. 23 we have $\mu' \vDash Q$.

Next, we prove that $\mu' \vDash \#(S \cup \{x\})$. Assuming $S = \{e_1, \ldots, e_l\}$, $\mu \vDash \#S$ implies that, for all $v_1, \ldots, v_l$,

$$
\Pr_{\sigma \sim \mu}\left[\bigwedge_{j \in [1,l]} \sigma \vDash e_j = v_j\right] = \prod_{j \in [1,l]} \Pr_{\sigma \sim \mu}\left[\sigma \vDash e_j = v_j\right].
$$

Let $e_{l+1} = x$, then by $x \notin \mathsf{fv}(S)$ and Lem. 22, for all v_{l+1}, (let $\mathcal{D}[i](\Lambda) = 0$)

$$
\begin{aligned}
&\Pr_{\sigma' \sim \mu'}\left[\bigwedge_{j \in [1,l+1]} \sigma' \vDash e_j = v_j\right] \\
&= \Pr_{\sigma \sim \mu, r \sim \mathcal{D}[i]}\left[\bigwedge_{j \in [1,l+1]} \sigma\{x \rightsquigarrow r\} \vDash e_j = v_j\right] \\
&= \sum_{r \in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \cdot \Pr_{\sigma \sim \mu}\left[\left(\bigwedge_{j \in [1,l]} \sigma\{x \rightsquigarrow r\} \vDash e_j = v_j\right)\right. \\
&\qquad\qquad\qquad\qquad \left. \wedge\, (\sigma\{x \rightsquigarrow r\} \vDash x = v_{l+1})\right] \\
&= \mathcal{D}[i](v_{l+1}) \cdot \Pr_{\sigma \sim \mu}\left[\bigwedge_{j \in [1,l]} \sigma \vDash e_j = v_j\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \mathcal{D}[i](v_{l+1}) \prod_{j\in[1,l]} \Pr_{\sigma\sim\mu}\left[\sigma \vDash e_j = v_j\right] \\
&= \left(\sum_{r\in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \cdot \Pr_{\sigma\sim\mu}\left[\sigma\{x \rightsquigarrow r\} \vDash x = v_{l+1}\right] \right) \\
&\quad \prod_{j\in[1,l]} \sum_{r\in supp(\mathcal{D}[i])} \mathcal{D}[i](r) \cdot \Pr_{\sigma\sim\mu}\left[\sigma\{x \rightsquigarrow r\} \vDash e_j = v_j\right] \\
&= \prod_{j\in[1,l+1]} \Pr_{\sigma\sim\mu, r\sim\mathcal{D}[i]}\left[\sigma\{x \rightsquigarrow r\} \vDash e_j = v_j\right] \\
&= \prod_{j\in[1,l+1]} \Pr_{\sigma'\sim\mu'}\left[\sigma' \vDash e_j = v_j\right],
\end{aligned}
$$

and thus $\mu' \vDash \#(S \cup \{x\})$.

It remains to prove $\mu' \vDash x \sim e$. By $x \notin \mathsf{fv}(e)$ and $\mu \vDash \lceil e = i \rceil$, we know that $\mu' \vDash \lceil e = i \rceil$ again by applying Lem. 23 twice. Now, since for all r we have

$$
\begin{aligned}
[\![\Pr[x = r]]\!]_{\mu'} &= \Pr_{\sigma'\sim\mu'}[\sigma' \vDash x = r] \\
&= \Pr_{\sigma\sim\mu, r'\sim\mathcal{D}[i]}[\sigma\{x \rightsquigarrow r'\} \vDash x = r] = \mathcal{D}[i](r),
\end{aligned}
$$

$\mu' \vDash x \sim e$ holds by definition. □

Lemma 32 (Csq-T-Sound). *For all P_1, P_2, C, Q_2 and Q_1, if*

- $\vDash [P_2]C[Q_2]$*;*
- $\vDash P_1 \Rightarrow P_2$*,* $\vDash Q_2 \Rightarrow Q_1$*;*

then

$$\vDash [P_1]C[Q_1].$$

Proof. Let $\mu \vDash P_1$. By $\vDash P_1 \Rightarrow P_2$ we know that $\mu \vDash P_2$, then from the premise we have $|[\![C]\!](\mu)| = 1$ and $[\![C]\!](\mu) \vDash Q_2$. Thus by $\vDash Q_2 \Rightarrow Q_1$ we have $[\![C]\!](\mu) \vDash Q_1$. □

Lemma 33 (Seq-T-Sound). *For all P, C_1, Q, C_2 and R, if*

- $\vDash [P]C_1[Q]$*;*
- $\vDash [Q]C_2[R]$*;*

then

$$\vDash [P]C_1; C_2[R].$$

Proof. Let $\mu \vDash P$. From the premise, $|[\![C_1]\!](\mu)| = 1$ and $[\![C_1]\!](\mu) \vDash Q$ holds. Thus, from the premise we know that $|[\![C_2]\!]([\![C_1]\!](\mu))| = 1$ and $[\![C_2]\!]([\![C_1]\!](\mu)) \vDash R$. By Lem. 29 we know that $[\![C_1; C_2]\!](\mu) = [\![C_2]\!]([\![C_1]\!](\mu))$, and thus $|[\![C_1; C_2]\!](\mu)| = 1$ and $[\![C_1; C_2]\!](\mu) \vDash R$. □

Lemma 34 (Skip-T-Sound). *For all Q,*

$$\vDash [Q]\textbf{skip}[Q].$$

Proof. Prove by applying Lem. 25. □

Lemma 35. *For all p, μ_1, μ_2 and C, if $|[\![C]\!](\mu_1)| = |[\![C]\!](\mu_2)| = 1$, then*

$$[\![C]\!](\mu_1 \oplus_p \mu_2) = [\![C]\!](\mu_1) \oplus_p [\![C]\!](\mu_2).$$

Proof. For all σ',

$$\begin{aligned}
&\sum_{\sigma}(\mu_1 \oplus_p \mu_2)(\sigma) \cdot [\![C]\!](\sigma)(\sigma') \\
&= p\sum_{\sigma} \mu_1(\sigma) \cdot [\![C]\!](\sigma)(\sigma') + (1-p)\sum_{\sigma} \mu_2(\sigma) \cdot [\![C]\!](\sigma)(\sigma') \\
&= p \cdot [\![C]\!](\mu_1)(\sigma') + (1-p) \cdot [\![C]\!](\mu_2)(\sigma').
\end{aligned}$$

Thus

$$[\![C]\!](\mu_1 \oplus_p \mu_2) = [\![C]\!](\mu_1) \oplus_p [\![C]\!](\mu_2).$$

□

Lemma 36. *For all P_1, p, P_2, C, Q_1 and Q_2, if*

- $\vDash [P_1]C[Q_1]$;
- $\vDash [P_2]C[Q_2]$;

then

$$\vDash [P_1 \oplus_p P_2]C[Q_1 \oplus_p Q_2].$$

Proof. Let $\mu \vDash P_1 \oplus_p P_2$. The cases of $p = 0$ and $p = 1$ are trivial. For $p \in (0,1)$, we know that there exist μ_1 and μ_2 such that $\mu = \mu_1 \oplus_p \mu_2$, $\mu_1 \vDash P_1$, and $\mu_2 \vDash P_2$. From the premise, $|[\![C]\!](\mu_1)| = |[\![C]\!](\mu_2)| = 1$, $[\![C]\!](\mu_1) \vDash Q_1$ and $[\![C]\!](\mu_2) \vDash Q_2$ hold. By Lem. 35,

$$\begin{aligned}
|[\![C]\!](\mu)| &= \sum_{\sigma'}[\![C]\!](\mu)(\sigma') \\
&= p\sum_{\sigma'}[\![C]\!](\mu_1)(\sigma') + (1-p)\sum_{\sigma'}[\![C]\!](\mu_2)(\sigma') \\
&= p \cdot |[\![C]\!](\mu_1)| + (1-p) \cdot |[\![C]\!](\mu_2)| = 1,
\end{aligned}$$

and $[\![C]\!](\mu) \vDash Q_1 \oplus_p Q_2$. □

Lemma 37 (Cond-T-Sound). *For all $P_1, p, P_2, b, C_1, C_2, Q_1$ and Q_2, if*

- $\vDash [P_1 \wedge \lceil b \rceil]C_1[Q_1]$;
- $\vDash [P_2 \wedge \lceil \neg b \rceil]C_2[Q_2]$;

then

$$\vDash [(P_1 \wedge \lceil b \rceil) \oplus_p (P_2 \wedge \lceil \neg b \rceil)] \textbf{if } (b) \textbf{ then } C_1 \textbf{ else } C_2 [Q_1 \oplus_p Q_2].$$

Proof. Let $C = \textbf{if } (b) \textbf{ then } C_1 \textbf{ else } C_2$. By Lem. 36, we only need to show that

$$\vDash [P_1 \wedge \lceil b \rceil] C [Q_1], \tag{20}$$

$$\vDash [P_2 \wedge \lceil \neg b \rceil] C [Q_2]. \tag{21}$$

Below we only prove (20), while the proof of (21) is similar. Let $\mu \vDash P_1 \wedge \lceil b \rceil$, then $\mu \vDash P_1$ and $\mu \vDash \lceil b \rceil$, and thus $[\![b]\!]_\sigma = \text{true}$ for all $\sigma \in \mathit{supp}(\mu)$. Furthermore, from the premise we know that $|[\![C_1]\!](\mu)| = 1$ and $[\![C_1]\!](\mu) \vDash Q_1$. Thus, from Lem. 26, for all σ' we have

$$\begin{aligned} [\![C]\!](\mu)(\sigma') &= \sum_\sigma \mu(\sigma) \cdot [\![C]\!](\sigma)(\sigma') \\ &= \sum_\sigma \mu(\sigma) \cdot [\![C_1]\!](\sigma)(\sigma') = [\![C_1]\!](\mu)(\sigma'), \end{aligned}$$

and then $[\![C]\!](\mu) = [\![C_1]\!](\mu)$. Thus $|[\![C]\!](\mu)| = |[\![C_1]\!](\mu)| = 1$ and $[\![C]\!](\mu) \vDash Q_1$, which directly implies (20). □

Lemma 38. *For all μ, x and C, if $x \notin \mathsf{fv}(C)$, then for all v we have*

$$([\![C]\!](\mu))\{x \rightsquigarrow v\} = [\![C]\!](\mu\{x \rightsquigarrow v\})$$

Proof. Note that for all σ' we have

$$\begin{aligned} (([\![C]\!](\mu))\{x \rightsquigarrow v\})(\sigma') &= \sum_\sigma \mu(\sigma) \sum_{\sigma'' \,:\, \sigma' = \sigma''\{x \rightsquigarrow v\}} [\![C]\!](\sigma)(\sigma''), \\ [\![C]\!](\mu\{x \rightsquigarrow v\})(\sigma') &= \sum_\sigma \mu(\sigma) \cdot [\![C]\!](\sigma\{x \rightsquigarrow v\})(\sigma'), \end{aligned}$$

and thus we only need to prove that, for all σ and σ',

$$\sum_{\sigma'' \,:\, \sigma' = \sigma''\{x \rightsquigarrow v\}} [\![C]\!](\sigma)(\sigma'') = [\![C]\!](\sigma\{x \rightsquigarrow v\})(\sigma').$$

If $\sigma'(x) \neq v$, by $x \notin \mathsf{fv}(C)$ we know that both sides of the above equation are 0. Below we assume that $\sigma'(x) = v$. Define $p_{n,C,\sigma,\sigma'}$ such that $(C, \sigma) \xrightarrow{p_{n,C,\sigma,\sigma'}}{}^n (\textbf{skip}, \sigma')$, then by induction we can prove that, for all σ and σ', if $\sigma(x) \neq \sigma'(x)$, then $p_{n,C,\sigma,\sigma'} = 0$. Thus, with $\sigma'' = \sigma'\{x \rightsquigarrow \sigma(x)\}$, it remains to prove that, for all n,

$$p_{n,C,\sigma,\sigma''} = p_{n,C,\sigma\{x \rightsquigarrow v\},\sigma''\{x \rightsquigarrow v\}}.$$

This can be proved by induction on n. □

Lemma 39. *For all b, C, σ, σ' and n,*

$$[\![C_{\mathsf{C}}^{n}; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \leq [\![C_{\mathsf{C}}^{n+1}; C_{\mathsf{CW}}]\!](\sigma)(\sigma'),$$

where

$$\begin{aligned} C_{\mathsf{C}} &= \textbf{if } (b) \textbf{ then } C, \\ C_{\mathsf{C}}^{0} &= \textbf{skip}, \\ C_{\mathsf{C}}^{n+1} &= C_{\mathsf{C}}^{n}; C_{\mathsf{C}}, \\ C_{\mathsf{CW}} &= \textbf{if } (b) \textbf{ then } (\textbf{while } (\text{true}) \textbf{ do skip}). \end{aligned}$$

Proof. Note that for all n, σ and σ', from Lem. 29 we have

$$\begin{aligned} [\![C_{\mathsf{C}}^{n}; C_{\mathsf{CW}}]\!](\sigma)(\sigma') &= \sum_{\sigma''} [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{CW}}]\!](\sigma'')(\sigma') \\ &= [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}], \end{aligned}$$

and thus for all n, σ and σ' we have

$$\begin{aligned} & [\![C_{\mathsf{C}}^{n+1}; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \\ = {} & [\![C_{\mathsf{C}}^{n+1}]\!](\sigma)(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}] \\ = {} & \sum_{\sigma''} [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}]\!](\sigma'')(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}] \\ = {} & \sum_{\sigma'' : [\![b]\!]_{\sigma''} = \text{true}} [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}]\!](\sigma'')(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}] \\ & + \sum_{\sigma'' : [\![b]\!]_{\sigma''} = \text{false}} [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}]\!](\sigma'')(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}] \\ = {} & \sum_{\sigma'' : [\![b]\!]_{\sigma''} = \text{true}} [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}]\!](\sigma'')(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}] \\ & + [\![C_{\mathsf{C}}^{n}]\!](\sigma)(\sigma') \cdot [[\![b]\!]_{\sigma'} = \text{false}] \\ \geq {} & [\![C_{\mathsf{C}}^{n}; C_{\mathsf{CW}}]\!](\sigma)(\sigma'). \end{aligned}$$

□

Lemma 40. *For all b, C and μ,*

$$[\![C_{\mathsf{W}}]\!](\mu) = \lim_{n \to \infty} [\![C_{\mathsf{C}}^{n}; C_{\mathsf{CW}}]\!](\mu),$$

where

$$\begin{aligned} C_{\mathsf{W}} &= \textbf{while } (b) \textbf{ do } C, \\ C_{\mathsf{C}} &= \textbf{if } (b) \textbf{ then } C, \\ C_{\mathsf{C}}^{0} &= \textbf{skip}, \\ C_{\mathsf{C}}^{n+1} &= C_{\mathsf{C}}^{n}; C_{\mathsf{C}}, \\ C_{\mathsf{CW}} &= \textbf{if } (b) \textbf{ then } (\textbf{while } (\text{true}) \textbf{ do skip}). \end{aligned}$$

Proof. Know that $[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \leq 1$ holds for all n, σ and σ', and thus from Lem. 39 we know that $\lim_{n\to\infty}[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\mu)$ exists. Therefore, for all σ', by Lem. 12, Lem. 39 and the monotone convergence theorem we have

$$\begin{aligned}\left(\lim_{n\to\infty}[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\mu)\right)(\sigma') &= \lim_{n\to\infty}[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\mu)(\sigma') \\ &= \lim_{n\to\infty}\sum_{\sigma}\mu(\sigma)\cdot[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') \\ &= \sum_{\sigma}\mu(\sigma)\lim_{n\to\infty}[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma'),\end{aligned}$$

and now we only need to prove the following: for all σ and σ',

$$[\![C_{\mathsf{W}}]\!](\sigma)(\sigma') = \lim_{n\to\infty}[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma').$$

For σ' such that $[\![b]\!]_{\sigma'} = \text{true}$, both sides of the above equation are 0. Below we suppose $[\![b]\!]_{\sigma'} = \text{false}$. Since (by Lem. 29)

$$\begin{aligned}[\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\sigma)(\sigma') &= \sum_{\sigma''}[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma'')\cdot[\![C_{\mathsf{CW}}]\!](\sigma'')(\sigma') \\ &= [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma')\end{aligned}$$

holds for all σ, we only need to prove that, for all σ,

$$[\![C_{\mathsf{W}}]\!](\sigma)(\sigma') = \lim_{n\to\infty}[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma').$$

We first show that, for all σ,

$$\lim_{n\to\infty}[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') \leq [\![C_{\mathsf{W}}]\!](\sigma)(\sigma'). \tag{22}$$

To prove the above, we only need to show that, for all n and σ,

$$[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') \leq [\![C_{\mathsf{W}}]\!](\sigma)(\sigma').$$

We prove by induction on n.

- $n = 0$. If $\sigma = \sigma'$, then by Lem. 25 we know that $[\![b]\!]_{\sigma} = \text{false}$ and $[\![\textbf{skip}]\!](\sigma)(\sigma') = 1$, and thus $[\![C_{\mathsf{W}}]\!](\sigma)(\sigma') = 1$. If $\sigma \neq \sigma'$, then by Lem. 25 we know that $[\![\textbf{skip}]\!](\sigma)(\sigma') = 0$.
- $n = k + 1$. If $[\![b]\!]_{\sigma} = \text{false}$, then $[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') = [\sigma = \sigma'] = [\![C_{\mathsf{W}}]\!](\sigma)(\sigma')$. If $[\![b]\!]_{\sigma} = \text{true}$, then by Lem. 26, Lem. 29 and the induction hypothesis,

$$\begin{aligned}[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') &= \sum_{\sigma''}[\![C_{\mathsf{C}}]\!](\sigma)(\sigma'')\cdot[\![C_{\mathsf{C}}^k]\!](\sigma'')(\sigma') \\ &\leq \sum_{\sigma''}[\![C_{\mathsf{C}}]\!](\sigma)(\sigma'')\cdot[\![C_{\mathsf{W}}]\!](\sigma'')(\sigma') \\ &= \sum_{\sigma''}[\![C]\!](\sigma)(\sigma'')\cdot[\![C_{\mathsf{W}}]\!](\sigma'')(\sigma') \\ &= [\![C; C_{\mathsf{W}}]\!](\sigma)(\sigma') \\ &= [\![C_{\mathsf{W}}]\!](\sigma)(\sigma').\end{aligned}$$

Thus (22) holds.

Then we show that, for all σ,

$$[\![C_{\mathsf{W}}]\!](\sigma)(\sigma') \leq \lim_{n\to\infty} [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma'). \tag{23}$$

Define $p_{m,C,\sigma}$ to satisfy $(C,\sigma) \xrightarrow{p_{m,C,\sigma}}{}^{n} (\mathbf{skip}, \sigma')$, we only need to prove that, for all m, σ,

$$p_{m,C_{\mathsf{W}},\sigma} \leq \lim_{n\to\infty} [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma').$$

Since $[\![b]\!]_{\sigma'} = \text{false}$, from Lem. 39 we know that

$$[\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma') \leq [\![C_{\mathsf{C}}^{n+1}]\!](\sigma)(\sigma') \tag{24}$$

holds for all n and σ, and thus we only need to show that, for all n and σ,

$$p_{n,C_{\mathsf{W}},\sigma} \leq [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma').$$

We prove by induction on n.

- $n = 0, 1$. Note that $p_{n,C_{\mathsf{W}},\sigma} = 0$.
- $n \geq 2$. Define $p_{n,C,\sigma,\sigma'}$ to satisfy $(C,\sigma) \xrightarrow{p_{n,C,\sigma,\sigma'}}{}^{n} (\mathbf{skip}, \sigma')$. If $[\![b]\!]_\sigma = \text{false}$, then $p_{n,C_{\mathsf{W}},\sigma} = [\sigma = \sigma'] = [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma')$. If $[\![b]\!]_\sigma = \text{true}$, then by (24), Lem. 24, Lem. 26, Lem. 28, Lem. 29 and the induction hypothesis, we have

$$\begin{aligned}
p_{n,C_{\mathsf{W}},\sigma} &= p_{n-2,C;C_{\mathsf{W}},\sigma} \\
&\leq \sum_{\sigma''} p_{n-2,C,\sigma,\sigma''} \cdot p_{n-2,C_{\mathsf{W}},\sigma'',\sigma'} \\
&\leq \sum_{\sigma''} p_{n-2,C,\sigma,\sigma''} \cdot [\![C_{\mathsf{C}}^{n-2}]\!](\sigma'')(\sigma') \\
&\leq \sum_{\sigma''} [\![C]\!](\sigma)(\sigma'') \cdot [\![C_{\mathsf{C}}^{n-1}]\!](\sigma'')(\sigma') \\
&= [\![C; C_{\mathsf{C}}^{n-1}]\!](\sigma)(\sigma') \\
&= [\![C_{\mathsf{C}}^n]\!](\sigma)(\sigma').
\end{aligned}$$

Thus (23) holds. □

Lemma 41. *For all b, C and μ, if $|[\![C_{\mathsf{W}}]\!](\mu)| = 1$, then*

$$[\![C_{\mathsf{W}}]\!](\mu) = \lim_{n\to\infty} [\![C_{\mathsf{C}}^n]\!](\mu),$$

where

$$\begin{aligned}
C_{\mathsf{W}} &= \mathbf{while}\ (b)\ \mathbf{do}\ C, \\
C_{\mathsf{C}} &= \mathbf{if}\ (b)\ \mathbf{then}\ C, \\
C_{\mathsf{C}}^0 &= \mathbf{skip}, \\
C_{\mathsf{C}}^{n+1} &= C_{\mathsf{C}}^n; C_{\mathsf{C}}.
\end{aligned}$$

Proof. Define C_{CW} as follows.

$$C_{\mathsf{CW}} = \textbf{if } (b) \textbf{ then } (\textbf{while } (\text{true}) \textbf{ do skip}).$$

By Lem. 40,

$$[\![C_{\mathsf{W}}]\!](\mu) = \lim_{n\to\infty} [\![C_{\mathsf{C}}^n ; C_{\mathsf{CW}}]\!](\mu). \tag{25}$$

From Lem. 25, Lem. 26 and Lem. 29, for all n and σ',

$$\begin{aligned}[\![C_{\mathsf{C}}^n ; C_{\mathsf{CW}}]\!](\mu)(\sigma') &= \sum_{\sigma''} [\![C_{\mathsf{C}}^n]\!](\mu)(\sigma'') \cdot [\![C_{\mathsf{CW}}]\!](\sigma'')(\sigma') \\ &\le [\![C_{\mathsf{C}}^n]\!](\mu)(\sigma'). \end{aligned}\tag{26}$$

Thus, by Lem. 12 and Lem. 39 we have

$$|[\![C_{\mathsf{W}}]\!](\mu)| = \lim_{n\to\infty} |[\![C_{\mathsf{C}}^n ; C_{\mathsf{CW}}]\!](\mu)| \le \lim_{n\to\infty} |[\![C_{\mathsf{C}}^n]\!](\mu)|.$$

Then, $\lim_{n\to\infty} |[\![C_{\mathsf{C}}^n]\!](\mu)| = |[\![C_{\mathsf{W}}]\!](\mu)| = 1$, which implies

$$\begin{aligned} 0 &= \lim_{n\to\infty} \sum_{\sigma'} [\![C_{\mathsf{C}}^n]\!](\mu)(\sigma') - \lim_{n\to\infty} \sum_{\sigma'} [\![C_{\mathsf{W}}]\!](\mu)(\sigma') \\ &= \lim_{n\to\infty} \left|[\![C_{\mathsf{C}}^n]\!](\mu) - [\![C_{\mathsf{W}}]\!](\mu)\right|, \end{aligned}$$

and thus

$$[\![C_{\mathsf{W}}]\!](\mu) = \lim_{n\to\infty} [\![C_{\mathsf{C}}^n]\!](\mu).$$

□

Lemma 42 (While-T-Sound). *For all P, Q, b, e, C, X and X', if*

- $\vDash [P \wedge \lceil b \wedge e = X\rceil] C [(P \wedge \lceil b \wedge e + 1 \le X\rceil) \vee (Q \wedge \lceil \neg b\rceil)]$;
- $\vDash P \wedge \lceil b\rceil \Rightarrow (\exists X'.\ \lceil 0 \le e \le X'\rceil)$;
- $X \notin \mathsf{fv}(P) \cup \mathsf{fv}(Q) \cup \mathsf{fv}(b) \cup \mathsf{fv}(e) \cup \mathsf{fv}(C)$;
- $X' \notin \mathsf{fv}(e)$;

then

$$\vDash [(P \wedge \lceil b\rceil) \vee (Q \wedge \lceil \neg b\rceil)] \textbf{while } (b) \textbf{ do } C [Q \wedge \lceil \neg b\rceil].$$

Proof. We use the following notations:

$$\begin{aligned} C_{\mathsf{W}} &= \textbf{while } (b) \textbf{ do } C, \\ C_{\mathsf{C}} &= \textbf{if } (b) \textbf{ then } C, \\ C_{\mathsf{C}}^0 &= \textbf{skip}, \\ C_{\mathsf{C}}^{n+1} &= C_{\mathsf{C}}^n ; C_{\mathsf{C}}, \\ C_{\mathsf{CW}} &= \textbf{if } (b) \textbf{ then } (\textbf{while } (\text{true}) \textbf{ do skip}). \end{aligned}$$

from the premises and Lem. 40, we only need to prove that: for all μ and $r \ge 0$, if $\mu \vDash (P \wedge \lceil b \wedge e \le r\rceil) \vee (Q \wedge \lceil \neg b\rceil)$, then

$$\left| \lim_{n\to\infty} [\![C_{\mathsf{C}}^n ; C_{\mathsf{CW}}]\!](\mu) \right| = 1, \tag{27}$$

$$\left(\lim_{n\to\infty} [\![C_{\mathsf{C}}^n; C_{\mathsf{CW}}]\!](\mu)\right) \vDash Q \wedge \lceil \neg b \rceil . \tag{28}$$

We first prove that: for all $\mu, r \geq 0$ and n, if $\mu \vDash (P \wedge \lceil b \wedge e \leq r \rceil) \vee (Q \wedge \lceil \neg b \rceil)$, then

$$|[\![C_{\mathsf{C}}^n]\!](\mu)| = 1, \tag{29}$$

$$[\![C_{\mathsf{C}}^n]\!](\mu) \vDash (P \wedge \lceil b \wedge e + n \leq r \rceil) \vee (Q \wedge \lceil \neg b \rceil). \tag{30}$$

We prove by induction on n. The case of $n = 0$ is trivial. Let $n = k + 1$. From the induction hypothesis, we know that $|[\![C_{\mathsf{C}}^k]\!](\mu)| = 1$ and $[\![C_{\mathsf{C}}^k]\!](\mu) \vDash (P \wedge \lceil b \wedge e + k \leq r \rceil) \vee (Q \wedge \lceil \neg b \rceil)$. Let $\mu' = [\![C_{\mathsf{C}}^k]\!](\mu)$. If $\mu' \vDash Q \wedge \lceil \neg b \rceil$, then $\mu' \vDash \lceil \neg b \rceil$, and thus from Lem. 25, Lem. 26 and Lem. 29 we have

$$[\![C_{\mathsf{C}}^n]\!](\mu) = [\![C_{\mathsf{C}}]\!](\mu') = \mu',$$

which implies (29) and (30). Otherwise $\mu' \vDash (P \wedge \lceil b \wedge e + k \leq r \rceil)$, and thus from Lem. 26 and Lem. 29 we have

$$[\![C_{\mathsf{C}}^n]\!](\mu) = [\![C_{\mathsf{C}}]\!](\mu') = [\![C]\!](\mu'). \tag{31}$$

Let $\mu'' = \mathbb{E}_{\sigma \sim \mu'}\{\delta(\sigma\{X \rightsquigarrow [\![e]\!]_\sigma\})\}$, then we know that $\mu'' \vDash \lceil e = X \rceil$ from the third premise, and for all $\sigma \in \mathit{supp}(\mu'')$ we have $[\![X]\!]_\sigma + k \leq r$. Besides, by applying Lem. 23 and the third premise twice on $\mu' \vDash P \wedge \lceil b \rceil$, we have $\mu'' \vDash P \wedge \lceil b \rceil$. Thus $\mu'' \vDash P \wedge \lceil b \wedge e = X \rceil$, then from the first premise we have

$$|[\![C]\!](\mu'')| = 1,$$

$$[\![C]\!](\mu'') \vDash (P \wedge \lceil b \wedge e + 1 \leq X \rceil) \vee (Q \wedge \lceil \neg b \rceil). \tag{32}$$

Then, from Lem. 38, (31) and the third premise we know that (29) holds, that is

$$\begin{aligned}
|[\![C_{\mathsf{C}}^n]\!](\mu)| &= |([\![C]\!](\mu'))\{X \rightsquigarrow 0\}| \\
&= |[\![C]\!](\mu'\{X \rightsquigarrow 0\})| \\
&= |[\![C]\!](\mu''\{X \rightsquigarrow 0\})| \\
&= |([\![C]\!](\mu''))\{X \rightsquigarrow 0\}| \\
&= |[\![C]\!](\mu'')| = 1.
\end{aligned}$$

Furthermore, since $[\![X]\!]_\sigma + k \leq r$ for all $\sigma \in \mathit{supp}(\mu'')$, from $X \notin \mathsf{fv}(C)$ we know $[\![X]\!]_\sigma + k \leq r$ holds for all $\sigma \in \mathit{supp}([\![C]\!](\mu''))$, and thus

$$[\![C]\!](\mu'') \vDash (P \wedge \lceil b \wedge e + n \leq r \rceil) \vee (Q \wedge \lceil \neg b \rceil)$$

holds from (32) and then we obtain

$$[\![C]\!](\mu') \vDash (P \wedge \lceil b \wedge e + n \leq r \rceil) \vee (Q \wedge \lceil \neg b \rceil),$$

by applying Lem. 38 and Lem. 23 both twice. Then we get (30) from (31).

Now, by taking $n = \lfloor r \rfloor + 1$ in (30), from the second premise we have

$$[\![C_{\mathsf{C}}^{\lfloor r \rfloor + 1}]\!](\mu) \vDash Q \wedge \lceil \neg b \rceil \, . \tag{33}$$

Thus by induction on n, for all $n \geq \lfloor r \rfloor + 1$, $[\![C_{\mathsf{C}}^{n}]\!](\mu) = [\![C_{\mathsf{C}}^{\lfloor r \rfloor + 1}]\!](\mu)$ holds from Lem. 26 and Lem. 29. Therefore, for all $n \geq \lfloor r \rfloor + 1$, we have

$$[\![C_{\mathsf{C}}^{n}; C_{\mathsf{CW}}]\!](\mu) = [\![C_{\mathsf{CW}}]\!]([\![C_{\mathsf{C}}^{n}]\!](\mu)) = [\![C_{\mathsf{C}}^{n}]\!](\mu) = [\![C_{\mathsf{C}}^{\lfloor r \rfloor + 1}]\!](\mu)$$

from Lem. 25, Lem. 26 and Lem. 29. Now (27) and (28) follow from (29) (by taking $n = \lfloor r \rfloor + 1$) and (33). □

Lemma 43 (While-TB-Sound). *For all $n, Q_0, \ldots, Q_n, b$ and C, if*

- *For all $i \in [0, n)$, $\vDash [Q_i]$**if** (b) **then** $C[Q_{i+1}]$;*
- *$\vDash Q_n \Rightarrow \lceil \neg b \rceil$;*

then

$$\vDash [Q_0]\textbf{while } (b) \textbf{ do } C[Q_n].$$

Proof. We use the following notations:

$$\begin{aligned}
C_{\mathsf{W}} &= \textbf{while } (b) \textbf{ do } C, \\
C_{\mathsf{C}} &= \textbf{if } (b) \textbf{ then } C, \\
C_{\mathsf{C}}^{0} &= \textbf{skip}, \\
C_{\mathsf{C}}^{m+1} &= C_{\mathsf{C}}^{m}; C_{\mathsf{C}}, \\
C_{\mathsf{CW}} &= \textbf{if } (b) \textbf{ then } (\textbf{while } (\text{true}) \textbf{ do skip}).
\end{aligned}$$

Let $\mu \vDash Q_0$. By Lem. 40, we only need to prove the following:

$$\left| \lim_{m \to \infty} [\![C_{\mathsf{C}}^{m}; C_{\mathsf{CW}}]\!](\mu) \right| = 1, \tag{34}$$

$$\left(\lim_{m \to \infty} [\![C_{\mathsf{C}}^{m}; C_{\mathsf{CW}}]\!](\mu) \right) \vDash Q_n. \tag{35}$$

We first prove that: for all $m \in [0, n]$,

$$|[\![C_{\mathsf{C}}^{m}]\!](\mu)| = 1, \tag{36}$$

$$[\![C_{\mathsf{C}}^{m}]\!](\mu) \vDash Q_m. \tag{37}$$

We prove by induction on m. The case of $m = 0$ is trivial. Let $m = k + 1$. From the induction hypothesis, we know that $|[\![C_{\mathsf{C}}^{k}]\!](\mu)| = 1$ and $[\![C_{\mathsf{C}}^{k}]\!](\mu) \vDash Q_k$. Let $\mu' = [\![C_{\mathsf{C}}^{k}]\!](\mu)$. From the first premise, we have $|[\![C_{\mathsf{C}}]\!](\mu')| = 1$ and $[\![C_{\mathsf{C}}]\!](\mu') \vDash Q_m$, and by Lem. 26 and Lem. 29 we know that

$$[\![C_{\mathsf{C}}^{m}]\!](\mu) = [\![C_{\mathsf{C}}]\!](\mu').$$

Thus (36) and (37) hold.

Now by taking $m = n$ in (37), we know that $[\![C_{\mathsf{C}}^{n}]\!](\mu) \vDash \lceil \neg b \rceil$ from the second premise. By Lem. 25, Lem. 26, Lem. 29 and induction, this implies the following: for all $m \geq n$,

$$[\![C_{\mathsf{C}}^{m}; C_{\mathsf{CW}}]\!](\mu) = [\![C_{\mathsf{C}}^{n}]\!](\mu).$$

Thus (34) and (35) follow from (36) and (37) by taking $m = n$.

□

Lemma 44 (Conj-T-Sound). *For all P_1, P_2, C, Q_1 and Q_2, if*

- $\vDash [P_1]C[Q_1]$;
- $\vDash [P_2]C[Q_2]$;

then

$$\vDash [P_1 \wedge P_2]C[Q_1 \wedge Q_2].$$

Proof. Let $\mu \vDash P_1 \wedge P_2$, then $\mu \vDash P_1$ and $\mu \vDash P_2$ holds. From the premise we know $|[\![C]\!](\mu)| = 1$, $[\![C]\!](\mu) \vDash Q_1$, and $[\![C]\!](\mu) \vDash Q_2$. Thus $[\![C]\!](\mu) \vDash Q_1 \wedge Q_2$. ☐

Lemma 45 (Disj-T-Sound). *For all P_1, P_2, C, Q_1 and Q_2, if*

- $\vDash [P_1]C[Q_1]$;
- $\vDash [P_2]C[Q_2]$;

then

$$\vDash [P_1 \vee P_2]C[Q_1 \vee Q_2].$$

Proof. Let $\mu \vDash P_1 \vee P_2$, then either $\mu \vDash P_1$ or $\mu \vDash P_2$ holds. If $\mu \vDash P_1$, then from the premise we have $|[\![C]\!](\mu)| = 1$ and $[\![C]\!](\mu) \vDash Q_1$. Thus $[\![C]\!](\mu) \vDash Q_1 \vee Q_2$. The case of $\mu \vDash P_2$ is simliar. ☐

Lemma 46 (Exists-T-Sound). *For all P, C, Q and X, if*

- $\vDash [P]C[Q]$;
- $X \notin \mathsf{fv}(C)$;

then

$$\vDash [\exists X.\, P]C[\exists X.\, Q].$$

Proof. Let $\mu \vDash \exists X.\, P$, then there exists v such that $\mu\{X \rightsquigarrow v\} \vDash P$. From the premise, we know that $|[\![C]\!](\mu\{X \rightsquigarrow v\})| = 1$ and $[\![C]\!](\mu\{X \rightsquigarrow v\}) \vDash Q$ hold. From $X \notin \mathsf{fv}(C)$ and Lem. 38 we have $[\![C]\!](\mu\{X \rightsquigarrow v\}) = ([\![C]\!](\mu))\{X \rightsquigarrow v\}$, and thus

$$|[\![C]\!](\mu)| = |([\![C]\!](\mu))\{X \rightsquigarrow v\}| = |[\![C]\!](\mu\{X \rightsquigarrow v\})| = 1$$

and $([\![C]\!](\mu))\{X \rightsquigarrow v\} \vDash Q$, then $[\![C]\!](\mu) \vDash \exists X.\, Q$. ☐

Lemma 47 (Forall-T-Sound). *For all P, C, Q and X, if*

- $\vDash [P]C[Q]$;
- $X \notin \mathsf{fv}(C)$;

then

$$\vDash [\forall X.\, P]C[\forall X.\, Q].$$

Proof. Let $\mu \vDash \forall X.\, P$, then for all v we have $\mu\{X \rightsquigarrow v\} \vDash P$. It remains to show that $|[\![C]\!](\mu)| = 1$ and $[\![C]\!](\mu) \vDash \forall X.\, Q$. For all v, from the premise we know that $|[\![C]\!](\mu\{X \rightsquigarrow v\})| = 1$ and $[\![C]\!](\mu\{X \rightsquigarrow v\}) \vDash Q$. By $X \notin \mathsf{fv}(C)$ and Lem. 38, $[\![C]\!](\mu\{X \rightsquigarrow v\}) = ([\![C]\!](\mu))\{X \rightsquigarrow v\}$, and thus

$$|[\![C]\!](\mu)| = |([\![C]\!](\mu))\{X \rightsquigarrow v\}| = |[\![C]\!](\mu\{X \rightsquigarrow v\})| = 1$$

and $([\![C]\!](\mu))\{X \rightsquigarrow v\} \vDash Q$. Therefore $[\![C]\!](\mu) \vDash \forall X.\, Q$. □

Lemma 48. *For all P, C_1, C_2, Q and R, if*

- $\vDash [P]C_1; C_2[Q]$*;*
- $\{\mu' \mid \mu' \vDash R\} = \{\mu' \mid \exists \mu.\ \mu \vDash P \wedge [\![C_1]\!](\mu) = \mu'\}$*;*

then

- $\vDash [P]C_1[R]$*;*
- $\vDash [R]C_2[Q]$*.*

Proof. The proof of $\vDash [P]C_1[R]$ is trivial. We show $\vDash [R]C_2[Q]$ below. Let $\mu' \vDash R$. From the second premise, there exists μ such that $\mu \vDash P$ and $[\![C_1]\!](\mu) = \mu'$. Thus, from the first premise, there exists μ'' such that $[\![C_1; C_2]\!](\mu) = \mu''$ (where $|\mu''| = 1$) and $\mu'' \vDash Q$. From Lem. 29, we know that $[\![C_2]\!](\mu') = [\![C_2]\!]([\![C_1]\!](\mu)) = [\![C_1; C_2]\!](\mu)$. Thus $|[\![C_2]\!](\mu')| = 1$ and $[\![C_2]\!](\mu') \vDash Q$. □

F An RT-Based Program Logic

When proving an inequality between probabilities involving two probabilistic programs, after applying the RT-based coupling, we are required to prove two Hoare triples in the RT-based semantics. In this section, we give a simple Hoare-style unary program logic for proving these Hoare triples, and prove its soundness. Logic rules are presented in Fig. 25 and Fig. 26.

Theorem 7. *For all P, C and Q,*

$$\vdash_{\mathrm{RT}} \{P\}C\{Q\} \implies \vDash_{\mathrm{RT}} \{P\}C\{Q\}$$

and

$$\vdash_{\mathrm{RT}} [P]C[Q] \implies \vDash_{\mathrm{RT}} [P]C[Q].$$

Proof. From Lem. 52, Lem. 53, Lem. 54, Lem. 55, Lem. 56, Lem. 57, Lem. 59, Lem. 60, Lem. 61, Lem. 62, Lem. 63, Lem. 64, Lem. 68, Lem. 65, Lem. 69, Lem. 70, Lem. 71, Lem. 72, Lem. 73, Lem. 74, Lem. 75 and Lem. 76. □

Lemma 49. *For all $\sigma, RT, \iota, \mathbf{Q}, E$ and x, if $(\sigma, RT, \iota) \vDash \mathbf{Q}[E/x]$, then $(\sigma', RT, \iota) \vDash \mathbf{Q}$, where $\sigma' = \sigma\{x \rightsquigarrow [\![E]\!]_{(\sigma, RT, \iota)}\}$.*

Proof. By induction on the structure of $\mathbf{Q}$. □

$$\frac{}{\vdash_{\text{RT}} \{\mathbf{Q}[e/x]\}x := e\{\mathbf{Q}\}} \quad (\text{RT-VAR})$$

$$\frac{\vDash_{\text{RT}} \mathbf{P} \Rightarrow \mathbf{Q}[(\mathsf{hd}_n + 1)/\mathsf{hd}_n][\mathsf{RT}[n][\mathsf{hd}_n]/x]}{\vdash_{\text{RT}} \{e = n \wedge \mathbf{P}\}x := \mathsf{Sample}(e)\{\mathbf{Q}\}} \quad (\text{RT-SMP})$$

$$\frac{\vDash_{\text{RT}} \mathbf{P}_1 \Rightarrow \mathbf{P}_2 \qquad \vdash_{\text{RT}} \{\mathbf{P}_2\}C\{\mathbf{Q}_2\} \qquad \vDash_{\text{RT}} \mathbf{Q}_2 \Rightarrow \mathbf{Q}_1}{\vdash_{\text{RT}} \{\mathbf{P}_1\}C\{\mathbf{Q}_1\}} \quad (\text{RT-CSQ})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{P}\}C_1\{\mathbf{Q}\} \qquad \vdash_{\text{RT}} \{\mathbf{Q}\}C_2\{\mathbf{R}\}}{\vdash_{\text{RT}} \{\mathbf{P}\}C_1; C_2\{\mathbf{R}\}} \quad (\text{RT-SEQ}) \qquad \frac{}{\vdash_{\text{RT}} \{\mathbf{Q}\}\mathbf{skip}\{\mathbf{Q}\}} \quad (\text{RT-SKIP})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{P} \wedge b\}C_1\{\mathbf{Q}\} \qquad \vdash_{\text{RT}} \{\mathbf{P} \wedge \neg b\}C_2\{\mathbf{Q}\}}{\vdash_{\text{RT}} \{\mathbf{P}\}\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\{\mathbf{Q}\}} \quad (\text{RT-COND})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{Q} \wedge b\}C\{\mathbf{Q}\}}{\vdash_{\text{RT}} \{\mathbf{Q}\}\mathbf{while}\ (b)\ \mathbf{do}\ C\{\mathbf{Q} \wedge \neg b\}} \quad (\text{RT-WHILE})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{P}_1\}C\{\mathbf{Q}_1\} \qquad \vdash_{\text{RT}} \{\mathbf{P}_2\}C\{\mathbf{Q}_2\}}{\vdash_{\text{RT}} \{\mathbf{P}_1 \wedge \mathbf{P}_2\}C\{\mathbf{Q}_1 \wedge \mathbf{Q}_2\}} \quad (\text{RT-CONJ})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{P}_1\}C\{\mathbf{Q}_1\} \qquad \vdash_{\text{RT}} \{\mathbf{P}_2\}C\{\mathbf{Q}_2\}}{\vdash_{\text{RT}} \{\mathbf{P}_1 \vee \mathbf{P}_2\}C\{\mathbf{Q}_1 \vee \mathbf{Q}_2\}} \quad (\text{RT-DISJ})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{P}\}C\{\mathbf{Q}\} \qquad X \notin \mathsf{fv}(C)}{\vdash_{\text{RT}} \{\exists X.\, \mathbf{P}\}C\{\exists X.\, \mathbf{Q}\}} \quad (\text{RT-EXISTS})$$

$$\frac{\vdash_{\text{RT}} \{\mathbf{P}\}C\{\mathbf{Q}\} \qquad X \notin \mathsf{fv}(C)}{\vdash_{\text{RT}} \{\forall X.\, \mathbf{P}\}C\{\forall X.\, \mathbf{Q}\}} \quad (\text{RT-FORALL})$$

Fig. 25. Selected rules of the resampling-table-based program logic (part I).

