# A Language for Quantifying Quantum Network Behavior

ANITA BUCKLEY, Università della Svizzera italiana, Switzerland
PAVEL CHUPRIKOV, Télécom Paris, Institut Polytechnique de Paris, France
RODRIGO OTONI, Università della Svizzera italiana, Switzerland
ROBERT SOULÉ, Yale University, USA
ROBERT RAND, University of Chicago, USA
PATRICK EUGSTER, Università della Svizzera italiana, Switzerland

Quantum networks have capabilities that are impossible to achieve using only classical information. They connect quantum capable nodes, with their fundamental unit of communication being the *Bell pair*, a pair of entangled quantum bits. Due to the nature of quantum phenomena, Bell pairs are fragile and difficult to transmit over long distances, thus requiring a network of repeaters along with dedicated hardware and software to ensure the desired results. The intrinsic challenges associated with quantum networks, such as competition over shared resources and high probabilities of failure, require quantitative reasoning about quantum network protocols. This paper develops PBKAT, an expressive language for specification, verification and optimization of quantum network protocols for Bell pair distribution. Our language is equipped with primitives for expressing probabilistic and possibilistic behaviors, and with semantics modeling protocol executions. We establish the properties of PBKAT's semantics, which we use for quantitative analysis of protocol behavior. We further implement a tool to automate PBKAT's usage, which we evaluated on real-world protocols drawn from the literature. Our results indicate that PBKAT is well suited for both expressing real-world quantum network protocols and reasoning about their quantitative properties.

CCS Concepts: • **Networks → Formal specifications**; • **Theory of computation → Semantics and reasoning**; • **Hardware → Quantum technologies**.

Additional Key Words and Phrases: entanglement distribution, probabilistic and possibilistic semantics

## 1 Introduction

Quantum networks are distributed systems that provide communication services to distributed quantum applications. They allow not only for the enhancement of existing applications' capabilities, but also the emergence of fundamentally new applications. Quantum networks bring many benefits over their classical counterparts, most notably their ability to increase communication security, e.g., by enabling unconditionally secure client-server communication, blind cloud computing, and secure multi-party computation [Gyongyosi and Imre 2022; Pirandola et al. 2020; Wang et al. 2023]. Furthermore, distribution is essential to expand quantum computation beyond the capabilities of individual quantum-enabled computers to quantum clusters [Kozlowski and Wehner 2019].

Quantum networks exploit non-classical phenomena governed by the laws of quantum mechanics, such as entanglement and superposition. The basic unit of quantum information is a quantum bit (*qubit*), a system that can be in one of two basis states, denoted $|0\rangle$ and $|1\rangle$, as well as in their linear combination, called *superposition*. The state of a composite (multipartite) quantum system can be *entangled*. Entanglement means that qubits are so tightly correlated that the subsystems cannot be described separately, and measuring (i.e., reading the state of) one effectively measures all the qubits. A pair of maximally entangled qubits, called a *Bell pair*, is the basic communication resource used by quantum network nodes (typically with each qubit of a Bell pair in a different node). Entanglement brings many benefits for communication, e.g., it prevents eavesdropping

Authors' Contact Information: Anita Buckley, anita.buckley@usi.ch, Università della Svizzera italiana, Switzerland; Pavel Chuprikov, pavel.chuprikov@telecom-paris.fr, Télécom Paris, Institut Polytechnique de Paris, France; Rodrigo Otoni, otonir@usi.ch, Università della Svizzera italiana, Switzerland; Robert Soulé, robert.soule@yale.edu, Yale University, USA; Robert Rand, rand@uchicago.edu, University of Chicago, USA; Patrick Eugster, eugstp@usi.ch, Università della Svizzera italiana, Switzerland.

or man-in-the-middle attacks [Pirandola et al. 2020]. However, there are also major obstacles to realizing long-distance quantum communication, including the *no-cloning* theorem [Nielsen and Chuang 2011] and *decoherence*. The no-cloning theorem in quantum mechanics states that it is impossible to create an identical copy of an unknown quantum state and decoherence means that the quantum state degrades quickly over time. These, along with noise and qubit loss, represent major obstacles to realizing long-distance quantum communication as done in classical store-and-forward networks. Improvements in hardware and error detection and correction mechanisms [Kenemer 2024] can help in the production of higher-quality quantum states, but entanglement *generation* and communication steps such as *distillation* – the creation of a single Bell state from two or more imperfect ones – have (intrinsically) high failure probabilities.