Lemma 50. *For all $\sigma, RT, \iota, \mathbf{Q}, E$ and i, if $(\sigma, RT, \iota) \vDash \mathbf{Q}[E/\mathsf{hd}_i]$, then $(\sigma, RT, \iota') \vDash \mathbf{Q}$, where*

$$\iota' = (\iota[1], \ldots, \iota[i-1], [\![E]\!]_{(\sigma, RT, \iota)}, \iota[i+1], \ldots, \iota[N]).$$

Proof. By induction on the structure of $\mathbf{Q}$. □

Lemma 51. *For all $\sigma, RT, \iota, \mathbf{Q}$ and x, if $x \notin \mathsf{fv}(\mathbf{Q})$, then for all v we have*

$$(\sigma, RT, \iota) \vDash \mathbf{Q} \iff (\sigma\{x \rightsquigarrow v\}, RT, \iota) \vDash \mathbf{Q}.$$

Proof. By induction on the structure of $\mathbf{Q}$. □

Lemma 52 (RT-Var-T-Sound). *For all $\mathbf{Q}, e$ and x,*

$$\vDash_{\text{RT}} [\mathbf{Q}[e/x]]x := e[\mathbf{Q}].$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{Q}[e/x]$, then there exists σ' such that $RT \vdash (x := e, \sigma, \iota) \to (\mathbf{skip}, \sigma', \iota)$, where $\sigma' = \sigma\{x \rightsquigarrow [\![e]\!]_\sigma\}$. From Lem. 49 we have $(\sigma', RT, \iota) \vDash \mathbf{Q}$. □

$$\frac{}{\vdash_{\mathrm{RT}} [\mathbf{Q}[e/x]]x := e[\mathbf{Q}]} \quad (\textsc{rt-var-t})$$

$$\frac{\vdash_{\mathrm{RT}} \mathbf{P} \Rightarrow \mathbf{Q}[(\mathsf{hd}_n + 1)/\mathsf{hd}_n][\mathsf{RT}[n][\mathsf{hd}_n]/x]}{\vdash_{\mathrm{RT}} [e = n \wedge \mathbf{P}]x := \mathsf{Sample}(e)[\mathbf{Q}]} \quad (\textsc{rt-smp-t})$$

$$\frac{\vdash_{\mathrm{RT}} \mathbf{P}_1 \Rightarrow \mathbf{P}_2 \qquad \vdash_{\mathrm{RT}} [\mathbf{P}_2]C[\mathbf{Q}_2] \qquad \vdash_{\mathrm{RT}} \mathbf{Q}_2 \Rightarrow \mathbf{Q}_1}{\vdash_{\mathrm{RT}} [\mathbf{P}_1]C[\mathbf{Q}_1]} \quad (\textsc{rt-csq-t})$$

$$\frac{\vdash_{\mathrm{RT}} [\mathbf{P}]C_1[\mathbf{Q}] \qquad \vdash_{\mathrm{RT}} [\mathbf{Q}]C_2[\mathbf{R}]}{\vdash_{\mathrm{RT}} [\mathbf{P}]C_1; C_2[\mathbf{R}]} \quad (\textsc{rt-seq-t}) \qquad \frac{}{\vdash_{\mathrm{RT}} [\mathbf{Q}]\mathbf{skip}[\mathbf{Q}]} \quad (\textsc{rt-skip-t})$$

$$\frac{\vdash_{\mathrm{RT}} [\mathbf{P} \wedge b]C_1[\mathbf{Q}] \qquad \vdash_{\mathrm{RT}} [\mathbf{P} \wedge \neg b]C_2[\mathbf{Q}]}{\vdash_{\mathrm{RT}} [\mathbf{P}]\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2[\mathbf{Q}]} \quad (\textsc{rt-cond-t})$$

$$\frac{\begin{array}{c}\vdash_{\mathrm{RT}} [\mathbf{Q} \wedge b \wedge e = X]C[\mathbf{Q} \wedge e + 1 \leq X] \\ X \notin \mathsf{fv}(\mathbf{Q}) \cup \mathsf{fv}(b) \cup \mathsf{fv}(e) \cup \mathsf{fv}(C) \qquad \vDash_{\mathrm{RT}} \mathbf{Q} \Rightarrow e \geq 0\end{array}}{\vdash_{\mathrm{RT}} [\mathbf{Q}]\mathbf{while}\ (b)\ \mathbf{do}\ C[\mathbf{Q} \wedge \neg b]} \quad (\textsc{rt-while-t})$$

$$\frac{\vdash_{\mathrm{RT}} [\mathbf{P}_1]C[\mathbf{Q}_1] \qquad \vdash_{\mathrm{RT}} [\mathbf{P}_2]C[\mathbf{Q}_2]}{\vdash_{\mathrm{RT}} [\mathbf{P}_1 \wedge \mathbf{P}_2]C[\mathbf{Q}_1 \wedge \mathbf{Q}_2]} \quad (\textsc{rt-conj-t})$$

$$\frac{\vdash_{\mathrm{RT}} [\mathbf{P}_1]C[\mathbf{Q}_1] \qquad \vdash_{\mathrm{RT}} [\mathbf{P}_2]C[\mathbf{Q}_2]}{\vdash_{\mathrm{RT}} [\mathbf{P}_1 \vee \mathbf{P}_2]C[\mathbf{Q}_1 \vee \mathbf{Q}_2]} \quad (\textsc{rt-disj-t})$$

$$\frac{\vdash_{\mathrm{RT}} [\mathbf{P}]C[\mathbf{Q}] \qquad X \notin \mathsf{fv}(C)}{\vdash_{\mathrm{RT}} [\exists X.\, \mathbf{P}]C[\exists X.\, \mathbf{Q}]} \quad (\textsc{rt-exists-t})$$

$$\frac{\vdash_{\mathrm{RT}} [\mathbf{P}]C[\mathbf{Q}] \qquad X \notin \mathsf{fv}(C)}{\vdash_{\mathrm{RT}} [\forall X.\, \mathbf{P}]C[\forall X.\, \mathbf{Q}]} \quad (\textsc{rt-forall-t})$$

Fig. 26. Selected rules of the resampling-table-based program logic (part II).

Lemma 53 (RT-Var-Sound). *For all* $\mathbf{Q}, e$ *and* x,

$$\vDash_{\mathrm{RT}} \{\mathbf{Q}[e/x]\}x := e\{\mathbf{Q}\}.$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{Q}[e/x]$ and $RT \vdash (x := e, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota')$, then $\sigma' = \sigma\{x \rightsquigarrow [\![e]\!]_\sigma\}$ and $\iota' = \iota$. From Lem. 49 we have $(\sigma', RT, \iota') \vDash \mathbf{Q}$. $\square$

Lemma 54 (RT-Smp-T-Sound). *For all* $\mathbf{P}, \mathbf{Q}, x, e$ *and* i, *if*

$$\vDash_{\mathrm{RT}} \mathbf{P} \Rightarrow \mathbf{Q}[(\mathsf{hd}_i + 1)/\mathsf{hd}_i][\mathsf{RT}[i][\mathsf{hd}_i]/x],$$

then

$$\vDash_{\mathrm{RT}} [e = i \wedge \mathbf{P}]x := \mathsf{Sample}(e)[\mathbf{Q}].$$

Proof. Let $(\sigma, RT, \iota) \vDash e = i \wedge \mathbf{P}$, then $[\![e]\!]_\sigma = i$ and

$$(\sigma, RT, \iota) \vDash \mathbf{Q}[(\mathsf{hd}_i + 1)/\mathsf{hd}_i][\mathsf{RT}[i][\mathsf{hd}_i]/x]$$

holds from the premise, and there exist σ' and ι' such that

$$RT \vdash (x := \mathsf{Sample}(e), \sigma, \iota) \to (\mathbf{skip}, \sigma', \iota'),$$

where $\sigma' = \sigma\{x \rightsquigarrow RT[i][\iota[i]]\}$ and $\iota' = (\iota[1], \ldots, \iota[i-1], \iota[i]+1, \iota[i+1], \ldots, \iota[N])$. Since $[\![\mathsf{RT}[i][\mathsf{hd}_i]]\!]_{(\sigma,RT,\iota)} = RT[i][\iota[i]]$, by Lem. 49 we know that

$$(\sigma', RT, \iota) \vDash \mathbf{Q}[(\mathsf{hd}_i + 1)/\mathsf{hd}_i].$$

Since $[\![\mathsf{hd}_i + 1]\!]_{(\sigma',RT,\iota)} = \iota[i]+1$, we then have $(\sigma', RT, \iota') \vDash \mathbf{Q}$ from Lem. 50. □

Lemma 55 (RT-Smp-Sound). *For all* $\mathbf{P}, \mathbf{Q}, x, e$ *and* i*, if*

$$\vDash_{\mathrm{RT}} \mathbf{P} \Rightarrow \mathbf{Q}[(\mathsf{hd}_i + 1)/\mathsf{hd}_i][\mathsf{RT}[i][\mathsf{hd}_i]/x],$$

then

$$\vDash_{\mathrm{RT}} \{e = i \wedge \mathbf{P}\} x := \mathsf{Sample}(e) \{\mathbf{Q}\}.$$

Proof. Let $(\sigma, RT, \iota) \vDash e = i \wedge \mathbf{P}$ and

$$RT \vdash (x := \mathsf{Sample}(e), \sigma, \iota) \to (\mathbf{skip}, \sigma', \iota'),$$

then we know that $[\![e]\!]_\sigma = i$, $\sigma' = \sigma\{x \rightsquigarrow RT[i][\iota[i]]\}$, $\iota' = (\iota[1], \ldots, \iota[i-1], \iota[i] + 1, \iota[i+1], \ldots, \iota[N])$, and from the premise

$$(\sigma, RT, \iota) \vDash \mathbf{Q}[(\mathsf{hd}_i + 1)/\mathsf{hd}_i][\mathsf{RT}[i][\mathsf{hd}_i]/x].$$

Since $[\![\mathsf{RT}[i][\mathsf{hd}_i]]\!]_{(\sigma,RT,\iota)} = RT[i][\iota[i]]$, by Lem. 49 we know that

$$(\sigma', RT, \iota) \vDash \mathbf{Q}[(\mathsf{hd}_i + 1)/\mathsf{hd}_i].$$

Since $[\![\mathsf{hd}_i + 1]\!]_{(\sigma',RT,\iota)} = \iota[i]+1$, we then have $(\sigma', RT, \iota') \vDash \mathbf{Q}$ from Lem. 50. □

Lemma 56 (RT-Csq-T-Sound). *For all* $\mathbf{P}_1, \mathbf{P}_2, C, \mathbf{Q}_2$ *and* $\mathbf{Q}_1$*, if*

- $\vDash_{\mathrm{RT}} [\mathbf{P}_2] C [\mathbf{Q}_2]$*;*
- $\vDash_{\mathrm{RT}} \mathbf{P}_1 \Rightarrow \mathbf{P}_2$*,* $\vDash_{\mathrm{RT}} \mathbf{Q}_2 \Rightarrow \mathbf{Q}_1$*;*

then

$$\vDash_{\mathrm{RT}} \{\mathbf{P}_1\} C \{\mathbf{Q}_1\}.$$

Proof. Let $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \vDash \mathbf{P}_1$. By $\vDash_{\mathrm{RT}} \mathbf{P}_1 \Rightarrow \mathbf{P}_2$ we know that $\Sigma \vDash \mathbf{P}_2$ and thus there exist σ' and ι' such that $RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota')$ and $(\sigma', RT, \iota') \vDash \mathbf{Q}_2$, then by $\vDash_{\mathrm{RT}} \mathbf{Q}_2 \Rightarrow \mathbf{Q}_1$ we have $(\sigma', RT, \iota') \vDash \mathbf{Q}_1$. □

Lemma 57 (RT-Csq-Sound). *For all* $\mathbf{P}_1, \mathbf{P}_2, C, \mathbf{Q}_2$ *and* $\mathbf{Q}_1$*, if*

- $\vDash_{\mathrm{RT}} \{\mathbf{P}_2\} C \{\mathbf{Q}_2\}$*;*
- $\vDash_{\mathrm{RT}} \mathbf{P}_1 \Rightarrow \mathbf{P}_2$*,* $\vDash_{\mathrm{RT}} \mathbf{Q}_2 \Rightarrow \mathbf{Q}_1$*;*

then

$$\vDash_{\mathrm{RT}} \{\mathbf{P}_1\} C \{\mathbf{Q}_1\}.$$

Proof. Let $\Sigma \vDash \mathbf{P}_1$ and $RT \vdash (C, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$. By $\vDash_{\mathrm{RT}} \mathbf{P}_1 \Rightarrow \mathbf{P}_2$ we know that $\Sigma \vDash \mathbf{P}_2$, and thus from the premise $(\sigma', RT, \iota') \vDash \mathbf{Q}_2$. Then we have $(\sigma', RT, \iota') \vDash \mathbf{Q}_1$ from $\vDash_{\mathrm{RT}} \mathbf{Q}_2 \Rightarrow \mathbf{Q}_1$. □

Lemma 58. *For all $\sigma, \sigma', \iota, \iota', RT, C_1$ and C_2,*

$$RT \vdash (C_1; C_2, \sigma, \iota) \rightarrow^n (\mathbf{skip}, \sigma', \iota')$$

holds iff there exist σ'', ι'', n_1 and n_2 such that

- *$n_1 + n_2 + 1 = n$;*
- *$RT \vdash (C_1, \sigma, \iota) \rightarrow^{n_1} (\mathbf{skip}, \sigma'', \iota'')$;*
- *$RT \vdash (C_2, \sigma'', \iota'') \rightarrow^{n_2} (\mathbf{skip}, \sigma', \iota')$.*

Proof. By induction on n. □

Lemma 59 (RT-Seq-T-Sound). *For all $\mathbf{P}, C_1, \mathbf{Q}, C_2$ and $\mathbf{R}$, if*

- *$\vDash_{\mathrm{RT}} [\mathbf{P}] C_1 [\mathbf{Q}]$;*
- *$\vDash_{\mathrm{RT}} [\mathbf{Q}] C_2 [\mathbf{R}]$;*

then

$$\vDash_{\mathrm{RT}} [\mathbf{P}] C_1; C_2 [\mathbf{R}].$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{P}$. From the premise, there exist σ'' and ι'' such that $(\sigma'', RT, \iota'') \vDash \mathbf{Q}$ and

$$RT \vdash (C_1, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma'', \iota''),$$

holds, and then from the premise we know that there exist σ' and ι' such that $(\sigma', RT, \iota') \vDash \mathbf{R}$ and

$$RT \vdash (C_2, \sigma'', \iota'') \rightarrow^* (\mathbf{skip}, \sigma', \iota')$$

holds. Now, by Lem. 58, we have

$$RT \vdash (C_1; C_2, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota').$$

□

Lemma 60 (RT-Seq-Sound). *For all $\mathbf{P}, C_1, \mathbf{Q}, C_2$ and $\mathbf{R}$, if*

- *$\vDash_{\mathrm{RT}} \{\mathbf{P}\} C_1 \{\mathbf{Q}\}$;*
- *$\vDash_{\mathrm{RT}} \{\mathbf{Q}\} C_2 \{\mathbf{R}\}$;*

then

$$\vDash_{\mathrm{RT}} \{\mathbf{P}\} C_1; C_2 \{\mathbf{R}\}.$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{P}$, and $RT \vdash (C_1; C_2, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$. By Lem. 58, there exist σ'' and ι'' such that

$$RT \vdash (C_1, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma'', \iota''),$$
$$RT \vdash (C_2, \sigma'', \iota'') \rightarrow^* (\mathbf{skip}, \sigma', \iota'),$$

and thus from the premises we have $(\sigma'', RT, \iota'') \vDash \mathbf{Q}$, and $(\sigma', RT, \iota') \vDash \mathbf{R}$ follows. □

Lemma 61 (RT-Skip-T-Sound). *For all* $\mathbf{Q}$*,*

$$\vDash_{\mathrm{RT}} [\mathbf{Q}]\mathbf{skip}[\mathbf{Q}].$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{Q}$, then $RT \vdash (\mathbf{skip}, \sigma, \iota) \rightarrow^0 (\mathbf{skip}, \sigma, \iota)$. □

Lemma 62 (RT-Skip-Sound). *For all* $\mathbf{Q}$*,*

$$\vDash_{\mathrm{RT}} \{\mathbf{Q}\}\mathbf{skip}\{\mathbf{Q}\}.$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{Q}$ and $RT \vdash (\mathbf{skip}, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$. Note that $\sigma' = \sigma$ and $\iota' = \iota$, and thus $(\sigma', RT, \iota') \vDash \mathbf{Q}$. □

Lemma 63 (RT-Cond-T-Sound). *For all* $\mathbf{P}, b, C_1, C_2$ *and* $\mathbf{Q}$*, if*

- $\vDash_{\mathrm{RT}} [\mathbf{P} \wedge b]C_1[\mathbf{Q}]$*;*
- $\vDash_{\mathrm{RT}} [\mathbf{P} \wedge \neg b]C_2[\mathbf{Q}]$*;*

then

$$\vDash_{\mathrm{RT}} [\mathbf{P}]\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2[\mathbf{Q}].$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{P}$. If $[\![b]\!]_\sigma = \text{true}$, then $(\sigma, RT, \iota) \vDash \mathbf{P} \wedge b$, and then from the premise we know that there exist σ' and ι' such that $RT \vdash (C_1, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$ and $(\sigma', RT, \iota') \vDash \mathbf{Q}$. Thus $RT \vdash (\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$. The case of $[\![b]\!]_\sigma = \text{false}$ is similar. □

Lemma 64 (RT-Cond-Sound). *For all* $\mathbf{P}, b, C_1, C_2$ *and* $\mathbf{Q}$*, if*

- $\vDash_{\mathrm{RT}} \{\mathbf{P} \wedge b\}C_1\{\mathbf{Q}\}$*;*
- $\vDash_{\mathrm{RT}} \{\mathbf{P} \wedge \neg b\}C_2\{\mathbf{Q}\}$*;*

then

$$\vDash_{\mathrm{RT}} \{\mathbf{P}\}\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\{\mathbf{Q}\}.$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{P}$ and

$$RT \vdash (\mathbf{if}\ (b)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota').$$

If $[\![b]\!]_\sigma = \text{true}$, then $RT \vdash (C_1, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$ and $(\sigma, RT, \iota) \vDash \mathbf{P} \wedge b$. From the premise, this implies $(\sigma', RT, \iota') \vDash \mathbf{Q}$. The case of $[\![b]\!]_\sigma = \text{false}$ is similar. □

Lemma 65 (RT-While-Sound). *For all* $\mathbf{Q}, b$ *and* C, *if*

$$\vDash_{\mathrm{RT}} \{\mathbf{Q} \wedge b\} C \{\mathbf{Q}\},$$

then

$$\vDash_{\mathrm{RT}} \{\mathbf{Q}\} \mathbf{while}\ (b)\ \mathbf{do}\ C\, \{\mathbf{Q} \wedge \neg b\}.$$

Proof. Let $(\sigma, RT, \iota) \vDash \mathbf{Q}$ and

$$RT \vdash (\mathbf{while}\ (b)\ \mathbf{do}\ C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota'),$$

that is, there exists some n such that

$$RT \vdash (\mathbf{while}\ (b)\ \mathbf{do}\ C, \sigma, \iota) \to^n (\mathbf{skip}, \sigma', \iota').$$

Below we prove that $(\sigma', RT, \iota') \vDash \mathbf{Q} \wedge \neg b$ by induction on n. If $[\![b]\!]_\sigma$ = false, then $(\sigma, RT, \iota) \vDash \mathbf{Q} \wedge \neg b$, $\sigma' = \sigma$ and $\iota' = \iota$, and thus $(\sigma', RT, \iota') \vDash \mathbf{Q} \wedge \neg b$. If $[\![b]\!]_\sigma$ = true, then $n \geq 2$, $(\sigma, RT, \iota) \vDash \mathbf{Q} \wedge b$ and

$$RT \vdash (C; \mathbf{while}\ (b)\ \mathbf{do}\ C, \sigma, \iota) \to^{n-2} (\mathbf{skip}, \sigma', \iota'),$$

and thus by Lem. 58 we know that there exist σ'', ι'' and $n' < n$ such that

$$RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma'', \iota''),$$
$$RT \vdash (\mathbf{while}\ (b)\ \mathbf{do}\ C, \sigma'', \iota'') \to^{n'} (\mathbf{skip}, \sigma', \iota').$$

The former implies $(\sigma'', RT, \iota'') \vDash \mathbf{Q}$ from the premise, and thus from the induction hypothesis we have $(\sigma', RT, \sigma') \vDash \mathbf{Q} \wedge \neg b$. $\square$

Lemma 66. *For all* $\sigma, \sigma', \iota, \iota', RT, x, C$ *and* n, *if* $x \notin \mathsf{fv}(C)$ *and* $RT \vdash (C, \sigma, \iota) \to^n (\mathbf{skip}, \sigma', \iota')$, *then for all* v *we have*

$$RT \vdash (C, \sigma\{x \rightsquigarrow v\}, \iota) \to^n (\mathbf{skip}, \sigma'\{x \rightsquigarrow v\}, \iota').$$

Proof. By induction on n. $\square$

Lemma 67. *For all* $C, \sigma, RT, \iota, \sigma', \iota', \sigma''$ *and* ι'', *if*

– $RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota')$;
– $RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma'', \iota'')$;

then $\sigma' = \sigma''$ *and* $\iota' = \iota''$.

Proof. Let the premises hold, then $RT \vdash (C, \sigma, \iota) \to^n (\mathbf{skip}, \sigma', \iota')$ for some n. Then we prove by induction on n. $\square$

Lemma 68 (RT-While-T-Sound). *For all* $\mathbf{Q}, b, e, C$ *and* X, *if*

– $\vDash_{\mathrm{RT}} [\mathbf{Q} \wedge b \wedge e = X] C [\mathbf{Q} \wedge e + 1 \leq X]$;
– $X \notin \mathsf{fv}(\mathbf{Q}) \cup \mathsf{fv}(b) \cup \mathsf{fv}(e) \cup \mathsf{fv}(C)$;
– $\vDash_{\mathrm{RT}} \mathbf{Q} \Rightarrow e \geq 0$;

then

$$\models_{\mathrm{RT}} [\mathbf{Q}]\textbf{while } (b) \textbf{ do } C\, [\mathbf{Q} \wedge \neg b].$$

Proof. Let $(\sigma, RT, \iota) \models \mathbf{Q}$. If $[\![b]\!]_\sigma =$ false, then $(\sigma, RT, \iota) \models \mathbf{Q} \wedge \neg b$ and $RT \vdash (\textbf{while } (b) \textbf{ do } C, \sigma, \iota) \to^2 (\textbf{skip}, \sigma, \iota)$. If $[\![b]\!]_\sigma =$ true and $[\![e]\!]_\sigma \notin \mathit{Real}$, we take $\sigma' = \sigma\{X \rightsquigarrow [\![e]\!]_\sigma\}$ and thus have

$$(\sigma', RT, \iota) \models \mathbf{Q} \wedge b \wedge e = X$$

from the second premise. From the first premise, there exist σ'' and ι'' such that $RT \vdash (C, \sigma', \iota) \to^* (\textbf{skip}, \sigma'', \iota'')$ and $(\sigma'', RT, \iota'') \models \mathbf{Q} \wedge e + 1 \leq X$, then by Lem. 66 and Lem. 67 we know that $[\![X]\!]_{\sigma''} = [\![X]\!]_{\sigma'} \notin \mathit{Real}$, which contradicts $[\![e + 1 \leq X]\!]_{\sigma''} =$ true. Thus it remains to prove the following: For all σ, ι, RT and r such that $r \geq 0$ and $(\sigma, RT, \iota) \models \mathbf{Q} \wedge b \wedge e = r$, there exist σ' and ι' such that

$$RT \vdash (\textbf{while } (b) \textbf{ do } C, \sigma, \iota) \to^* (\textbf{skip}, \sigma', \iota')$$

and $(\sigma', RT, \iota') \models \mathbf{Q} \wedge \neg b$. We prove by induction on $\lfloor r \rfloor$.

- $\lfloor r \rfloor = 0$. Assuming $\sigma''' = \sigma\{X \rightsquigarrow r\}$, we have $(\sigma''', RT, \iota) \models \mathbf{Q} \wedge b \wedge e = X$, and then from the premises we know that there exist σ'' and ι'' such that $RT \vdash (C, \sigma''', \iota) \to^* (\textbf{skip}, \sigma'', \iota'')$ and $(\sigma'', RT, \iota'') \models \mathbf{Q} \wedge 1 \leq e + 1 \leq X$. Since $X \notin \mathsf{fv}(C)$, by Lem. 66 and Lem. 67 we have $[\![X]\!]_{\sigma''} = [\![X]\!]_{\sigma'''} = r$, and thus $(\sigma'', RT, \iota'') \models 1 \leq r$, which contradicts with $\lfloor r \rfloor = 0$.
- $\lfloor r \rfloor > 0$. Assuming $\sigma''' = \sigma\{X \rightsquigarrow r\}$, we have $(\sigma''', RT, \iota) \models \mathbf{Q} \wedge b \wedge e = X$, and then from the premise we know that there exist σ'' and ι'' such that $RT \vdash (C, \sigma''', \iota) \to^* (\textbf{skip}, \sigma'', \iota'')$ and $(\sigma'', RT, \iota'') \models \mathbf{Q} \wedge e + 1 \leq X$. Since $X \notin \mathsf{fv}(C)$, by Lem. 66 and Lem. 67 we have $[\![X]\!]_{\sigma''} = [\![X]\!]_{\sigma'''} = r$, and thus there exists $r' \leq r - 1$ such that $(\sigma'', RT, \iota'') \models \mathbf{Q} \wedge e = r'$. Besides, from Lem. 66 we know that

$$RT \vdash (C, \sigma, \iota) \to^* (\textbf{skip}, \sigma''\{X \rightsquigarrow \sigma(X)\}, \iota''). \tag{38}$$

  Since $X \notin \mathsf{fv}(\mathbf{Q}) \cup \mathsf{fv}(e) \cup \mathsf{fv}(b)$, by Lem. 51 we have $(\sigma''\{X \rightsquigarrow \sigma(X)\}, RT, \iota'') \models \mathbf{Q} \wedge e = r'$. If $[\![b]\!]_{\sigma''\{X \rightsquigarrow \sigma(X)\}} =$ false, then $(\sigma''\{X \rightsquigarrow \sigma(X)\}, RT, \iota'') \models \mathbf{Q} \wedge \neg b$ and by $[\![b]\!]_\sigma =$ true and (38) we have

$$RT \vdash (\textbf{while } (b) \textbf{ do } C, \sigma, \iota) \to^* (\textbf{skip}, \sigma''\{X \rightsquigarrow \sigma(X)\}, \iota'').$$

  If $[\![b]\!]_{\sigma''\{X \rightsquigarrow \sigma(X)\}} =$ true, then $(\sigma''\{X \rightsquigarrow \sigma(X)\}, RT, \iota'') \models \mathbf{Q} \wedge b \wedge e = r'$, and then by the induction hypothesis there exist σ' and ι' such that

$$RT \vdash (\textbf{while } (b) \textbf{ do } C, \sigma''\{X \rightsquigarrow \sigma(X)\}, \iota'') \to^* (\textbf{skip}, \sigma', \iota'),$$

  and $(\sigma', RT, \iota') \models \mathbf{Q} \wedge \neg b$. Thus, by $[\![b]\!]_\sigma =$ true and (38) we obtain that

$$RT \vdash (\textbf{while } (b) \textbf{ do } C, \sigma, \iota) \to^* (\textbf{skip}, \sigma', \iota').$$

□

Lemma 69 (RT-Conj-T-Sound). *For all* $\mathbf{P}_1, \mathbf{P}_2, C, \mathbf{Q}_1$ *and* $\mathbf{Q}_2$*, if*

– $\models_{\mathrm{RT}} [\mathbf{P}_1]C[\mathbf{Q}_1]$;
– $\models_{\mathrm{RT}} [\mathbf{P}_2]C[\mathbf{Q}_2]$;

then

$$\models_{\mathrm{RT}} [\mathbf{P}_1 \wedge \mathbf{P}_2]C[\mathbf{Q}_1 \wedge \mathbf{Q}_2].$$

Proof. Define $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \models \mathbf{P}_1 \wedge \mathbf{P}_2$, then $\Sigma \models \mathbf{P}_1$ and $\Sigma \models \mathbf{P}_2$. From the premises, we know that there exist σ' and ι' such that $RT \vdash (C, \sigma, \iota) \twoheadrightarrow^* (\mathbf{skip}, \sigma', \iota')$ and $(\sigma', RT, \iota') \models \mathbf{Q}_1$, and there exist σ'' and ι'' such that $RT \vdash (C, \sigma, \iota) \twoheadrightarrow^* (\mathbf{skip}, \sigma'', \iota'')$ and $(\sigma'', RT, \iota'') \models \mathbf{Q}_2$. From Lem. 67 we have $\sigma' = \sigma''$ and $\iota' = \iota''$, and thus $(\sigma', RT, \iota') \models \mathbf{Q}_1 \wedge \mathbf{Q}_2$. □

Lemma 70 (RT-Conj-Sound). *For all* $\mathbf{P}_1, \mathbf{P}_2, C, \mathbf{Q}_1$ *and* $\mathbf{Q}_2$*, if*

– $\models_{\mathrm{RT}} \{\mathbf{P}_1\}C\{\mathbf{Q}_1\}$;
– $\models_{\mathrm{RT}} \{\mathbf{P}_2\}C\{\mathbf{Q}_2\}$;

then

$$\models_{\mathrm{RT}} \{\mathbf{P}_1 \wedge \mathbf{P}_2\}C\{\mathbf{Q}_1 \wedge \mathbf{Q}_2\}.$$

Proof. Let $\Sigma = (\sigma, RT, \iota)$, σ' and ι' satisfy $\Sigma \models \mathbf{P}_1 \wedge \mathbf{P}_2$ and $RT \vdash (C, \sigma, \iota) \twoheadrightarrow^* (\mathbf{skip}, \sigma', \iota')$, then $\Sigma \models \mathbf{P}_1$ and $\Sigma \models \mathbf{P}_2$. From the premises we know that $(\sigma', RT, \iota') \models \mathbf{Q}_1$ and $(\sigma', RT, \iota') \models \mathbf{Q}_2$, and thus $(\sigma', RT, \iota') \models \mathbf{Q}_1 \wedge \mathbf{Q}_2$. □

Lemma 71 (RT-Disj-T-Sound). *For all* $\mathbf{P}_1, \mathbf{P}_2, C, \mathbf{Q}_1$ *and* $\mathbf{Q}_2$*, if*

– $\models_{\mathrm{RT}} [\mathbf{P}_1]C[\mathbf{Q}_1]$;
– $\models_{\mathrm{RT}} [\mathbf{P}_2]C[\mathbf{Q}_2]$;

then3

$$\models_{\mathrm{RT}} [\mathbf{P}_1 \vee \mathbf{P}_2]C[\mathbf{Q}_1 \vee \mathbf{Q}_2].$$

Proof. Define $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \models \mathbf{P}_1 \vee \mathbf{P}_2$, now either $\Sigma \models \mathbf{P}_1$ or $\Sigma \models \mathbf{P}_2$ holds. If $\Sigma \models \mathbf{P}_1$, then from the premise we know that there exist σ' and ι' such that $RT \vdash (C, \sigma, \iota) \twoheadrightarrow^* (\mathbf{skip}, \sigma', \iota')$ and $(\sigma', RT, \iota') \models \mathbf{Q}_1$, and thus $(\sigma', RT, \iota') \models \mathbf{Q}_1 \vee \mathbf{Q}_2$. The case of $\Sigma \models \mathbf{P}_2$ is similar. □

Lemma 72 (RT-Disj-Sound). *For all* $\mathbf{P}_1, \mathbf{P}_2, C, \mathbf{Q}_1$ *and* $\mathbf{Q}_2$*, if*

– $\models_{\mathrm{RT}} \{\mathbf{P}_1\}C\{\mathbf{Q}_1\}$;
– $\models_{\mathrm{RT}} \{\mathbf{P}_2\}C\{\mathbf{Q}_2\}$;

then

$$\models_{\mathrm{RT}} \{\mathbf{P}_1 \vee \mathbf{P}_2\}C\{\mathbf{Q}_1 \vee \mathbf{Q}_2\}.$$

Proof. Let $\Sigma = (\sigma, RT, \iota)$, σ' and ι' satisfy $\Sigma \models \mathbf{P}_1 \vee \mathbf{P}_2$ and $RT \vdash (C, \sigma, \iota) \twoheadrightarrow^* (\mathbf{skip}, \sigma', \iota')$, then either $\Sigma \models \mathbf{P}_1$ or $\Sigma \models \mathbf{P}_2$ holds. If $\Sigma \models \mathbf{P}_1$, then from the premise we have $(\sigma', RT, \iota') \models \mathbf{Q}_1$ and thus $(\sigma', RT, \iota') \models \mathbf{Q}_1 \vee \mathbf{Q}_2$. The case of $\Sigma \models \mathbf{P}_2$ is simliar. □

Lemma 73 (RT-Exists-T-Sound). *For all* $\mathbf{P}, C, \mathbf{Q}$ *and* X*, if*

- $\vDash_{\mathrm{RT}} [\mathbf{P}]C[\mathbf{Q}]$*;*
- $X \notin \mathsf{fv}(C)$*;*

then

$$\vDash_{\mathrm{RT}} [\exists X.\, \mathbf{P}]C[\exists X.\, \mathbf{Q}].$$

Proof. Define $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \vDash \exists X.\, \mathbf{P}$, then there exists v such that $(\sigma\{X \rightsquigarrow v\}, RT, \iota) \vDash \mathbf{P}$. From the premise, there exist σ' and ι' such that $(\sigma', RT, \iota') \vDash \mathbf{Q}$ and

$$RT \vdash (C, \sigma\{X \rightsquigarrow v\}, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota').$$

From $X \notin \mathsf{fv}(C)$ and Lem. 66 we know that $RT \vdash (C, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma'\{X \rightsquigarrow [\![X]\!]_\sigma\}, \iota')$. Since $\sigma' = (\sigma'\{X \rightsquigarrow [\![X]\!]_\sigma\})\{X \rightsquigarrow [\![X]\!]_{\sigma'}\}$, we have $(\sigma'\{X \rightsquigarrow [\![X]\!]_\sigma\}, RT, \iota') \vDash \exists X.\, \mathbf{Q}$. ☐

Lemma 74 (RT-Exists-Sound). *For all* $\mathbf{P}, C, \mathbf{Q}$ *and* X*, if*

- $\vDash_{\mathrm{RT}} \{\mathbf{P}\}C\{\mathbf{Q}\}$*;*
- $X \notin \mathsf{fv}(C)$*;*

then

$$\vDash_{\mathrm{RT}} \{\exists X.\, \mathbf{P}\}C\{\exists X.\, \mathbf{Q}\}.$$

Proof. Let $\Sigma = (\sigma, RT, \iota)$, σ' and ι' satisfy $\Sigma \vDash \exists X.\, \mathbf{P}$ and $RT \vdash (C, \sigma, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota')$, then there exists v such that $(\sigma\{X \rightsquigarrow v\}, RT, \iota) \vDash \mathbf{P}$. From $X \notin \mathsf{fv}(C)$ and Lem. 66 we have

$$RT \vdash (C, \sigma\{X \rightsquigarrow v\}, \iota) \rightarrow^* (\mathbf{skip}, \sigma'\{X \rightsquigarrow v\}, \iota'),$$

and then from the premise $(\sigma'\{X \rightsquigarrow v\}, RT, \iota') \vDash \mathbf{Q}$, which implies $(\sigma', RT, \iota') \vDash \exists X.\, \mathbf{Q}$. ☐

Lemma 75 (RT-Forall-T-Sound). *For all* $\mathbf{P}, C, \mathbf{Q}$ *and* X*, if*

- $\vDash_{\mathrm{RT}} [\mathbf{P}]C[\mathbf{Q}]$*;*
- $X \notin \mathsf{fv}(C)$*;*

then

$$\vDash_{\mathrm{RT}} [\forall X.\, \mathbf{P}]C[\forall X.\, \mathbf{Q}].$$

Proof. Define $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \vDash \forall X.\, \mathbf{P}$, then for all v we have $(\sigma\{X \rightsquigarrow v\}, RT, \iota) \vDash \mathbf{P}$. For some v_0, from the premise, there exist σ' and ι' such that

$$RT \vdash (C, \sigma\{X \rightsquigarrow v_0\}, \iota) \rightarrow^* (\mathbf{skip}, \sigma', \iota').$$

For all v, from the premise, we know that there exist σ'' and ι'' such that $(\sigma'', RT, \iota'') \vDash \mathbf{Q}$ and

$$RT \vdash (C, \sigma\{X \rightsquigarrow v\}, \iota) \rightarrow^* (\mathbf{skip}, \sigma'', \iota''),$$

then from $X \notin \mathsf{fv}(C)$, Lem. 66 and Lem. 67 we have $\sigma'' = \sigma'\{X \rightsquigarrow v\}$ and $\iota'' = \iota'$. Thus $(\sigma', RT, \iota') \vDash \forall X.\, \mathbf{Q}$. ☐

Lemma 76 (RT-Forall-Sound). *For all* $\mathbf{P}, C, \mathbf{Q}$ *and* X*, if*

- $\models \{\mathbf{P}\}C\{\mathbf{Q}\}$*;*
- $X \notin \mathsf{fv}(C)$*;*

then

$$\models \{\forall X.\, \mathbf{P}\}C\{\forall X.\, \mathbf{Q}\}.$$

Proof. Let $\Sigma = (\sigma, RT, \iota)$, σ' and ι' satisfy $\Sigma \models \forall X.\, \mathbf{P}$ and $RT \vdash (C, \sigma, \iota) \to^* (\mathbf{skip}, \sigma', \iota')$, then for all v we have $(\sigma\{X \rightsquigarrow v\}, RT, \iota) \models \mathbf{P}$. By $X \notin \mathsf{fv}(C)$ and Lem. 66 we know that

$$RT \vdash (C, \sigma\{X \rightsquigarrow v\}, \iota) \to^* (\mathbf{skip}, \sigma'\{X \rightsquigarrow v\}, \iota'),$$

then $(\sigma'\{X \rightsquigarrow v\}, RT, \iota') \models \mathbf{Q}$ from the premise. Thus $(\sigma', RT, \iota') \models \forall X.\, \mathbf{Q}$. □

G Auxiliary Lemmas

In this section, we give several auxiliary lemmas for the verification of ALLLs and other results.

Lemma 77. *For all* P_1, P_2, C, Q_2 *and* Q_1*, if*

- $\models [P_2]C[Q_2]$*;*
- $\models P_1 \Rightarrow P_2$*,* $\models Q_2 \Rightarrow Q_1$*;*

then

$$\models [P_1]C[Q_1].$$

Proof. By Lem. 32. □

Lemma 78. *For all* P_1, P_2, C, Q_1 *and* Q_2*, if*

- $\models [P_1]C[Q_1]$*;*
- $\models \{P_2\}C\{Q_2\}$*;*

then

$$\models [P_1 \wedge P_2]C[Q_1 \wedge Q_2].$$

Proof. Similar to the proof of Lem. 44. □

Lemma 79. *For all* $P, C_1, C_2, \mathbf{q}_1, \mathbf{q}_2$ *and* r*, if*

- $\models \{P\}C_1 \leq C_2\{\mathbf{q}_1, \mathbf{q}_2\}$*;*
- $\models [P]C_2[\Pr[\mathbf{q}_2] \leq r]$*;*

then

$$\models \{P\}C_1\{\Pr[\mathbf{q}_1] \leq r\}.$$

Proof. Let $\mu \vDash P$ and $|[\![C_1]\!](\mu)| = 1$. From the second premise, we know that $[\![C_2]\!](\mu) = 1$ and $[\![C_2]\!](\mu) \vDash \Pr[\mathbf{q}_2] \leq r$, and thus

$$\Pr_{\sigma \sim [\![C_2]\!](\mu)}[\sigma \vDash \mathbf{q}_2] \leq r.$$

Then, from the first premise and $\mu \vDash P$, we have

$$\Pr_{\sigma \sim [\![C_1]\!](\mu)}[\sigma \vDash \mathbf{q}_1] \leq \Pr_{\sigma \sim [\![C_2]\!](\mu)}[\sigma \vDash \mathbf{q}_2] \leq r,$$

which implies $[\![C_1]\!](\mu) \vDash \Pr[\mathbf{q}_1] \leq r$. ☐

Lemma 80. *For all* $\mathbf{p}, C$ *and* $\mathbf{q}$*, if*

$$\vDash_{\mathrm{RT}} [\mathbf{p} \wedge \mathsf{hdinit}] C [\mathbf{q}],$$

then

$$\vDash [\lceil \mathbf{p} \rceil] C [\lceil \mathbf{q} \rceil].$$

Proof. Let μ satisfy $\mu \vDash \lceil \mathbf{p} \rceil$. We first show that $|[\![C]\!](\mu)| = 1$. From Thm. 1, we only need to prove that $|[\![C]\!]_{\mathrm{RT}}(\mu)| = 1$, that is, $|[\![C]\!]_{\mathrm{RT}}(\sigma)| = 1$ holds for all $\sigma \in \mathit{supp}(\mu)$. From the premise, we know that $\mathit{RT} \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \_, \_)$ holds for all RT, thus

$$\begin{aligned} |[\![C]\!]_{\mathrm{RT}}(\sigma)| &= \sum_{\sigma'} \mathcal{M}(\{\mathit{RT} \mid \mathit{RT} \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\}) \\ &= \mathcal{M}\left(\biguplus_{\sigma'} \{\mathit{RT} \mid \mathit{RT} \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_)\} \right) \\ &= \mathcal{M}(\{\mathit{RT} \mid \mathit{RT} \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \_, \_)\}) \\ &= \mathcal{M}(\mathit{RTable}) = 1. \end{aligned}$$

Then we show that $[\![C]\!](\mu) \vDash \lceil \mathbf{q} \rceil$. Again, by applying Thm. 1, we only need to prove that $[\![C]\!]_{\mathrm{RT}}(\mu) \vDash \lceil \mathbf{q} \rceil$, that is, $\sigma' \vDash \mathbf{q}$ holds for all $\sigma \in \mathit{supp}(\mu)$ and σ' such that $[\![C]\!]_{\mathrm{RT}}(\sigma)(\sigma') > 0$. Assuming that $\sigma \in \mathit{supp}(\mu)$ and $[\![C]\!]_{\mathrm{RT}}(\sigma)(\sigma') > 0$, we know that $\sigma \vDash \mathbf{p}$, and by definition there must exist an RT such that

$$\mathit{RT} \vdash (C, \sigma, \iota_{\mathsf{init}}) \to^* (\mathbf{skip}, \sigma', \_),$$

then from the premise we have $\sigma' \vDash \mathbf{q}$. ☐

Lemma 81. *For all* e *and* r*,* t-$\mathbf{closed}(\mathbb{E}[e] \leq r \wedge \lceil e \geq 0 \rceil)$*.*

Proof. Let $\lim \vec{\mu} = \mu$, and $\vec{\mu}[i] \vDash \mathbb{E}[e] \leq r \wedge \lceil e \geq 0 \rceil$ for all $i \geq 0$. We first prove that $\mu \vDash \lceil e \geq 0 \rceil$. For all $\sigma \in \mathit{supp}(\mu)$, since $\lim \vec{\mu} = \mu$, there must exist some $i \geq 0$ such that $\sigma \in \mathit{supp}(\vec{\mu}[i])$. Then from $\vec{\mu}[i] \vDash \lceil e \geq 0 \rceil$ we have $\sigma \vDash e \geq 0$. Thus $\mu \vDash \lceil e \geq 0 \rceil$.

Then we show that $\mu \vDash \mathbb{E}[e] \leq r$. From $\mu \vDash \lceil e \geq 0 \rceil$, we know that $[\![e]\!]_\sigma \geq 0$ for all $\sigma \in \mathit{supp}(\mu)$. For $\sigma \in \mathit{supp}(\mu)$, by Lem. 12 we have

$$\mu(\sigma) \cdot [\![e]\!]_\sigma = \left(\lim_{n \to \infty} \vec{\mu}[n](\sigma) \right) \cdot [\![e]\!]_\sigma = \lim_{n \to \infty} \left(\vec{\mu}[n](\sigma) \cdot [\![e]\!]_\sigma \right).$$

Then by definition and Fatou's lemma, we have

$$\begin{aligned}
[\![\mathbb{E}[e]]\!]_\mu = \mathbb{E}_{\sigma\sim\mu}[[\![e]\!]_\sigma] &= \sum_\sigma \mu(\sigma)\cdot[\![e]\!]_\sigma \\
&= \sum_\sigma \lim_{n\to\infty}(\vec{\mu}[n](\sigma)\cdot[\![e]\!]_\sigma) \\
&\le \liminf_{n\to\infty}\sum_\sigma \vec{\mu}[n](\sigma)\cdot[\![e]\!]_\sigma \\
&= \liminf_{n\to\infty}[\![\mathbb{E}[e]]\!]_{\vec{\mu}[n]} \le r.
\end{aligned}$$

Thus $\mu \vDash \mathbb{E}[e] \le r$. □

Lemma 82. *For all* $\mathbf{q}$ *and* r*,* t-$\mathbf{closed}(\Pr[\mathbf{q}] \le r)$*.*

Proof. Let $\lim\vec{\mu} = \mu$, and $\vec{\mu}[i] \vDash \Pr[\mathbf{q}] \le r$ for all $i \ge 0$. We show that $\mu \vDash \Pr[\mathbf{q}] \le r$. By definition and Lem. 12, we have

$$[\![\Pr[\mathbf{q}]]\!]_\mu = \Pr_{\sigma\sim\mu}[\sigma \vDash \mathbf{q}] = \lim_{n\to\infty}\Pr_{\sigma\sim\vec{\mu}[n]}[\sigma\vDash\mathbf{q}] \le r.$$

Thus $\mu \vDash \Pr[\mathbf{q}] \le r$. □

Lemma 83. *For all* P *and* Q*, if* t-$\mathbf{closed}(P)$ *and* t-$\mathbf{closed}(Q)$*, then* t-$\mathbf{closed}(P\wedge Q)$*.*

Proof. Let $\lim\vec{\mu} = \mu$, and $\vec{\mu}[i] \vDash P\wedge Q$ for all $i \ge 0$. Thus $\vec{\mu}[i]\vDash P$ and $\vec{\mu}[i]\vDash Q$ for all $i \ge 0$. From the premises, we have $\mu\vDash P$ and $\mu \vDash Q$, and thus $\mu\vDash P\wedge Q$. □

H Proofs of Ex. 1 and Ex. 2

In this section, we give the proofs of Ex. 1 and Ex. 2.

H.1 Proof of Ex. 1

By applying Lem. 77, Thm. 2 and Lem. 81, we only need to prove

$$\vDash [\lceil cnt = 0 \wedge y = 1\rceil]\, C'_{\text{flip}}(K)\, [\mathbb{E}[cnt] \le 2 \wedge \lceil cnt \ge 0\rceil]$$

for each K. By applying Thm. 6, it remains to prove

$$\vdash [\lceil cnt = 0 \wedge y = 1\rceil]\, C'_{\text{flip}}(K)\, [\mathbb{E}[cnt] \le 2 \wedge \lceil cnt \ge 0\rceil]\,.$$

The proof sketch of the above judgment is presented below. Here we apply the (WHILE-TB) rule, where $n = K$ and

$$Q_i = \left((\lceil y = 1 \wedge cnt = i\rceil) \oplus_{\frac{1}{2^i}} \left(\lceil y = 0 \wedge cnt \ge 0\rceil \wedge \mathbb{E}[cnt] = 2 - \frac{i}{2^i - 1}\right)\right).$$

```
$[\lceil cnt = 0 \wedge y = 1 \rceil]$
$[Q_0]$
while $(y = 1 \wedge cnt < K)$ do
    $y := \mathsf{Sample}(1);$
    $cnt := cnt + 1;$
$[Q_K]$
$[\mathbb{E}[cnt] \leq 2 \wedge \lceil cnt \geq 0 \rceil]$

$[Q_i]$
if $(y = 1 \wedge cnt < K)$ then
    $[\lceil cnt = i \rceil]$
    $y := \mathsf{Sample}(1);$
    $[\lceil cnt = i \rceil \wedge y \sim 1]$
    $cnt := cnt + 1;$
    $[\lceil cnt = i + 1 \rceil \wedge y \sim 1]$
$\left[(\lceil cnt = i + 1 \rceil \wedge y \sim 1) \oplus_{\frac{1}{2^i}} \left(\lceil y = 0 \wedge cnt \geq 0 \rceil \wedge \mathbb{E}[cnt] = 2 - \frac{i}{2^i - 1}\right)\right]$
$\Big[\Big((\lceil y = 1 \wedge cnt = i + 1 \rceil) \oplus_{\frac{1}{2}} (\lceil y = 0 \wedge cnt = i + 1 \rceil)\Big)$
  $\oplus_{\frac{1}{2^i}} \left(\lceil y = 0 \wedge cnt \geq 0 \rceil \wedge \mathbb{E}[cnt] = 2 - \frac{i}{2^i - 1}\right)\Big]$
$[Q_{i+1}]$
```

Informally, Q_i captures the quantitative properties of *cnt* after the i-th iteration. With probability $\frac{1}{2^i}$, the program samples 1 from $y := \mathsf{Sample}(1)$ for i times in a row, and thus $y = 1$ and $cnt = i$. Otherwise $y = 0$, and the weighted sum of *cnt* is

$$\sum_{j=1}^{i} \frac{j}{2^j} = 2 - \frac{2+i}{2^i}.$$

Conditioning on $y = 0$, the expectation of *cnt* is $2 - \frac{i}{2^i - 1}$. Furthermore, for $i = K$, by Q_K we know that the expectation of *cnt* is

$$i \cdot \frac{1}{2^i} + \left(2 - \frac{i}{2^i - 1}\right)\left(1 - \frac{1}{2^i}\right) = 2 - \frac{1}{2^{i-1}} \leq 2.$$

H.2 Proof of Ex. 2

We give relevant definitions in Fig. 27. Take $\mathbf{R} = \mathsf{coll}(k)$, where

$$\mathsf{coll}(k) \triangleq \bigvee\nolimits_{0 \leq i < j < m(k)} . \, \mathsf{RT}[1][i] = \mathsf{RT}[1][j],$$
$$m(i) \triangleq |\{n_j : j \in [1, i]\}|.$$

By applying Thm. 3, we only need to prove

$$\models_{\mathrm{RT}} \{\mathsf{inp} \wedge \mathsf{hdinit}\} \, C_{\mathrm{PRF}}^{\mathsf{bad}} \, \{bad = 1 \Rightarrow \mathbf{R}\} \tag{39}$$

$$
\begin{aligned}
(\mathit{Seq})\ \Lambda &::= [\,] \mid n :: \Lambda \mid (n,r) :: \Lambda \\
(\mathit{Val})\ v &::= \ldots \mid (n,r) \\
(\mathit{Expr})\ e &::= \ldots \mid (e_1,e_2) \\
(\mathit{Bexp})\ b &::= \ldots \mid \mathsf{find}(e_1,e_2) \mid \mathsf{findkey}(e_1,e_2) \mid \mathsf{findval}(e_1,e_2)
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathsf{find}(e_1,(e_2,e_3))]\!]_\sigma &\triangleq \begin{cases}\text{true} & \text{if } [\![e_1]\!]_\sigma = \Lambda \wedge \Lambda\langle\_\rangle = ([\![e_2]\!]_\sigma, [\![e_3]\!]_\sigma) \\ \text{false} & \text{otherwise}\end{cases} \\
[\![\mathsf{findkey}(e_1,e_2)]\!]_\sigma &\triangleq \begin{cases}\text{true} & \text{if } [\![e_1]\!]_\sigma = \Lambda \wedge \Lambda\langle\_\rangle = ([\![e_2]\!]_\sigma, \_) \\ \text{false} & \text{otherwise}\end{cases} \\
[\![\mathsf{findval}(e_1,e_2)]\!]_\sigma &\triangleq \begin{cases}\text{true} & \text{if } [\![e_1]\!]_\sigma = \Lambda \wedge \Lambda\langle\_\rangle = (\_, [\![e_2]\!]_\sigma) \\ \text{false} & \text{otherwise}\end{cases}
\end{aligned}
$$

Fig. 27. Definitions in Ex. 2.

and

$$
\begin{aligned}
\vdash_{\mathrm{RT}}\ &[\mathsf{inp} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathrm{PRF}} \\
&[\exists X_1, X_2, Y.\ X_1 \neq X_2 \wedge \mathsf{find}(L,(X_1,Y)) \wedge \mathsf{find}(L,(X_2,Y))].
\end{aligned}
\tag{40}
$$

Then, by applying Thm. 7, it remains to prove

$$
\vdash_{\mathrm{RT}} \{\mathsf{inp} \wedge \mathsf{hdinit}\}\, C_{\mathrm{PRF}}^{\mathsf{bad}}\, \{\mathit{bad} = 1 \Rightarrow \mathbf{R}\} \tag{41}
$$

and

$$
\begin{aligned}
\vdash_{\mathrm{RT}}\ &[\mathsf{inp} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathrm{PRF}} \\
&[\exists X_1, X_2, Y.\ X_1 \neq X_2 \wedge \mathsf{find}(L,(X_1,Y)) \wedge \mathsf{find}(L,(X_2,Y))].
\end{aligned}
\tag{42}
$$

We use the following definitions:

$$
\begin{aligned}
l(i) &\triangleq \min\{m(j) : n_j = n_i\} \\
\mathsf{flx}(n) &\triangleq \textstyle\bigwedge_{i\in[1,n]} .\ \mathsf{find}(L,(x[i], \mathsf{RT}[1][l(i)-1])) \\
\mathsf{bad}(n) &\triangleq \mathit{bad} = 1 \wedge \mathsf{coll}(n) \\
\mathsf{good}(n) &\triangleq \mathit{bad} = 0 \wedge \mathsf{flx}(n) \wedge \mathsf{len}(L) = m(n)
\end{aligned}
$$

The proofs of (41) and (42) are sketched in Fig. 28 and Fig. 29 respectively.

I Witness-Tree-Like Structures

In this section, we give definitions and lemmas related to the following four witness-tree-like structures: witness trees [51], lopsided witness trees [51], strong witness trees [54], and independent set sequences [44].

```
{$\mathsf{inp} \wedge \mathsf{hdinit}$}
$L := []$; $d := 1$; $bad := 0$;
{$\mathsf{inp} \wedge \mathsf{hdinit} \wedge L = [] \wedge d = 1 \wedge bad = 0$}
{$\mathsf{inp} \wedge 1 \le d \le k+1 \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d-1) \vee \mathsf{bad}(d-1))$}
while $(d \le k)$ do
    {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d-1) \vee \mathsf{bad}(d-1))$}
    if $(\neg\mathsf{findkey}(L, x[d]))$ then
        {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d-1)$
                                        $\wedge \neg\mathsf{findkey}(L, x[d]) \vee \mathsf{bad}(d-1))$}
        {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d) - 1 \wedge l(d) = m(d)$
                                        $\wedge m(d) = m(d-1) + 1 \vee \mathsf{bad}(d-1))$}
        $y := \mathsf{Sample}(1)$;
        {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d) \wedge l(d) = m(d)$
                                        $\wedge m(d) = m(d-1) + 1 \wedge y = \mathsf{RT}[1][l(d)-1]$
                                        $\vee \mathsf{bad}(d-1))$}
        if $(\mathsf{findval}(L, y))$ then
            {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d) \wedge l(d) = m(d)$
                                            $\wedge y = \mathsf{RT}[1][l(d)-1] \wedge \mathsf{findval}(L, y)$
                                            $\vee \mathsf{bad}(d-1))$}
            {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d) \wedge y = \mathsf{RT}[1][l(d)-1]$
                                            $\wedge \mathsf{coll}(d) \vee \mathsf{bad}(d-1))$}
            $bad := 1$;
            {$\mathsf{inp} \wedge 1 \le d \le k \wedge \mathsf{bad}(d)$}
        {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d) \wedge m(d) = m(d-1) + 1$
                                        $\wedge y = \mathsf{RT}[1][l(d)-1] \vee \mathsf{bad}(d))$}
        $L := \mathsf{app}(L, (x[d], y))$;
        {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d) \wedge \mathsf{hd}_1 = m(d) \vee \mathsf{bad}(d))$}
    {$\mathsf{inp} \wedge 1 \le d \le k \wedge (\mathsf{good}(d) \wedge \mathsf{hd}_1 = m(d) \vee \mathsf{bad}(d))$}
    $d := d + 1$;
    {$\mathsf{inp} \wedge 1 \le d \le k+1 \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d-1) \vee \mathsf{bad}(d-1))$}
{$\mathsf{inp} \wedge d = k+1 \wedge (\mathsf{good}(d-1) \wedge \mathsf{hd}_1 = m(d-1) \vee \mathsf{bad}(d-1))$}
{$bad = 1 \Rightarrow \mathsf{coll}(k)$}
```

Fig. 28. Proof of (41).

I.1 Witness Trees

Below we define witness trees and prove some of their important properties.

We define WT, $WTMap$, $f_{\mathsf{WT}}, g_{\mathsf{WT}}$ in Fig. 30. The set of all witness trees, denoted as WT, is defined as follows, where $\{\cdots\}$ represents a multiset.

$$(WT)\ wt ::= (m, \{wt_1, \ldots, wt_n\})$$

Define $WTMap(K)$ as the set of all (proper) witness trees with size no more than K. Informally, a tree wt is "proper" iff

– For each node in wt, all of its child nodes have distinct labels from $[1, M]$.
– For each node in wt, if the node has label m, then all of its child nodes have labels from $\Gamma^+(m)$.

```
$[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge \mathsf{hdinit}]$
$L := []; \; d := 1;$
$[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge \mathsf{hdinit} \wedge L = [] \wedge d = 1]$
$[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k+1 \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d-1)]$
$\textbf{while } (d \leq k) \textbf{ do}$
    $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d-1) \wedge k+1-d = X]$
    $\textbf{if } (\neg \mathsf{findkey}(L, x[d])) \textbf{ then}$
        $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d-1)$
            $\wedge \neg \mathsf{findkey}(L, x[d]) \wedge k+1-d = X]$
        $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d) - 1$
            $\wedge l(d) = m(d) \wedge k+1-d = X]$
        $y := \mathsf{Sample}(1);$
        $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d)$
            $\wedge y = \mathsf{RT}[1][l(d)-1] \wedge k+1-d = X]$
        $L := \mathsf{app}(L, (x[d], y));$
        $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k \wedge \mathsf{flx}(d) \wedge \mathsf{hd}_1 = m(d) \wedge k+1-d = X]$
    $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k \wedge \mathsf{flx}(d) \wedge \mathsf{hd}_1 = m(d) \wedge k+1-d = X]$
    $d := d+1;$
    $[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge 1 \leq d \leq k+1 \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d-1) \wedge k+1-d+1 \leq X]$
$[\mathsf{inp} \wedge \mathsf{coll}(k) \wedge d = k+1 \wedge \mathsf{flx}(d-1) \wedge \mathsf{hd}_1 = m(d-1)]$
$[\exists X_1, X_2, Y.\ X_1 \neq X_2 \wedge \mathsf{find}(L, (X_1, Y)) \wedge \mathsf{find}(L, (X_2, Y))]$
```

Fig. 29. Proof of (42).

For execution log $\Lambda \in \mathit{ExLog}$, we define $f_{\mathsf{WT}}(\Lambda)$ as the witness tree constructed from Λ. Informally, $\mathsf{GPar}(\Lambda, i, j)$ holds iff the parent node of $\Lambda\langle i\rangle$ on the witness tree is $\Lambda\langle j\rangle$. To define $\mathsf{GPar}(\Lambda, i, j)$, we treat the witness tree as a longest path spanning tree of the "witness DAG". $\mathsf{GPath}(\Lambda, i, l)$ holds iff there exists a path of length l from $\Lambda\langle i\rangle$ to $\Lambda\langle|\Lambda|\rangle$ on the "witness DAG", and $\mathsf{GDep}(\Lambda, i, l)$ holds iff the longest such path is of length l.

For a witness tree $wt \in WT$, we define $g_{\mathsf{WT}}(wt)$ as a reversed BFS ordering of wt.

Other auxiliary definitions related to witness trees, for example the definition of the index version of $GWT(\Lambda, i)$ ($GWTI(\Lambda, i)$), are also given in Fig. 30.

The following lemmas capture the properties of witness trees.

Lemma 84. *For all $\Lambda \in \mathit{ExLog}, i, l$ and l', if $\mathsf{GDep}(\Lambda, i, l)$ and $\mathsf{GDep}(\Lambda, i, l')$, then $l = l'$.*

Proof. Let $\mathsf{GDep}(\Lambda, i, l)$ and $\mathsf{GDep}(\Lambda, i, l')$ hold. We prove by contradiction. Assume that $l \neq l'$. Without loss of generality, let $l < l'$, then by $\mathsf{GDep}(\Lambda, i, l')$ we have $\mathsf{GPath}(\Lambda, i, l')$, and by $\mathsf{GDep}(\Lambda, i, l)$ we know that $\neg\mathsf{GPath}(\Lambda, i, l'')$ for all $l'' > l$, which leads to a contradiction. Thus $l = l'$. □

Lemma 85. *For all $\Lambda \in \mathit{ExLog}, i, j$ and k, if $\mathsf{GPar}(\Lambda, i, j)$ and $\mathsf{GPar}(\Lambda, i, k)$, then $j = k$.*

$$(\mathit{WT})\ wt ::= (m, \{wt_1, \ldots, wt_n\})$$

$$\begin{aligned}
\mathit{WTMap}(K) &\triangleq \{wt \in \mathit{WT} \mid \mathsf{Proper}(wt) \wedge |wt| \leq K\}\\
|(m, \{wt_1, \ldots, wt_n\})| &\triangleq 1 + \textstyle\sum_{i\in[1,n]} |wt_i|\\
\mathit{root}((m, \{wt_1, \ldots, wt_n\})) &\triangleq m\\
\mathsf{Proper}((m, \{wt_1, \ldots, wt_n\})) \text{ iff } &\left(\textstyle\bigwedge_{i\in[1,n]} .\ \mathsf{Proper}(wt_i)\right)\\
&\wedge |\{\mathit{root}(wt_1), \ldots, \mathit{root}(wt_n)\}| = n\\
&\wedge m \in [1, M] \wedge \{\mathit{root}(wt_1), \ldots, \mathit{root}(wt_n)\} \subseteq \Gamma^+(m)
\end{aligned}$$

$$\begin{aligned}
f_{\mathsf{WT}}(\Lambda) &\triangleq \mathit{GWT}(\Lambda, |\Lambda|)\\
\mathit{GWT}(\Lambda, i) &\triangleq (\Lambda\langle i\rangle, \{\mathit{GWT}(\Lambda, j) \mid \mathsf{GPar}(\Lambda, j, i)\})
\end{aligned}$$

$$\begin{aligned}
g_{\mathsf{WT}}(wt) &\triangleq \mathit{LYWT}(\mathsf{id})(wt)\\
\mathit{LYWT}(h)(wt) &\triangleq \mathit{Lay}(h)(wt, \mathit{Hgh}(wt) - 1) \parallel \cdots \parallel \mathit{Lay}(h)(wt, 0)\\
\mathit{Hgh}((m, \{wt_1, \ldots, wt_n\})) &\triangleq 1 + \max\{\mathit{Hgh}(wt_i) \mid i \in [1, n]\}
\end{aligned}$$

$$\begin{aligned}
n \in \Lambda \text{ iff } &\exists i \in [1, |\Lambda|].\ \Lambda\langle i\rangle = n\\
\#_m(\Lambda) &\triangleq \textstyle\sum_{i\in[1,|\Lambda|]} [\Lambda\langle i\rangle = m]\\
\#_{m,\Lambda'}(\Lambda) &\triangleq \textstyle\sum_{i\in[1,|\Lambda|]} [\Lambda'\langle \Lambda\langle i\rangle\rangle = m]\\
\#_{m'}((m, \{wt_1, \ldots, wt_n\})) &\triangleq [m' = m] + \textstyle\sum_{i\in[1,n]} \#_{m'}(wt_i)\\
\mathit{GWTS}(\Lambda, i) &\triangleq i :: (\mathit{GWTS}(\Lambda, j_1) \parallel \cdots \parallel \mathit{GWTS}(\Lambda, j_n))\\
&\qquad \text{where } \{j_1, \ldots, j_n\} = \{j \mid \mathsf{GPar}(\Lambda, j, i)\}\\
\mathit{GWTI}(\Lambda, i) &\triangleq (i, \{\mathit{GWTI}(\Lambda, j) \mid \mathsf{GPar}(\Lambda, j, i)\})\\
\mathit{Lay}(h)(wt, l) &\triangleq \mathsf{seq}(h)(\mathit{LayS}(wt, l))\\
\mathsf{seq}(h)(I) &\triangleq i_n :: \cdots :: i_1 :: []\\
&\quad \text{where } I = \{i_1, \ldots, i_n\} \wedge (h(i_1), i_1) \leq \cdots \leq (h(i_n), i_n)\\
\mathit{LayS}((m, \{wt_1, \ldots, wt_n\}), l) &\triangleq \begin{cases} \{m\} \quad \text{(multiset)} & \text{if } l = 0\\ \mathit{LayS}(wt_1, l-1) \cup \cdots \cup \mathit{LayS}(wt_n, l-1) & \text{if } l \geq 1 \end{cases}
\end{aligned}$$

$$\frac{}{\mathsf{GPath}(\Lambda, |\Lambda|, 0)} \qquad \frac{i < j \leq |\Lambda| \qquad \Lambda\langle j\rangle \in \Gamma^+(\Lambda\langle i\rangle) \qquad \mathsf{GPath}(\Lambda, j, l)}{\mathsf{GPath}(\Lambda, i, l+1)}$$

$$\frac{\mathsf{GPath}(\Lambda, i, l) \qquad \forall l' > l.\ \neg\mathsf{GPath}(\Lambda, i, l')}{\mathsf{GDep}(\Lambda, i, l)}$$

$$\frac{\begin{array}{c}\mathsf{GDep}(\Lambda, i, l+1)\\ i < j \leq |\Lambda| \qquad \Lambda\langle j\rangle \in \Gamma^+(\Lambda\langle i\rangle) \qquad \mathsf{GPath}(\Lambda, j, l)\\ \forall k.\ i < k < j \wedge \Lambda\langle k\rangle \in \Gamma^+(\Lambda\langle i\rangle) \implies \neg\mathsf{GPath}(\Lambda, k, l)\end{array}}{\mathsf{GPar}(\Lambda, i, j)}$$

Fig. 30. Definitions related to witness trees.

Proof. Let $\mathsf{GPar}(\Lambda, i, j)$ and $\mathsf{GPar}(\Lambda, i, k)$ hold, then $i < j$ and $i < k$. Without loss of generality, assume that $k \leq j$. By definition, there exist l and l' such that:

$$\Lambda\langle k\rangle \in \Gamma^+(\Lambda\langle i\rangle), \mathsf{GPath}(\Lambda, j, l), \mathsf{GDep}(\Lambda, i, l+1),$$

$$\mathsf{GPath}(\Lambda, k, l'), \mathsf{GDep}(\Lambda, i, l'+1),$$

and $\neg\mathsf{GPath}(\Lambda, j', l)$ for all j' such that $\Lambda\langle j'\rangle \in \Gamma^+(\Lambda\langle i\rangle)$ and $i < j' < j$. By Lem. 84 we know that $l = l'$, and thus $j = k$. □

Lemma 86. *For all $\Lambda \in ExLog$ and $i \in [1, |\Lambda|]$, $root(GWT(\Lambda, i)) = \Lambda\langle i\rangle$.*

Proof. By definition. □

Lemma 87. *For all $\Lambda \in ExLog, i$, taking $\Lambda' = GWTS(\Lambda, i)$, if $\mathsf{GPath}(\Lambda, i, l)$ holds for some l, then*

- *For all $j \in [1, |\Lambda'|]$, $\Lambda'\langle j\rangle \in [1, i]$;*
- *For all $j \neq k \in [1, |\Lambda'|]$, $\Lambda'\langle j\rangle \neq \Lambda'\langle k\rangle$;*
- *For all $j \in [1, |\Lambda'|]$, either $\Lambda'\langle j\rangle = i$ or there exists $k \in [j+1, |\Lambda'|]$ such that $\mathsf{GPar}(\Lambda, \Lambda'\langle j\rangle, \Lambda'\langle k\rangle))$.*

Proof. We prove by induction on i. The first and the third properties are trivial. Below we prove the second property by contradiction. Assume that there exist $j \neq k \in [1, |\Lambda'|]$ such that $\Lambda'\langle j\rangle = \Lambda'\langle k\rangle$, and (j, k) has the largest lexicographical order among such pairs. From the induction hypothesis, there exist $j' \neq k' < i$ such that $\Lambda'\langle j\rangle \in GWTS(\Lambda, j')$, $\Lambda'\langle k\rangle \in GWTS(\Lambda, k')$, $\mathsf{GPar}(\Lambda, j', i)$ and $\mathsf{GPar}(\Lambda, k', i)$. If $\Lambda'\langle j\rangle = j'$ and $\Lambda'\langle k\rangle = k'$, then it leads to a contradiction. If $\Lambda'\langle j\rangle = j'$ and $\Lambda'\langle k\rangle \neq k'$, then from the induction hypothesis there exists $i' \in [1, k']$ such that $\mathsf{GPar}(\Lambda, \Lambda'\langle k\rangle, i')$, but from $\mathsf{GPar}(\Lambda, j', i)$ we know that $\mathsf{GPar}(\Lambda, \Lambda'\langle k\rangle, i)$, which contradicts Lem. 85. The case of $\Lambda'\langle k\rangle = k'$ is similar. If $\Lambda'\langle j\rangle \neq j'$, $\Lambda'\langle k\rangle \neq k'$, then from the induction hypothesis we know that there exist $j'' > j$ and $k'' > k$ such that $\mathsf{GPar}(\Lambda, \Lambda'\langle j\rangle, \Lambda'\langle j''\rangle)$ and $\mathsf{GPar}(\Lambda, \Lambda'\langle k\rangle, \Lambda'\langle k''\rangle)$, and then by Lem. 85 this implies $\Lambda'\langle j''\rangle = \Lambda'\langle k''\rangle$, which contradicts that (j, k) has the largest lexicographical order. □

Lemma 88. *For all $\Lambda \in ExLog$, $|f_{\mathsf{WT}}(\Lambda)| \leq |\Lambda|$.*

Proof. From Lem. 87 and $\mathsf{GPath}(\Lambda, |\Lambda|, 0)$, $|GWTS(\Lambda, |\Lambda|)| \leq |\Lambda|$. Then by induction, for all i we have $|GWT(\Lambda, i)| = |GWTS(\Lambda, i)|$, and thus $|f_{\mathsf{WT}}(\Lambda)| = |GWT(\Lambda, |\Lambda|)| = |GWTS(\Lambda, |\Lambda|)| \leq |\Lambda|$. □

Lemma 89. *For all $\Lambda \in ExLog, i, j$ and l, if $\mathsf{GPar}(\Lambda, i, j)$, then*

$$\mathsf{GDep}(\Lambda, j, l) \iff \mathsf{GDep}(\Lambda, i, l+1).$$

Proof. Assuming $\mathsf{GPar}(\Lambda, i, j)$, we know that there exists l' such that $\mathsf{GPath}(\Lambda, j, l')$, $\mathsf{GDep}(\Lambda, i, l'+1)$, $i < j \leq |\Lambda|$, and $\Lambda\langle j\rangle \in \Gamma^+(\Lambda\langle i\rangle)$.

If $\mathsf{GDep}(\Lambda, j, l)$, then by $\mathsf{GPath}(\Lambda, j, l')$ we know that $l' \leq l$. Since $\mathsf{GPath}(\Lambda, i, l+1)$, from $\mathsf{GDep}(\Lambda, i, l'+1)$ we know that $l \leq l'$, thus $l = l'$ and $\mathsf{GDep}(\Lambda, i, l+1)$.

If $\mathsf{GDep}(\Lambda, i, l+1)$, then from Lem. 84 we have $l = l'$, and thus $\mathsf{GPath}(\Lambda, j, l)$. If $\mathsf{GPath}(\Lambda, j, l'')$ for some $l'' > l$, then from $\Lambda\langle j\rangle \in \Gamma^+(\Lambda\langle i\rangle)$ we know that $\mathsf{GPath}(\Lambda, i, l''+1)$, which contradicts $\mathsf{GDep}(\Lambda, i, l'+1)$. Thus $\mathsf{GDep}(\Lambda, j, l)$. □

Lemma 90. *For all $\Lambda \in ExLog, i, j$ and k, if $j \neq k$, $\mathsf{GPar}(\Lambda, j, i)$ and $\mathsf{GPar}(\Lambda, k, i)$, then $\Lambda\langle k\rangle \notin \Gamma^+(\Lambda\langle j\rangle)$.*

Proof. We prove by contradiction. Let $\Lambda\langle k\rangle \in \Gamma^+(\Lambda\langle j\rangle)$. Without loss of generality, let $j < k$. Assume that $\mathsf{GPar}(\Lambda, j, i)$ and $\mathsf{GPar}(\Lambda, k, i)$ hold, then by definition we know that there must exist l and l' such that $\mathsf{GPath}(\Lambda, i, l)$, $\mathsf{GDep}(\Lambda, j, l+1)$, $\mathsf{GPath}(\Lambda, i, l')$ and $\mathsf{GDep}(\Lambda, k, l'+1)$. From Lem. 89, this implies $\mathsf{GDep}(\Lambda, i, l)$ and $\mathsf{GDep}(\Lambda, i, l')$, then from Lem. 84 we have $l = l'$. Thus, since $\mathsf{GDep}(\Lambda, j, l'+1)$, for all $l'' > l'+1$ we have $\neg\mathsf{GPath}(\Lambda, j, l'')$. However, by $\mathsf{GDep}(\Lambda, k, l'+1)$ and $\Lambda\langle k\rangle \in \Gamma^+(\Lambda\langle j\rangle)$ we have $\mathsf{GPath}(\Lambda, j, l'+2)$, which leads to a contradiction. Thus $\Lambda\langle k\rangle \notin \Gamma^+(\Lambda\langle j\rangle)$. □

Lemma 91. *For all $\Lambda \in ExLog, i$ and $l > |\Lambda|$, $\neg\mathsf{GPath}(\Lambda, i, l)$.*

Proof. By definition. □

Lemma 92. *For all $\Lambda \in ExLog, i$ and l, if $\mathsf{GPath}(\Lambda, i, l)$, then $i \in GWTS(\Lambda, |\Lambda|)$.*

Proof. Assume that $\mathsf{GPath}(\Lambda, i, l)$, then from Lem. 91 there exists $l' \geq l$ such that $\mathsf{GDep}(\Lambda, i, l')$. By Lem. 89 and induction, there exist $|\Lambda| = i_0, i_1, \ldots, i_{l'} = i$ such that: $\mathsf{GDep}(\Lambda, i_{l''}, l'')$ holds for all $l'' \in [0, l']$, and $\mathsf{GPar}(\Lambda, i_{l''+1}, i_{l''})$ holds for all $l'' \in [0, l')$. By induction we know that $i \in GWTS(\Lambda, i_{l''})$ holds for all $l'' \in [0, l']$, and thus $i \in GWTS(\Lambda, |\Lambda|)$. □

Lemma 93. *For all $\Lambda \in ExLog$,*

$$\#_{\Lambda\langle|\Lambda|\rangle}(GWT(\Lambda, |\Lambda|)) = \#_{\Lambda\langle|\Lambda|\rangle}(\Lambda).$$

Proof. By induction, we can prove that: for all m and $i \in [1, |\Lambda|]$, $\#_m(GWT(\Lambda, i)) = \#_{m,\Lambda}(GWTS(\Lambda, i))$. Below we only need to prove that $\#_{\Lambda\langle|\Lambda|\rangle,\Lambda}(GWTS(\Lambda, |\Lambda|)) = \#_{\Lambda\langle|\Lambda|\rangle}(\Lambda)$. From Lem. 87, it remains to prove that, for all i such that $\Lambda\langle i\rangle = \Lambda\langle|\Lambda|\rangle$ we have $i \in GWTS(\Lambda, |\Lambda|)$. Assuming that $\{i \mid \Lambda\langle i\rangle = \Lambda\langle|\Lambda|\rangle\} = \{i_0, \ldots, i_l\}$ where $|\Lambda| = i_0 > \cdots > i_l$, by induction we know that, $\mathsf{GPath}(\Lambda, i_{l'}, l')$ for all $l' \in [0, l]$, and thus for all i satisfying $\Lambda\langle i\rangle = \Lambda\langle|\Lambda|\rangle$ there exists l' such that $\mathsf{GPath}(\Lambda, i, l')$, and then from Lem. 92 we obtain that $i \in GWTS(\Lambda, |\Lambda|)$. □

Lemma 94. *For all $\Lambda \in ExLog$ and i,*

$$\#_i(LYWT(\lambda i.\, \Lambda\langle i\rangle)(GWTI(\Lambda, |\Lambda|))) \leq 1.$$

Proof. By induction,

$$\#_i(LYWT(\lambda i.\, \Lambda\langle i\rangle)(GWTI(\Lambda, |\Lambda|))) = \#_i(GWTS(\Lambda, |\Lambda|)),$$

and then from Lem. 87 we have $\#_i(GWTS(\Lambda, |\Lambda|)) \leq 1$. Thus $\#_i(LYWT(\lambda i.\, \Lambda\langle i\rangle)(GWTI(\Lambda, |\Lambda|))) \leq 1$. □

Lemma 95. *For all $\Lambda \in$ ExLog, i and l, if* $\mathsf{GPath}(\Lambda, i, l)$*, then* $i \in \mathit{Lay}(\lambda i.\, \Lambda\langle i\rangle)(\mathit{GWTI}(\Lambda, |\Lambda|), l')$ *for some l'.*

Proof. Assuming that $\mathsf{GPath}(\Lambda, i, l)$, from Lem. 91 we know that there exists $l' \geq l$ such that $\mathsf{GDep}(\Lambda, i, l')$. By Lem. 89 and induction, there exist $|\Lambda| = i_0, i_1, \ldots, i_{l'} = i$ such that: $\mathsf{GDep}(\Lambda, i_{l''}, l'')$ for all $l'' \in [0, l']$, and $\mathsf{GPar}(\Lambda, i_{l''+1}, i_{l''})$ for all $l'' \in [0, l')$. By induction, we can prove that

$$i_{l'} \in \mathit{Lay}(\lambda i.\, \Lambda\langle i\rangle)(\mathit{GWTI}(\Lambda, i_{l''}), l' - l'')$$

for all $l'' \leq l'$. Thus $i \in \mathit{Lay}(\lambda i.\, \Lambda\langle i\rangle)(\mathit{GWTI}(\Lambda, |\Lambda|), l')$. □

Lemma 96. *For all $\Lambda \in$ ExLog, l and i, if*

$$i \in \mathit{Lay}(\lambda i.\, \Lambda\langle i\rangle)(\mathit{GWTI}(\Lambda, |\Lambda|), l),$$

then $\mathsf{GDep}(\Lambda, i, l)$.

Proof. For all $j \in |\Lambda|$, l and $i \in \mathit{Lay}(\lambda i.\, \Lambda\langle i\rangle)(\mathit{GWTI}(\Lambda, j), l)$, by induction on l we can prove that, there exist $j = i_0, i_1, \ldots, i_l = i$ such that $\mathsf{GPar}(\Lambda, i_{l'}, i_{l'+1})$ holds for all $l' \in [0, l)$. Thus for $i \in \mathit{Lay}(\lambda i.\, \Lambda\langle i\rangle)(\mathit{GWTI}(\Lambda, |\Lambda|), l)$, by Lem. 89 and induction we have $\mathsf{GDep}(\Lambda, i, l)$. □

Lemma 97. *For all wt and h, $|\mathit{LYWT}(h)(wt)| = |wt|$.*

Proof. By induction on the structure of wt. □

Lemma 98. *For all $\Lambda \in$ ExLog and K such that $|\Lambda| \leq K$,*

$$f_{\mathsf{WT}}(\Lambda) \in \mathit{WTMap}(K).$$

Proof. Let $\Lambda \in \mathit{ExLog}$ satisfy $|\Lambda| \leq K$, then by Lem. 88 we know that $|f_{\mathsf{WT}}(\Lambda)| \leq |\Lambda| \leq K$. From Lem. 86 and Lem. 90, by induction we have $\mathsf{Proper}(f_{\mathsf{WT}}(\Lambda))$. Thus by definition we obtain that $f_{\mathsf{WT}}(\Lambda) \in \mathit{WTMap}(K)$. □

Lemma 99. *For all $\Lambda \prec \Lambda' \in$ ExLog,*

$$f_{\mathsf{WT}}(\Lambda) \neq f_{\mathsf{WT}}(\Lambda').$$

Proof. We show that $\mathit{GWT}(\Lambda, |\Lambda|) \neq \mathit{GWT}(\Lambda', |\Lambda'|)$. If $\Lambda\langle|\Lambda|\rangle \neq \Lambda'\langle|\Lambda'|\rangle$, then $\mathit{GWT}(\Lambda, |\Lambda|) \neq \mathit{GWT}(\Lambda', |\Lambda'|)$, otherwise it contradicts Lem. 86. If $\Lambda\langle|\Lambda|\rangle = \Lambda'\langle|\Lambda'|\rangle$, then by $\Lambda \prec \Lambda'$ we have $\#_{\Lambda\langle|\Lambda|\rangle}(\Lambda) = \sum_{i\in[1,|\Lambda|]}[\Lambda\langle i\rangle = \Lambda'\langle|\Lambda'|\rangle] < \#_{\Lambda'\langle|\Lambda'|\rangle}(\Lambda')$. Thus, from Lem. 93 we get $\mathit{GWT}(\Lambda, |\Lambda|) \neq \mathit{GWT}(\Lambda', |\Lambda'|)$. □

Lemma 100. *For all $\Lambda, \Lambda' \in$ ExLog, if*

$$(g_{\mathsf{WT}} \circ f_{\mathsf{WT}})(\Lambda) = \Lambda',$$

then for each $l \in [1, |\Lambda'|]$ there exists k such that $\Lambda\langle k\rangle = \Lambda'\langle l\rangle$ and

$$\sum_{k'<k}[\mathrm{vbl}(\Lambda\langle k'\rangle, i)] = \sum_{l'<l}[\mathrm{vbl}(\Lambda'\langle l'\rangle, i)]$$

for all $i \in [1, N]$ such that $\mathrm{vbl}(\Lambda'\langle l\rangle, i)$.

Proof. Suppose that $f_{\mathsf{WT}}(\Lambda) = wt$, $g_{\mathsf{WT}}(wt) = \Lambda'$ and $l \in [1, |\Lambda'|]$. Taking $\Lambda'' = LYWT(\lambda i.\, \Lambda\langle i\rangle)(GWTI(\Lambda, |\Lambda|))$, by induction we have $|\Lambda'| = |\Lambda''|$, and for all $l' \in [1, |\Lambda'|]$ we have $\Lambda'\langle l'\rangle = \Lambda\langle \Lambda''\langle l'\rangle\rangle$. Take $j = \Lambda''\langle l\rangle$, then by Lem. 96 there exists l'' such that $\mathsf{GPath}(\Lambda, j, l'')$. For $i \in [1, N]$ such that $\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l\rangle], i)$, we only need to prove that

$$\sum_{j'<j} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle j'\rangle], i)] = \sum_{l'<l} [\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)].$$

From Lem. 94, we know that all elements in Λ'' are distinct, and from Lem. 95 we have $j' \in \Lambda''$ for all $j' < j$ such that $\mathrm{vbl}(\mathcal{E}[\Lambda\langle j'\rangle], i)$ (which implies $\mathsf{GPath}(\Lambda, j', l'' + 1)$). Thus

$$\sum_{j'<j} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle j'\rangle], i)] = \sum_{l' : \Lambda''\langle l'\rangle < \Lambda''\langle l\rangle} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle \Lambda''\langle l'\rangle\rangle], i)],$$

and it remains to prove the following: for all l', if $\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)$, then

$$l' < l \iff \Lambda''\langle l'\rangle < \Lambda''\langle l\rangle.$$

For $l' < l$ such that $\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)$, we prove $\Lambda''\langle l'\rangle < \Lambda''\langle l\rangle$ by contradiction. Assume that $\Lambda''\langle l'\rangle > \Lambda''\langle l\rangle$. Since $l' < l$, there must exist $d' \geq d$ such that $\Lambda''\langle l'\rangle \in Lay(\lambda i.\, \Lambda\langle i\rangle)(GWTI(\Lambda, |\Lambda|), d')$ and $\Lambda''\langle l\rangle \in Lay(\lambda i.\, \Lambda\langle i\rangle)(GWTI(\Lambda, |\Lambda|), d)$. By Lem. 96, this implies $\mathsf{GDep}(\Lambda, \Lambda''\langle l'\rangle, d')$ and $\mathsf{GDep}(\Lambda, \Lambda''\langle l\rangle, d)$, and thus from $\Lambda''\langle l'\rangle > \Lambda''\langle l\rangle$ and $\Lambda\langle \Lambda''\langle l'\rangle\rangle \in \Gamma^+(\Lambda\langle \Lambda''\langle l\rangle\rangle)$ we know that $\mathsf{GPath}(\Lambda, \Lambda''\langle l\rangle, d' + 1)$, a contradiction. Therefore $l' < l \implies \Lambda''\langle l'\rangle < \Lambda''\langle l\rangle$. Similarly, if $\Lambda''\langle l'\rangle < \Lambda''\langle l\rangle$, one can show that $l' < l$. □

Lemma 101. *For all reals $\alpha_1, \ldots, \alpha_M \in (0, 1)$, if the Erdös-Lovász condition*

$$\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \leq \alpha_i \prod_{j \in \Gamma(i)} (1 - \alpha_j)$$

holds, then for all K we have

$$\sum_{wt \in WTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \leq r_{\mathsf{EL}},$$

where

$$r_{\mathsf{EL}} = \sum_{i \in [1,M]} \alpha_i (1 - \alpha_i)^{-1}.$$

Proof. From the Erdös-Lovász condition, we only need to prove that, for all K,

$$\sum_{wt \in WTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \cdot \prod_{j \in \Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1 - \alpha_j) \leq \sum_{k \in [1,M]} \alpha_k \cdot (1 - \alpha_k)^{-1}.$$

Define $h_j(K) = \{wt \in \mathit{WTMap}(K) \mid \mathit{root}(wt) = j\}$, then we only need to prove that, for all $k \in [1, M]$ and K,

$$\sum_{wt \in h_k(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j \in \Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j) \leq \alpha_k \cdot (1-\alpha_k)^{-1}.$$

We prove by induction on K. The case of $K = 0$ if trivial. For the induction step,

$$\begin{aligned}
&\sum_{wt \in h_k(K+1)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j \in \Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j) \\
&= \sum_{(k,\{wt_1,\ldots,wt_n\}) \in h_k(K+1)} \Bigg(\bigg(\alpha_k \prod_{j \in \Gamma(k)} (1-\alpha_j)\bigg) \\
&\quad \cdot \prod_{l \in [1,n]} \prod_{i=1}^{|g_{\mathsf{WT}}(wt_l)|} \alpha_{g_{\mathsf{WT}}(wt_l)[i]} \prod_{j \in \Gamma(g_{\mathsf{WT}}(wt_l)[i])} (1-\alpha_j)\Bigg) \\
&\leq \alpha_k \cdot (1-\alpha_k)^{-1} \prod_{j \in \Gamma^+(k)} (1-\alpha_j) \\
&\quad \cdot \left(1 + \sum_{wt \in h_j(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{l \in \Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_l)\right) \\
&\leq \alpha_k \cdot (1-\alpha_k)^{-1} \prod_{j \in \Gamma^+(k)} (1-\alpha_j)\left(1 + \alpha_j \cdot (1-\alpha_j)^{-1}\right) \\
&= \alpha_k \cdot (1-\alpha_k)^{-1}.
\end{aligned}$$

□

Lemma 102. *For all reals $\alpha_1, \ldots, \alpha_M, \epsilon \in (0,1)$, if the Erdös-Lovász condition (with ϵ slack)*

$$\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \leq (1-\epsilon)\alpha_i \prod_{j \in \Gamma(i)} (1-\alpha_j)$$

holds, then for all K and m we have

$$\sum_{\substack{wt \in \mathit{WTMap}(K) \\ |wt| \geq m}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \leq (1-\epsilon)^m r_{\mathsf{EL}},$$

where

$$r_{\mathsf{EL}} = \sum_{i \in [1,M]} \alpha_i (1-\alpha_i)^{-1}.$$

Proof. From the Erdös-Lovász condition (with ϵ slack), Lem. 97 and Lem. 101, for all K and m,

$$\begin{aligned}
&\sum_{\substack{wt\in WTMap(K)\\|wt|\geq m}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\\
\leq &\sum_{\substack{wt\in WTMap(K)\\|wt|\geq m}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} (1-\epsilon)\alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j\in\Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j)\\
= &\sum_{\substack{wt\in WTMap(K)\\|wt|\geq m}} (1-\epsilon)^{|g_{\mathsf{WT}}(wt)|} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j\in\Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j)\\
\leq &(1-\epsilon)^m \sum_{\substack{wt\in WTMap(K)\\|wt|\geq m}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j\in\Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j)\\
\leq &(1-\epsilon)^m \sum_{wt\in WTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j\in\Gamma(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j)\\
\leq &(1-\epsilon)^m \sum_{i\in[1,M]} \alpha_i(1-\alpha_i)^{-1}.
\end{aligned}$$

□

Lemma 103. *For all reals* $\alpha_1,\ldots,\alpha_{M-1}\in(0,1)$*, if*

$$\forall i\in[1,M).\ \mathrm{P}(\mathcal{E}[i])\leq\alpha_i \prod_{j\in\Gamma(i)\setminus\{M\}}(1-\alpha_j),$$

then for all K *we have*

$$\sum_{\substack{wt\in WTMap(K)\\\#_M(wt)=0}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\leq r_{\mathsf{HSS}},$$

where

$$r_{\mathsf{HSS}}=\sum_{i\in[1,M)}\alpha_i(1-\alpha_i)^{-1},$$

Proof. Similar to Lem. 101. □

Lemma 104. *For all reals* $\alpha_1,\ldots,\alpha_{M-1}\in(0,1)$*, if*

$$\forall i\in[1,M).\ \mathrm{P}(\mathcal{E}[i])\leq\alpha_i \prod_{j\in\Gamma(i)\setminus\{M\}}(1-\alpha_j),$$

then for all K we have

$$\sum_{\substack{wt\in WTMap(K)\\ root(wt)=M\wedge \#_M(wt)=1}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \le \gamma_{\mathsf{HSS}},$$

where

$$\gamma_{\mathsf{HSS}} = \mathrm{P}(\mathcal{E}[M]) \prod_{i\in\mathit{\Gamma}(M)} (1-\alpha_i)^{-1}.$$

Proof. The case of $K=0$ is trivial. For $K\ge 1$,

$$\begin{aligned}
&\sum_{\substack{wt\in WTMap(K)\\ root(wt)=M\wedge \#_M(wt)=1}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\\
&= \sum_{\substack{(M,wt_1,\dots,wt_n)\in WTMap(K)\\ \#_M(wt_1)=\dots=\#_M(wt_n)=0}} \mathrm{P}(\mathcal{E}[M])\cdot \prod_{l\in[1,n]} \prod_{i=1}^{|g_{\mathsf{WT}}(wt_l)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt_l)\langle i\rangle])\\
&\le \mathrm{P}(\mathcal{E}[M]) \prod_{j\in\mathit{\Gamma}(M)} \left(1+ \prod_{\substack{wt\in WTMap(K-1)\\ root(wt)=j\wedge \#_M(wt)=0}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\right)\\
&\le \mathrm{P}(\mathcal{E}[M]) \prod_{j\in\mathit{\Gamma}(M)} \left(1+\alpha_j(1-\alpha_j)^{-1}\right) = \gamma_{\mathsf{HSS}}.
\end{aligned}$$

The last inequality can be derived in a similar way as in Lem. 103. □

I.2 Lopsided Witness Trees

In this subsection, we define lopsided witness trees, and prove some of their important properties. The lopsided witness tree is similar to the witness tree in App. I.1, with the only difference being that, for each node with label m, all of its child nodes have labels from $\mathit{\Gamma}'^{+}(m)$, not $\mathit{\Gamma}^{+}(m)$.

We define $\mathit{LWTMap}, f_{\mathsf{LWT}}$ in Fig. 31. $\mathit{LWTMap}(K)$ is the set of all (proper) lopsided witness trees with size no more than K. Informally, a lopsided witness tree wt is "proper" iff

– For each node in wt, all of its child nodes have distinct labels from $[1,M]$.
– For each node in wt, if the node has label m, then all of its child nodes have labels from $\mathit{\Gamma}'^{+}(m)$.

For execution log $\mathit{\Lambda}\in \mathit{ExLog}$, we define $f_{\mathsf{LWT}}(\mathit{\Lambda})$ as the lopsided witness tree constructed from $\mathit{\Lambda}$, which is similar to f_{WT} in App. I.1.

The following lemmas capture the properties of lopsided witness trees.

Lemma 105. *For all $\mathit{\Lambda}\in \mathit{ExLog}, i, l$ and l', if $\mathsf{GDep}'(\mathit{\Lambda},i,l)$ and $\mathsf{GDep}'(\mathit{\Lambda},i,l')$, then $l=l'$.*

$$
\begin{aligned}
\mathit{LWTMap}(K) &\triangleq \{wt \in WT \mid \mathsf{Proper}'(wt) \wedge |wt| \le K\} \\
\mathsf{Proper}'((m, \{wt_1, \dots, wt_n\})) \text{ iff } & \left(\bigwedge\nolimits_{i \in [1,n]} . \mathsf{Proper}'(wt_i)\right) \\
& \wedge |\{root(wt_1), \dots, root(wt_n)\}| = n \\
& \wedge m \in [1, M] \wedge \{root(wt_1), \dots, root(wt_n)\} \subseteq \Gamma'^{+}(m)
\end{aligned}
$$

$$
\begin{aligned}
f_{\mathsf{LWT}}(\Lambda) &\triangleq GWT'(\Lambda, |\Lambda|) \\
GWT'(\Lambda, i) &\triangleq (\Lambda\langle i\rangle, \{GWT'(\Lambda, j) \mid \mathsf{GPar}'(\Lambda, j, i)\})
\end{aligned}
$$

$$
\begin{aligned}
GWTS'(\Lambda, i) &\triangleq i :: (GWTS'(\Lambda, j_1) \parallel \cdots \parallel GWTS'(\Lambda, j_n)) \\
&\qquad \text{where } \{j_1, \dots, j_n\} = \{j \mid \mathsf{GPar}'(\Lambda, j, i)\} \\
GWTI'(\Lambda, i) &\triangleq (i, \{GWTI'(\Lambda, j) \mid \mathsf{GPar}'(\Lambda, j, i)\})
\end{aligned}
$$

$$
\frac{}{\mathsf{GPath}'(\Lambda, |\Lambda|, 0)} \qquad \frac{i < j \le |\Lambda| \qquad \Lambda\langle j\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle) \qquad \mathsf{GPath}'(\Lambda, j, l)}{\mathsf{GPath}'(\Lambda, i, l+1)}
$$

$$
\frac{\mathsf{GPath}'(\Lambda, i, l) \qquad \forall l' > l.\ \neg\mathsf{GPath}'(\Lambda, i, l')}{\mathsf{GDep}'(\Lambda, i, l)}
$$

$$
\frac{\begin{array}{c}\mathsf{GDep}'(\Lambda, i, l+1) \\ i < j \le |\Lambda| \qquad \Lambda\langle j\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle) \qquad \mathsf{GPath}'(\Lambda, j, l) \\ \forall k.\ i < k \wedge \Lambda\langle k\rangle < \Lambda\langle j\rangle \wedge \Lambda\langle k\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle) \implies \neg\mathsf{GPath}'(\Lambda, k, l)\end{array}}{\mathsf{GPar}'(\Lambda, i, j)}
$$

Fig. 31. Definitions related to lopsided witness trees.

Proof. Similar to Lem. 84. □

Lemma 106. *For all $\Lambda \in ExLog, i, j$ and k, if $\mathsf{GPar}'(\Lambda, i, j)$ and $\mathsf{GPar}'(\Lambda, i, k)$, then $j = k$.*

Proof. Similar to Lem. 85. □

Lemma 107. *For all $\Lambda \in ExLog$ and $i \in [1, |\Lambda|]$,*

$$root(GWT'(\Lambda, i)) = \Lambda\langle i\rangle.$$

Proof. By definition. □

Lemma 108. *For all $\Lambda \in ExLog$ and i, taking $\Lambda' = GWTS'(\Lambda, i)$, if $\mathsf{GPath}'(\Lambda, i, l)$ holds for some l, then*

- *For all $j \in [1, |\Lambda'|]$, $\Lambda'\langle j\rangle \in [1, i]$;*
- *For all $j \ne k \in [1, |\Lambda'|]$, $\Lambda'\langle j\rangle \ne \Lambda'\langle k\rangle$;*
- *For all $j \in [1, |\Lambda'|]$, either $\Lambda'\langle j\rangle = i$ or there exists $k \in [j+1, |\Lambda'|]$ such that $\mathsf{GPar}'(\Lambda, \Lambda'\langle j\rangle, \Lambda'\langle k\rangle))$.*

Proof. Similar to Lem. 87. □

Lemma 109. *For all $\Lambda \in ExLog$, $|f_{\mathsf{LWT}}(\Lambda)| \le |\Lambda|$.*

Proof. Similar to Lem. 88. □

Lemma 110. *For all $\mathit{\Lambda} \in \mathit{ExLog}, i, j$ and l, if $\mathsf{GPar}'(\mathit{\Lambda}, i, j)$, then $\mathsf{GDep}'(\mathit{\Lambda}, j, l) \iff \mathsf{GDep}'(\mathit{\Lambda}, i, l+1)$.*

Proof. Similar to Lem. 89. □

Lemma 111. *For all $\mathit{\Lambda} \in \mathit{ExLog}, i, j, k$, if $j \neq k$, $\mathsf{GPar}'(\mathit{\Lambda}, j, i)$ and $\mathsf{GPar}'(\mathit{\Lambda}, k, i)$, then $\mathit{\Lambda}\langle k\rangle \notin \mathit{\Gamma}'^{+}(\mathit{\Lambda}\langle j\rangle)$.*

Proof. Similar to Lem. 90. □

Lemma 112. *For all $\mathit{\Lambda} \in \mathit{ExLog}, i$ and $l > |\mathit{\Lambda}|$, $\neg\mathsf{GPath}'(\mathit{\Lambda}, i, l)$.*

Proof. By definition. □

Lemma 113. *For all $\mathit{\Lambda} \in \mathit{ExLog}, i$ and l, if $\mathsf{GPath}'(\mathit{\Lambda}, i, l)$, then $i \in \mathit{GWTS}'(\mathit{\Lambda}, |\mathit{\Lambda}|)$.*

Proof. Similar to Lem. 92. □

Lemma 114. *For all $\mathit{\Lambda} \in \mathit{ExLog}$,*

$$\#_{\mathit{\Lambda}\langle|\mathit{\Lambda}|\rangle}(\mathit{GWT}'(\mathit{\Lambda}, |\mathit{\Lambda}|)) = \#_{\mathit{\Lambda}\langle|\mathit{\Lambda}|\rangle}(\mathit{\Lambda}).$$

Proof. Similar to Lem. 93. □

Lemma 115. *For all $\mathit{\Lambda} \in \mathit{ExLog}$ and i,*

$$\#_i(\mathit{LYWT}(\lambda i.\, \mathit{\Lambda}\langle i\rangle)(\mathit{GWTI}'(\mathit{\Lambda}, |\mathit{\Lambda}|))) \leq 1.$$

Proof. Similar to Lem. 94. □

Lemma 116. *For all $\mathit{\Lambda} \in \mathit{ExLog}, i$ and l, if $\mathsf{GPath}'(\mathit{\Lambda}, i, l)$, then $i \in \mathit{Lay}(\lambda i.\, \mathit{\Lambda}\langle i\rangle)(\mathit{GWTI}'(\mathit{\Lambda}, |\mathit{\Lambda}|), l')$ for some l'.*

Proof. Similar to Lem. 95. □

Lemma 117. *For all $\mathit{\Lambda} \in \mathit{ExLog}, l$ and i, if*

$$i \in \mathit{Lay}(\lambda i.\, \mathit{\Lambda}\langle i\rangle)(\mathit{GWTI}'(\mathit{\Lambda}, |\mathit{\Lambda}|), l),$$

then $\mathsf{GDep}'(\mathit{\Lambda}, i, l)$.

Proof. Similar to Lem. 96. □

Lemma 118. *For all $\mathit{\Lambda}, \mathit{\Lambda}', j, i$ and l such that $j \in [1, |\mathit{\Lambda}| - 1)$, $i \in [1, |\mathit{\Lambda}|]$ and $|\mathit{\Lambda}| = |\mathit{\Lambda}'|$, if*

- *For all $k \in [1, |\mathit{\Lambda}|] \setminus \{j, j+1\}$, $\mathit{\Lambda}\langle k\rangle = \mathit{\Lambda}'\langle k\rangle$;*
- *$\mathit{\Lambda}\langle j\rangle = \mathit{\Lambda}'\langle j+1\rangle$, $\mathit{\Lambda}\langle j+1\rangle = \mathit{\Lambda}'\langle j\rangle$;*
- *$\mathit{\Lambda}\langle j\rangle \notin \mathit{\Gamma}'^{+}(\mathit{\Lambda}\langle j+1\rangle)$;*

then

– *If* $i \notin \{j, j+1\}$*, then* $\mathsf{GPath}'(\varLambda, i, l) \iff \mathsf{GPath}'(\varLambda', i, l)$*;*
– $\mathsf{GPath}'(\varLambda, j, l) \iff \mathsf{GPath}'(\varLambda', j+1, l)$*;*
– $\mathsf{GPath}'(\varLambda, j+1, l) \iff \mathsf{GPath}'(\varLambda', j, l)$*.*

Proof. The case of $i = |\varLambda|$ is trivial. We then prove by induction and case analysis on i.

– $i \notin \{j, j+1\}$. From the premise we know that $\varLambda\langle i\rangle = \varLambda'\langle i\rangle$. We prove

$$\mathsf{GPath}'(\varLambda, i, l) \implies \mathsf{GPath}'(\varLambda', i, l),$$

and the other direction is similar. Let $\mathsf{GPath}'(\varLambda, i, l)$, then $l \geq 1$, and there exists $k \in (i, |\varLambda|]$ such that $\varLambda\langle k\rangle \in \varGamma'^{+}(\varLambda\langle i\rangle)$ and $\mathsf{GPath}'(\varLambda, k, l-1)$.
  - If $k \notin \{j, j+1\}$, then $\varLambda\langle k\rangle = \varLambda'\langle k\rangle$, and from the induction hypothesis we have $\mathsf{GPath}'(\varLambda', k, l-1)$. Then $\varLambda'\langle k\rangle \in \varGamma'^{+}(\varLambda'\langle i\rangle)$, and thus $\mathsf{GPath}'(\varLambda', i, l)$.
  - If $k = j$, then $\varLambda\langle k\rangle = \varLambda'\langle k+1\rangle$, and from the induction hypothesis we have $\mathsf{GPath}'(\varLambda', k+1, l-1)$. Then $\varLambda'\langle k+1\rangle \in \varGamma'^{+}(\varLambda\langle i\rangle)$, and thus $\mathsf{GPath}'(\varLambda', i, l)$.
  - If $k = j+1$, then $\varLambda\langle k\rangle = \varLambda'\langle k-1\rangle$, and from the induction hypothesis we have $\mathsf{GPath}'(\varLambda', k-1, l-1)$. Then $\varLambda'\langle k-1\rangle \in \varGamma'^{+}(\varLambda\langle i\rangle)$. Note that $k - 1 > i$ since $i \neq j$, and thus $\mathsf{GPath}'(\varLambda', i, l)$.

– We first prove

$$\mathsf{GPath}'(\varLambda, j, l) \implies \mathsf{GPath}'(\varLambda', j+1, l).$$

From the premise we have $\varLambda\langle j\rangle = \varLambda'\langle j+1\rangle$. Assume that $\mathsf{GPath}'(\varLambda, j, l)$, then $l \geq 1$, and there exists $k \in (j, |\varLambda|]$ such that $\varLambda\langle k\rangle \in \varGamma'^{+}(\varLambda\langle j\rangle)$ and $\mathsf{GPath}'(\varLambda, k, l-1)$. From $\varLambda\langle j\rangle \notin \varGamma'^{+}(\varLambda\langle j+1\rangle)$, this implies $k > j+1$. Thus $\varLambda\langle k\rangle = \varLambda'\langle k\rangle$, and from the induction hypothesis we have $\mathsf{GPath}'(\varLambda', k, l-1)$. Then $\varLambda'\langle k\rangle \in \varGamma'^{+}(\varLambda'\langle j+1\rangle)$, and thus $\mathsf{GPath}'(\varLambda', j+1, l)$. We then prove

$$\mathsf{GPath}'(\varLambda', j+1, l) \implies \mathsf{GPath}'(\varLambda, j, l).$$

Assume that $\mathsf{GPath}'(\varLambda', j+1, l)$, then $l \geq 1$, and there exists $k \in (j+1, |\varLambda|]$ such that $\varLambda'\langle k\rangle \in \varGamma'^{+}(\varLambda'\langle j+1\rangle)$ and $\mathsf{GPath}'(\varLambda', k, l-1)$. Thus $\varLambda\langle k\rangle = \varLambda'\langle k\rangle$, and from the induction hypothesis we have $\mathsf{GPath}'(\varLambda, k, l-1)$. Then $\varLambda\langle k\rangle \in \varGamma'^{+}(\varLambda\langle j\rangle)$, and thus $\mathsf{GPath}'(\varLambda, j, l)$.

– $\mathsf{GPath}'(\varLambda, j+1, l) \iff \mathsf{GPath}'(\varLambda', j, l)$. The proof is similar to the previous case.

□

Lemma 119. *For all* $\varLambda, \varLambda', j, i$ *and* l *such that* $j \in [1, |\varLambda| - 1)$*,* $i \in [1, |\varLambda|]$ *and* $|\varLambda| = |\varLambda'|$*, if*

– *For all* $k \in [1, |\varLambda|] \setminus \{j, j+1\}$*,* $\varLambda\langle k\rangle = \varLambda'\langle k\rangle$*;*
– $\varLambda\langle j\rangle = \varLambda'\langle j+1\rangle$*,* $\varLambda\langle j+1\rangle = \varLambda'\langle j\rangle$*;*

– $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$;

then

– *If* $i \notin \{j, j+1\}$, *then* $\mathsf{GDep}'(\Lambda, i, l) \iff \mathsf{GPath}'(\Lambda', i, l)$;
– $\mathsf{GDep}'(\Lambda, j, l) \iff \mathsf{GDep}'(\Lambda', j+1, l)$;
– $\mathsf{GDep}'(\Lambda, j+1, l) \iff \mathsf{GDep}'(\Lambda', j, l)$.

Proof. Directly from Lem. 118. □

Lemma 120. *For all* Λ, Λ', j, i *and* i' *such that* $j \in [1, |\Lambda| - 1)$, $i \in [1, |\Lambda|]$ *and* $|\Lambda| = |\Lambda'|$, *if*

– *For all* $k \in [1, |\Lambda|] \setminus \{j, j+1\}$, $\Lambda\langle k\rangle = \Lambda'\langle k\rangle$;
– $\Lambda\langle j\rangle = \Lambda'\langle j+1\rangle$, $\Lambda\langle j+1\rangle = \Lambda'\langle j\rangle$;
– $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$;

then

– *If* $i, i' \notin \{j, j+1\}$, *then* $\mathsf{GPar}'(\Lambda, i, i') \iff \mathsf{GPar}'(\Lambda', i, i')$;
– $\mathsf{GPar}'(\Lambda, j, i') \iff \mathsf{GPar}'(\Lambda', j+1, i')$;
– $\mathsf{GPar}'(\Lambda, j+1, i') \iff \mathsf{GPar}'(\Lambda', j, i')$;
– $\mathsf{GPar}'(\Lambda, i, j) \iff \mathsf{GPar}'(\Lambda', i, j+1)$;
– $\mathsf{GPar}'(\Lambda, i, j+1) \iff \mathsf{GPar}'(\Lambda', i, j)$.

Proof. Assume that $\mathsf{GPar}'(\Lambda, i, i')$. From $\mathsf{GPar}'(\Lambda, i, i')$ we know that $i < i'$, and there exists l such that $\mathsf{GDep}'(\Lambda, i, l+1)$, $\Lambda\langle i'\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle)$, $\mathsf{GPath}'(\Lambda, i', l)$, and for all $k > i$ such that $\Lambda\langle k\rangle < \Lambda\langle i'\rangle$ and $\Lambda\langle k\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle)$ we have $\neg\mathsf{GPath}'(\Lambda, k, l)$. We then prove the above five cases by instantiating i and i'.

The case of $i, i' \notin \{j, j+1\}$ is trivial.

Let $i = j$. Since $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$, we have $i' > j+1$. We prove that $\mathsf{GPar}'(\Lambda', j+1, i')$. From Lem. 118, Lem. 119 and premises, we have $\mathsf{GDep}'(\Lambda', j+1, l+1)$, $\Lambda'\langle i'\rangle \in \Gamma'^{+}(\Lambda'\langle j+1\rangle)$ and $\mathsf{GPath}'(\Lambda', i', l)$. For all $k > j+1$ such that $\Lambda'\langle k\rangle < \Lambda'\langle i'\rangle$ and $\Lambda'\langle k\rangle \in \Gamma'^{+}(\Lambda'\langle j+1\rangle)$, we have $\Lambda\langle k\rangle < \Lambda\langle i'\rangle$ and $\Lambda\langle k\rangle \in \Gamma'^{+}(\Lambda\langle j\rangle)$, and thus $\neg\mathsf{GPath}'(\Lambda, k, l)$. From Lem. 118, this implies $\neg\mathsf{GPath}'(\Lambda', k, l)$. Thus $\mathsf{GPar}'(\Lambda', j+1, i')$. The other direction is similar to the third case below.

Let $i = j+1$. We prove that $\mathsf{GPar}'(\Lambda', j, i')$. From Lem. 118, Lem. 119 and premises, $\mathsf{GDep}'(\Lambda', j, l+1)$, $\Lambda'\langle i'\rangle \in \Gamma'^{+}(\Lambda'\langle j\rangle)$ and $\mathsf{GPath}'(\Lambda', i', l)$. For all $k > j$ such that $\Lambda'\langle k\rangle < \Lambda'\langle i'\rangle$ and $\Lambda'\langle k\rangle \in \Gamma'^{+}(\Lambda'\langle j\rangle)$, we have $k > j+1$ from $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$, then $\Lambda\langle k\rangle < \Lambda\langle i'\rangle$ and $\Lambda\langle k\rangle \in \Gamma'^{+}(\Lambda\langle j+1\rangle)$. Thus $\neg\mathsf{GPath}'(\Lambda, k, l)$, which implies $\neg\mathsf{GPath}'(\Lambda', k, l)$ by Lem. 118. Thus $\mathsf{GPar}'(\Lambda', j, i')$. The other direction is similar to the second case above.

Let $i' = j$. We prove that $\mathsf{GPar}'(\Lambda', i, j+1)$. From Lem. 118, Lem. 119 and premises, we have $\mathsf{GDep}'(\Lambda', i, l+1)$, $\Lambda'\langle j+1\rangle \in \Gamma'^{+}(\Lambda'\langle i\rangle)$ and $\mathsf{GPath}'(\Lambda', j+1, l)$. For all $k > i$ such that $\Lambda'\langle k\rangle < \Lambda'\langle j+1\rangle$ and $\Lambda'\langle k\rangle \in \Gamma'^{+}(\Lambda'\langle i\rangle)$, we have $k \neq j+1$.

– If $k = j$, then $\Lambda\langle j+1\rangle < \Lambda\langle j\rangle$ and $\Lambda\langle j+1\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle)$, and thus $\neg\mathsf{GPath}'(\Lambda, j+1, l)$. From Lem. 118, this implies $\neg\mathsf{GPath}'(\Lambda', j, l)$.

- If $k \neq j$, then $\Lambda\langle k\rangle < \Lambda\langle j\rangle$ and $\Lambda\langle k\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle)$, and thus $\neg\mathsf{GPath}'(\Lambda, k, l)$. Then $\neg\mathsf{GPath}'(\Lambda', k, l)$ from Lem. 118.

Thus $\neg\mathsf{GPath}'(\Lambda', k, l)$ and then $\mathsf{GPar}'(\Lambda', i, j+1)$. The other direction is similar to the fifth case below.

Let $i' = j+1$. Since $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$, we have $i < j$. We prove that $\mathsf{GPar}'(\Lambda', i, j)$. From Lem. 118, Lem. 119 and premises, we have $\mathsf{GDep}'(\Lambda', i, l{+}1)$, $\Lambda'\langle j\rangle \in \Gamma'^{+}(\Lambda'\langle i\rangle)$ and $\mathsf{GPath}'(\Lambda', j, l)$. For all $k > i$ such that $\Lambda'\langle k\rangle < \Lambda'\langle j\rangle$ and $\Lambda'\langle k\rangle \in \Gamma'^{+}(\Lambda'\langle i\rangle)$, we have $k \neq j$.

- If $k = j+1$, then $\Lambda\langle j\rangle < \Lambda\langle j+1\rangle$ and $\Lambda\langle j\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle)$, and thus $\neg\mathsf{GPath}'(\Lambda, j, l)$. Then $\neg\mathsf{GPath}'(\Lambda', j+1, l)$ from Lem. 118.
- If $k \neq j+1$, then $\Lambda\langle k\rangle < \Lambda\langle j+1\rangle$ and $\Lambda\langle k\rangle \in \Gamma'^{+}(\Lambda\langle i\rangle)$, and thus $\neg\mathsf{GPath}'(\Lambda, k, l)$. Then $\neg\mathsf{GPath}'(\Lambda', k, l)$ from Lem. 118.

Thus $\neg\mathsf{GPath}'(\Lambda', k, l)$ and then $\mathsf{GPar}'(\Lambda', i, j)$. The other direction is similar to the forth case above.

□

Lemma 121. *For all Λ, Λ', j and i such that $j \in [1, |\Lambda| - 1)$, $i \in [1, |\Lambda|]$ and $|\Lambda| = |\Lambda'|$, if*

- *For all $k \in [1, |\Lambda|] \setminus \{j, j+1\}$, $\Lambda\langle k\rangle = \Lambda'\langle k\rangle$;*
- *$\Lambda\langle j\rangle = \Lambda'\langle j+1\rangle$, $\Lambda\langle j+1\rangle = \Lambda'\langle j\rangle$;*
- *$\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$;*

then

- *If $i \notin \{j, j+1\}$, then $GWT'(\Lambda, i) = GWT'(\Lambda', i)$;*
- *$GWT'(\Lambda, j) = GWT'(\Lambda', j+1)$;*
- *$GWT'(\Lambda, j+1) = GWT'(\Lambda', j)$.*

Proof. We prove by induction and case analysis on i.

- If $i \notin \{j, j+1\}$, then $\Lambda\langle i\rangle = \Lambda'\langle i\rangle$. Assuming that $S = \{k \mid \mathsf{GPar}'(\Lambda, k, i)\}$ and $S' = \{k \mid \mathsf{GPar}'(\Lambda', k, i)\}$, from Lem. 120 we know that
  - For all $k \notin \{j, j+1\}$, $k \in S \iff k \in S'$;
  - $j \in S \iff j+1 \in S'$;
  - $j+1 \in S \iff j \in S'$.

  Then, since $\Lambda\langle i\rangle = \Lambda'\langle i\rangle$, from Lem. 111 and the induction hypothesis we have $GWT'(\Lambda, i) = GWT'(\Lambda', i)$ by definition.
- We prove $GWT'(\Lambda, j) = GWT'(\Lambda', j{+}1)$. Assuming $S = \{k \mid \mathsf{GPar}'(\Lambda, k, j)\}$ and $S' = \{k \mid \mathsf{GPar}'(\Lambda', k, j+1)\}$, from Lem. 120 and $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$ we know that
  - For all $k \notin \{j, j+1\}$, $k \in S \iff k \in S'$;
  - $j, j+1 \notin S \cup S'$;

  Then, since $\Lambda\langle j\rangle = \Lambda'\langle j+1\rangle$, from Lem. 111 and the induction hypothesis we have $GWT'(\Lambda, i) = GWT'(\Lambda', i)$ by definition.
- $GWT'(\Lambda, j+1) = GWT'(\Lambda', j)$. The proof is similar to the previous case.

□

Lemma 122. *For all Λ, Λ' and j such that $j \in [1, |\Lambda| - 1)$ and $|\Lambda| = |\Lambda'|$, if*

- *For all $k \in [1, |\Lambda|] \setminus \{j, j+1\}$, $\Lambda\langle k\rangle = \Lambda'\langle k\rangle$;*
- *$\Lambda\langle j\rangle = \Lambda'\langle j+1\rangle$, $\Lambda\langle j+1\rangle = \Lambda'\langle j\rangle$;*
- *$\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$;*

then $f_{\mathsf{LWT}}(\Lambda) = f_{\mathsf{LWT}}(\Lambda')$.

Proof. Directly from Lem. 121. □

Lemma 123. *For all $\Lambda_1, \Lambda_2, \Lambda'_1, \Lambda'_2$ and j such that $j \in [1, |\Lambda_1| - 1)$ and $|\Lambda_1| = |\Lambda_2|$, if*

- *For all $k \in [1, |\Lambda_1|] \setminus \{j, j+1\}$, $\Lambda_1\langle k\rangle = \Lambda_2\langle k\rangle$;*
- *$\Lambda_1\langle j\rangle = \Lambda_2\langle j+1\rangle$, $\Lambda_1\langle j+1\rangle = \Lambda_2\langle j\rangle$;*
- *$\Lambda_1\langle j\rangle \notin \Gamma'^{+}(\Lambda_1\langle j+1\rangle)$;*
- *$\Lambda'_1 = LYWT(\lambda i.\, \Lambda_1\langle i\rangle)(GWTI'(\Lambda_1, |\Lambda_1|))$,
  $\Lambda'_2 = LYWT(\lambda i.\, \Lambda_2\langle i\rangle)(GWTI'(\Lambda_2, |\Lambda_2|))$;*

then $|\Lambda'_1| = |\Lambda'_2|$, and for all $i \in [1, |\Lambda'_1|]$,

- *If $\Lambda'_1\langle i\rangle \notin \{j, j+1\}$, then $\Lambda'_1\langle i\rangle = \Lambda'_2\langle i\rangle$;*
- *If $\Lambda'_1\langle i\rangle = j$, then $\Lambda'_2\langle i\rangle = j+1$;*
- *If $\Lambda'_1\langle i\rangle = j+1$, then $\Lambda'_2\langle i\rangle = j$.*

Proof. Let $h_1 = \lambda i.\, \Lambda_1\langle i\rangle$ and $h_2 = \lambda i.\, \Lambda_2\langle i\rangle$. By Lem. 122, Lem. 97 and induction, we have

$$\begin{aligned}|\Lambda'_1| &= |GWTI'(\Lambda_1, |\Lambda_1|)| = |f_{\mathsf{LWT}}(\Lambda_1)| \\ &= |f_{\mathsf{LWT}}(\Lambda_2)| = |GWTI'(\Lambda_2, |\Lambda_2|)| = |\Lambda'_2|.\end{aligned}$$

Now we only need to prove that, for all l:

- If $\Lambda''_1\langle k\rangle \notin \{j, j+1\}$, then $\Lambda''_1\langle k\rangle = \Lambda''_2\langle i\rangle$;
- If $\Lambda''_1\langle k\rangle = j$, then $\Lambda''_2\langle k\rangle = j+1$;
- If $\Lambda''_1\langle k\rangle = j+1$, then $\Lambda''_2\langle k\rangle = j$;
- $|\Lambda''_1| = |\Lambda''_2|$

for $i \in [1, |\Lambda|] \setminus \{j, j+1\}$, $\Lambda''_1 = \mathit{Lay}(h_1)(GWTI'(\Lambda_1, i), l)$, $\Lambda''_2 = \mathit{Lay}(h_2)(GWTI'(\Lambda_2, i), l)$ and $k \in [1, |\Lambda''_1|]$;

$$\mathit{Lay}(h_1)(GWTI'(\Lambda_1, j), l) = \mathit{Lay}(h_2)(GWTI'(\Lambda_2, j+1), l);$$

and

$$\mathit{Lay}(h_1)(GWTI'(\Lambda_1, j+1), l) = \mathit{Lay}(h_2)(GWTI'(\Lambda_2, j), l).$$

We prove by induction on l. The case of $l = 0$ is trivial. Below we suppose $l > 0$.

If $i \notin \{j, j+1\}$, assuming that $S_1 = \{k \mid \mathsf{GPar}'(\Lambda_1, k, i)\}$ and $S_2 = \{k \mid \mathsf{GPar}'(\Lambda_2, k, i)\}$, from Lem. 120 we have

- For all $k \notin \{j, j+1\}$, $k \in S_1 \iff k \in S_2$;
- $j \in S_1 \iff j+1 \in S_2$;
- $j+1 \in S_1 \iff j \in S_2$.

Since $h_1(j) = h_2(j+1)$ and $h_1(j+1) = h_2(j)$, the proof then follows from the induction hypothesis and Lem. 111.

Then we prove

$$Lay(h_1)(GWTI'(\Lambda_1, j), l) = Lay(h_2)(GWTI'(\Lambda_2, j+1), l).$$

Let $S_1 = \{k \mid \mathsf{GPar}'(\Lambda_1, k, j)\}$ and $S_2 = \{k \mid \mathsf{GPar}'(\Lambda_2, k, j+1)\}$, then from Lem. 120 and $\Lambda_1\langle j\rangle \notin \Gamma'^{+}(\Lambda_1\langle j+1\rangle)$ we know that

- For all $k \notin \{j, j+1\}$, $k \in S_1 \iff k \in S_2$;
- $j, j+1 \notin S_1 \cup S_2$;

Then the proof follows from the induction hypothesis and Lem. 111.

The proof of

$$Lay(h_1)(GWTI'(\Lambda_1, j+1), l) = Lay(h_2)(GWTI'(\Lambda_2, j), l)$$

is similar to the previous case. □

Lemma 124. *For all $\Lambda \in ExLog$ and K such that $|\Lambda| \leq K$,*

$$f_{\mathsf{LWT}}(\Lambda) \in LWTMap(K).$$

Proof. Let $\Lambda \in ExLog$ and $|\Lambda| \leq K$. By Lem. 109, we know that $|f_{\mathsf{LWT}}(\Lambda)| \leq |\Lambda| \leq K$. From Lem. 107 and Lem. 111, by induction we have $\mathsf{Proper}'(f_{\mathsf{LWT}}(\Lambda))$. Thus by definition we obtain that $f_{\mathsf{LWT}}(\Lambda) \in LWTMap(K)$. □

Lemma 125. *For all $\Lambda \prec \Lambda' \in ExLog$,*

$$f_{\mathsf{LWT}}(\Lambda) \neq f_{\mathsf{LWT}}(\Lambda').$$

Proof. We show that $GWT'(\Lambda, |\Lambda|) \neq GWT'(\Lambda', |\Lambda'|)$. If $\Lambda\langle|\Lambda|\rangle \neq \Lambda'\langle|\Lambda'|\rangle$, then $GWT'(\Lambda, |\Lambda|) \neq GWT'(\Lambda', |\Lambda'|)$, otherwise it contradicts Lem. 107. If $\Lambda\langle|\Lambda|\rangle = \Lambda'\langle|\Lambda'|\rangle$, then by $\Lambda \prec \Lambda'$ we have $\#_{\Lambda\langle|\Lambda|\rangle}(\Lambda) = \sum_{i\in[1,|\Lambda|]}[\Lambda\langle i\rangle = \Lambda'\langle|\Lambda'|\rangle] < \#_{\Lambda'\langle|\Lambda'|\rangle}(\Lambda')$. Thus, from Lem. 114 we get $GWT'(\Lambda, |\Lambda|) \neq GWT'(\Lambda', |\Lambda'|)$. □

Lemma 126. *For all reals $\alpha_1, \ldots, \alpha_M \in (0,1)$, if*

$$\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \leq \alpha_i \prod_{j\in\Gamma'(i)} (1-\alpha_j)$$

holds, then for all K we have

$$\sum_{wt\in LWTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \leq r_{\mathsf{EL}},$$

where

$$r_{\mathsf{EL}} = \sum_{i\in[1,M]} \alpha_i(1-\alpha_i)^{-1}.$$

Proof. From the premise, we only need to prove that, for all K,

$$\sum_{wt \in LWTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j \in \Gamma'(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j) \le \sum_{k\in[1,M]} \alpha_k \cdot (1-\alpha_k)^{-1}.$$

Define $h_j(K) = \{wt \in LWTMap(K) \mid root(wt) = j\}$, then we only need to prove that, for all $k \in [1, M]$ and K,

$$\sum_{wt \in h_k(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j \in \Gamma'(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j) \le \alpha_k \cdot (1-\alpha_k)^{-1}.$$

We prove by induction on K. The case of $K = 0$ if trivial. For the induction step,

$$\begin{aligned}
&\sum_{wt \in h_k(K+1)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{j \in \Gamma'(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_j) \\
&= \sum_{(k,\{wt_1,\dots,wt_n\})\in h_k(K+1)} \Bigg(\Bigg(\alpha_k \prod_{j\in\Gamma'(k)} (1-\alpha_j)\Bigg) \\
&\quad \cdot \prod_{l\in[1,n]} \prod_{i=1}^{|g_{\mathsf{WT}}(wt_l)|} \alpha_{g_{\mathsf{WT}}(wt_l)[i]} \prod_{j\in\Gamma'(g_{\mathsf{WT}}(wt_l)[i])} (1-\alpha_j)\Bigg) \\
&\le \alpha_k \cdot (1-\alpha_k)^{-1} \prod_{j\in\Gamma'^{+}(k)} (1-\alpha_j) \\
&\quad \cdot \left(1 + \sum_{wt\in h_j(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \alpha_{g_{\mathsf{WT}}(wt)\langle i\rangle} \prod_{l\in\Gamma'(g_{\mathsf{WT}}(wt)\langle i\rangle)} (1-\alpha_l)\right) \\
&\le \alpha_k \cdot (1-\alpha_k)^{-1} \prod_{j\in\Gamma'^{+}(k)} (1-\alpha_j)\left(1+\alpha_j\cdot(1-\alpha_j)^{-1}\right) \\
&= \alpha_k \cdot (1-\alpha_k)^{-1}.
\end{aligned}$$

□

I.3 Strong Witness Trees

A witness tree wt is called a strong witness tree, if the following condition holds: for each node in wt, all of its child nodes form an independent set on the dependency graph. We define the map *SWTMap* in Fig. 32.

Lemma 127. *For all $\Lambda \in ExLog$ and K such that $|\Lambda| \le K$,*

$$f_{\mathsf{WT}}(\Lambda) \in SWTMap(K).$$

$$\begin{aligned}
&SWTMap(K) \triangleq \{wt \in WT \mid \mathsf{Proper}(wt) \wedge \mathsf{Strong}(wt) \wedge |wt| \le K\} \\
\mathsf{Strong}((m, \{wt_1, \ldots, wt_n\})) \text{ iff } &\left(\textstyle\bigwedge_{i\in[1,n]} . \ \mathsf{Strong}(wt_i)\right) \\
&\wedge \left(\forall i < j \in [1,n]. \ root(wt_i) \notin \mathit{\Gamma}^+(root(wt_j))\right)
\end{aligned}$$

Fig. 32. Definitions related to strong witness trees.

Proof. Let $\mathit{\Lambda} \in ExLog$ satisfy $|\mathit{\Lambda}| \le K$, then by Lem. 88 we know that $|f_{\mathsf{WT}}(\mathit{\Lambda})| \le |\mathit{\Lambda}| \le K$. Similar to Lem. 98, we have $\mathsf{Proper}(f_{\mathsf{WT}}(\mathit{\Lambda}))$ and $\mathsf{Strong}(f_{\mathsf{WT}}(\mathit{\Lambda}))$ by Lem. 86, Lem. 90 and induction. Thus by definition we obtain that $f_{\mathsf{WT}}(\mathit{\Lambda}) \in SWTMap(K)$. □

Lemma 128. *For all reals $\beta_1, \ldots, \beta_M \in (0, \infty)$, if the cluster expansion condition*

$$\forall i \in [1, M]. \ \mathrm{P}(\mathcal{E}[i]) \le \beta_i \left(\sum_{\substack{I \subseteq \mathit{\Gamma}^+(i) \\ \mathrm{Indep}(I)}} \prod_{j \in I} \beta_j \right)^{-1}$$

holds, then for all K we have

$$\sum_{wt \in SWTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \le \sum_{i \in [1,M]} \beta_i.$$

Proof. From the cluster expansion condition, we only need to prove that, for all K,

$$\sum_{wt \in SWTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \beta_{g_{\mathsf{WT}}(wt)\langle i \rangle} \left(\sum_{\substack{I \subseteq \mathit{\Gamma}^+(i) \\ \mathrm{Indep}(I)}} \prod_{j \in I} \beta_j \right)^{-1} \le \sum_{k \in [1,M]} \beta_k.$$

Define $h_j(K) = \{wt \in SWTMap(K) \mid root(wt) = j\}$, then we only need to prove that, for all $k \in [1, M]$ and K,

$$\sum_{wt \in h_k(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \beta_{g_{\mathsf{WT}}(wt)\langle i \rangle} \left(\sum_{\substack{I \subseteq \mathit{\Gamma}^+(i) \\ \mathrm{Indep}(I)}} \prod_{j \in I} \beta_j \right)^{-1} \le \beta_k.$$

We prove by induction on K. The case of $K = 0$ if trivial. For the induction step,

$$\sum_{wt \in h_k(K+1)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \beta_{g_{\mathsf{WT}}(wt)\langle i \rangle} \left(\sum_{\substack{I \subseteq \mathit{\Gamma}^+(g_{\mathsf{WT}}(wt)\langle i \rangle) \\ \mathrm{Indep}(I)}} \prod_{j \in I} \beta_j \right)^{-1}$$

$$
\begin{aligned}
&= \sum_{(k,\{wt_1,\ldots,wt_n\})\in h_k(K+1)} \beta_k \left(\sum_{\substack{I\subseteq\Gamma^+(k)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \beta_j \right)^{-1} \\
&\quad \cdot \prod_{l\in[1,n]} \prod_{i=1}^{|g_{\mathsf{WT}}(wt_l)|} \beta_{g_{\mathsf{WT}}(wt_l)\langle i\rangle} \left(\sum_{\substack{I\subseteq\Gamma^+(g_{\mathsf{WT}}(wt_l)\langle i\rangle)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \beta_j \right)^{-1} \\
&\le \beta_k \left(\sum_{\substack{I\subseteq\Gamma^+(k)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \beta_j \right)^{-1} \sum_{\substack{I\subseteq\Gamma^+(k)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \\
&\quad \sum_{wt\in h_j(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \beta_{g_{\mathsf{WT}}(wt)\langle i\rangle} \left(\sum_{\substack{I\subseteq\Gamma^+(g_{\mathsf{WT}}(wt)\langle i\rangle)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \beta_j \right)^{-1} \\
&\le \beta_k \left(\sum_{\substack{I\subseteq\Gamma^+(k)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \beta_j \right)^{-1} \sum_{\substack{I\subseteq\Gamma^+(k)\\ \mathrm{Indep}(I)}} \prod_{j\in I} \beta_j \\
&= \beta_k.
\end{aligned}
$$

□

I.4 Stable Set Sequences

Below we define the stable set sequences.

We define $\mathit{SSS}, \mathit{SSSMap}, f_{\mathsf{SSS}}, g_{\mathsf{SSS}}$ in Fig. 33. SSS represents the set of all stable set sequences. A stable set sequence $\mathcal{I} = (I_0, \ldots, I_n)$ is a sequence consisting of a series of independent sets on the dependency graph, where each set in $\mathcal{I}$ is "covered" by the preceding set, that is: if $m \in I_{i+1}$, then there exists some element in I_i from $\Gamma^+(m)$. We define SSSMap as the set of all stable set sequences with total size no more than K and the first set a singleton.

For execution log $\Lambda \in \mathit{ExLog}$, we define $f_{\mathsf{SSS}}(\Lambda)$ as the stable set sequence constructed from Λ, where the auxiliary definitions can be found in Fig. 30.

For a stable set sequence $\mathcal{I} \in \mathit{SSS}$, we define $g_{\mathsf{SSS}}(\mathcal{I})$ as the sequence obtaining by repeatedly concatenating the sets (each in the form of a sequence) in $\mathcal{I}$ in a reversed order.

Other auxiliary definitions related to stable set sequences are also given in Fig. 33.

The following lemmas capture the properties of stable set sequences.

Lemma 129. *For all $\Lambda \in \mathit{ExLog}, i, j$ and l, if $i \neq j, \mathsf{GDep}(\Lambda, i, l)$ and $\mathsf{GDep}(\Lambda, j, l)$, then $\Lambda\langle j\rangle \notin \Gamma^+(\Lambda\langle i\rangle)$.*

$$(SSS)\ \mathcal{I} ::= (I_0, \ldots, I_n) \qquad \text{where } (\forall i \in I_0.\ i \in [1, M]) \wedge (\forall j \in [0, n].\ \text{Indep}(I_j)) \wedge \big(\forall j \in [0, n).\ I_{j+1} \subseteq \Gamma^+(I_j)\big)$$

$$\begin{aligned}
\mathit{SSSMap}(K) &\triangleq \{(I_0, \ldots, I_n) \in \mathit{SSS} \mid |I_0| = 1 \wedge |(I_0, \ldots, I_n)| \leq K\} \\
|(I_0, \ldots, I_n)| &\triangleq |I_0| + \cdots + |I_n|
\end{aligned}$$

$$\begin{aligned}
f_{\mathsf{SSS}}(\varLambda) &\triangleq \mathit{GS}(\varLambda) \\
\mathit{GS}(\varLambda) &\triangleq (I_0, \ldots, I_n) \quad \text{where } \forall j \in [0, n].\ I_j = \{\varLambda\langle i\rangle \mid \mathsf{GDep}(\varLambda, i, j)\}, \\
&\qquad\qquad n = \max\{l \mid \mathsf{GDep}(\varLambda, \_, l)\}
\end{aligned}$$

$$\begin{aligned}
g_{\mathsf{SSS}}((I_0, \ldots, I_n)) &\triangleq \mathit{SS}(\mathsf{id})((I_0, \ldots, I_n)) \\
\mathit{SS}(h)((I_0, \ldots, I_n)) &\triangleq \mathsf{seq}(h)(I_n) \parallel \cdots \parallel \mathsf{seq}(h)(I_0)
\end{aligned}$$

$$\begin{aligned}
m \in (I_0, \ldots, I_n) &\text{ iff } \exists j \in [0, n].\ m \in I_j \\
\#_m((I_0, \ldots, I_n)) &\triangleq \textstyle\sum_{i \in [0,n]} [m \in I_i] \\
\#_{m,\varLambda}((I_0, \ldots, I_n)) &\triangleq \textstyle\sum_{i \in [0,n]} \sum_{m' \in I_i} [\varLambda\langle m'\rangle = m] \\
\mathit{GSI}(\varLambda) &\triangleq (I_0, \ldots, I_n) \quad \text{where } \forall j \in [0, n].\ I_j = \{i \mid \mathsf{GDep}(\varLambda, i, j)\}, \\
&\qquad\qquad n = \max\{l \mid \mathsf{GDep}(\varLambda, \_, l)\}
\end{aligned}$$

Fig. 33. Definitions related to stable set sequences.

Proof. We prove by contradiction. Let $\varLambda\langle j\rangle \in \Gamma^+(\varLambda\langle i\rangle)$. Without loss of generality, we assume that $i < j$. Let $\mathsf{GDep}(\varLambda, i, l)$ and $\mathsf{GDep}(\varLambda, j, l)$ hold, then $\neg\mathsf{GPath}(\varLambda, i, l')$ for all $l' > l$. However, by $\mathsf{GDep}(\varLambda, j, l)$ and $\varLambda\langle j\rangle \in \Gamma^+(\varLambda\langle i\rangle)$ we know that $\mathsf{GPath}(\varLambda, i, l+1)$, a contradiction. Thus $\varLambda\langle j\rangle \notin \Gamma^+(\varLambda\langle i\rangle)$. □

Lemma 130. *For all $\varLambda \in \mathit{ExLog}$, $\#_{\varLambda\langle|\varLambda|\rangle}(\mathit{GS}(\varLambda)) = \#_{\varLambda\langle|\varLambda|\rangle}(\varLambda)$.*

Proof. By Lem. 129, we know that

$$\#_m(\mathit{GS}(\varLambda)) = \#_{m,\varLambda}(\mathit{GSI}(\varLambda))$$

holds for all m. Let $\mathit{GSI}(\varLambda) = (I_0, \ldots, I_n)$. Below we only need to prove that $\#_{\varLambda\langle|\varLambda|\rangle,\varLambda}(\mathit{GSI}(\varLambda)) = \#_{\varLambda\langle|\varLambda|\rangle}(\varLambda)$, that is, we prove that for all i such that $\varLambda\langle i\rangle = \varLambda\langle|\varLambda|\rangle$ there exists j such that $i \in I_j$. Assume that $\{i \mid \varLambda\langle i\rangle = \varLambda\langle|\varLambda|\rangle\} = \{i_0, \ldots, i_l\}$, where $|\varLambda| = i_0 > \cdots > i_l$. Then, by induction, we know that $\mathsf{GPath}(\varLambda, i_{l'}, l')$ holds for all $l' \in [0, l]$, and thus for all i such that $\varLambda\langle i\rangle = \varLambda\langle|\varLambda|\rangle$ there exists l' such that $\mathsf{GPath}(\varLambda, i, l')$, and then From Lem. 91 there exists j such that $\mathsf{GDep}(\varLambda, i, j)$, which implies $i \in I_j$. □

Lemma 131. *For all $\varLambda \in \mathit{ExLog}$ and K such that $|\varLambda| \leq K$,*

$$f_{\mathsf{SSS}}(\varLambda) \in \mathit{SSSMap}(K).$$

Proof. Let $\varLambda \in \mathit{ExLog}$ satisfy $|\varLambda| \leq K$, then by Lem. 89 and Lem. 129 we know that $f_{\mathsf{SSS}}(\varLambda) \in \mathit{SSS}$, and by definition we have $|f_{\mathsf{SSS}}(\varLambda)| \leq |\varLambda| \leq K$. Thus $f_{\mathsf{SSS}}(\varLambda) \in \mathit{SSSMap}(K)$. □

Lemma 132. *For all $\Lambda \prec \Lambda' \in ExLog$,*

$$f_{\mathsf{SSS}}(\Lambda) \neq f_{\mathsf{SSS}}(\Lambda').$$

Proof. Let $\Lambda \prec \Lambda' \in ExLog$. Assuming that $f_{\mathsf{SSS}}(\Lambda) = (I_0, \ldots, I_n)$ and $f_{\mathsf{SSS}}(\Lambda') = (J_0, \ldots, J_m)$, below we prove that $(I_0, \ldots, I_n) \neq (J_0, \ldots, J_m)$. If $\Lambda\langle|\Lambda|\rangle \neq \Lambda'\langle|\Lambda'|\rangle$, then $I_0 = \{\Lambda\langle|\Lambda|\rangle\} \neq \{\Lambda'\langle|\Lambda'|\rangle\} = J_0$. If $\Lambda\langle|\Lambda|\rangle = \Lambda'\langle|\Lambda'|\rangle$, then from $\Lambda \prec \Lambda'$ we know that

$$\#_{\Lambda\langle|\Lambda|\rangle}(\Lambda) = \sum_{i \in [1,|\Lambda|]} [\Lambda\langle i\rangle = \Lambda'\langle|\Lambda'|\rangle] < \#_{\Lambda'\langle|\Lambda'|\rangle}(\Lambda'),$$

thus from Lem. 130 we have $GS(\Lambda) \neq GS(\Lambda')$. ◻

Lemma 133. *For all $\Lambda, \Lambda' \in ExLog$, if*

$$(g_{\mathsf{SSS}} \circ f_{\mathsf{SSS}})(\Lambda) = \Lambda',$$

then for each $l \in [1, |\Lambda'|]$ there exists k such that $\Lambda\langle k\rangle = \Lambda'\langle l\rangle$ and

$$\sum_{k'<k} [\mathrm{vbl}(\Lambda\langle k'\rangle, i)] = \sum_{l'<l} [\mathrm{vbl}(\Lambda'\langle l'\rangle, i)]$$

for all $i \in [1, N]$ such that $\mathrm{vbl}(\Lambda'\langle l\rangle, i)$.

Proof. Let $\Lambda, \mathcal{I} = (I_0, \ldots, I_n), \Lambda'$ and l satisfy $f_{\mathsf{SSS}}(\Lambda) = \mathcal{I}$, $g_{\mathsf{SSS}}(\mathcal{I}) = \Lambda'$, $l \in [1, |\Lambda'|]$. Let $\mathcal{J} = GSI(\Lambda) = (J_0, \ldots, J_{n'})$, $\Lambda'' = SS(\lambda i.\, \Lambda\langle i\rangle)(\mathcal{J})$, then by induction we have $|\Lambda'| = |\Lambda''|$, and for all $l' \in [1, |\Lambda'|]$ we have $\Lambda'\langle l'\rangle = \Lambda\langle \Lambda''\langle l'\rangle\rangle$. Take $j = \Lambda''\langle l\rangle$, then by definition we know that there exists l'' such that $\mathsf{GPath}(\Lambda, j, l'')$. For i such that $\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l\rangle], i)$, we only need to prove that

$$\sum_{j'<j} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle j'\rangle], i)] = \sum_{l'<l} [\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)].$$

By definition, all elements in Λ'' are distinct, and $j' \in \Lambda''$ for all $j' < j$ such that $\mathrm{vbl}(\mathcal{E}[\Lambda\langle j'\rangle], i)$ (this implies $\mathsf{GPath}(\Lambda, j', l'' + 1)$), and thus

$$\sum_{j'<j} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle j'\rangle], i)] = \sum_{l' : \Lambda''\langle l'\rangle < \Lambda''\langle l\rangle} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle \Lambda''\langle l'\rangle\rangle], i)].$$

Then it remains to prove that, for all l' such that $\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)$,

$$l' < l \iff \Lambda''\langle l'\rangle < \Lambda''\langle l\rangle.$$

Let $l' < l$ satisfy $\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)$, then there exists $d' \geq d$ such that $\Lambda''\langle l'\rangle \in J_{d'}$ and $\Lambda''\langle l\rangle \in J_d$, which implies that $\mathsf{GDep}(\Lambda, \Lambda''\langle l'\rangle, d')$ and $\mathsf{GDep}(\Lambda, \Lambda''\langle l\rangle, d)$. Suppose that $\Lambda''\langle l'\rangle > \Lambda''\langle l\rangle$, then from $\Lambda\langle \Lambda''\langle l'\rangle\rangle \in \Gamma^+(\Lambda\langle \Lambda''\langle l\rangle\rangle)$ we have $\mathsf{GPath}(\Lambda, \Lambda''\langle l\rangle, d' + 1)$, a contradiction. Thus $l' < l \implies \Lambda''\langle l'\rangle < \Lambda''\langle l\rangle$. Similarly, if $\Lambda''\langle l'\rangle < \Lambda''\langle l\rangle$, one can show that $l' < l$. ◻

Lemma 134. *If the Shearer's condition*

$$\forall I \subseteq [1,M].\ \mathrm{Indep}(I) \implies q_I > 0$$

holds, where

$$q_I = \sum_{\substack{I \subseteq J \subseteq [1,M] \\ \mathrm{Indep}(J)}} (-1)^{|J|-|I|} \prod_{j \in J} \mathrm{P}(\mathcal{E}[j]),$$

then for all K we have

$$\sum_{\mathcal{I} \in SSSMap(K)} \prod_{i=1}^{|g_{\mathsf{SSS}}(\mathcal{I})|} \mathrm{P}(\mathcal{E}[g_{\mathsf{SSS}}(\mathcal{I})\langle i \rangle]) \leq \sum_{i \in [1,M]} \frac{q_{\{i\}}}{q_\varnothing}.$$

Proof. Define

$$h_I(K) = \{(I_0, \ldots, I_n) \mid I_0 = I \wedge |(I_0, \ldots, I_n)| \leq K\} \cup (I = \varnothing\ ?\ \{()\} : \varnothing),$$

then we only need to prove that, for all K and I such that $\mathrm{Indep}(I)$,

$$\sum_{\mathcal{I} \in h_I(K)} \prod_{i=1}^{|g_{\mathsf{SSS}}(\mathcal{I})|} \mathrm{P}(\mathcal{E}[g_{\mathsf{SSS}}(\mathcal{I})\langle i \rangle]) \leq \frac{q_I}{q_\varnothing}.$$

We prove by induction on K. The case of $K = 0$ is trivial. For the induction step,

$$\begin{aligned}
&\sum_{\mathcal{I} \in h_I(K+1)} \prod_{i=1}^{|g_{\mathsf{SSS}}(\mathcal{I})|} \mathrm{P}(\mathcal{E}[g_{\mathsf{SSS}}(\mathcal{I})\langle i \rangle]) \\
&\leq \left(\prod_{i \in I} \mathrm{P}(\mathcal{E}[i])\right) \sum_{\substack{J \subseteq \Gamma^+(I) \\ \mathrm{Indep}(J)}} \sum_{\mathcal{J} \in h_J(K)} \prod_{j=1}^{|g_{\mathsf{SSS}}(\mathcal{J})|} \mathrm{P}(\mathcal{E}[g_{\mathsf{SSS}}(\mathcal{J})[j]]) \\
&\leq \left(\prod_{i \in I} \mathrm{P}(\mathcal{E}[i])\right) \sum_{\substack{J \subseteq \Gamma^+(I) \\ \mathrm{Indep}(J)}} \frac{q_J}{q_\varnothing} \\
&= \left(\prod_{i \in I} \mathrm{P}(\mathcal{E}[i])\right) \cdot \frac{1}{q_\varnothing} \cdot \sum_{\substack{J \subseteq [1,M] \\ J \cap ([1,M] \setminus \Gamma^+(I)) = \varnothing \\ \mathrm{Indep}(J)}} q_J \\
&= \left(\prod_{i \in I} \mathrm{P}(\mathcal{E}[i])\right) \cdot \frac{1}{q_\varnothing} \cdot \sum_{\substack{J \subseteq [1,M] \setminus \Gamma^+(I) \\ \mathrm{Indep}(J)}} (-1)^{|J|} \prod_{j \in J} \mathrm{P}(\mathcal{E}[j]) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ([60], \text{Eq. } 5.5) \\
&= \frac{q_I}{q_\varnothing}.
\end{aligned}$$

□

```
$C_{\mathsf{HSS}} \triangleq$

$d := 1$;
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d)$;
    $x[d] := a$;
    $d := d + 1$;
$\mathit{flag} := 0$;
$\mathit{cnt} := 0$;
$\mathit{lst} := []$;
while $(\mathit{flag} = 0)$ do
    $z := 0$;
    $h := 1$;
    while $(h < M)$ do
        if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then
            $z := h$;
        $h := h + 1$;
    if $(z = 0)$ then $\mathit{flag} := 1$;
    else
        $\mathit{cnt} := \mathit{cnt} + 1$;
        $\mathit{lst} := \mathsf{app}(\mathit{lst}, z)$;
        $d := 1$;
        while $(d \leq N)$ do
            if $(\mathsf{vbl}(z, d))$ then
                $a := \mathsf{Sample}(d)$;
                $x[d] := a$;
            $d := d + 1$;
```

```
$C_{\mathsf{MTpar}} \triangleq$

$d := 1$;
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d)$;
    $x[d] := a$;
    $d := d + 1$;
$\mathit{flag} := 0$;
$\mathit{cnt} := 0$;
$\mathit{lst} := []$;
while $(\mathit{flag} = 0)$ do
    $mis := \mathsf{MIS}(x[1], \ldots, x[N])$;
    if $(mis = [])$ then $\mathit{flag} := 1$;
    else
        $\mathit{cnt} := \mathit{cnt} + 1$;
        $\mathit{lst} := \mathsf{concat}(\mathit{lst}, mis)$;
        $L[\mathit{cnt}] := mis$;
        $C_{\mathsf{par}}(1)$;
        $\cdots$ ;
        $C_{\mathsf{par}}(M)$;

$C_{\mathsf{par}}(i) \triangleq$
if $(\mathsf{len}(mis) \geq i)$ then
    $d := 1$;
    while $(d \leq N)$ do
        if $(\mathsf{vbl}(mis[i], d))$ then
            $a := \mathsf{Sample}(d)$;
            $x[d] := a$;
        $d := d + 1$;
```

Fig. 34. Codes of variants of the MT algorithm.

J Verified ALLL-related results

This section gives the statements and formal proofs of all ALLL-results we verify.

In our proofs, we use witness-tree-like structures. Definitions and lemmas related to these structures are presented in App. I.

We give the following notation, which will be used through this section. The set of possible execution logs (and their prefixes), denoted as *ExLog*, is defined as follows:

$$\mathit{ExLog} \triangleq \{\Lambda \in \mathit{Seq} \mid \Lambda \neq [] \wedge (\forall i \in [1, |\Lambda|].\ \Lambda\langle i\rangle \in [1, M])\}.$$

Informally, an execution log is a non-empty list Λ where all elements in Λ belong to $[1, M]$.

J.1 Theorem 1.2 of [51] (Thm. 4)

Proof of Thm. 4. Let the Erdős-Lovász condition hold. By applying Lem. 77, Thm. 2 and Lem. 81, with the auxiliary code $C'_{\mathsf{MT}}(\mathit{cnt}, K)$ defined in Fig. 35, we

```
$C'_{\mathsf{MT}}(cnt, K) \triangleq$

$d := 1$;
while $(d \le N)$ do
    $a := \mathsf{Sample}(d)$;
    $x[d] := a$;
    $d := d + 1$;
$flag := 0$;
$cnt := 0$;
$lst := []$;
while $(flag = 0 \wedge cnt < K)$ do
    $z := 0$;
    $h := 1$;
    while $(h \le M)$ do
        if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then
            $z := h$;
        $h := h + 1$;
    if $(z = 0)$ then $flag := 1$;
    else
        $cnt := cnt + 1$;
        $lst := \mathsf{app}(lst, z)$;
        $d := 1$;
        while $(d \le N)$ do
            if $(\mathsf{vbl}(z, d))$ then
                $a := \mathsf{Sample}(d)$;
                $x[d] := a$;
            $d := d + 1$;
```

```
$C_{\mathsf{check}}(\Lambda) \triangleq$

$succ := 1$;
$h := 1$;
while $(h \le |\Lambda|)$ do
    $z := \Lambda\langle h \rangle$;
    $d := 1$;
    while $(d \le N)$ do
        if $(\mathsf{vbl}(z, d))$ then
            $a := \mathsf{Sample}(d)$;
            $x[d] := a$;
        $d := d + 1$;
    if $(\neg\mathsf{hold}(z, x[1], \ldots, x[N]))$ then
        $succ := 0$;
    $h := h + 1$;
```

Fig. 35. Auxiliary codes for Thm. 4.

only need to prove that, for all K,

$$\models [\textbf{true}]\, C'_{\mathsf{MT}}(cnt, K)\, [\mathbb{E}[cnt] \le r_{\mathsf{EL}} \wedge \lceil cnt \ge 0 \rceil]\,. \tag{43}$$

Since the Erdős-Lovász condition holds, by Lem. 101 we know that

$$\models \sum_{wt \in WTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \le r_{\mathsf{EL}}. \tag{44}$$

The above inequation corresponds to (4), the inequation in the third stage of the proof sketch in Sec. 2.1. Informally, $WTMap(K)$ is the set of all witness trees with size no more than K. Note that $WTMap(K)$ is a finite set, thus (44) only contains a finite series. $g_{\mathsf{WT}}(wt)$ represents a reversed BFS ordering of wt, and thus the product in (44) enumerates all events in wt by traversing wt with respect to its reversed BFS ordering $g_{\mathsf{WT}}(wt)$. We define $WTMap$ in App. I.1. With (44), to prove (43), by Lem. 77, we only need to prove that

$$\models [\textbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \Bigg[\lceil cnt \ge 0 \rceil \wedge$$

$$\mathbb{E}[cnt] \leq \sum_{wt \in WTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle])\Bigg]. \tag{45}$$

Then we define $\mathsf{FWT}(e, wt, i)$ as follows:

$$\mathsf{FWT}(e, wt, i) \triangleq \bigvee_{\Lambda \in f_{\mathsf{WT}}^{-1}(wt) \,:\, |\Lambda| = i} .\ e = \Lambda.$$

For $\Lambda \in \mathit{ExLog}$, $f_{\mathsf{WT}}(\Lambda)$ is the witness tree constructed from Λ, as defined in App. I.1. Informally, $\mathsf{FWT}(e, wt, i)$ holds iff the witness tree wt can be constructed from the list e, where the length of e is i (this makes the disjunction in $\mathsf{FWT}(e, wt, i)$ finite). Thus, $\mathsf{FWT}(\mathsf{pf}(lst, i), wt, i)$ holds iff the witness tree wt can be constructed from the execution log's prefix with length i. Now, to prove (45), by repeatedly applying Lem. 77 and Lem. 78, we only need to prove the following two subgoals:

$$\models [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \Bigg[\lceil cnt \geq 0 \rceil \wedge \mathbb{E}[cnt] = \sum_{wt \in WTMap(K)} \Pr\Bigg[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i) \Bigg] \Bigg], \tag{46}$$

and for all $wt \in WTMap(K)$

$$\models \{\mathbf{true}\}\, C'_{\mathsf{MT}}(cnt, K) \Bigg\{ \Pr\Bigg[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i) \Bigg] \leq \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \Bigg\}. \tag{47}$$

(46) and (47) correspond to (2) and (3), which are the goals in the first and the second stages of the proof sketch in Sec. 2.1, respectively.

For (46), from Lem. 77 and the linearity of expectation, we only need to prove that

$$\models [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \Bigg[\Bigg\lceil cnt = \sum_{wt \in WTMap(K)} \Bigg[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i) \Bigg] \Bigg\rceil \Bigg]; \tag{48}$$

then by Lem. 80, to prove (48), we only need to prove the following:

$$\models_{\mathrm{RT}} [\mathrm{true} \wedge \mathsf{hdinit}]\, C'_{\mathsf{MT}}(cnt, K) \Bigg[cnt = \sum_{wt \in WTMap(K)}$$

$$\left[\bigvee_{i\in[1,K]} i \le cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst,i),wt,i)\right]\Bigg]. \tag{49}$$

For (47), from Lem. 79 and Lem. 77, with the auxiliary code $C_{\mathsf{check}}(\varLambda)$ (the code of check(wt) in Sec. 2.1, where $\varLambda = g_{\mathsf{WT}}(wt)$) defined in Fig. 35, following the informal proof in the second stage in Sec. 2.1, we only need to prove the following two subgoals that respectively correspond to (b) and (a):

$$\begin{aligned}&\vDash \{\mathbf{true}\}\, C'_{\mathsf{MT}}(cnt,K), C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\\ &\quad\left\{\left(\bigvee_{i\in[1,K]} i \le cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst,i),wt,i)\right), succ = 1\right\},\end{aligned} \tag{50}$$

and

$$\begin{aligned}&\vDash [\mathbf{true}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\\ &\quad\left[\Pr[succ = 1] = \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\right].\end{aligned} \tag{51}$$

For (50), by RT-based coupling (Thm. 3), we only need to prove

$$\begin{aligned}&\vDash_{\mathrm{RT}} \{\mathrm{true} \wedge \mathsf{hdinit}\}\, C'_{\mathsf{MT}}(cnt,K)\\ &\quad\left\{\left(\bigvee_{i\in[1,K]} i \le cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst,i),wt,i)\right) \Rightarrow \mathbf{R}\right\}\end{aligned} \tag{52}$$

and

$$\vDash_{\mathrm{RT}} [\mathrm{true} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\, [succ = 1], \tag{53}$$

where $\mathbf{R}$ is defined below.

$$\begin{aligned}\mathbf{R} \triangleq\ & \textstyle\bigwedge_{l\in[1,|g_{\mathsf{WT}}(wt)|]} .\ \forall V_1,\ldots V_N.\\ &\quad (\textstyle\bigwedge_{i\in[1,N]} .\ \mathsf{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)\\ &\qquad\quad \Rightarrow V_i = \mathsf{RT}[i][\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)])\\ &\qquad \Rightarrow \mathsf{hold}(g_{\mathsf{WT}}(wt)\langle l\rangle, V_1,\ldots,V_N)\\ \mathsf{ve}(i,\varLambda,l) \triangleq\ & \textstyle\sum_{l'\in[1,l]}[\mathsf{vbl}(\varLambda\langle l'\rangle, i)]\end{aligned}$$

Informally, $\mathbf{R}$ says that, for all events in wt, at the time the event is chosen at the beginning of the outer loop in $C'_{\mathsf{MT}}(K)$, it must hold under the current assignment formed by some of the resampling table's entries. The exact column numbers of these entries are computed purely based on wt, with the help of Lem. 100. Moreover, for all resampling tables RT, if RT satisfies $\mathbf{R}$, then we have the following: for witness tree wt, when we test all the events in wt according to the reversed BFS ordering of wt ($g_{\mathsf{WT}}(wt)$) with respect to the resampling table RT, all tests pass. $\mathsf{ve}(i,\varLambda,l)$ represents the position of the i-th head after events

$\Lambda\langle 1\rangle, \ldots, \Lambda\langle l\rangle$ all being sampled, and in $\mathbf{R}$ we take $\Lambda = g_{\mathsf{WT}}(wt)$ as the reversed BFS ordering of wt.

Given that $\mathbf{R}$ is defined, (52) and (53) says that, for all resampling tables RT:

– If wt can be constructed from some prefix of the execution log generated by the MT algorithm ($C'_{\mathsf{MT}}(cnt, K)$) using RT, then $\mathbf{R}$ holds on RT.
– If $\mathbf{R}$ holds on RT, then all tests in the check(wt) program ($C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))$) pass when the program is executed using RT.

Now it remains to prove (49), (52), (53) and (51). For (49), (52) and (53), we apply Thm. 7, and use inference rules of the resampling-table-based program logic (listed in Fig. 25 and Fig. 26) to complete the proof. For (51), we apply Thm. 6, and use inference rules listed in Fig. 24 to complete the proof. Proofs of these four judgments are presented in Fig. 37, Fig. 38, Fig. 39, Fig. 40, Fig. 41 and Fig. 42, while the auxiliary assertions used by these proofs are presented in Fig. 36. In Fig. 40, Fig. 41 and Fig. 42, we write Λ for $g_{\mathsf{WT}}(wt)$.

In Fig. 37, Fig. 38, Fig. 39, Fig. 40, Fig. 41 and Fig. 42, we omit the proofs of side conditions when applying (CSQ-T), (RT-CSQ) and (RT-CSQ-T). Below we show the proofs of three non-trivial side conditions.

1. The side condition in the last line of Fig. 37:

$$\models_{\mathrm{RT}} \mathsf{CL}(K+1) \Rightarrow cnt = \sum_{wt \in WTMap(K)} \left[\bigvee_{i \in [1,K]} i \le cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i) \right].$$

   In the above side condition, the assertion $\mathsf{CL}(K+1)$ only says that lst is indeed an execution log with length no more than K.
   Proof: Let $\Sigma \models \mathsf{CL}(K+1)$, then there exists $m \in [0, K]$ such that:
   – $[\![cnt]\!]_{\Sigma} = |[\![lst]\!]_{\Sigma}| = m$;
   – For all $k \in [1, m]$, $[\![\mathsf{pf}(lst, k)]\!]_{\Sigma} \in ExLog$;
   – For all $k \in [1, m]$, $|[\![\mathsf{pf}(lst, k)]\!]_{\Sigma}| = k \le K$.
   Thus, by Lem. 98, we have $f_{\mathsf{WT}}([\![\mathsf{pf}(lst, k)]\!]_{\Sigma}) \in WTMap(K)$ for all $k \in [1, m]$. Now, let

$$wt_k = f_{\mathsf{WT}}([\![\mathsf{pf}(lst, k)]\!]_{\Sigma}),$$

   then we know that $wt_1 \ne \cdots \ne wt_m$ from Lem. 99, and for all $k \in [1, m]$ we have the following:
   – $\Sigma \models \mathsf{FWT}(\mathsf{pf}(lst, k), wt_k, k)$;
   – For all $wt \ne wt_k$, $\Sigma \models \neg\mathsf{FWT}(\mathsf{pf}(lst, k), wt, k)$.
   Thus, one can verify that

$$\Sigma \models cnt = \sum_{wt \in WTMap(K)} \left[\bigvee_{i \in [1,K]} i \le cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i) \right].$$

2. The side condition in the last line of Fig. 38:

$$\vDash_{\mathrm{RT}} \left(\bigvee_{k\in[0,K]} cnt = k \wedge \mathsf{len}(lst) = k \wedge \mathsf{L}(lst, k) \right) \Rightarrow$$
$$\left(\left(\bigvee_{i\in[1,K]} i \leq cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i) \right) \Rightarrow \mathbf{R} \right).$$

Proof: Define $\Sigma = (\sigma, RT, \iota)$ such that

$$\Sigma \vDash \bigvee_{k\in[0,K]} cnt = k \wedge \mathsf{len}(lst) = k \wedge \mathsf{L}(lst, k),$$

then there exists $m \in [0, K]$ such that $[\![cnt]\!]_\Sigma = |[\![lst]\!]_\Sigma| = m$, and

(a) For all $k \in [1, m]$, $r_1, \ldots, r_N$ and Λ, if $\Lambda = [\![lst]\!]_\Sigma$ and $r_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma]$ for all $i \in [1, N]$, then $\mathcal{E}[\Lambda\langle k\rangle](r_1, \ldots, r_N) =$ true.

Then suppose

$$\Sigma \vDash \bigvee_{i\in[1,K]} i \leq cnt \wedge \mathsf{FWT}(\mathsf{pf}(lst, i), wt, i).$$

Know that $f_{\mathsf{WT}}([\![\mathsf{pf}(lst, j)]\!]_\Sigma) = wt$ for some $j \in [1, m]$, and we only need to prove that $\Sigma \vDash \mathbf{R}$:

(b) For all $l \in [1, |g_{\mathsf{WT}}(wt)|]$ and $r_1, \ldots, r_N$, if for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$ we have $r_i = RT[i][[\![\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)]\!]_\Sigma]$, then

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \ldots, r_N) = \text{true}.$$

Let l and $r_1, \ldots, r_N$ satisfy the premise of (2b), then from Lem. 100, we have the following: if $\Lambda = [\![\mathsf{pf}(lst, j)]\!]_\Sigma$, then there exists k such that $\Lambda\langle k\rangle = g_{\mathsf{WT}}(wt)\langle l\rangle$, and for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$ we have

$$[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma = [\![\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)]\!]_\Sigma.$$

Let $r'_1, \ldots, r'_N$ satisfy $r'_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma]$ for all $i \in [1, N]$, then from (2a) we know that

$$\mathcal{E}[\Lambda\langle k\rangle](r'_1, \ldots, r'_N) = \text{true},$$

which implies

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r'_1, \ldots, r'_N) = \text{true}$$

by $\Lambda\langle k\rangle = g_{\mathsf{WT}}(wt)\langle l\rangle$. Since $(r'_1, \ldots, r'_N)$ and $(r_1, \ldots, r_N)$ agree on all positions i such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$, by definition we can prove that

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r'_1, \ldots, r'_N) = \mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \ldots, r_N),$$

and thus $\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \ldots, r_N) = \text{true}$.

$$\begin{aligned}
\mathsf{LM}(m) &\triangleq \textstyle\bigwedge_{l\in[1,m]} .\ 1 \leq \mathit{lst}\langle l\rangle \leq M\\
\mathsf{CL}(n) &\triangleq 0 \leq \mathit{cnt} < n \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge \mathsf{LM}(\mathit{cnt})\\
\mathsf{ve}(i,\Lambda,l) &\triangleq \textstyle\sum_{l'\in[1,l]}[\mathsf{vbl}(\Lambda\langle l'\rangle, i)]\\
\mathsf{L}(\Lambda,m) &\triangleq \textstyle\bigwedge_{l\in[0,m)} .\ \forall V_1,\ldots,V_N .\ \Big(\bigwedge_{i\in[1,N]} .\ V_i = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l)]\Big)\\
&\qquad\qquad\qquad\qquad\qquad \Rightarrow \mathsf{hold}(\Lambda\langle l+1\rangle, V_1,\ldots,V_N)\\
\mathsf{U}(\Lambda,l) &\triangleq \textstyle\bigwedge_{i\in[1,N]} .\ x[i] = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l)] \wedge \mathsf{hd}_i = \mathsf{ve}(i,\Lambda,l)+1\\
\mathsf{U}'(\Lambda,l,l',j,j') &\triangleq \Big(\textstyle\bigwedge_{i\in[1,j)} .\ x[i] = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l')] \wedge \mathsf{hd}_i = \mathsf{ve}(i,\Lambda,l')+1\Big)\\
&\quad\wedge \Big(\textstyle\bigwedge_{i\in[j',N]} .\ x[i] = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l)] \wedge \mathsf{hd}_i = \mathsf{ve}(i,\Lambda,l)+1\Big)\\
\mathsf{CLU}(n) &\triangleq 0 \leq \mathit{cnt} < n \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge \mathsf{L}(\mathit{lst},\mathit{cnt}) \wedge \mathsf{U}(\mathit{lst},\mathit{cnt})\\
\mathsf{UG}(\Lambda,l) &\triangleq \textstyle\bigwedge_{i\in[1,N]} .\ \mathsf{hd}_i = \mathsf{ve}(i,\Lambda,l) \wedge (\mathsf{hd}_i = 0 \vee x[i] = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l)-1])\\
\mathsf{UG}'(\Lambda,l,l',j,j') &\triangleq\\
&\Big(\textstyle\bigwedge_{i\in[1,j)} .\ \mathsf{hd}_i = \mathsf{ve}(i,\Lambda,l') \wedge (\mathsf{hd}_i = 0 \vee x[i] = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l')-1])\Big)\\
&\wedge \Big(\textstyle\bigwedge_{i\in[j',N]} .\ \mathsf{hd}_i = \mathsf{ve}(i,\Lambda,l) \wedge (\mathsf{hd}_i = 0 \vee x[i] = \mathsf{RT}[i][\mathsf{ve}(i,\Lambda,l)-1])\Big)\\
S(k,j) &\triangleq \textstyle\biguplus_{i:1\leq i\leq j\wedge \mathrm{vbl}(k,i)}\{x[i]\}\\
S^{+}(k,j) &\triangleq \{\mathit{succ}\} \uplus S_{k,j}\\
\mathsf{D}(S) &\triangleq \textstyle\bigwedge_{x[i]\in S} .\ x[i] \sim i\\
\mathsf{P}(n) &\triangleq \Pr[\mathit{succ} = 1] = \textstyle\prod_{i\in[1,n]} \mathrm{P}(\mathcal{E}[\Lambda\langle i\rangle])
\end{aligned}$$

Fig. 36. Auxiliary assertions for Thm. 4.

3. The side condition that precedes the last **if** in Fig. 40: for $\Lambda = g_{\mathsf{WT}}(wt)$ and j,
$$\vDash_{\mathrm{RT}} \mathbf{R} \wedge \mathsf{UG}(\Lambda,j) \wedge z = \Lambda\langle j\rangle \Rightarrow \mathsf{hold}(z, x[1],\ldots,x[N]).$$
Proof: Define $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \vDash \mathbf{R} \wedge \mathsf{UG}(\Lambda,j) \wedge z = \Lambda\langle j\rangle$. We only need to prove that $\Sigma \vDash \mathsf{hold}(z, x[1],\ldots,x[N])$. For each $i \in [1,N]$ such that $\mathrm{vbl}(\mathcal{E}[\Lambda\langle j\rangle], i)$, we have $[\![\mathsf{ve}(i,\Lambda,j)]\!]_\Sigma \geq 1$. By $\Sigma \vDash \mathsf{UG}(\Lambda,j)$, this implies that $[\![\mathsf{hd}_i]\!]_\Sigma \neq 0$, and thus
$$\begin{aligned}
[\![x[i]]\!]_\Sigma &= RT[i][[\![\mathsf{ve}(i,\Lambda,j)]\!]_\Sigma - 1]\\
&= RT[i][[\![\mathsf{ve}(i,\Lambda,j-1)]\!]_\Sigma].
\end{aligned}$$
Now, from $\Sigma \vDash \mathbf{R}$ we know that $\Sigma \vDash \mathsf{hold}(z, x[1],\ldots,x[N])$ must hold.

□

```
$[\mathsf{true} \wedge \mathsf{hdinit}]$
$d := 1;$
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d);$
    $x[d] := a;$
    $d := d + 1;$
$[\mathsf{true}]$
$\mathit{flag} := 0;\ \mathit{cnt} := 0;\ \mathit{lst} := [];$
$[\mathit{cnt} = 0 \wedge \mathit{lst} = []]$
$[\mathsf{CL}(K+1)]$
while $(\mathit{flag} = 0 \wedge \mathit{cnt} < K)$ do
    $[\mathsf{CL}(K) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    $z := 0;\ h := 1;$
    $[\mathsf{CL}(K) \wedge 0 \leq z \leq M \wedge 1 \leq h \leq M + 1 \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    while $(h \leq M)$ do
        $[\mathsf{CL}(K) \wedge 0 \leq z \leq M \wedge 1 \leq h \leq M \wedge \mathit{flag} = 0$
                $\wedge K - \mathit{cnt} - \mathit{flag} = X \wedge M + 1 - h = X']$
        if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then $z := h;$
        $h := h + 1;$
        $[\mathsf{CL}(K) \wedge 0 \leq z \leq M \wedge 1 \leq h \leq M + 1 \wedge \mathit{flag} = 0$
                $\wedge K - \mathit{cnt} - \mathit{flag} = X \wedge M + 1 - h + 1 \leq X']$
    $[\mathsf{CL}(K) \wedge 0 \leq z \leq M \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    if $(z = 0)$ then
        $[\mathsf{CL}(K) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $\mathit{flag} := 1;$
        $[\mathsf{CL}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
    else
        $[\mathsf{CL}(K) \wedge 1 \leq z \leq M \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $[0 \leq \mathit{cnt} < K \wedge \mathsf{LM}(\mathit{cnt}) \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge 1 \leq z \leq M$
                    $\wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $\mathit{cnt} := \mathit{cnt} + 1;\ \mathit{lst} := \mathsf{app}(\mathit{lst}, z);$
        $[1 \leq \mathit{cnt} \leq K \wedge \mathsf{LM}(\mathit{cnt}) \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $[\mathsf{CL}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $d := 1;$
        while $(d \leq N)$ do ;
            if $(\mathsf{vbl}(z, d))$ then
                $a := \mathsf{Sample}(d);$
                $x[d] := a;$
            $d := d + 1;$
        $[\mathsf{CL}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
    $[\mathsf{CL}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
$[\mathsf{CL}(K+1)]$
```

$$\left[\mathit{cnt} = \sum_{wt \in WTMap(K)} \left[\bigvee_{i \in [1,K]} i \leq \mathit{cnt} \wedge \mathsf{FWT}(\mathsf{pf}(\mathit{lst}, i), wt, i)\right]\right]$$

Fig. 37. Proof of (49).

```
$\{\mathsf{true} \wedge \mathsf{hdinit}\}$
$d := 1;$
$\Big\{1 \le d \le N+1 \wedge \Big(\bigwedge_{i\in[1,N]} \cdot (i \ge d \Rightarrow \mathsf{hd}_i = 0)$
$\qquad \wedge (i < d \Rightarrow x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1)\Big)\Big\}$
while $(d \le N)$ do
    $\Big\{1 \le d \le N \wedge \Big(\bigwedge_{i\in[1,N]} \cdot (i \ge d \Rightarrow \mathsf{hd}_i = 0)$
    $\qquad \wedge (i < d \Rightarrow x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1)\Big)\Big\}$
    $a := \mathsf{Sample}(d);$
    $\Big\{1 \le d \le N \wedge \Big(\bigwedge_{i\in[1,N]} \cdot (i > d \Rightarrow \mathsf{hd}_i = 0) \wedge (i = d \Rightarrow a = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1)$
    $\qquad \wedge (i < d \Rightarrow x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1)\Big)\Big\}$
    $x[d] := a;$
    $\Big\{1 \le d \le N \wedge \Big(\bigwedge_{i\in[1,N]} \cdot (i > d \Rightarrow \mathsf{hd}_i = 0)$
    $\qquad \wedge (i \le d \Rightarrow x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1)\Big)\Big\}$
    $d := d + 1;$
    $\Big\{1 \le d \le N+1 \wedge \Big(\bigwedge_{i\in[1,N]} \cdot (i \ge d \Rightarrow \mathsf{hd}_i = 0)$
    $\qquad \wedge (i < d \Rightarrow x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1)\Big)\Big\}$
$\Big\{\bigwedge_{i\in[1,N]} \cdot x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1\Big\}$
$\mathit{flag} := 0;\ \mathit{cnt} := 0;\ \mathit{lst} := [];$
$\{\mathit{cnt} = 0 \wedge \mathit{lst} = [] \wedge \mathsf{U}([], 0)\}$
$\{\mathsf{CLU}(K+1)\}$
while $(\mathit{flag} = 0 \wedge \mathit{cnt} < K)$ do
    $\{\mathsf{CLU}(K)\}$
    $z := 0;\ h := 1;$
    $\{\mathsf{CLU}(K) \wedge (z = 0 \vee \mathsf{hold}(z, x[1], \ldots, x[N]))\}$
    while $(h \le M)$ do
        if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then $z := h;$
        $h := h + 1;$
    $\{\mathsf{CLU}(K) \wedge (z = 0 \vee \mathsf{hold}(z, x[1], \ldots, x[N]))\}$
    if $(z = 0)$ then
        $\{\mathsf{CLU}(K)\}$   $\mathit{flag} := 1;$   $\{\mathsf{CLU}(K+1)\}$
    else
        $\{\mathsf{CLU}(K) \wedge \mathsf{hold}(z, x[1], \ldots, x[N])\}$
        $\ldots$
        $\{\mathsf{CLU}(K+1)\}$
    $\{\mathsf{CLU}(K+1)\}$
$\Big\{\bigvee_{k\in[0,K]} \cdot \mathit{cnt} = k \wedge \mathsf{len}(\mathit{lst}) = k \wedge \mathsf{L}(\mathit{lst}, k)\Big\}$
$\Big\{\Big(\bigvee_{i\in[1,K]} \cdot i \le \mathit{cnt} \wedge \mathsf{FWT}(\mathsf{pf}(\mathit{lst}, i), \mathit{ds}, i)\Big) \Rightarrow \mathbf{R}\Big\}$
```

Fig. 38. Proof of (52) (part I).

```
{$\mathsf{CLU}(K) \wedge \mathsf{hold}(z, x[1], \ldots, x[N])$}
{$0 \le cnt < K \wedge \mathsf{len}(lst) = cnt \wedge \mathsf{L}(lst, cnt) \wedge \mathsf{U}(lst, cnt) \wedge \mathsf{hold}(z, x[1], \ldots, x[N])$}
$cnt := cnt + 1;\ lst := \mathsf{app}(lst, z);$
{$1 \le cnt \le K \wedge \mathsf{len}(lst) = cnt \wedge lst\langle cnt\rangle = z \wedge \mathsf{L}(lst, cnt - 1)$
            $\wedge\, \mathsf{U}(lst, cnt - 1) \wedge \mathsf{hold}(z, x[1], \ldots, x[N])$}
{$1 \le cnt \le K \wedge \mathsf{len}(lst) = cnt \wedge lst\langle cnt\rangle = z \wedge \mathsf{L}(lst, cnt) \wedge \mathsf{U}(lst, cnt - 1)$}
$d := 1;$
{$1 \le cnt \le K \wedge \mathsf{len}(lst) = cnt \wedge lst\langle cnt\rangle = z \wedge \mathsf{L}(lst, cnt) \wedge \mathsf{U}(lst, cnt - 1) \wedge d = 1$}
{$1 \le d \le N + 1 \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d) \wedge \cdots$}
while ($d \le N$) do
    {$1 \le d \le N \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d)$}
    if ($\mathsf{vbl}(z, d)$) then
        {$1 \le d \le N \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d + 1)$
                  $\wedge\, \mathsf{hd}_d = \mathsf{ve}(d, lst, cnt - 1) + 1 \wedge \mathsf{vbl}(z, d)$}
        {$1 \le d \le N \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d + 1)$
                  $\wedge\, \mathsf{hd}_d = \mathsf{ve}(d, lst, cnt)$}
        $a := \mathsf{Sample}(d);$
        {$1 \le d \le N \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d + 1)$
                  $\wedge\, \mathsf{hd}_d = \mathsf{ve}(d, lst, cnt) + 1 \wedge a = \mathsf{RT}[d][\mathsf{hd}_d - 1]$}
        $x[d] := a;$
        {$1 \le d \le N \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d + 1)$
                  $\wedge\, \mathsf{hd}_d = \mathsf{ve}(d, lst, cnt) + 1 \wedge x[d] = \mathsf{RT}[d][\mathsf{hd}_d - 1]$}
    {$1 \le d \le N \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d + 1, d + 1)$}
    $d := d + 1;$
    {$1 \le d \le N + 1 \wedge \mathsf{len}(lst) = cnt \ge 1 \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, d, d)$}
{$\mathsf{len}(lst) = cnt \wedge lst\langle cnt\rangle = z \wedge \mathsf{U}'(lst, cnt - 1, cnt, N + 1, N + 1) \wedge \cdots$}
{$1 \le cnt \le K \wedge \mathsf{len}(lst) = cnt \wedge \mathsf{L}(lst, cnt) \wedge \mathsf{U}(lst, cnt)$}
{$\mathsf{CLU}(K + 1)$}
```

Fig. 39. Proof of (52) (part II).

```
$[\mathsf{true} \wedge \mathbf{R} \wedge \mathsf{hdinit}]$
$succ := 1;\ h := 1;$
$[1 \le h \le |\Lambda| + 1 \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h-1)]$
while $(h \le |\Lambda|)$ do
    $[1 \le h \le |\Lambda| \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h-1) \wedge |\Lambda| + 1 - h = X]$
    $z := \Lambda\langle h\rangle;$
    $[1 \le h \le |\Lambda| \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h-1) \wedge |\Lambda| + 1 - h = X \wedge z = \Lambda\langle h\rangle]$
    $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h-1) \wedge z = \Lambda\langle h\rangle \wedge \cdots]$
    $d := 1;$
    $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h-1) \wedge z = \Lambda\langle h\rangle \wedge d = 1]$
    $[1 \le d \le N+1 \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}'(\Lambda, h-1, h, d, d) \wedge z = \Lambda\langle h\rangle]$
    while $(d \le N)$ do
        $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}'(\Lambda, h-1, h, d, d)$
                  $\wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X']$
        if $(\mathsf{vbl}(z, d))$ then
            $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X' \wedge \mathsf{vbl}(z, d)$
                      $\wedge \mathsf{UG}'(\Lambda, h-1, h, d, d+1) \wedge \mathsf{hd}_d = \mathsf{ve}(d, \Lambda, h-1)]$
            $a := \mathsf{Sample}(d);$
            $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X' \wedge \mathsf{vbl}(z, d)$
                      $\wedge \mathsf{UG}'(\Lambda, h-1, h, d, d+1)$
                      $\wedge \mathsf{hd}_d = \mathsf{ve}(d, \Lambda, h-1) + 1 \wedge a = \mathsf{RT}[d][\mathsf{hd}_d - 1]]$
            $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X'$
                      $\wedge \mathsf{UG}'(\Lambda, h-1, h, d, d+1)$
                      $\wedge \mathsf{hd}_d = \mathsf{ve}(d, \Lambda, h) \wedge a = \mathsf{RT}[d][\mathsf{hd}_d - 1]]$
            $x[d] := a;$
            $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X'$
                      $\wedge \mathsf{UG}'(\Lambda, h-1, h, d, d+1)$
                      $\wedge \mathsf{hd}_d = \mathsf{ve}(d, \Lambda, h) \wedge x[d] = \mathsf{RT}[d][\mathsf{hd}_d - 1]]$
            $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X'$
                      $\wedge \mathsf{UG}'(\Lambda, h-1, h, d+1, d+1)]$
        $[1 \le d \le N \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d = X'$
                  $\wedge \mathsf{UG}'(\Lambda, h-1, h, d+1, d+1)]$
        $d := d + 1;$
        $[1 \le d \le N+1 \wedge \mathbf{R} \wedge succ = 1 \wedge z = \Lambda\langle h\rangle \wedge N + 1 - d + 1 \le X'$
                  $\wedge \mathsf{UG}'(\Lambda, h-1, h, d, d)]$
    $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}'(\Lambda, h-1, h, N+1, N+1) \wedge z = \Lambda\langle h\rangle]$
    $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h) \wedge z = \Lambda\langle h\rangle]$
    $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h) \wedge z = \Lambda\langle h\rangle \wedge \mathsf{hold}(z, x[1], \ldots, x[N])]$
    if $(\neg\mathsf{hold}(z, x[1], \ldots, x[N]))$ then
        $[\mathsf{false}]$
        $succ := 0;$
        $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h)]$
    $[\mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h) \wedge \cdots]$
    $[1 \le h \le |\Lambda| \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h) \wedge |\Lambda| + 1 - h = X]$
    $h := h + 1;$
    $[1 \le h \le |\Lambda| + 1 \wedge \mathbf{R} \wedge succ = 1 \wedge \mathsf{UG}(\Lambda, h-1) \wedge |\Lambda| + 1 - h + 1 \le X]$
$[succ = 1]$
```

Fig. 40. Proof of (53).

```
$[\mathbf{true}]$
$succ := 1;\ h := 1;\quad [\lceil succ = 1 \wedge h = 1\rceil]$
$\left[\left(\left(\bigvee_{j\in[1,|\Lambda|]}.\lceil h = j\rceil \wedge \mathsf{P}(j-1)\right) \wedge \lceil h \le |\Lambda|\rceil\right) \vee (\mathsf{P}(|\Lambda|) \wedge \lceil\neg(h \le |\Lambda|)\rceil)\right]$
$\mathbf{while}\ (h \le |\Lambda|)\ \mathbf{do}$
    $\left[\left(\bigvee_{j\in[1,|\Lambda|]}.\lceil h = j\rceil \wedge \mathsf{P}(j-1)\right) \wedge \lceil h \le |\Lambda| \wedge |\Lambda| + 1 - h = X\rceil\right]$
    $z := \Lambda\langle h\rangle;$
    $\left[\left(\bigvee_{j\in[1,|\Lambda|]}.\lceil h = j \wedge z = \Lambda\langle h\rangle\rceil \wedge \mathsf{P}(j-1)\right) \wedge \lceil h \le |\Lambda| \wedge |\Lambda| + 1 - h = X\rceil\right]$
    $[\cdots \lceil h = j \wedge z = \Lambda\langle h\rangle\rceil \wedge \mathsf{P}(j-1) \wedge \cdots]$
    $d := 1;$
    $[\lceil z = \Lambda\langle j\rangle \wedge d = 1\rceil \wedge \cdots]$
    $\left[\left(\left(\bigvee_{i\in[1,N]}.\lceil z = \Lambda\langle j\rangle \wedge d = i\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1})\right) \wedge \lceil d \le N\rceil\right)\right.$
      $\left.\vee \left(\lceil z = \Lambda\langle j\rangle\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,N}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,N}) \wedge \lceil\neg(d \le N)\rceil\right)\right]$
    $\mathbf{while}\ (d \le N)\ \mathbf{do}$
        $\left[\left(\bigvee_{i\in[1,N]}.\lceil z = \Lambda\langle j\rangle \wedge d = i\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1})\right)\right.$
          $\left.\wedge \lceil d \le N \wedge N + 1 - d = X\rceil\right]$
        $\ldots$
        $\left[\left(\left(\bigvee_{i\in[1,N]}.\lceil z = \Lambda\langle j\rangle \wedge d = i\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1})\right)\right.\right.$
            $\left.\wedge \lceil d \le N \wedge N + 1 - d + 1 \le X\rceil\right)$
          $\left.\vee \left(\lceil z = \Lambda\langle j\rangle\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,N}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,N}) \wedge \lceil\neg(d \le N)\rceil\right)\right]$
    $\left[\lceil z = \Lambda\langle j\rangle\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,N}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,N}) \wedge \cdots\right]$
    $\left[\mathsf{P}(j-1) \wedge \lceil h = j \wedge z = \Lambda\langle h\rangle\rceil \wedge \#(S^+_{\Lambda\langle j\rangle,N}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,N})\right]$
    $\left[(\mathsf{P}(j-1) \wedge \lceil h = j\rceil \wedge \lceil\neg\mathsf{hold}(z, x[1], \ldots, x[N])\rceil)\right.$
      $\left.\oplus_{1-\mathrm{P}(\mathcal{E}[\Lambda\langle j\rangle])} (\mathsf{P}(j-1) \wedge \lceil h = j\rceil \wedge \lceil\neg\neg\mathsf{hold}(z, x[1], \ldots, x[N])\rceil)\right]$
    $\mathbf{if}\ (\neg\mathsf{hold}(z, x[1], \ldots, x[N]))\ \mathbf{then}$
        $[\mathsf{P}(j-1) \wedge \lceil h = j\rceil \wedge \lceil\neg\mathsf{hold}(z, x[1], \ldots, x[N])\rceil]$
        $succ := 0;$
        $[\Pr[succ = 1] = 0 \wedge \lceil h = j\rceil]$
    $\left[(\Pr[succ = 1] = 0 \wedge \lceil h = j\rceil) \oplus_{1-\mathrm{P}(\mathcal{E}[\Lambda\langle j\rangle])} (\mathsf{P}(j-1) \wedge \lceil h = j\rceil)\right]$
    $[\cdots \mathsf{P}(j) \wedge \lceil h = j\rceil \wedge \cdots]$
    $\left[\left(\bigvee_{j\in[1,|\Lambda|]}.\lceil h = j\rceil \wedge \mathsf{P}(j)\right) \wedge \lceil h \le |\Lambda| \wedge |\Lambda| + 1 - h = X\rceil\right]$
    $h := h + 1;$
    $\left[\left(\bigvee_{j\in[1,|\Lambda|]}.\lceil h = j + 1\rceil \wedge \mathsf{P}(j)\right) \wedge \lceil h \le |\Lambda| + 1 \wedge |\Lambda| + 2 - h = X\rceil\right]$
    $\left[\left(\left(\bigvee_{j\in[1,|\Lambda|]}.\lceil h = j\rceil \wedge \mathsf{P}(j-1)\right) \wedge \lceil h \le |\Lambda| \wedge |\Lambda| + 1 - h + 1 \le X\rceil\right)\right.$
      $\left.\vee (\mathsf{P}(|\Lambda|) \wedge \lceil\neg(h \le |\Lambda|)\rceil)\right]$
$[\mathsf{P}(|\Lambda|)]$
```

Fig. 41. Proof of (51) (part I).

```
$\Big[\Big(\bigvee_{i \in [1,N]} . \lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1})\Big)$
  $\wedge \lceil d \leq N \wedge N+1-d = X \rceil\Big]$
$\Big[\cdots \lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1}) \wedge \cdots\Big]$
$\Big[\Big(\lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1}) \wedge \lceil \mathsf{vbl}(z,d) \rceil\Big) \oplus_1 \cdots\Big]$
**if** $(\mathsf{vbl}(z,d))$ **then**
    $\Big[\lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge (\exists X.\, \lceil d = X \rceil) \wedge \#(S^{+}_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1}) \wedge \lceil \mathsf{vbl}(z,d) \rceil\Big]$
    $a := \mathsf{Sample}(d);$
    $\Big[\lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i-1} \cup \{a\}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1}) \wedge \lceil \mathsf{vbl}(z,d) \rceil \wedge a \sim d\Big]$
    $x[d] := a;$
    $\Big[\lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i})\Big]$
$\Big[\cdots \lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i}) \wedge \cdots\Big]$
$\Big[\Big(\bigvee_{i \in [1,N]} . \lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i})\Big)$
  $\wedge \lceil d \leq N \wedge N+1-d = X \rceil\Big]$
$d := d + 1;$
$\Big[\Big(\bigvee_{i \in [1,N]} . \lceil z = \Lambda\langle j\rangle \wedge d = i+1 \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i})\Big)$
  $\wedge \lceil d \leq N+1 \wedge N+2-d = X \rceil\Big]$
$\Big[\Big(\Big(\bigvee_{i \in [1,N]} . \lceil z = \Lambda\langle j\rangle \wedge d = i \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,i-1}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,i-1})\Big)$
    $\wedge \lceil d \leq N \wedge N+1-d+1 \leq X \rceil\Big)$
  $\vee \Big(\lceil z = \Lambda\langle j\rangle \rceil \wedge \#(S^{+}_{\Lambda\langle j\rangle,N}) \wedge \mathsf{D}(S_{\Lambda\langle j\rangle,N}) \wedge \lceil \neg(d \leq N) \rceil\Big)\Big]$
```

Fig. 42. Proof of (51) (part II).

J.2 Our New Result

Definition 7. *For all reals r,* $\mathrm{MTpre}(r)$ *holds iff there exist*

- *Set DS;*
- *Function $DSMap \in Nat \to \mathcal{P}_{\mathsf{fin}}(DS)$;*
- *Function $f \in ExLog \to DS$;*
- *Function $g \in DS \to ExLog$;*

such that

1. *For all K and Λ, if $\Lambda \in ExLog$ and $|\Lambda| \leq K$, then*
$$f(\Lambda) \in DSMap(K);$$
2. $\mathsf{Exclusive}(f)$ *and* $\mathsf{Iterable}(f, g)$ *hold;*
3. *The following inequality holds;*
$$\sum_{ds \in DSMap(K)} \prod_{i=1}^{|g(ds)|} \mathrm{P}(\mathcal{E}[g(ds)\langle i\rangle]) \leq r$$

where $\mathsf{Exclusive}(f)$ *iff*

$$\forall \Lambda \prec \Lambda' \in ExLog.\ \ f(\Lambda) \neq f(\Lambda'),$$

and $\mathsf{Iterable}(f, g)$ *iff the following holds: for all $\Lambda, \Lambda' \in ExLog$, if $(g \circ f)(\Lambda) = \Lambda'$, then for each $l \in [1, |\Lambda'|]$ there exists k such that $\Lambda\langle k\rangle = \Lambda'\langle l\rangle$, and*

$$\forall i \in [1, N].\ \ \mathrm{vbl}(\mathcal{E}[\Lambda'\langle l\rangle], i) \implies \sum_{k' < k} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle k'\rangle], i)] = \sum_{l' < l} [\mathrm{vbl}(\mathcal{E}[\Lambda'\langle l'\rangle], i)].$$

Theorem 8. *For all reals r, if* $\mathrm{MTpre}(r)$ *holds, then*

$$\models [\mathbf{true}]\, C_{\mathsf{MT}}(cnt)\, [\mathbb{E}[cnt] \leq r]\,.$$

We explain the idea of Def. 7. DS is the set of all instances of some witness-tree-like structures. We can rewrite $|\Lambda|$, the length of the execution log in the MT algorithm, as some intermediate expression related to the structures in DS, which is simpler to analyze. More precisely, we rewrite $|\Lambda|$ as the number of $ds \in DS$ such that, ds can be constructed from some prefix of the execution log. This construction is described by f, which is similar to f_{WT}. Also, if the length of the execution log is no more K, then in the above intermediate expression we can restrict ds to a structure with "size" no more than K (that is, $ds \in DSMap(K)$), with the help of the first premise. This makes the expression well-defined, as the number of possible ds from $DSMap(K)$ must be finite.

Then, to bound the probability of ds being constructed from some prefix of the execution log, similar to Sec. 2.1, we introduce a program check(ds) to enumerate the events in ds in a specific order. This order is described by g, which is similar to g_{WT}. With check(ds), we can reduce the proof of the bound to a coupling proof with the MT algorithm and check(ds) involved.

$\mathsf{Exclusive}(f)$ requires that the structures constructed from all prefixes of the execution log are pairwise distinct. $\mathsf{Iterable}(f, g)$ captures an important property of the witness-tree-like structures that, for all events η in the structure ds, and for all variables X_i that η depends on, X_i has been resampled in check(ds) before η being picked as many times as X_i has been resampled in the MT algorithm before η being picked. The third premise is similar to (4).

Below we prove Thm. 8. The proof is analogy to Thm. 4, except that we extend the witness tree to general witness-tree-like structures.

Proof. Assume that MTpre(r) holds. By applying Lem. 77, Thm. 2 and Lem. 81, with the auxiliary code $C'_{\mathsf{MT}}(cnt, K)$ defined in Fig. 35, we only need to prove that, for all K,

$$\models [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K)\, [\mathbb{E}[cnt] \le r \wedge \lceil cnt \ge 0 \rceil]\,. \tag{54}$$

From MTpre(r), we know that

$$\models \sum_{ds \in DSMap(K)} \prod_{i=1}^{|g(ds)|} \mathrm{P}(\mathcal{E}[g(ds)]\langle i \rangle) \le r;$$

thus, to prove (54), by Lem. 77, we only need to prove that

$$\models [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \left[\lceil cnt \ge 0 \rceil \wedge \mathbb{E}[cnt] \le \sum_{ds \in DSMap(K)} \prod_{i=1}^{|g(ds)|} \mathrm{P}(\mathcal{E}[g(ds)]\langle i \rangle) \right]. \tag{55}$$

Informally, $DSMap(K)$ is the set of all data structures with size no more than K, and $g(ds)$ is a reversed BFS ordering of data structure ds.

Then we define $\mathsf{FDS}(e, ds, i)$ as follows:

$$\mathsf{FDS}(e, ds, i) \triangleq \bigvee\nolimits_{\Lambda \in f^{-1}(ds) \,:\, |\Lambda| = i} \,.\; e = \Lambda.$$

Informally, $\mathsf{FDS}(e, ds, i)$ holds iff ds can be constructed from the execution log e, where e is of length i. For execution log $\Lambda \in ExLog$, $f(\Lambda)$ is the data structure constructed from Λ. Now, to prove (55), by repeatedly applying Lem. 78, we only need to prove the following two subgoals:

$$\models [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \left[\lceil cnt \ge 0 \rceil \wedge \mathbb{E}[cnt] = \sum_{ds \in DSMap(K)} \Pr\left[\bigvee_{i \in [1,K]} i \le cnt \wedge \mathsf{FDS}(\mathsf{pf}(lst, i), ds, i) \right] \right], \tag{56}$$

and for all $ds \in \mathit{DSMap}(K)$

$$\vDash \{\mathbf{true}\}\, C'_{\mathsf{MT}}(\mathit{cnt}, K) \left\{ \Pr\left[\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right] \leq \prod_{i=1}^{|g(ds)|} \mathrm{P}(\mathcal{E}[g(ds)]\langle i\rangle) \right\}. \tag{57}$$

For (56), from Lem. 77 and the linearity of expectation, we only need to prove that

$$\vDash [\mathbf{true}]\, C'_{\mathsf{MT}}(\mathit{cnt}, K) \left[\left[\mathit{cnt} = \sum_{ds\in \mathit{DSMap}(K)} \left[\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right]\right]\right]; \tag{58}$$

then by Lem. 80, to prove (58), we only need to prove the following:

$$\vDash_{\mathrm{RT}} [\mathrm{true} \wedge \mathsf{hdinit}]\, C'_{\mathsf{MT}}(\mathit{cnt}, K) \left[\mathit{cnt} = \sum_{ds\in \mathit{DSMap}(K)} \left[\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right]\right]. \tag{59}$$

For (57), by Lem. 79, with the auxiliary code $C_{\mathsf{check}}(\Lambda)$ defined in Fig. 35, we only need to prove the following two subgoals:

$$\vDash \{\mathbf{true}\}\, C'_{\mathsf{MT}}(\mathit{cnt}, K), C_{\mathsf{check}}(g(ds)) \left\{ \left(\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right), \mathit{succ} = 1 \right\}, \tag{60}$$

and

$$\vDash [\mathbf{true}]\, C_{\mathsf{check}}(g(ds)) \left[\Pr[\mathit{succ} = 1] = \prod_{i=1}^{|g(ds)|} \mathrm{P}(\mathcal{E}[g(ds)]\langle i\rangle)\right]. \tag{61}$$

For (60), by RT-based coupling (Thm. 3), we only need to prove the following two subgoals:

$$\vDash_{\mathrm{RT}} \{\mathrm{true} \wedge \mathsf{hdinit}\}\, C'_{\mathsf{MT}}(\mathit{cnt}, K) \left\{ \left(\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right) \Rightarrow \mathbf{R} \right\}, \tag{62}$$

and

$$\vDash_{\mathrm{RT}} [\mathrm{true} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathsf{check}}(g(ds))\, [\mathit{succ} = 1]\,, \tag{63}$$

where $\mathbf{R}$ is defined below.

$$\begin{aligned}\mathbf{R} \triangleq {} & \textstyle\bigwedge_{l\in[1,|g(ds)|]} .\ \forall V_1, \dots V_N . \\ & \quad (\textstyle\bigwedge_{i\in[1,N]} .\ \mathsf{vbl}(g(ds)\langle l\rangle, i) \\ & \qquad\qquad \Rightarrow V_i = \mathsf{RT}[i][\mathsf{ve}(i, g(ds), l-1)]) \\ & \qquad \Rightarrow \mathsf{hold}(g(ds)\langle l\rangle, V_1, \dots, V_N) \\ \mathsf{ve}(i, \Lambda, l) \triangleq {} & \textstyle\sum_{l'\in[1,l]} [\mathsf{vbl}(\Lambda\langle l'\rangle, i)]\end{aligned}$$

This $\mathbf{R}$ is analogy to the one defined in the proof of Thm. 4, except that the wt there is now replaced with ds, and g_{WT} is replaced with g.

Now it remains to prove (59), (62), (63) and (61). The proof of (63) and (61) are sketched in Fig. 40, Fig. 41 and Fig. 42, where we take $\Lambda = g(ds)$. For (59) and (62), we apply Thm. 7, and use inference rules of the resampling-table-based program logic (listed in Fig. 25 and Fig. 26) to complete the proof. Proofs of these two judgments are presented in Fig. 43 and Fig. 44, while the auxiliary assertions used by these proofs are again the ones defined in Fig. 36, and we omit the common parts with Fig. 37, Fig. 38 and Fig. 39.

We then show the proofs of the side conditions in Fig. 43 and Fig. 44.

1. The side condition in Fig. 43:

$$\vDash_{\mathrm{RT}} \mathsf{CL}(K+1) \Rightarrow \mathit{cnt} = \sum_{ds\in \mathit{DSMap}(K)} \left[\bigvee_{i\in[1,K]} i \le \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right].$$

   Proof: Let $\Sigma \vDash \mathsf{CL}(K+1)$, then there exists $m \in [0, K]$ such that
   - $[\![\mathit{cnt}]\!]_\Sigma = |[\![\mathit{lst}]\!]_\Sigma| = m$;
   - For all $k \in [1, m]$, $[\![\mathsf{pf}(\mathit{lst}, k)]\!]_\Sigma \in \mathit{ExLog}$;
   - For all $k \in [1, m]$, $|[\![\mathsf{pf}(\mathit{lst}, k)]\!]_\Sigma| = k \le K$.

   Thus, by MTpre(r), for all $k \in [1, m]$ we have

$$f([\![\mathsf{pf}(\mathit{lst}, k)]\!]_\Sigma) \in \mathit{DSMap}(K).$$

   Now, let

$$ds_k = f([\![\mathsf{pf}(\mathit{lst}, k)]\!]_\Sigma),$$

   then from $\mathsf{Exclusive}(f)$ we know that $ds_1 \ne \cdots \ne ds_m$, and for all $k \in [1, m]$ we have the following:
   - $\Sigma \vDash \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, k), ds_k, k)$;
   - For all $ds \ne ds_k$, $\Sigma \vDash \neg\mathsf{FDS}(\mathsf{pf}(\mathit{lst}, k), ds, k)$.

   Thus, one can verify that

$$\Sigma \vDash \mathit{cnt} = \sum_{ds\in \mathit{DSMap}(K)} \left[\bigvee_{i\in[1,K]} i \le \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i)\right].$$

2. The side condition in Fig. 44:

$$\models_{\mathrm{RT}} \left(\bigvee_{k\in[0,K]} \mathit{cnt} = k \wedge \mathsf{len}(\mathit{lst}) = k \wedge \mathsf{L}(\mathit{lst}, k) \right) \Rightarrow$$
$$\left(\left(\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i) \right) \Rightarrow \mathbf{R} \right).$$

Proof: Define $\Sigma = (\sigma, RT, \iota)$ such that

$$\Sigma \models \bigvee_{k\in[0,K]} \mathit{cnt} = k \wedge \mathsf{len}(\mathit{lst}) = k \wedge \mathsf{L}(\mathit{lst}, k),$$

then there exists $m \in [0, K]$ such that $[\![\mathit{cnt}]\!]_\Sigma = |[\![\mathit{lst}]\!]_\Sigma| = m$, and

(a) For all $k \in [1, m]$, $r_1, \ldots, r_N$ and Λ, if $\Lambda = [\![\mathit{lst}]\!]_\Sigma$ and $r_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma]$ for all $i \in [1, N]$, then $\mathcal{E}[\Lambda\langle k\rangle](r_1, \ldots, r_N) =$ true.

Then suppose

$$\Sigma \models \bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FDS}(\mathsf{pf}(\mathit{lst}, i), ds, i).$$

Know that $f([\![\mathsf{pf}(\mathit{lst}, j)]\!]_\Sigma) = ds$ for some $j \in [1, m]$, and we only need to prove that $\Sigma \models \mathbf{R}$:

(b) For all $l \in [1, |g(ds)|]$ and $r_1, \ldots, r_N$, if

$$r_i = RT[i][[\![\mathsf{ve}(i, g(ds), l-1)]\!]_\Sigma]$$

for all $i \in [1, N]$ such that $\mathrm{vbl}(g(ds)\langle l\rangle, i)$, then

$$\mathcal{E}[g(ds)\langle l\rangle](r_1, \ldots, r_N) = \text{true}.$$

Let l and $r_1, \ldots, r_N$ satisfy the premise of (2b), then from $\mathsf{Iterable}(f, g)$ we have: with $\Lambda = [\![\mathsf{pf}(\mathit{lst}, j)]\!]_\Sigma$, there exists k such that $\Lambda\langle k\rangle = g(ds)\langle l\rangle$, and for all $i \in [1, N]$ such that $\mathrm{vbl}(g(ds)\langle l\rangle, i)$ we have

$$[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma = [\![\mathsf{ve}(i, g(ds), l-1)]\!]_\Sigma.$$

Define $r'_1, \ldots, r'_N$ such that $r'_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma]$ for all $i \in [1, N]$, then from (2a) we know that

$$\mathcal{E}[\Lambda\langle k\rangle](r'_1, \ldots, r'_N) = \text{true},$$

which implies

$$\mathcal{E}[g(ds)\langle l\rangle](r'_1, \ldots, r'_N) = \text{true}$$

by $\Lambda\langle k\rangle = g(ds)\langle l\rangle$. Since $(r'_1, \ldots, r'_N)$ and $(r_1, \ldots, r_N)$ agree on all positions i such that $\mathrm{vbl}(g(ds)\langle l\rangle, i)$, by definition we can prove that

$$\mathcal{E}[g(ds)\langle l\rangle](r'_1, \ldots, r'_N) = \mathcal{E}[g(ds)\langle l\rangle](r_1, \ldots, r_N),$$

and thus $\mathcal{E}[g(ds)\langle l\rangle](r_1, \ldots, r_N) = \text{true}$.

□

$[\text{true} \wedge \mathsf{hdinit}]$
$C'_{\mathsf{MT}}(cnt, K)$
$[\mathsf{CL}(K+1)]$
$$\left[cnt = \sum_{ds \in DSMap(K)} \left[\bigvee_{i \in [1,K]} i \le cnt \wedge \mathsf{FDS}(\mathsf{pf}(lst, i), ds, i)\right]\right]$$

Fig. 43. Proof of (59).

$\{\text{true} \wedge \mathsf{hdinit}\}$
$C'_{\mathsf{MT}}(cnt, K)$
$$\left\{\bigvee_{k \in [0,K]} cnt = k \wedge \mathsf{len}(lst) = k \wedge \mathsf{L}(lst, k)\right\}$$
$$\left\{\left(\bigvee_{i \in [1,K]} i \le cnt \wedge \mathsf{FDS}(\mathsf{pf}(lst, i), ds, i)\right) \Rightarrow \mathbf{R}\right\}$$

Fig. 44. Proof of (62).

J.3 Theorem 1.4 of [54] and Theorem 4 of [44]

We prove Thm. 4, Thm. 9 and Thm. 10 by directly applying Thm. 8.

An Alternative Proof of Thm. 4. Fixing $\alpha_1, \dots, \alpha_M$ below, we prove that MTpre(r_{EL}) holds. Take $DS = WT$, $DSMap = WTMap$, $f = f_{\mathsf{WT}}$, $g = g_{\mathsf{WT}}$, then the proof follows from Lem. 98, Lem. 99, Lem. 100 and Lem. 101. □

Theorem 9. *For all reals $\beta_1, \dots, \beta_M \in (0, \infty)$, if the cluster expansion condition [10]*

$$\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \le \beta_i \left(\sum_{\substack{I \subseteq \Gamma^+(i) \\ \mathrm{Indep}(I)}} \prod_{j \in I} \beta_j\right)^{-1}$$

holds, then

$$\models [\mathbf{true}]\, C_{\mathsf{MT}}(cnt)\, [\mathbb{E}[cnt] \le r_{\mathsf{CE}}]\,,$$

where

$$r_{\mathsf{CE}} = \sum_{i \in [1,M]} \beta_i.$$

Proof. Fix $\beta_1, \dots, \beta_M$ below. We prove that MTpre(r_{CE}) holds. Take $DS = WT$, $DSMap = SWTMap$, $f = f_{\mathsf{WT}}$, $g = g_{\mathsf{WT}}$, where $SWTMap$ is defined in App. I.3. Then the proof follows from Lem. 127, Lem. 99, Lem. 100 and Lem. 128. □

Theorem 10. *If the Shearer's condition [57]*

$$\forall I \subseteq [1, M].\ \mathrm{Indep}(I) \implies q_I > 0$$

holds, then

$$\vDash [\mathbf{true}]\, C_{\mathsf{MT}}(cnt)\, [\mathbb{E}[cnt] \leq r_{\mathsf{S}}]\,,$$

where

$$r_{\mathsf{S}} = \sum_{i\in[1,M]} \frac{q_{\{i\}}}{q_{\varnothing}},$$
$$q_I = \sum_{\substack{I\subseteq J\subseteq[1,M]\\ \mathrm{Indep}(J)}} (-1)^{|J|-|I|} \prod_{j\in J} \mathrm{P}(\mathcal{E}[j]).$$

Proof. We prove that MTpre(r_{S}) holds. Take $DS = SSS$, $DSMap = SSSMap$, $f = f_{\mathsf{SSS}}$, $g = g_{\mathsf{SSS}}$; see App. I.4 for detailed definitions of $SSS, SSSMap, f_{\mathsf{SSS}}, g_{\mathsf{SSS}}$. Then the proof follows from Lem. 131, Lem. 132, Lem. 133 and Lem. 134. □

J.4 Theorem 6.1 of [51]

Theorem 11. *For all reals $\alpha_1, \ldots, \alpha_M \in (0,1)$, if*

$$\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \leq \alpha_i \prod_{j\in\Gamma'(i)} (1-\alpha_j),$$

then

$$\vDash [\mathbf{true}]\, C_{\mathsf{MT}}(cnt)\, [\mathbb{E}[cnt] \leq r_{\mathsf{EL}}]\,,$$

where

$$r_{\mathsf{EL}} = \sum_{i\in[1,M]} \alpha_i(1-\alpha_i)^{-1}.$$

Below we use the following notation:

$$ve(i, \Lambda, l) \triangleq \sum_{l'<l} [\mathrm{vbl}(\mathcal{E}[\Lambda\langle l'\rangle], i)]$$

Lemma 135. *For all Λ, Λ', j, RT such that $j \in [1, |\Lambda|)$ and $|\Lambda| = |\Lambda'|$, if*

- *For all $k \in [1, |\Lambda|] \setminus \{j, j+1\}$, $\Lambda\langle k\rangle = \Lambda'\langle k\rangle$;*
- *$\Lambda\langle j\rangle = \Lambda'\langle j+1\rangle$, $\Lambda\langle j+1\rangle = \Lambda'\langle j\rangle$;*
- *$\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$;*
- *$RT \vDash \mathsf{L}(\Lambda, |\Lambda|)$;*

then $RT \vDash \mathsf{L}(\Lambda', |\Lambda'|)$.

Proof. From the premise, we know that

- For all $k \in [1, |\Lambda|]$ and $q_1, \ldots, q_N$, if

$$q_i = RT[i][ve(i, \Lambda, k-1)]$$

for all $i \in [1, N]$, then $\mathcal{E}[\Lambda\langle k\rangle](q_1, \ldots, q_N) = \text{true}$.

We only need to prove that

– For all $k \in [1, |\Lambda'|]$ and $r_1, \ldots, r_N$, if

$$r_i = RT[i][ve(i, \Lambda', k-1)]$$

for all $i \in [1, N]$, then $\mathcal{E}[\Lambda'\langle k\rangle](r_1, \ldots, r_N) = \text{true}$.

The case of $k \notin \{j, j+1\}$ is trivial. Below we prove the case of $k = j$, and the case of $k = j+1$ is similar. Let

– $r_i = RT[i][ve(i, \Lambda', j-1)]$;
– $r'_i = RT[i][ve(i, \Lambda', j)]$;
– $q_i = RT[i][ve(i, \Lambda, j-1)]$;
– $q'_i = RT[i][ve(i, \Lambda, j)]$

for all $i \in [1, N]$, then

$$\mathcal{E}[\Lambda\langle j\rangle](q_1, \ldots, q_N) = \mathcal{E}[\Lambda\langle j+1\rangle](q'_1, \ldots, q'_N) = \text{true},$$

Note that $r_i = q'_i$ for all $i \in [1, N]$ such that $\neg\text{vbl}(\mathcal{E}[\Lambda\langle j\rangle], i)$. Assume that $q''_1, \ldots, q''_N$ satisfy that $q''_i = q'_i$ for all $i \in [1, N]$ such that $\text{vbl}(\mathcal{E}[\Lambda\langle j+1\rangle], i)$, and $q''_i = q_i$ for all $i \in [1, N]$ such that $\neg\text{vbl}(\mathcal{E}[\Lambda\langle j+1\rangle], i)$, then $r_i = q''_i$ for all $i \in [1, N]$ such that $\neg\text{vbl}(\mathcal{E}[\Lambda\langle j\rangle], i) \vee \neg\text{vbl}(\mathcal{E}[\Lambda\langle j+1\rangle], i)$, and

$$\mathcal{E}[\Lambda\langle j+1\rangle](q''_1, \ldots, q''_N) = \mathcal{E}[\Lambda\langle j+1\rangle](q'_1, \ldots, q'_N) = \text{true}.$$

Moreover, since $r_i = q_i$ for all $i \in [1, N]$, we have

$$\mathcal{E}[\Lambda\langle j\rangle](r_1, \ldots, r_N) = \mathcal{E}[\Lambda\langle j\rangle](q_1, \ldots, q_N) = \text{true}.$$

Thus, from the definition of $\Lambda\langle j\rangle \notin \Gamma'^{+}(\Lambda\langle j+1\rangle)$, we have

$$\mathcal{E}[\Lambda\langle j+1\rangle](r_1, \ldots, r_N) = \text{true};$$

then from $\Lambda'\langle j\rangle = \Lambda\langle j+1\rangle$ we have

$$\mathcal{E}[\Lambda'\langle j\rangle](r_1, \ldots, r_N) = \text{true}.$$

□

Proof of Thm. 11. By applying Lem. 77, Thm. 2 and Lem. 81, with the auxiliary code $C'_{\mathsf{MT}}(cnt, K)$ defined in Fig. 35, we only need to prove that, for all K,

$$\models [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K)\, [\lceil cnt \geq 0\rceil \wedge \mathbb{E}[cnt] \leq r_{\mathsf{EL}}]\,. \tag{64}$$

From the premise and Lem. 126, we know that

$$\models \sum_{wt \in LWTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \leq r_{\mathsf{EL}}. \tag{65}$$

Informally, $LWTMap(K)$ is the set of all lopsided witness trees with size no more than K, which is defined in App. I.2. With (65), to prove (64), by Lem. 77, we only need to prove that

$$\vDash [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \left[\lceil cnt \geq 0 \rceil \wedge \mathbb{E}[cnt] \leq \sum_{wt \in LWTMap(K)} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \right]. \tag{66}$$

Then we define $\mathsf{FLWT}(e, wt, i)$ as follows:

$$\mathsf{FLWT}(e, wt, i) \triangleq \bigvee\nolimits_{\Lambda \in (f_{\mathsf{LWT}})^{-1}(wt) \,:\, |\Lambda| = i} \cdot\ e = \Lambda.$$

For $\Lambda \in ExLog$, $f_{\mathsf{LWT}}(\Lambda)$ is the lopsided witness tree constructed from Λ, as defined in App. I.2. $\mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i)$ holds iff the lopsided witness tree wt can be constructed from the execution log's prefix with length i. Now, to prove (66), by repeatedly applying Lem. 77 and Lem. 78, we only need to prove the following two subgoals:

$$\vDash [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \left[\lceil cnt \geq 0 \rceil \wedge \mathbb{E}[cnt] = \sum_{wt \in LWTMap(K)} \Pr\left[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i) \right] \right], \tag{67}$$

and for all $wt \in LWTMap(K)$

$$\vDash \{\mathbf{true}\}\, C'_{\mathsf{MT}}(cnt, K) \left\{ \Pr\left[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i) \right] \leq \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \right\}. \tag{68}$$

For (67), from Lem. 77 and the linearity of expectation, we only need to prove that

$$\vDash [\mathbf{true}]\, C'_{\mathsf{MT}}(cnt, K) \left[\left[cnt = \sum_{wt \in LWTMap(K)} \left[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i) \right] \right] \right]; \tag{69}$$

then by Lem. 80, to prove (69), we only need to prove the following:

$$\vDash_{\mathrm{RT}} [\mathrm{true} \wedge \mathsf{hdinit}]\, C'_{\mathsf{MT}}(\mathit{cnt}, K) \left[\mathit{cnt} = \sum_{wt \in LWTMap(K)} \left[\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FLWT}(\mathsf{pf}(\mathit{lst}, i), wt, i)\right]\right]. \tag{70}$$

For (68), from Lem. 79 and Lem. 77, with the auxiliary code $C_{\mathsf{check}}(\mathit{\Lambda})$ defined in Fig. 35, we only need to prove the following two subgoals:

$$\vDash \{\mathbf{true}\}\, C'_{\mathsf{MT}}(\mathit{cnt}, K), C_{\mathsf{check}}(g_{\mathsf{WT}}(wt)) \left\{\left(\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FLWT}(\mathsf{pf}(\mathit{lst}, i), wt, i)\right), \mathit{succ} = 1\right\}, \tag{71}$$

and

$$\vDash [\mathbf{true}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt)) \left[\Pr[\mathit{succ} = 1] = \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\right]. \tag{72}$$

For (71), by RT-based coupling (Thm. 3), we only need to prove

$$\vDash_{\mathrm{RT}} \{\mathrm{true} \wedge \mathsf{hdinit}\}\, C'_{\mathsf{MT}}(\mathit{cnt}, K) \left\{\left(\bigvee_{i\in[1,K]} i \leq \mathit{cnt} \wedge \mathsf{FLWT}(\mathsf{pf}(\mathit{lst}, i), wt, i)\right) \Rightarrow \mathbf{R}\right\} \tag{73}$$

and

$$\vDash_{\mathrm{RT}} [\mathrm{true} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\, [\mathit{succ} = 1], \tag{74}$$

where $\mathbf{R}$ is defined below.

$$\begin{aligned}\mathbf{R} \triangleq \textstyle\bigwedge_{l\in[1,|g_{\mathsf{WT}}(wt)|]} .\ & \forall V_1, \ldots V_N. \\ & (\textstyle\bigwedge_{i\in[1,N]} .\ \mathsf{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i) \\ & \qquad \Rightarrow V_i = \mathsf{RT}[i][\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)]) \\ & \quad \Rightarrow \mathsf{hold}(g_{\mathsf{WT}}(wt)\langle l\rangle, V_1, \ldots, V_N) \\ \mathsf{ve}(i, \mathit{\Lambda}, l) \triangleq & \textstyle\sum_{l'\in[1,l]} [\mathsf{vbl}(\mathit{\Lambda}\langle l'\rangle, i)]\end{aligned}$$

Now it remains to prove (70), (73), (74) and (72). The proof of (74) and (72) are sketched in Fig. 40, Fig. 41 and Fig. 42, where we take $\mathit{\Lambda} = g_{\mathsf{WT}}(wt)$. For (70) and (73), we apply Thm. 7, and use inference rules of the resampling-table-based program logic (listed in Fig. 25 and Fig. 26) to complete the proof. Proofs of these two judgments are presented in Fig. 45 and Fig. 46, while the auxiliary assertions used by these proofs are again the ones defined in Fig. 36, and we omit the common parts with Fig. 37, Fig. 38 and Fig. 39.

We then show the proofs of side conditions in Fig. 45 and Fig. 46.

1. The side condition in the last line of Fig. 45:

$$\vDash_{\mathrm{RT}} \mathsf{CL}(K+1) \Rightarrow$$
$$cnt = \sum_{wt \in LWTMap(K)} \left[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i) \right].$$

Proof: Let $\Sigma \vDash \mathsf{CL}(K+1)$, then there exists $m \in [0, K]$ such that:
 – $[\![cnt]\!]_\Sigma = |[\![lst]\!]_\Sigma| = m$;
 – For all $k \in [1, m]$, $[\![\mathsf{pf}(lst, k)]\!]_\Sigma \in ExLog$;
 – For all $k \in [1, m]$, $|[\![\mathsf{pf}(lst, k)]\!]_\Sigma| = k \leq K$.
Thus, by Lem. 124, $f_{\mathsf{LWT}}([\![\mathsf{pf}(lst, k)]\!]_\Sigma) \in LWTMap(K)$ for all $k \in [1, m]$. Now, let

$$wt_k = f_{\mathsf{LWT}}([\![\mathsf{pf}(lst, k)]\!]_\Sigma),$$

then we know that $wt_1 \neq \cdots \neq wt_m$ from Lem. 125, and for all $k \in [1, m]$ we have the following:
 – $\Sigma \vDash \mathsf{FLWT}(\mathsf{pf}(lst, k), wt_k, k)$;
 – For all $wt \neq wt_k$, $\Sigma \vDash \neg\mathsf{FLWT}(\mathsf{pf}(lst, k), wt, k)$.
Thus, one can verify that

$$\Sigma \vDash cnt = \sum_{wt \in LWTMap(K)} \left[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i) \right].$$

2. The side condition in the last line of Fig. 46:

$$\vDash_{\mathrm{RT}} \left(\bigvee_{k \in [0,K]} cnt = k \wedge \mathsf{len}(lst) = k \wedge \mathsf{L}(lst, k) \right) \Rightarrow$$
$$\left(\left(\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i) \right) \Rightarrow \mathbf{R} \right).$$

Proof: Define $\Sigma = (\sigma, RT, \iota)$ such that

$$\Sigma \vDash \bigvee_{k \in [0,K]} cnt = k \wedge \mathsf{len}(lst) = k \wedge \mathsf{L}(lst, k),$$

then there exists Λ_0 such that $[\![cnt]\!]_\Sigma = |\Lambda_0| \leq K$, $[\![lst]\!]_\Sigma = \Lambda_0$ and $RT \vDash \mathsf{L}(\Lambda_0, |\Lambda_0|)$, where $RT \vDash \mathsf{L}(\Lambda, |\Lambda|)$ holds iff
 – For all $k \in [1, |\Lambda|]$ and $r_1, \ldots, r_N$, if

$$r_i = RT[i][ve(i, \Lambda, k-1)]$$

 for all $i \in [1, N]$, then $\mathcal{E}[\Lambda\langle k \rangle](r_1, \ldots, r_N) = \text{true}$.

Then suppose

$$\Sigma \vDash \bigvee_{i\in[1,K]} i \le cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i).$$

Know that $f_{\mathsf{LWT}}(\llbracket \mathsf{pf}(lst, j) \rrbracket_{\Sigma}) = wt$ for some $j \in [1, |\Lambda_0|]$, and thus there exists some prefix of Λ_0, say Λ_1, such that $f_{\mathsf{LWT}}(\Lambda_1) = wt$ and $RT \vDash \mathsf{L}(\Lambda_1, |\Lambda_1|)$. Now we only need to prove that $\Sigma \vDash \mathbf{R}$:

(b) For all $l \in [1, |g_{\mathsf{WT}}(wt)|]$ and $r_1, \dots, r_N$, if for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$ we have $r_i = RT[i][ve(i, g_{\mathsf{WT}}(wt), l-1)]$, then

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \dots, r_N) = \text{true}.$$

Let

$$S = \{\Lambda : f_{\mathsf{LWT}}(\Lambda) = wt \wedge RT \vDash \mathsf{L}(\Lambda, |\Lambda|)\},$$

then from $\Lambda_1 \in S$ we have $S \neq \varnothing$. Now, define $w(\Lambda)$ as

$$\sum_{i=1}^{|g_{\mathsf{WT}}(wt)|} (|g_{\mathsf{WT}}(wt)| + 1 - i) \cdot LYWT(\lambda i.\, \Lambda\langle i\rangle)(GWTI'(\Lambda, |\Lambda|))\langle i\rangle,$$

then we take $\Lambda_2 = \mathrm{argmin}_{\Lambda\in S}\, w(\Lambda)$. Let

$$\Lambda_2' = LYWT(\lambda i.\, \Lambda_2\langle i\rangle)(GWTI'(\Lambda_2, |\Lambda_2|));$$

by Lem. 97 and induction,

$$\begin{aligned}|g_{\mathsf{WT}}(wt)| = |wt| &= |GWT'(\Lambda_2, |\Lambda_2|)| \\ &= |GWTI'(\Lambda_2, |\Lambda_2|)| = |\Lambda_2'|,\end{aligned}$$

and for all $l' \in |g_{\mathsf{WT}}(wt)|$

$$\Lambda_2\langle \Lambda_2'\langle l'\rangle\rangle = g_{\mathsf{WT}}(wt)\langle l'\rangle. \tag{75}$$

From Lem. 109, Lem. 97 and $f_{\mathsf{LWT}}(\Lambda_2) = wt$, we have $|g_{\mathsf{WT}}(wt)| \le |\Lambda_2|$. Below we prove that

$$\Lambda_2'\langle l\rangle = l \tag{76}$$

holds for all $l \in [1, |g_{\mathsf{WT}}(wt)|]$, from which we have

$$g_{\mathsf{WT}}(wt)\langle l\rangle = \Lambda_2\langle l\rangle$$

from (75) for all l and then from $RT \vDash \mathsf{L}(\Lambda_2, |\Lambda_2|)$ we have (2b). We prove (76) by contradiction. Assume that there exist l and j such that $\Lambda_2'\langle l\rangle = j+1$, and $\Lambda_2'\langle l'\rangle \neq j$ for all $l' < l$. From Lem. 115, we have the following three cases:

- $\#_j(\Lambda_2') = 0$, $l = |\Lambda_2'|$;
- $\#_j(\Lambda_2') = 0$, $l < |\Lambda_2'|$;
- $\#_j(\Lambda_2') = 1$, and there exists $l' > l$ such that $\Lambda_2'\langle l'\rangle = j$.

Since $\Lambda'_2\langle l\rangle = j+1$, from Lem. 117 there exists d such that $\mathsf{GDep}'(\Lambda_2, j+1, d)$. We then prove

$$\Lambda_2\langle j\rangle \notin \Gamma'^{+}(\Lambda_2\langle j+1\rangle) \tag{77}$$

by contradiction. Assuming $\Lambda_2\langle j\rangle \in \Gamma'^{+}(\Lambda_2\langle j+1\rangle)$, we have $\mathsf{GPath}'(\Lambda_2, j, d+1)$ from $\mathsf{GDep}'(\Lambda_2, j+1, d)$. If $\#_j(\Lambda'_2) = 0$, then $\neg\mathsf{GPath}'(\Lambda_2, j, d+1)$ follows from Lem. 116, a contradiction. If there exists $l' > l$ such that $\Lambda'_2\langle l'\rangle = j$, then from Lem. 117 there must exist some $d' \le d$ such that $\mathsf{GDep}'(\Lambda_2, j, d')$, which again contradicts $\mathsf{GPath}'(\Lambda_2, j, d+1)$. Thus (77) holds. Now it remains to show that

$$\Lambda_2 \neq \operatorname*{argmin}_{\Lambda\in S} w(\Lambda), \tag{78}$$

which leads to a direct contradiction and thus (76) follows. To see this, we then construct $\Lambda_3 \in S$ such that $w(\Lambda_3) < w(\Lambda_2)$. Below we let

$$\Lambda'_3 = LYWT(\lambda i.\, \Lambda_3\langle i\rangle)(GWTI'(\Lambda_3, |\Lambda_3|)).$$

- $\#_j(\Lambda'_2) = 0$, $l = |\Lambda'_2|$. Now $|\Lambda_2| = \Lambda'_2\langle l\rangle = j+1$. Let Λ and Λ_3 satisfy $\Lambda_2 = \Lambda_2\langle j+1\rangle :: \Lambda_2\langle j\rangle :: \Lambda$ and $\Lambda_3 = \Lambda_2\langle j+1\rangle :: \Lambda$. From Lem. 135 and $RT \vDash \mathsf{L}(\Lambda_2, |\Lambda_2|)$ we have $RT \vDash \mathsf{L}(\Lambda_2\langle j\rangle :: \Lambda_3, |\Lambda_3|+1)$, and thus $RT \vDash \mathsf{L}(\Lambda_3, |\Lambda_3|)$. Since $\Lambda_2\langle j\rangle \notin \Gamma'^{+}(\Lambda_2\langle j+1\rangle)$, by induction, for all $i \in [1, |\Lambda_3|)$ and l, $\mathsf{GPath}'(\Lambda_2, i, l)$ holds iff $\mathsf{GPath}'(\Lambda_3, i, l)$. Thus we can prove that
  - $f_{\mathsf{LWT}}(\Lambda_2) = f_{\mathsf{LWT}}(\Lambda_3)$;
  - $|\Lambda'_2| = |\Lambda'_3|$, $\Lambda'_2\langle|\Lambda'_2|\rangle = j+1$, and $\Lambda'_3\langle|\Lambda'_3|\rangle = j$;
  - $\Lambda'_2\langle i\rangle = \Lambda'_3\langle i\rangle$ for all $i \in [1, |\Lambda'_3|)$;

  then $\Lambda_3 \in S$, and $w(\Lambda_3) = w(\Lambda_2) - (j+1) + j < w(\Lambda_2)$.
- $\#_j(\Lambda'_2) = 0$, $l < |\Lambda'_2|$. Since $\Lambda'_2\langle l\rangle = j+1$, $\Lambda'_2\langle|\Lambda'_2|\rangle = |\Lambda_2|$ and $l < |\Lambda'_2|$, from Lem. 115 we have $j+1 < |\Lambda_2|$. We construct Λ_3 by swapping $\Lambda_2\langle j\rangle$ and $\Lambda_2\langle j+1\rangle$ in Λ_2. From $RT \vDash \mathsf{L}(\Lambda_2, |\Lambda_2|)$ and Lem. 135 we know that $RT \vDash \mathsf{L}(\Lambda_3, |\Lambda_3|)$. Moreover, from Lem. 122 and $j+1 < |\Lambda_2|$ we have $f_{\mathsf{LWT}}(\Lambda_3) = f_{\mathsf{LWT}}(\Lambda_2) = wt$. Thus $\Lambda_3 \in S$. Now from Lem. 123 and Lem. 115,

$$\begin{aligned} w(\Lambda_3) = w(\Lambda_2) &- (|g_{\mathsf{WT}}(wt)| + 1 - l)\cdot(j+1) \\ &+ (|g_{\mathsf{WT}}(wt)| + 1 - l)\cdot j \\ < w(\Lambda_2)&. \end{aligned}$$

- $\#_j(\Lambda'_2) = 1$, and there exists $l' > l$ such that $\Lambda'_2\langle l'\rangle = j$. Since $\Lambda'_2\langle l\rangle = j+1$, $\Lambda'_2\langle|\Lambda'_2|\rangle = |\Lambda_2|$ and $l < l' \le |\Lambda'_2|$, from Lem. 115 we have $j+1 < |\Lambda_2|$. We construct Λ_3 by swapping $\Lambda_2\langle j\rangle$ and $\Lambda_2\langle j+1\rangle$ in Λ_2. From $RT \vDash \mathsf{L}(\Lambda_2, |\Lambda_2|)$ and Lem. 135 we know that $RT \vDash \mathsf{L}(\Lambda_3, |\Lambda_3|)$. Moreover, from Lem. 122 and $j+1 < |\Lambda_2|$ we have $f_{\mathsf{LWT}}(\Lambda_3) = f_{\mathsf{LWT}}(\Lambda_2) = wt$. Thus $\Lambda_3 \in S$. Now from Lem. 123 and Lem. 115,

$$w(\Lambda_3) = w(\Lambda_2) - (|g_{\mathsf{WT}}(wt)| + 1 - l)\cdot(j+1)$$

$$\begin{array}{l}[\mathsf{true} \wedge \mathsf{hdinit}]\\ C'_{\mathsf{MT}}(cnt, K)\\ [\mathsf{CL}(K+1)]\\ \left[cnt = \sum_{wt \in LWTMap(K)} \left[\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i)\right]\right]\end{array}$$

Fig. 45. Proof of (70).

$$\begin{array}{l}\{\mathsf{true} \wedge \mathsf{hdinit}\}\\ C'_{\mathsf{MT}}(cnt, K)\\ \left\{\bigvee_{k \in [0,K]} cnt = k \wedge \mathsf{len}(lst) = k \wedge \mathsf{L}(lst, k)\right\}\\ \left\{\left(\bigvee_{i \in [1,K]} i \leq cnt \wedge \mathsf{FLWT}(\mathsf{pf}(lst, i), wt, i)\right) \Rightarrow \mathbf{R}\right\}\end{array}$$

Fig. 46. Proof of (73).

$$\begin{aligned} &- (|g_{\mathsf{WT}}(wt)| + 1 - l') \cdot j \\ &+ (|g_{\mathsf{WT}}(wt)| + 1 - l) \cdot j \\ &+ (|g_{\mathsf{WT}}(wt)| + 1 - l') \cdot (j+1) \\ = w(\Lambda_2) + l - l' < w(\Lambda_2). \end{aligned}$$

□

J.5 Theorem 2.2 of [32]

In the previous subsections, we focus on the termination property and the expected iteration numbers of the MT algorithm. Below we turn to the *MT-distribution* problem: how does the output (an assignment of the N variables) of the MT algorithm distribute? As we will see later, this problem is not only significant in itself, but also closely related to a useful variant of the MT algorithm which samples on only the "core events".

The first work that considers the MT-distribution is [32]. In the second part of Theorem 2.2 of [32], they upper bound the probability of an event other than $\mathcal{E}[1], \ldots, \mathcal{E}[M]$ occurring under the output of the MT algorithm. With this result in hand, they derive several important results, for example the first constant-factor approximation algorithm for the Santa Claus problem [4].

From another perspective, this result is closely related to a variant of the MT algorithm. That is, for events $\mathcal{E}[1], \ldots, \mathcal{E}[M]$, we apply the MT algorithm only on some of these events, for example $\mathcal{E}[1], \ldots, \mathcal{E}[M']$, where $M' < M$ is a constant. Here we call $\mathcal{E}[1], \ldots, \mathcal{E}[M']$ the "core events". Of course there might be some event belongs to $\mathcal{E}[M'+1], \ldots, \mathcal{E}[M]$ that does not hold on the output of the algorithm, but with the above-mentioned result we can prove that this happens with only a low probability, since for each event in $\mathcal{E}[M'+1], \ldots, \mathcal{E}[M]$ the probability of occurrence has an upper bound. Thus, by sacrificing some precision, the variant on core events obtains better performance than the original MT algorithm, since it runs on less events.

We formally verify the result we mentioned before, that is, the second part of Theorem 2.2 of [32]. The result is formally stated as Thm. 12. The code C_{HSS} is defined in Fig. 34, where we take the first $M-1$ events as the events to be sampled, and leave the event $\mathcal{E}[M]$ as the one that the occurrence probability is concerned. C_{HSS} differs from $C_{\mathsf{MT}}(cnt)$ only in the bound of the inner loop that chooses the event to be resampled. Moreover, C_{HSS} can be also regarded as a variant of the MT algorithm on core events, since $\mathcal{E}[M'+1], \ldots, \mathcal{E}[M]$ in the previous description can be combined into a single event by disjunctions.

Theorem 12. *For all reals* $\alpha_1, \ldots, \alpha_{M-1} \in (0,1)$, *if*

$$\forall i \in [1, M).\ \mathrm{P}(\mathcal{E}[i]) \le \alpha_i \prod_{j \in \Gamma(i) \setminus \{M\}} (1 - \alpha_j),$$

then

$$\models [\mathbf{true}]\, C_{\mathsf{HSS}} \begin{bmatrix} \Pr[\mathsf{hold}(M, x[1], \ldots, x[N])] \le \gamma_{\mathsf{HSS}} \\ \wedge\, \mathbb{E}[cnt] \le r_{\mathsf{HSS}} \end{bmatrix},$$

where

$$r_{\mathsf{HSS}} = \sum_{i \in [1,M)} \alpha_i (1 - \alpha_i)^{-1},$$

$$\gamma_{\mathsf{HSS}} = \mathrm{P}(\mathcal{E}[M]) \prod_{i \in \Gamma(M)} (1 - \alpha_i)^{-1}.$$

The proof of Thm. 12 relies on the intuition that, if $\mathcal{E}[M]$ holds after the algorithm terminates, then $\mathcal{E}[M]$ appears as if it is being chosen as an event to be resampled, and thus it can be inserted in the witness tree as we insert the events in the execution log.

Proof of Thm. 12. Let the premise (the Erdős-Lovász condition on $M-1$ events) hold. By applying Lem. 77, Thm. 2, Lem. 81, Lem. 82 and Lem. 83, with the auxiliary code $C'_{\mathsf{HSS}}(K)$ defined in Fig. 47, we only need to prove that, for all K,

$$\models [\mathbf{true}]\, C'_{\mathsf{HSS}}(K) \begin{bmatrix} \Pr[\mathsf{hold}(M, x[1], \ldots, x[N])] \le \gamma_{\mathsf{HSS}} \\ \wedge \mathbb{E}[cnt] \le r_{\mathsf{HSS}} \wedge \lceil cnt \ge 0 \rceil \end{bmatrix}. \tag{79}$$

From Lem. 103 and Lem. 104, we have

$$\models \sum_{\substack{wt \in WTMap(K) \\ \#_M(wt) = 0}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \le r_{\mathsf{HSS}}$$

and

$$\models \sum_{\substack{wt \in WTMap(K+1) \\ root(wt)=M \,\wedge\, \#_M(wt)=1}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \le \gamma_{\mathsf{HSS}};$$

thus, to prove (79), by Lem. 77 and Lem. 78, we only need to prove that

$$\models [\mathbf{true}]\, C'_{\mathsf{HSS}}(K) \left[\lceil cnt \ge 0 \rceil \wedge \mathbb{E}[cnt] \le \sum_{\substack{wt \in WTMap(K) \\ \#_M(wt)=0}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \right] \tag{80}$$

and

$$\models [\mathbf{true}]\, C'_{\mathsf{HSS}}(K) \left[\begin{array}{l} \Pr[\mathsf{hold}(M, x[1], \ldots, x[N])] \\ \le \displaystyle\sum_{\substack{wt \in WTMap(K+1) \\ root(wt)=M \,\wedge\, \#_M(wt)=1}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \end{array} \right]. \tag{81}$$

The proof of (80) is almost the same as (45). Below we focus on the proof of (81).

Similar to Thm. 4, we define $\mathsf{FWT}(e, wt, i)$ as follows:

$$\mathsf{FWT}(e, wt, i) \triangleq \bigvee\nolimits_{\Lambda \in f_{\mathsf{WT}}^{-1}(wt) \,:\, |\Lambda|=i} .\ e = \Lambda.$$

Now, to prove (81), by repeatedly applying Lem. 77 and Lem. 78, we only need to prove the following two subgoals:

$$\begin{array}{l} \models [\mathbf{true}]\, C'_{\mathsf{HSS}}(K) \\ \quad \left[\begin{array}{l} \Pr[\mathsf{hold}(M, x[1], \ldots, x[N])] \\ \le \Pr\left[\displaystyle\bigvee_{\substack{wt \in WTMap(K+1) \\ root(wt)=M \,\wedge\, \#_M(wt)=1}} \begin{array}{l} \mathsf{hold}(M, x[1], \ldots, x[N]) \\ \wedge\, \mathsf{FWT}(\mathsf{app}(lst, M), wt, cnt+1) \end{array} \right] \end{array} \right]. \end{array} \tag{82}$$

and for all $wt \in WTMap(K+1)$

$$\models \{\mathbf{true}\}\, C'_{\mathsf{HSS}}(K) \left\{ \begin{array}{l} \Pr\left[\begin{array}{l} \mathsf{hold}(M, x[1], \ldots, x[N]) \\ \wedge\, \mathsf{FWT}(\mathsf{app}(lst, M), wt, cnt+1) \end{array} \right] \\ \qquad \le \displaystyle\prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle]) \end{array} \right\}. \tag{83}$$

Informally, $\mathsf{FWT}(\mathsf{app}(lst, M), wt, cnt+1)$ holds if the witness tree wt can be constructed from the whole execution log (lst) with $\mathcal{E}[M]$ appended.

For (82), from Lem. 77 we only need to prove that

$$\models [\mathbf{true}]\, C'_{\mathsf{HSS}}(K)$$

$$\left[\left[\bigvee_{\substack{wt\in WTMap(K+1)\\ root(wt)=M \wedge \#_M(wt)=1}} \mathsf{FWT}(\mathsf{app}(lst,M),wt,cnt+1)\right]\right], \tag{84}$$

then by Lem. 80, to prove (84), we only need to prove the following:

$$\begin{aligned}&\models_{\mathrm{RT}} [\mathrm{true}\wedge \mathsf{hdinit}]\, C'_{\mathsf{HSS}}(K)\\ &\quad\left[\bigvee_{\substack{wt\in WTMap(K+1)\\ root(wt)=M \wedge \#_M(wt)=1}} \mathsf{FWT}(\mathsf{app}(lst,M),wt,cnt+1)\right].\end{aligned} \tag{85}$$

For (83), by Lem. 79, with the auxiliary code $C_{\mathsf{check}}(\Lambda)$ defined in Fig. 35, we only need to prove the following two subgoals:

$$\begin{aligned}&\models \{\mathbf{true}\}\, C'_{\mathsf{HSS}}(K), C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\\ &\quad\left\{\begin{matrix}\mathsf{hold}(M,x[1],\ldots,x[N])\\ \wedge\,\mathsf{FWT}(\mathsf{app}(lst,M),wt,cnt+1)\end{matrix}, succ=1\right\},\end{aligned} \tag{86}$$

and

$$\begin{aligned}&\models [\mathbf{true}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\\ &\quad\left[\Pr[succ=1]=\prod_{i=1}^{|g_{\mathsf{WT}}(wt)|}\mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i\rangle])\right].\end{aligned} \tag{87}$$

For (86), by RT-based coupling (Thm. 3), we only need to prove the following two subgoals:

$$\begin{aligned}&\models_{\mathrm{RT}} \{\mathrm{true}\wedge \mathsf{hdinit}\}\, C'_{\mathsf{HSS}}(K)\\ &\quad\left\{\begin{matrix}\mathsf{hold}(M,x[1],\ldots,x[N])\\ \wedge\,\mathsf{FWT}(\mathsf{app}(lst,M),wt,cnt+1)\end{matrix} \Rightarrow \mathbf{R}\right\},\end{aligned} \tag{88}$$

and

$$\models_{\mathrm{RT}} [\mathrm{true}\wedge \mathbf{R}\wedge \mathsf{hdinit}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\, [succ=1], \tag{89}$$

where $\mathbf{R}$ is the same as that in Thm. 4.

$$\begin{aligned}\mathbf{R} \triangleq\ & \textstyle\bigwedge_{l\in[1,|g_{\mathsf{WT}}(wt)|]} .\ \forall V_1,\ldots V_N.\\ &\quad (\textstyle\bigwedge_{i\in[1,N]} .\ \mathsf{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)\\ &\qquad\quad \Rightarrow V_i=\mathsf{RT}[i][\mathsf{ve}(i,g_{\mathsf{WT}}(wt),l-1)])\\ &\qquad \Rightarrow \mathsf{hold}(g_{\mathsf{WT}}(wt)\langle l\rangle, V_1,\ldots,V_N)\\ \mathsf{ve}(i,\Lambda,l) \triangleq\ & \textstyle\sum_{l'\in[1,l]}[\mathsf{vbl}(\Lambda\langle l'\rangle, i)]\end{aligned}$$

Given that $\mathbf{R}$ is defined, (88) and (89) says that, for all resampling tables RT:

– If, using RT, wt can be constructed from the execution log generated by the MT algorithm ($C'_{\mathsf{HSS}}(K)$) with $\mathcal{E}[M]$ appended, and $\mathcal{E}[M]$ holds immediately after the execution log being generated, then $\mathbf{R}$ holds on RT. Note that $\mathsf{hold}(M, x[1], \ldots, x[N])$ is a necessary precondition for $\mathbf{R}$ to hold, since M is the root of wt.
– If $\mathbf{R}$ holds on RT, then all tests in the check(wt) program ($C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))$) pass when the program is executed using RT.

Now it remains to prove (85), (88), (89) and (87). The proofs of (89) and (87) are exactly the same as those of (53) and (51). Below we prove (85) and (88) by applying Thm. 7. We use inference rules of the resampling-table-based program logic (listed in Fig. 25 and Fig. 26) to derive these two judgments, which are sketched in Fig. 48 and Fig. 49. We use the following auxiliary assertions (and those listed in Fig. 36):

$$\begin{aligned}\mathsf{LM2}(m) &\triangleq \textstyle\bigwedge_{l\in[1,m]} .\ 1 \le \mathit{lst}[l] < M \\ \mathsf{CL2}(n) &\triangleq 0 \le \mathit{cnt} < n \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge \mathsf{LM2}(\mathit{cnt})\end{aligned}$$

The proof of (88) is mostly similar to Fig. 38 and Fig. 39, and we thus omit the common part in Fig. 49.

Below we show the proofs of two non-trivial side conditions in Fig. 48 and Fig. 49.

1. The side condition in the last line of Fig. 48:

$$\models_{\mathrm{RT}} \mathsf{CL2}(K+1) \Rightarrow \bigvee_{\substack{wt\in WTMap(K+1)\\ root(wt)=M\ \wedge\ \#_M(wt)=1}} \mathsf{FWT}(\mathsf{app}(\mathit{lst}, M), wt, \mathit{cnt}+1).$$

Proof: Let $\Sigma \models \mathsf{CL2}(K+1)$, then there exists $m \in [0, K]$ such that $[\![\mathit{cnt}]\!]_\Sigma = |[\![\mathit{lst}]\!]_\Sigma| = m$, and for all $i \in [1, m]$ we have $([\![\mathit{lst}]\!]_\Sigma)\langle i\rangle \in [1, M)$. Let

$$wt = f_{\mathsf{WT}}([\![\mathsf{app}(\mathit{lst}, M)]\!]_\Sigma),$$

from Lem. 86 and Lem. 93 we have $root(wt) = M$ and $\#_M(wt) = 1$. Since $|[\![\mathsf{app}(\mathit{lst}, M)]\!]_\Sigma| = m+1 \le K+1$, from Lem. 98 we have $wt \in WTMap(K+1)$, and thus

$$\Sigma \models \mathsf{FWT}(\mathsf{app}(\mathit{lst}, M), wt, \mathit{cnt}+1).$$

2. The side condition in the last line of Fig. 49:

$$\models_{\mathrm{RT}} \begin{array}{c} \mathit{cnt} \le K \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge \mathsf{L}(\mathit{lst}, \mathit{cnt}) \wedge \mathsf{U}(\mathit{lst}, \mathit{cnt}) \ \Rightarrow \\ \left(\begin{array}{l} \mathsf{hold}(M, x[1], \ldots, x[N]) \\ \wedge\, \mathsf{FWT}(\mathsf{app}(\mathit{lst}, M), wt, \mathit{cnt}+1) \end{array} \Rightarrow \mathbf{R}\right). \end{array}$$

Proof: Define $\Sigma = (\sigma, RT, \iota)$ such that

$$\Sigma \models \mathit{cnt} \le K \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge \mathsf{L}(\mathit{lst}, \mathit{cnt}) \wedge \mathsf{U}(\mathit{lst}, \mathit{cnt}),$$

then there exists $m \in [0, K]$ and Λ such that $[\![\mathit{cnt}]\!]_\Sigma = m$, $[\![\mathit{lst}]\!]_\Sigma = \Lambda$, $|\Lambda| = m$, and

(a) For all $k \in [1, m]$, $\mathcal{E}[\Lambda\langle k\rangle](r_1, \ldots, r_N) = \text{true}$, where $r_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma]$ for each $i \in [1, N]$.
(b) For all $i \in [1, N]$, $[\![x[i]]\!]_\Sigma = RT[i][[\![\mathsf{ve}(i, \Lambda, m)]\!]_\Sigma]$.

Then suppose

$$\Sigma \vDash \mathsf{hold}(M, x[1], \ldots, x[N]) \wedge \mathsf{FWT}(\mathsf{app}(\mathit{lst}, M), \mathit{wt}, \mathit{cnt}+1).$$

From $\Sigma \vDash \mathsf{hold}(M, x[1], \ldots, x[N])$, (2a) and (2b), we have:

(c) For all $k \in [1, m+1]$, $\mathcal{E}[(M :: \Lambda)\langle k\rangle](r_1, \ldots, r_N) = \text{true}$, where $r_i = RT[i][[\![\mathsf{ve}(i, (M :: \Lambda), k-1)]\!]_\Sigma]$ for each $i \in [1, N]$.

Know that

$$f_{\mathsf{WT}}([\![\mathsf{app}(\mathit{lst}, M)]\!]_\Sigma) = f_{\mathsf{WT}}(M :: \Lambda) = \mathit{wt},$$

and we only need to prove that $\Sigma \vDash \mathbf{R}$:

(d) For all $l \in [1, |g_{\mathsf{WT}}(\mathit{wt})|]$ and $r_1, \ldots, r_N$, if for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle, i)$ we have $r_i = RT[i][[\![\mathsf{ve}(i, g_{\mathsf{WT}}(\mathit{wt}), l-1)]\!]_\Sigma]$, then

$$\mathcal{E}[g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle](r_1, \ldots, r_N) = \text{true}.$$

Let l and $r_1, \ldots, r_N$ satisfy the premise of (2d), then from Lem. 100, we have the following: there exists k such that $(M :: \Lambda)\langle k\rangle = g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle$, and for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle, i)$ we have

$$[\![\mathsf{ve}(i, M :: \Lambda, k-1)]\!]_\Sigma = [\![\mathsf{ve}(i, g_{\mathsf{WT}}(\mathit{wt}), l-1)]\!]_\Sigma.$$

Define $r'_1, \ldots, r'_N$ such that

$$r'_i = RT[i][[\![\mathsf{ve}(i, M :: \Lambda, k-1)]\!]_\Sigma]$$

for all $i \in [1, N]$, then from (2c) we know that

$$\mathcal{E}[(M :: \Lambda)\langle k\rangle](r'_1, \ldots, r'_N) = \text{true},$$

which implies

$$\mathcal{E}[g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle](r'_1, \ldots, r'_N) = \text{true}$$

by $(M :: \Lambda)\langle k\rangle = g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle$. We can prove that

$$\mathcal{E}[g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle](r'_1, \ldots, r'_N) = \mathcal{E}[g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle](r_1, \ldots, r_N),$$

since $(r_1, \ldots, r_N)$ and $(r'_1, \ldots, r'_N)$ agree on all positions i such that $\mathrm{vbl}(g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle, i)$. Thus

$$\mathcal{E}[g_{\mathsf{WT}}(\mathit{wt})\langle l\rangle](r_1, \ldots, r_N) = \text{true}.$$

□

```
$C'_{\mathsf{HSS}}(K) \triangleq$

$d := 1$;
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d)$;
    $x[d] := a$;
    $d := d + 1$;
$flag := 0$;
$cnt := 0$;
$lst := []$;
while $(flag = 0 \wedge cnt < K)$ do
    $z := 0$;
    $h := 1$;
    while $(h < M)$ do
        if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then
            $z := h$;
        $h := h + 1$;
    if $(z = 0)$ then $flag := 1$;
    else
        $cnt := cnt + 1$;
        $lst := \mathsf{app}(lst, z)$;
        $d := 1$;
        while $(d \leq N)$ do
            if $(\mathsf{vbl}(z, d))$ then
                $a := \mathsf{Sample}(d)$;
                $x[d] := a$;
            $d := d + 1$;
```

Fig. 47. Auxiliary code for Thm. 12.

J.6 Theorem 1.3 of [51]

Besides the MT algorithm in Theorem 1.2, Moser and Tardos also propose other algorithmic versions of Lovász Local Lemma in [51]. The parallel version (or rather, a "parallelizable version") of the MT algorithm, with the code C_{MTpar} shown in Fig. 34, is probably the most important one of them.

The idea of the algorithm is to "compress" the MT algorithm into a compact version, which is suited for parallelization. In each iteration, the loop in the algorithm first generates a maximal independent set (MIS) mis, which is an event set with the largest size such that all events in the set occur under the current assignment and do not depend on the variables that some others depend on. Then, all variables that the events in mis depend on are resampled in one round. This algorithm can be parallelized: one can generate the MIS by applying those parallel algorithms, e.g. the Luby's algorithm [47], and resample the variables parallelly, e.g. by replacing the sequential composition of $C_{\mathsf{par}}(1), \ldots, C_{\mathsf{par}}(M)$ with parallel composition.

Moser and Tardos give a tail bound for the iteration numbers of the algorithm's main loop. By allowing an ϵ-slack in the Erdős-Lovász condition, they

```
$[\mathsf{true} \wedge \mathsf{hdinit}]$
$d := 1;$
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d);$
    $x[d] := a;$
    $d := d + 1;$
$[\mathsf{true}]$
$\mathit{flag} := 0;\ \mathit{cnt} := 0;\ \mathit{lst} := [];$
$[\mathit{cnt} = 0 \wedge \mathit{lst} = []]$
$[\mathsf{CL2}(K+1)]$
while $(\mathit{flag} = 0 \wedge \mathit{cnt} < K)$ do
    $[\mathsf{CL2}(K) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    $z := 0;\ h := 1;$
    $[\mathsf{CL2}(K) \wedge 0 \leq z < M \wedge 1 \leq h \leq M \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    while $(h < M)$ do
        $[\mathsf{CL2}(K) \wedge 0 \leq z < M \wedge 1 \leq h < M \wedge \mathit{flag} = 0$
                $\wedge K - \mathit{cnt} - \mathit{flag} = X \wedge M - h = X']$
        if $(\mathsf{hold}(h, x[1], \ldots, x[N]))$ then $z := h;$
        $h := h + 1;$
        $[\mathsf{CL2}(K) \wedge 0 \leq z < M \wedge 1 \leq h \leq M \wedge \mathit{flag} = 0$
                $\wedge K - \mathit{cnt} - \mathit{flag} = X \wedge M - h + 1 \leq X']$
    $[\mathsf{CL2}(K) \wedge 0 \leq z < M \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    if $(z = 0)$ then
        $[\mathsf{CL2}(K) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $\mathit{flag} := 1;$
        $[\mathsf{CL2}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
    else
        $[\mathsf{CL2}(K) \wedge 1 \leq z < M \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $[0 \leq \mathit{cnt} < K \wedge \mathsf{LM2}(\mathit{cnt}) \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge 1 \leq z < M$
                $\wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $\mathit{cnt} := \mathit{cnt} + 1;\ \mathit{lst} := \mathsf{app}(\mathit{lst}, z);$
        $[1 \leq \mathit{cnt} \leq K \wedge \mathsf{LM2}(\mathit{cnt}) \wedge \mathsf{len}(\mathit{lst}) = \mathit{cnt} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $[\mathsf{CL2}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $d := 1;$
        while $(d \leq N)$ do ;
            if $(\mathsf{vbl}(z, d))$ then
                $a := \mathsf{Sample}(d);$
                $x[d] := a;$
            $d := d + 1;$
        $[\mathsf{CL2}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
    $[\mathsf{CL2}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
$[\mathsf{CL2}(K+1)]$
$\left[\bigvee_{\substack{wt \in WTMap(K+1) \\ root(wt) = M \wedge \#_M(wt) = 1}} \mathsf{FWT}(\mathsf{app}(\mathit{lst}, M), wt, \mathit{cnt} + 1)\right]$
```

Fig. 48. Proof of (85).

$\{\mathsf{true} \wedge \mathsf{hdinit}\}$
$C'_{\mathsf{HSS}}(K)$
$\{\mathsf{CLU}(K+1)\}$
$\{cnt \le K \wedge \mathsf{len}(lst) = cnt \wedge \mathsf{L}(lst, cnt) \wedge \mathsf{U}(lst, cnt)\}$
$\{\mathsf{hold}(M, x[1], \ldots, x[N]) \wedge \mathsf{FWT}(\mathsf{app}(lst, M), wt, cnt+1) \Rightarrow \mathbf{R}\}$

Fig. 49. Proof of (88).

prove that the probability of the iteration numbers exceeding n decreases exponentially as n grows.

We formally verify the above result. Below we first present the required definitions for generating MIS. We extend the definition of expressions:

$$(\mathit{Expr})\ e ::= \ldots \ \mid\ \mathsf{MIS}(e_1, \ldots, e_N)$$

$[\![\mathsf{MIS}(e_1, \ldots, e_N)]\!]_\sigma$ is defined as

$$\max \left\{ \begin{array}{l} \Lambda : (\forall j \in [1, |\Lambda|].\ \Lambda\langle j\rangle \in [1, M] \\ \qquad\qquad \wedge\, \mathcal{E}[\Lambda\langle j\rangle](r_1, \ldots, r_N) = \mathrm{true}) \\ \quad\ \wedge\ (\forall j < k \in [1, |\Lambda|].\ \Lambda\langle k\rangle \notin \Gamma^+(\Lambda\langle j\rangle)) \end{array} \right\},$$

where $[\![e_i]\!]_\sigma = r_i$ for each $i \in [1, N]$. In the above definition, $\max\{\Lambda : \cdots\}$ is one of the lists with the maximum length that represent independent sets of the dependency graph and contain only occurring events. Formally, $\max\{\Lambda : \cdots\}$ is the list which is greater than all other lists in the set, and for two lists Λ, Λ', $\Lambda < \Lambda'$ iff one of the following conditions holds:

– $|\Lambda| < |\Lambda'|$;
– $|\Lambda| = |\Lambda'|$, and there exists some Λ'' such that $\Lambda'' \prec \Lambda$, $\Lambda'' \prec \Lambda'$, and $\Lambda\langle|\Lambda''|+1\rangle < \Lambda'\langle|\Lambda''|+1\rangle$.

Moser and Tardos's result is then formally stated in Thm. 13.

Theorem 13. *For all reals $\alpha_1, \ldots, \alpha_M, \epsilon \in (0, 1)$, if*

$$\forall i \in [1, M].\ \mathrm{P}(\mathcal{E}[i]) \le (1-\epsilon)\alpha_i \left(\prod_{j \in \Gamma(i)} (1 - \alpha_j) \right),$$

then for all n we have

$$\models [\mathbf{true}]\, C_{\mathsf{MTpar}} \left[\Pr[cnt \ge n] \le (1-\epsilon)^n r_{\mathsf{EL}} \wedge \mathbb{E}[cnt] \le (\epsilon^{-1} - 1) r_{\mathsf{EL}} \right].$$

Proof. Let the premise (the Erdős-Lovász condition with ϵ slack) hold. By applying Lem. 77, Thm. 2, Lem. 81, Lem. 82 and Lem. 83, with the auxiliary code $C'_{\mathsf{MTpar}}(K)$ defined in Fig. 50, we only need to prove that, for all K and n,

$$\models [\mathbf{true}]\, C'_{\mathsf{MTpar}}(K) \left[\begin{array}{l} \Pr[cnt \ge n] \le (1-\epsilon)^n r_{\mathsf{EL}} \\ \wedge\, \mathbb{E}[cnt] \le (\epsilon^{-1} - 1) r_{\mathsf{EL}} \wedge \lceil cnt \ge 0 \rceil \end{array} \right]. \qquad (90)$$

From Lem. 102, we have

$$\models \sum_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \leq (1-\epsilon)^m r_{\mathsf{EL}}$$

for all m. Note that

$$\sum_{m \in [1,K]} (1-\epsilon)^m \leq \epsilon^{-1} - 1$$

and

$$\models \lceil 0 \leq \mathit{cnt} \leq K \rceil \Rightarrow \mathbb{E}[\mathit{cnt}] = \sum_{m \in [1,K]} \Pr[\mathit{cnt} \geq m];$$

thus, to prove (90), by applying Lem. 77 and Lem. 78, we only need to prove the following: for all $m \in [1,K] \cup \{n\}$,

$$\models [\mathbf{true}]\, C'_{\mathsf{MTpar}}(K) \left[\Pr[\mathit{cnt} \geq m] \leq \sum_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \right], \tag{91}$$

and

$$\models \{\mathbf{true}\}\, C'_{\mathsf{MTpar}}(K)\, \{\lceil 0 \leq \mathit{cnt} \leq K \rceil\}\,. \tag{92}$$

Similar to Thm. 4, we define $\mathsf{FWT}(e, wt, i)$ as follows:

$$\mathsf{FWT}(e, wt, i) \triangleq \bigvee\nolimits_{\Lambda \in f_{\mathsf{WT}}^{-1}(wt)\,:\,|\Lambda| = i} .\ e = \Lambda.$$

Now, to prove (91), by repeatedly applying Lem. 77 and Lem. 78, we only need to prove the following two subgoals:

$$\models [\mathbf{true}]\, C'_{\mathsf{MTpar}}(K) \left[\Pr[\mathit{cnt} \geq m] \leq \Pr\left[\bigvee_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \mathsf{FWT}(\mathit{lst}, wt, \mathsf{len}(\mathit{lst})) \right] \right], \tag{93}$$

and for all $wt \in WTMap(K \cdot M)$ such that $|wt| \geq m$

$$\models \{\mathbf{true}\}\, C'_{\mathsf{MTpar}}(K) \left\{ \Pr[\mathsf{FWT}(\mathit{lst}, wt, \mathsf{len}(\mathit{lst}))] \leq \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \right\}. \tag{94}$$

Informally, $\mathsf{FWT}(\mathit{lst}, wt, \mathsf{len}(\mathit{lst}))$ holds if the witness tree wt can be constructed from the execution log lst.

For (93), from Lem. 77 we only need to prove that

$$\models [\mathbf{true}]\, C'_{\mathsf{MTpar}}(K) \left[\left[cnt \geq m \Rightarrow \bigvee_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \mathsf{FWT}(lst, wt, \mathsf{len}(lst)) \right]\right]. \tag{95}$$

By Lem. 77, to prove (92) and (95), we only need to prove that

$$\models [\mathbf{true}]\, C'_{\mathsf{MTpar}}(K)\, [\lceil 0 \leq cnt \leq K \wedge \left(cnt \geq m \Rightarrow \bigvee_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \mathsf{FWT}(lst, wt, \mathsf{len}(lst)) \right) \rceil].$$

Then, by Lem. 80, to prove the above formula, we only need to prove the following:

$$\models_{\mathrm{RT}} [\mathrm{true} \wedge \mathsf{hdinit}]\, C'_{\mathsf{MTpar}}(K)\, [0 \leq cnt \leq K \wedge \left(cnt \geq m \Rightarrow \bigvee_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \mathsf{FWT}(lst, wt, \mathsf{len}(lst)) \right)]. \tag{96}$$

For (94), by Lem. 79, with the auxiliary code $C_{\mathsf{check}}(\Lambda)$ defined in Fig. 35, we only need to prove the following two subgoals:

$$\models \{\mathbf{true}\}\, C'_{\mathsf{MTpar}}(K), C_{\mathsf{check}}(g_{\mathsf{WT}}(wt)) \\ \{\mathsf{FWT}(lst, wt, \mathsf{len}(lst)), succ = 1\}, \tag{97}$$

and

$$\models [\mathbf{true}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt)) \left[\Pr[succ = 1] = \prod_{i=1}^{|g_{\mathsf{WT}}(wt)|} \mathrm{P}(\mathcal{E}[g_{\mathsf{WT}}(wt)\langle i \rangle]) \right]. \tag{98}$$

For (97), by RT-based coupling (Thm. 3), we only need to prove the following two subgoals:

$$\models_{\mathrm{RT}} \{\mathrm{true} \wedge \mathsf{hdinit}\}\, C'_{\mathsf{MTpar}}(K)\, \{\mathsf{FWT}(lst, wt, \mathsf{len}(lst)) \Rightarrow \mathbf{R}\}, \tag{99}$$

and

$$\models_{\mathrm{RT}} [\mathrm{true} \wedge \mathbf{R} \wedge \mathsf{hdinit}]\, C_{\mathsf{check}}(g_{\mathsf{WT}}(wt))\, [succ = 1], \tag{100}$$

where $\mathbf{R}$ is the same as that in Thm. 4.

$$\begin{aligned} \mathbf{R} \triangleq {} & \textstyle\bigwedge_{l \in [1, |g_{\mathsf{WT}}(wt)|]} .\ \forall V_1, \ldots V_N. \\ & (\textstyle\bigwedge_{i \in [1,N]} .\ \mathsf{vbl}(g_{\mathsf{WT}}(wt)\langle l \rangle, i) \\ & \quad \Rightarrow V_i = \mathsf{RT}[i][\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)]) \\ & \Rightarrow \mathsf{hold}(g_{\mathsf{WT}}(wt)\langle l \rangle, V_1, \ldots, V_N) \end{aligned}$$

$$\mathsf{ve}(i, \Lambda, l) \triangleq \textstyle\sum_{l' \in [1,l]} [\mathsf{vbl}(\Lambda\langle l' \rangle, i)]$$

Now it remains to prove (96), (99), (100) and (98). The proofs of (100) and (98) are exactly the same as the proofs of (53) and (51). Below we prove (96) and (99) by applying Thm. 7. We use inference rules of the resampling-table-based program logic (listed in Fig. 25 and Fig. 26) to derive the two corresponding judgments, and the proofs are sketched in Fig. 52, Fig. 53, Fig. 54 and Fig. 55. The auxiliary assertions are shown in Fig. 51 and Fig. 36. For simplicity, in Fig. 55, we use the following shorthands:

$$\begin{aligned}\mathsf{la} &\triangleq \mathsf{concat}(lst', \mathsf{pf}(mis, i-1))\\ \mathsf{lb} &\triangleq \mathsf{concat}(lst', \mathsf{pf}(mis, i))\\ \mathsf{na} &\triangleq \mathsf{len}(lst') + i - 1\\ \mathsf{nb} &\triangleq \mathsf{len}(lst') + i\end{aligned}$$

Below we show the proofs of two non-trivial side conditions in Fig. 52, Fig. 53, Fig. 54 and Fig. 55.

1. The side condition in Fig. 52:

$$\models_{\mathrm{RT}} \mathsf{CL3}(K+1) \Rightarrow cnt \leq K \wedge \left(cnt \geq m \implies \bigvee_{\substack{wt \in WTMap(K\cdot M)\\ |wt| \geq m}} \mathsf{FWT}(lst, wt, \mathsf{len}(lst)) \right).$$

Proof: Let $\Sigma \models \mathsf{CL3}(K+1)$, then by definition we have $\Sigma \models cnt \leq K$. Assuming that $\Sigma \models cnt \geq m$, it remains to prove that, if $m \leq K$, then there exists $wt \in WTMap(K \cdot M)$ such that $|wt| \geq m$ and $f_{\mathsf{WT}}([\![lst]\!]_\Sigma) = wt$. From $\Sigma \models \mathsf{CL3}(K+1)$, we have $\Sigma \models lst = \mathsf{pfl}(cnt)$, $\Sigma \models \mathsf{LM3}(cnt)$ and $\Sigma \models \mathsf{CPL}(cnt)$; thus, with $l = [\![cnt]\!]_\Sigma \geq m$ and $\Lambda = [\![lst]\!]_\Sigma$, we have:
(a) $\Lambda \in ExLog$, and $|\Lambda| \leq l \cdot M \leq K \cdot M$;
(b) There exist $1 \leq i_1 < \cdots < i_l \leq |\Lambda|$ such that $\Lambda\langle i_{j+1}\rangle \in \Gamma^+(\Lambda\langle i_j\rangle)$ for all $j \in [1, l)$.
From (1a) and Lem. 98, we have $f_{\mathsf{WT}}([\![lst]\!]_\Sigma) \in WTMap(K \cdot M)$. From (1b) we know that there exists $i \in |\Lambda|$ such that $\mathsf{GPath}(\Lambda, i, l)$, and thus by Lem. 95 and induction we have $|wt| \geq l \geq m$.

2. The side condition in Fig. 54:

$$\models_{\mathrm{RT}} \mathsf{L}(lst, \mathsf{len}(lst)) \Rightarrow (\mathsf{FWT}(lst, wt, \mathsf{len}(lst)) \Rightarrow \mathbf{R}).$$

Proof: Define $\Sigma = (\sigma, RT, \iota)$ such that $\Sigma \models \mathsf{L}(lst, \mathsf{len}(cnt))$, then with $[\![lst]\!]_\Sigma = \Lambda$ and $|\Lambda| = m$ we have
(a) For all $k \in [1, m]$, $\mathcal{E}[\Lambda\langle k\rangle](r_1, \ldots, r_N) = \text{true}$, where $r_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_\Sigma]$ for each $i \in [1, N]$.
Then suppose $\Sigma \models \mathsf{FWT}(lst, wt, \mathsf{len}(lst))$, which implies $f_{\mathsf{WT}}([\![lst]\!]_\Sigma) = f_{\mathsf{WT}}(\Lambda) = wt$. Now we only need to prove that $\Sigma \models \mathbf{R}$:
(b) For all $l \in [1, |g_{\mathsf{WT}}(wt)|]$ and $r_1, \ldots, r_N$, if for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$ we have $r_i = RT[i][[\![\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)]\!]_\Sigma]$, then

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \ldots, r_N) = \text{true}.$$

```
$C'_{\mathsf{MTpar}}(K) \triangleq$

$d := 1;$
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d);$
    $x[d] := a;$
    $d := d + 1;$
$flag := 0;$
$cnt := 0;$
$lst := [];$
while $(flag = 0 \wedge cnt < K)$ do
    $mis := \mathsf{MIS}(x[1], \ldots, x[N]);$
    if $(mis = [])$ then $flag := 1;$
    else
        $cnt := cnt + 1;$
        $lst := \mathsf{concat}(lst, mis);$
        $L[cnt] := mis;$
        $C_{\mathsf{par}}(1);$
        $\cdots;$
        $C_{\mathsf{par}}(M);$
```

Fig. 50. Auxiliary code for Thm. 13.

Let l and $r_1, \ldots, r_N$ satisfy the premise of (2b), then from Lem. 100, there exists k such that $\Lambda\langle k\rangle = g_{\mathsf{WT}}(wt)\langle l\rangle$, and for all $i \in [1, N]$ such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$ we have

$$[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_{\Sigma} = [\![\mathsf{ve}(i, g_{\mathsf{WT}}(wt), l-1)]\!]_{\Sigma}.$$

Define $r'_1, \ldots, r'_N$ such that

$$r'_i = RT[i][[\![\mathsf{ve}(i, \Lambda, k-1)]\!]_{\Sigma}]$$

for all $i \in [1, N]$, then from (2a) we know that

$$\mathcal{E}[\Lambda\langle k\rangle](r'_1, \ldots, r'_N) = \text{true},$$

which implies

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r'_1, \ldots, r'_N) = \text{true}$$

by $\Lambda\langle k\rangle = g_{\mathsf{WT}}(wt)\langle l\rangle$. Since $(r_1, \ldots, r_N)$ and $(r'_1, \ldots, r'_N)$ agree on all positions i such that $\mathrm{vbl}(g_{\mathsf{WT}}(wt)\langle l\rangle, i)$, we can prove that

$$\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r'_1, \ldots, r'_N) = \mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \ldots, r_N),$$

and thus $\mathcal{E}[g_{\mathsf{WT}}(wt)\langle l\rangle](r_1, \ldots, r_N) = \text{true}$.

□

$$
\begin{aligned}
\mathsf{SM}(\Lambda) &\triangleq \mathsf{len}(\Lambda) \leq M \wedge \left(\textstyle\bigwedge_{j\in[1,|\Lambda|]} .\ 1 \leq \Lambda\langle j\rangle \leq M\right)\\
\mathsf{LM3}(m) &\triangleq \textstyle\bigwedge_{l\in[1,m]} .\ \mathsf{SM}(L[l]) \wedge \mathsf{len}(L[l]) \geq 1\\
\mathsf{CP}(\Lambda,\Lambda') &\triangleq \textstyle\bigwedge_{j\in[1,M]} .\ \Lambda'\langle \_\rangle = j \Rightarrow \Lambda\langle \_\rangle \in \Gamma^{+}(j)\\
\mathsf{CPL}(m) &\triangleq \textstyle\bigwedge_{l\in[1,m)} .\ \mathsf{CP}(L[l], L[l+1])\\
\mathsf{pfl}(m) &\triangleq \begin{cases}\epsilon & \text{if } m = 0\\ \mathsf{concat}(\mathsf{pfl}(m-1), L[m]) & \text{if } m \geq 1\end{cases}\\
\mathsf{IND}(\Lambda) &\triangleq \textstyle\bigwedge_{j,k\in[1,M]} .\ 1 \leq j < k \leq \mathsf{len}(\Lambda) \Rightarrow \Lambda\langle j\rangle \notin \Gamma^{+}(\Lambda\langle k\rangle)\\
\mathsf{HD}(\Lambda) &\triangleq \textstyle\bigwedge_{j\in[1,M]} .\ \Lambda\langle \_\rangle = j \Rightarrow \mathsf{hold}(j, x[1], \ldots, x[N])\\
\mathsf{HD}'(\Lambda, n) &\triangleq \textstyle\bigwedge_{j\in[1,M]} .\ n < j \leq \mathsf{len}(\Lambda) \Rightarrow \mathsf{hold}(\Lambda\langle j\rangle, x[1], \ldots, x[N])\\
\mathsf{MI}(\Lambda) &\triangleq \textstyle\bigwedge_{j\in[1,M]} .\ \left(\textstyle\bigwedge_{k\in[1,M]} .\ k \in \Gamma^{+}(j) \Rightarrow \neg(\Lambda\langle \_\rangle = k)\right)\\
&\qquad\qquad \Rightarrow \neg\mathsf{hold}(j, x[1], \ldots, x[N])\\
\mathsf{CL3}(n) &\triangleq 0 \leq cnt < n \wedge lst = \mathsf{pfl}(cnt) \wedge \mathsf{LM3}(cnt)\\
&\qquad\qquad \wedge \mathsf{CPL}(cnt) \wedge (cnt = 0 \vee \mathsf{MI}(L[cnt]))\\
\mathsf{LU}(\Lambda) &\triangleq \mathsf{L}(\Lambda, \mathsf{len}(\Lambda)) \wedge \mathsf{U}(\Lambda, \mathsf{len}(\Lambda))\\
\mathsf{LUI}(i) &\triangleq \exists lst'.\ lst = \mathsf{concat}(lst', mis) \wedge \mathsf{LU}(\mathsf{lb}) \wedge \mathsf{SM}(mis)\\
&\qquad\qquad \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i)
\end{aligned}
$$

Fig. 51. Auxiliary assertions for Thm. 13.

```
$[\mathsf{true} \wedge \mathsf{hdinit}]$
$d := 1;$
while $(d \leq N)$ do
    $a := \mathsf{Sample}(d);$
    $x[d] := a;$
    $d := d + 1;$
$[\mathsf{true}]$
$\mathit{flag} := 0;\ \mathit{cnt} := 0;\ \mathit{lst} := [];$
$[\mathit{cnt} = 0 \wedge \mathit{lst} = []]$
$[\mathsf{CL3}(K+1)]$
while $(\mathit{flag} = 0 \wedge \mathit{cnt} < K)$ do
    $[\mathsf{CL3}(K) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    $\mathit{mis} := \mathsf{MIS}(x[1], \ldots, x[N]);$
    $[\mathsf{CL3}(K) \wedge \mathsf{SM}(\mathit{mis}) \wedge \mathsf{HD}(\mathit{mis}) \wedge \mathsf{MI}(\mathit{mis}) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
    if $(\mathit{mis} = [])$ then
        $[\mathsf{CL3}(K) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $\mathit{flag} := 1;$
        $[\mathsf{CL3}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
    else
        $[\mathsf{CL3}(K) \wedge \mathsf{SM}(\mathit{mis}) \wedge \mathsf{HD}(\mathit{mis}) \wedge \mathsf{MI}(\mathit{mis}) \wedge \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X]$
        $[0 \leq \mathit{cnt} < K \wedge \mathit{lst} = \mathsf{pfl}(\mathit{cnt}) \wedge \mathsf{LM3}(\mathit{cnt}) \wedge \mathsf{CPL}(\mathit{cnt})$
                    $\wedge\, \mathit{flag} = 0 \wedge K - \mathit{cnt} - \mathit{flag} = X$
                    $\wedge\, \mathsf{SM}(\mathit{mis}) \wedge \mathsf{MI}(\mathit{mis}) \wedge (\mathit{cnt} = 0 \vee \mathsf{CP}(L[\mathit{cnt}], \mathit{mis}))]$
        $\mathit{cnt} := \mathit{cnt} + 1;$
        $[1 \leq \mathit{cnt} \leq K \wedge \mathit{lst} = \mathsf{pfl}(\mathit{cnt} - 1) \wedge \mathsf{LM3}(\mathit{cnt} - 1) \wedge \mathsf{CPL}(\mathit{cnt} - 1)$
                    $\wedge\, K - \mathit{cnt} - \mathit{flag} + 1 \leq X$
                    $\wedge\, \mathsf{SM}(\mathit{mis}) \wedge \mathsf{MI}(\mathit{mis}) \wedge (\mathit{cnt} = 1 \vee \mathsf{CP}(L[\mathit{cnt} - 1], \mathit{mis}))]$
        $\mathit{lst} := \mathsf{concat}(\mathit{lst}, \mathit{mis});\ L[\mathit{cnt}] := \mathit{mis};$
        $[1 \leq \mathit{cnt} \leq K \wedge \mathit{lst} = \mathsf{pfl}(\mathit{cnt}) \wedge \mathsf{LM3}(\mathit{cnt}) \wedge \mathsf{CPL}(\mathit{cnt})$
                    $\wedge\, \mathsf{MI}(L[\mathit{cnt}]) \wedge L[\mathit{cnt}] = \mathit{mis} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $C_{\mathsf{par}}(1);$
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $\cdots$
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $C_{\mathsf{par}}(M);$
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
        $[\mathsf{CL3}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
    $[\mathsf{CL3}(K+1) \wedge K - \mathit{cnt} - \mathit{flag} + 1 \leq X]$
$[\mathsf{CL3}(K+1)]$
$\left[0 \leq \mathit{cnt} \leq K \wedge \left(\mathit{cnt} \geq m \implies \bigvee_{\substack{wt \in WTMap(K \cdot M) \\ |wt| \geq m}} \mathsf{FWT}(\mathit{lst}, wt, \mathsf{len}(\mathit{lst}))\right)\right]$
```

Fig. 52. Proof of (96) (part I).

```
$[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis}]$
if $(\mathsf{len}(\mathit{mis}) \geq i)$ then
    $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis})]$
    $d := 1;$
    $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis}) \wedge 1 \leq d \leq N+1]$
    while $(d \leq N)$ do
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis})$
                    $\wedge 1 \leq d \leq N \wedge N+1-d = X']$
        if $(\mathsf{vbl}(\mathit{mis}\langle i \rangle, d))$ then
            $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis})$
                        $\wedge 1 \leq d \leq N \wedge \mathsf{vbl}(\mathit{mis}\langle i \rangle, d) \wedge N+1-d = X']$
            $a := \mathsf{Sample}(d);\ x[d] := a;$
            $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis})$
                        $\wedge 1 \leq d \leq N \wedge N+1-d = X']$
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis})$
                    $\wedge 1 \leq d \leq N \wedge N+1-d = X']$
        $d := d+1;$
        $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis} \wedge 1 \leq i \leq \mathsf{len}(\mathit{mis})$
                    $\wedge 1 \leq d \leq N+1 \wedge N+1-d+1 \leq X']$
    $[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis}]$
$[\mathsf{CL3}(K+1) \wedge \mathit{cnt} \geq 1 \wedge L[\mathit{cnt}] = \mathit{mis}]$
```

Fig. 53. Proof of (96) (part II).

```
$\{\mathsf{true} \wedge \mathsf{hdinit}\}$
$d := 1;$
while $(d \leq N)$ do $(a := \mathsf{Sample}(d);\ x[d] := a;\ d := d + 1);$
$\left\{\bigwedge_{i \in [1,N]} .\ x[i] = \mathsf{RT}[i][0] \wedge \mathsf{hd}_i = 1\right\}$
$\mathit{flag} := 0;\ \mathit{cnt} := 0;\ \mathit{lst} := [];$
$\{\mathit{cnt} = 0 \wedge \mathit{lst} = [] \wedge \mathsf{U}([], 0)\}$
$\{\mathsf{LU}(\mathit{lst})\}$
while $(\mathit{flag} = 0 \wedge \mathit{cnt} < K)$ do
    $\{\mathsf{LU}(\mathit{lst})\}$
    $\mathit{mis} := \mathsf{MIS}(x[1], \ldots, x[N]);$
    $\{\mathsf{LU}(\mathit{lst}, \mathsf{len}(\mathit{lst})) \wedge \mathsf{IND}(\mathit{mis}) \wedge \mathsf{HD}(\mathit{mis})\}$
    if $(\mathit{mis} = [])$ then
        $\{\mathsf{LU}(\mathit{lst})\}$
        $\mathit{flag} := 1;$
        $\{\mathsf{LU}(\mathit{lst})\}$
    else
        $\{\mathsf{LU}(\mathit{lst}) \wedge \mathsf{IND}(\mathit{mis}) \wedge \mathsf{HD}(\mathit{mis})\}$
        $\mathit{cnt} := \mathit{cnt} + 1;$
        $\{\mathsf{LU}(\mathit{lst}) \wedge \mathsf{IND}(\mathit{mis}) \wedge \mathsf{HD}(\mathit{mis})\}$
        $\mathit{lst} := \mathsf{concat}(\mathit{lst}, \mathit{mis});\ L[\mathit{cnt}] := \mathit{mis};$
        $\{\exists \mathit{lst}'.\ \mathit{lst} = \mathsf{concat}(\mathit{lst}', \mathit{mis}) \wedge \mathsf{LU}(\mathit{lst}') \wedge \mathsf{IND}(\mathit{mis}) \wedge \mathsf{HD}(\mathit{mis})\}$
        $\{\mathsf{LUI}(0)\}$
        $C_{\mathsf{par}}(1);$
        $\{\mathsf{LUI}(1)\}$
        $\ldots$
        $\{\mathsf{LUI}(M-1)\}$
        $C_{\mathsf{par}}(M);$
        $\{\mathsf{LUI}(M)\}$
        $\{\mathsf{LU}(\mathit{lst})\}$
    $\{\mathsf{LU}(\mathit{lst})\}$
$\{\mathsf{L}(\mathit{lst}, \mathsf{len}(\mathit{lst}))\}$
$\{\mathsf{FWT}(\mathit{lst}, \mathit{wt}, \mathsf{len}(\mathit{lst})) \Rightarrow \mathbf{R}\}$
```

Fig. 54. Proof of (99) (part I).

```
{$\mathsf{LUI}(i-1)$}
if $(\mathsf{len}(mis) \geq i)$ then
    {$\mathsf{LUI}(i-1) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
    {$\exists lst'.\ lst = \mathsf{concat}(lst', mis) \wedge \mathsf{L}(\mathsf{la}, \mathsf{na}) \wedge \mathsf{U}(\mathsf{la}, \mathsf{na}) \wedge \mathsf{hold}(mis\langle i\rangle, x[1], \ldots, x[N])$
                                  $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
    {$\exists lst'.\ lst = \mathsf{concat}(lst', mis) \wedge \mathsf{L}(\mathsf{lb}, \mathsf{nb}) \wedge \mathsf{U}(\mathsf{lb}, \mathsf{na})$
                                  $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
    $d := 1;$
    {$\exists lst'.\ lst = \mathsf{concat}(lst', mis) \wedge \mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d) \wedge 1 \leq d \leq N+1$
                                  $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
    {$\cdots \mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d) \wedge 1 \leq d \leq N+1$
                           $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis) \wedge \cdots$}
    while $(d \leq N)$ do
        {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d) \wedge 1 \leq d \leq N$
                       $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
        {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d+1) \wedge \mathsf{hd}_i = \mathsf{ve}(d, \mathsf{lb}, \mathsf{na}) + 1 \wedge 1 \leq d \leq N$
                           $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
        if $(\mathsf{vbl}(mis\langle i\rangle, d))$ then
            {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d+1) \wedge \mathsf{hd}_d = \mathsf{ve}(d, \mathsf{lb}, \mathsf{na}) + 1 \wedge \mathsf{vbl}(\mathsf{lb}[\mathsf{nb}], d)$
              $\wedge\, 1 \leq d \leq N \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
            {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d+1) \wedge \mathsf{hd}_d = \mathsf{ve}(d, \mathsf{lb}, \mathsf{nb})$
              $\wedge\, 1 \leq d \leq N \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
            $a := \mathsf{Sample}(d);$
            {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d+1)$
              $\wedge\, a = \mathsf{RT}[d][\mathsf{ve}(d, \mathsf{lb}, \mathsf{nb})] \wedge \mathsf{hd}_d = \mathsf{ve}(d, \mathsf{lb}, \mathsf{nb}) + 1$
              $\wedge\, 1 \leq d \leq N \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
            $x[d] := a;$
            {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d+1)$
              $\wedge\, x[d] = \mathsf{RT}[d][\mathsf{ve}(d, \mathsf{lb}, \mathsf{nb})] \wedge \mathsf{hd}_d = \mathsf{ve}(d, \mathsf{lb}, \mathsf{nb}) + 1$
              $\wedge\, 1 \leq d \leq N \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
            {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d+1, d+1) \wedge 1 \leq d \leq N$
              $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
        {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d+1, d+1) \wedge 1 \leq d \leq N$
          $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
        $d := d + 1;$
        {$\mathsf{U}'(\mathsf{lb}, \mathsf{na}, \mathsf{nb}, d, d) \wedge 1 \leq d \leq N+1$
          $\wedge\, \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis)$}
    {$\cdots \mathsf{U}(\mathsf{lb}, \mathsf{nb}) \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i) \wedge 1 \leq i \leq \mathsf{len}(mis) \wedge \cdots$}
    {$\exists lst'.\ lst = \mathsf{concat}(lst', mis) \wedge \mathsf{L}(\mathsf{lb}, \mathsf{nb}) \wedge \mathsf{U}(\mathsf{lb}, \mathsf{nb}) \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i)$}
    {$\exists lst'.\ lst = \mathsf{concat}(lst', mis) \wedge \mathsf{LU}(\mathsf{lb}) \wedge \mathsf{IND}(mis) \wedge \mathsf{HD}'(mis, i)$}
    {$\mathsf{LUI}(i)$}
{$\mathsf{LUI}(i)$}
```

Fig. 55. Proof of (99) (part II).

K More about [29]

The recent work [29] proposes a relational program logic for proving contextual refinement of two probabilistic programs. We argue that, while this logic can be applied to the formal proof of a subgoal that occurs in the original proofs of ALLLs, our proof using RT-based coupling would be less complicated, thanks to the immutability of resampling tables.

Below we first state the intermediate proof goal occurring in the original proof of the MT algorithm, which, as shown later, can be proved by applying the relational program logic in [29]. This goal is stated as contextual refinement between two programs. Then we briefly introduce the relational program logic in [29]. Finally, we outline several proof paths for the intermediate goal with the relational program logic applied, and compare them with our proof using RT-based coupling.

K.1 Proof Goal as Contextual Refinement

The relational program logic in [29] can be used to prove the following goal: the probability of $\mathbf{q}_1$ holding after the (K-truncated) MT algorithm does not exceed the probability of $\mathbf{q}_2$ holding after the wt-check program, where $\mathbf{q}_1$ and $\mathbf{q}_2$ represent the two post-conditions in (8). This is one of the intermediate proof goals occurring in the original proof of the MT algorithm (see (b)).

As shown later, we state this proof goal as contextual refinement between two probabilistic programs. A program e (written in an ML-like language) contextually refines a program e', denoted by

$$\vDash e \precsim e',$$

is defined as follows: for any context $\mathcal{C}$, the termination probability of $\mathcal{C}[e]$ is no more than the termination probability of $\mathcal{C}[e']$.

We state the proof goal as an instance of the contextual refinement defined above. We construct a program e_1 roughly in the form of

$$\mathsf{let}\ \_ = e_{\mathsf{MT}}\ \mathsf{in} \qquad \mathsf{if}\ \mathbf{q}_1\ \mathsf{then}\ e_1'\ \mathsf{else}\ \varOmega$$

and e_2 in the form of

$$\mathsf{let}\ \_ = e_{\mathsf{check}}\ \mathsf{in} \qquad \mathsf{if}\ \mathbf{q}_2\ \mathsf{then}\ e_2'\ \mathsf{else}\ \varOmega,$$

where e_{MT} and e_{check} are codes of the (truncated) MT algorithm and the wt-check program respectively, $\varOmega = (\mathsf{rec}\ f\ x = f\ x)()$ diverges, and e_1' and e_2' clear the program state (e.g. heaps) so that, if e_1 and e_2 both terminate, then they terminate at the same state and with the same return value 0.

Then, our proof goal can be stated as

$$\vDash e_1 \precsim e_2. \tag{101}$$

To see this, note that e_1 contextually refines e_2 implies that the probability of e_1 terminating, which is also the probability of $\mathbf{q}_1$ holding after e_{MT}, is no more than the probability of e_2 terminating, which is also the probability of $\mathbf{q}_2$ holding after e_{check}.

K.2 The Relational Program Logic in [29]

In [29], they propose a relational program logic to prove contextual refinement like (101). That is, to prove $\vDash e \precsim e'$, one only needs to derive

$$\vdash e \precsim e'$$

by applying rules provided by the logic. At the core of these rules is a set of coupling rules. Below we give a brief introduction of these coupling rules (with slight adaptation).

The common goal of these rules is to pair the sampling operations in the two programs, and let the results of each pair of operations be the same. This ensures that each execution path of the left program (e) corresponds to a path of the right program (e') with the same probability, and thus establishes the contextual refinement. But the way these rules achieve this common goal varies.

As an example, the following SYNC rule simply couples two samplings that are both evaluated next:

$$\frac{\forall n \leq D. \quad \vdash E[n] \precsim E'[n]}{\vdash E[\mathrm{rand}(D)] \precsim E'[\mathrm{rand}(D)]} \quad (\textsc{sync})$$

Here E and E' are evaluation contexts, and $\mathrm{rand}(D)$ uniformly samples from $\{0, 1, \ldots, D\}$. This rule synchronously pairs the sampling operations in the two evaluation contexts, and let their results be the same. Similar rules are proposed in [8, 6].

One may want to couple two samplings, where one is to be evaluated next, but the other may be evaluated much later. In this scenario, SYNC cannot be applied. To support this, the logic provide rules CP-L, CP-R, RD-L and RD-R, relying on the *presampling tapes* mechanism. Tapes are ghost variables which belong to one of the two programs. Each tape maintains a queue of sample values, and is empty before the program is executed. Informally, when we want to asynchronously couple two samplings, we can "cache" the sample value of the earlier sampling in the other program's tape, so that the later sampling can fetch that value from its own tape as the result, ensuring that the results of the two samplings coincide.

We introduce these rules below. If the sampling in the left program is to be evaluated next but the one in the right program is not, then by applying CP-R we can add the result (n) of the left sampling to the tape of the right program (with tag ι):

$$\frac{\iota \hookrightarrow_{\mathsf{s}} (D, \vec{n}) \qquad \forall n \leq D. \;\; \iota \hookrightarrow_{\mathsf{s}} (D, \vec{n} \cdot n) \twoheadrightarrow \vdash E[n] \precsim e_2}{\vdash E[\mathrm{rand}(D)] \precsim e_2} \quad (\textsc{cp-r})$$

The assertion "$\iota \hookrightarrow_{\mathsf{s}} (D, \vec{n})$" says, the right program has a tape with tag ι, with its content being $\vec{n}$, which are all drawn from $\{0, \ldots, D\}$. Later, when the right program executes to the sampling that should be coupled with the left one, by applying RD-R we read the sample value from the right tape, which was added by the left program before:

$$\frac{\iota \hookrightarrow_{\mathsf{s}} (D, n \cdot \vec{n}) \qquad \iota \hookrightarrow_{\mathsf{s}} (D, \vec{n}) -\!\!* \vdash e_1 \precsim E[n]}{\vdash e_1 \precsim E[\text{rand}(D)]} \quad (\text{RD-R})$$

This way we let the results of the two coupled samplings be the same, even when they are not evaluated at the same time. CP-L and RD-L are similar to CP-R and RD-R, which we omit here.

The logic also provides the following CP-LR rule:

$$\frac{\begin{array}{c}\iota \hookrightarrow (D, \vec{n}) \qquad \iota' \hookrightarrow_{\mathsf{s}} (D, \vec{n}') \\ \forall n \leq D.\ \ \iota \hookrightarrow (D, \vec{n} \cdot n) * \iota' \hookrightarrow_{\mathsf{s}} (D, \vec{n}' \cdot n) -\!\!* \vdash e_1 \precsim e_2\end{array}}{\vdash e_1 \precsim e_2} \quad (\text{CP-LR})$$

The assertion "$\iota \hookrightarrow (D, \vec{n})$" is similar to "$\iota' \hookrightarrow_{\mathsf{s}} (D, \vec{n}')$", but describes the left tape. By applying CP-LR, we add the sample value (n) to both the left tape (with tag ι) and the right tape (with tag ι'). Later, if this value is popped from the two tapes by applying RD-L and RD-R respectively, then the two corresponding samplings are coupled with the same result.

K.3 Several Proof Paths

From the soundness of the relational program logic, it remains to complete the proof of (101) by deriving

$$\vdash e_1 \precsim e_2. \tag{102}$$

This judgment can be derived in various ways, by applying different sets of coupling rules described in the previous subsection. We outline two proof paths of (102), compare them with our proof using RT-based coupling, and discuss other possible paths.

For simplicity, we assume that $\mathit{supp}(\mathcal{D}[i]) = \{0, \ldots, i\}$ for each $i \in [1, N]$, and the two programs use rand(i) to sample from $\mathcal{D}[i]$. We also assume that each program has N initially empty tapes, where tapes of the left program are tagged with $\iota_1, \ldots, \iota_N$, and tapes from the right program are tagged with $\iota'_1, \ldots, \iota'_N$. Tapes ι_i and ι'_i store sample values drawn from $\mathcal{D}[i]$. That is, $\iota_i \hookrightarrow (i, \epsilon)$ and $\iota'_i \hookrightarrow_{\mathsf{s}} (i, \epsilon)$ hold for all $i \in [1, N]$.

Below we sketch the first proof path, which can be divided into three stages.

Stage 1. For all $i \in [1, N]$, we repeatedly apply CP-LR for K times, where we take $\iota = \iota_i$, $\iota' = \iota'_i$ and $D = i$. That is, we generate NK numbers, and add them to the tapes of both programs. Now, the contents of the left tapes are identical of those of the right tapes. Tapes of either program are similar to our resampling table. The judgment to be derived is still $\vdash e_1 \precsim e_2$.

Stage 2. This stage includes two steps. In this stage, we reason about the left program (e_1), and leave the right program (e_2) unchanged.

First, by repeatedly applying one-sided rules, we reduce the judgment $\vdash e_1 \precsim e_2$ to

$$\vdash \mathsf{if}\ \mathbf{q}_1\ \mathsf{then}\ e_1'\ \mathsf{else}\ \mathit{\Omega} \preceq e_2. \tag{103}$$

In particular, when $\vdash e_1 \precsim e_2$ is reduced to a judgment

$$\vdash E[\mathrm{rand}(i)] \precsim e_2, \tag{104}$$

we apply the rule RD-L and reduce the judgment to

$$\vdash E[n] \precsim e_2, \tag{105}$$

where n is the number popped from the tape ι_i.

During this process, some invariants should be established to describe properties involving the state of the left program and the values popped from the left tapes. However, these values can only be found in the right tapes[6]. Thus we should establish invariants involving the state of the left program and the right tapes.

Also, to maintain these invariants, we need to additionally establish invariants capturing the correspondence between left and right tapes. The reason is that, to find the popped values, which are fetched from left tapes at samplings, on right tapes, we must track something like "each value on the remaining left tapes equals to some value on right tapes with certain index".

Second, we do case analysis on $\mathbf{q}$. For the case that $\mathbf{q}$ holds (on the state of the left program), we reduce (103) to

$$\vdash 0 \preceq e_2, \tag{106}$$

which will be derived in stage 3. For the case that $\mathbf{q}$ does not hold, we reduce (103) to

$$\vdash \mathit{\Omega} \preceq e_2, \tag{107}$$

which can be directly derived.

Stage 3. We derive (106). This stage also includes two steps. In this stage, we reason about the right program (e_2), and leave the left program (0) unchanged.

First, since $\mathbf{q}$ holds on the state of the left program, together with other invariants established before (e.g. "invariants involving the state of the left program and the right tapes" in stage 2), we obtain certain property of the right

[6] An alternative approach is to store these popped values in auxiliary variables, and establish invariants involving the state of the left program and these variables. However, we still need to write invariants capturing the correspondence between these variables and the right tapes, since we need to translate the properties of these variables to properties of the right tapes, which will be used in stage 3. Also, to maintain these invariants, we again need to capture the correspondence between the remaining left tapes and the right tapes, as explained below.

tapes. This property is similar to our "**R**"; however, it will be invalidated later, since the right tapes will be shortened, and thus an invariant that is more complex than our "**R**" is needed in the following reasoning.

By repeatedly applying one-sided rules, we reduce (106) to

$$\vdash 0 \precsim \mathsf{if}\ \mathbf{q}_2\ \mathsf{then}\ e_2'\ \mathsf{else}\ \Omega. \tag{108}$$

In particular, when (106) is reduced to a judgment

$$\vdash 0 \precsim E[\mathrm{rand}(i)], \tag{109}$$

we apply the rule RD-R and reduce the judgment to

$$\vdash 0 \precsim E[n], \tag{110}$$

where n is the number popped from the tape ι_i'.

During this process, we use auxiliary variables to store these popped numbers, and establish invariants involving the state of the right program, the right tapes, and these auxiliary variables. For example, these invariants should capture that, by concatenating the popped values in the auxiliary variables and the remaining contents of the right tapes, we obtain "full" tapes satisfying the aforementioned property (which is similar to our "**R**").

Second, from the invariants established before, we know that $\mathbf{q}_2$ holds (on the state of the right program). Hence we reduce (108) to

$$\vdash 0 \precsim 0, \tag{111}$$

which can be directly derived.

Compare with our proof. Compared with our proof using RT-based coupling, the above proof path is somewhat more complicated. In this proof path:

– One needs to use about NK number of auxiliary variables to store the values popped from the tapes (stage 3).
– One needs to write invariants, which is more complex than our "**R**", to describe the tapes (stage 3). These invariants involve the state of the right program, the dynamically changing right tapes, and those auxiliary variables.
– One needs to write invariants to describe the correspondence between the tapes used by the two programs (stage 2). These invariants involve the state of the left program, the dynamically changing left tapes, and the right tapes.

In contrast, our proof using RT-based coupling avoids these drawbacks. Since our RT's are immutable, it is not needed to use auxiliary variables to store popped values, and we can use **R** to describe the immutable RT throughout the entire process of reasoning about the right program. Meanwhile, there is no need to describe the correspondence between the tables used by the two programs, attributing to the fact that used sample values can be found in the immutable RT, which is shared by the two programs.

The second proof path. We then sketch the second proof path. The idea of this proof path is similar to the previous one. However, here we do not apply CP-LR at the beginning of the reasoning. We first reason about the left program. At the time a sampling operation is to be evaluated, instead of applying RD-L, we apply CP-R to add the result of that sampling to the corresponding right tape. We also establish invariants involving the state of the left program and the right tapes. Then we reason about the right program, following the same steps as in the third stage of the previous path.

This proof path seems simpler than the previous one, since when reasoning about the left program we ignore the left tapes, and there is thus no need to track the correspondence between the left and the right tapes. However, it faces the same drawbacks as the previous proof path when reasoning about the right program. Indeed, auxiliary variables are still needed to store the values popped from the right tapes, and complicated invariants should be established to describe properties involving these auxiliary variables and the dynamically changing right tapes.

Other proof paths. Other proof paths may also be available, but they would not be simpler than the above two paths.

For example, instead of first reasoning about the left program and then reasoning about the right program, one may want to reason about these two programs by applying the rules CP-R and RD-R *alternately* (so that the proof can be more "compositional", since we can divide the whole reasoning process into several pieces, and apply coupling rules in these pieces separately). However, in this proof path, one should write invariants involving the right tapes, those popped values from the right tapes, and states of *both* programs. These invariants can be much more complicated than the ones occurring in the previous two proof paths, since they should describe the interleaving executions of the (truncated) MT algorithm and the *wt*-check program, which can be extremely complex due to the disparity between these two programs.