We consider quantum network *protocols* as distributed programs that govern end-to-end Bell pair distribution among remote nodes, in line with Illiano et al. [2022]; Kozlowski et al. [2023]; Pant et al. [2019]. The necessity for protocols to handle distributed coordination and failure-prone primitive operations, combined with scarcity of resources (e.g., memory and communication qubits) in quantum networks, lead to highly complex protocol behavior. This makes *formal reasoning* critical to enable protocol optimization, efficient compilation to hardware, and safe coexistence of multiple protocols over the same network, in addition to the verification of correctness properties and quantitative analysis of individual protocols. Importantly, any meaningful reasoning approach needs to handle both *probabilistic behaviors*, inherent to quantum communication primitives, and *nondeterminism*, arising from protocols running in parallel.

With quantum networks becoming a reality [Knaut et al. 2024; Liu et al. 2024; Stolk et al. 2024] and entering practice, as illustrated by companies like Cisco and Aliro[1], and by the recent development of an operating system for network applications [Donne et al. 2025], the need for formal specification and verification becomes more pressing. The ability to reason in possibilistic, i.e., (non)deterministic, terms about protocols (e.g., *can a given protocol create a Bell pair between two specified network nodes?*) is an important first step towards network verification, which was recently addressed by the BellKAT language [Buckley et al. 2024]. However, BellKAT does not capture the probabilistic nature of quantum mechanics, and thus can not provide quantitative accounts that are crucial in real-world scenarios. Practical protocols have to deal with the inherently probabilistic nature of certain network operations, where it becomes important to analyze quantitative properties (e.g., *what is the probability that a Bell pair between two end nodes is created?*). The probability of entanglement creation is the key metric relevant to practitioners, and simulators such as NetSquid [Coopmans et al. 2021] and SeQUeNCe [Wu et al. 2021] give estimates for it. Rigorous analysis of protocols thus requires a model that captures probabilistic behavior in real-world quantum networks.

We propose PBKAT, a language that enables quantitative analysis of real-world quantum network protocols. As the name suggests, PBKAT (Probabilistic BellKAT) is inspired by BellKAT and the extensive body of work applied to the practical verification of classical networks, particularly McNetKAT [Smolka et al. 2019b] (based on ProbNetKAT [Foster et al. 2016] and GKAT [Smolka et al. 2019a]), but it has distinct features that cater to the way quantum communication occurs in practice. Our language is designed to tackle round-based behavior in realistic quantum network protocols as currently envisioned by the Quantum Internet Research Group (QIRG)[2] of the Internet Research Task Force (IRTF), allowing for the encoding of a wide range of practical quantum network protocols. PBKAT addresses the combination of nondeterminism and probability, as both are innate to quantum networks, by adapting the work of Bonchi et al. [2021; 2022] to Bell pair distribution.

While PBKAT's semantics is theoretically rigorous, it also faithfully models real-world quantum network protocol executions. In particular, we designed the semantics to provide a formalism

[1] See https://research.cisco.com and https://www.aliroquantum.com.

[2] See https://irtf.org/qirg.

capable of analyzing quantitative properties of quantum network protocols, catering to practical applications. PBKAT's semantics is based on the composition of two protocol semantics, the abstract semantics representing a PBKAT expression (i.e., protocol specification) as a set of guarded strings of (uninterpreted) quantum network actions, and the probabilistic interpretation modeling protocol executions. The abstract semantics can be seen as a guarded variation of synchronous Kleene algebra (SKA) [Prisacariu 2010; Wagemaker et al. 2019], replacing the union and iteration constructs with guarded versions $e +_\beta f$ and $e^{(\beta)}$, respectively, conditioned on Boolean predicates $\beta$ over *network states* (i.e., the Bell pairs present in the network). The probabilistic interpretation uses guarded strings to generate probability distributions over protocol execution outcomes, which are multisets of Bell pairs. We establish the properties of PBKAT semantics which we prove correct, enabling us to perform quantitative analysis of real-world quantum network protocols.

We implemented a prototype tool capable of reasoning about quantum network protocols specified in PBKAT. With it, practitioners can specify and check quantitative properties of protocols, such as resilience to failure, as well as optimize protocols and manage network resources by predicting the occurrence and effects of race conditions. We evaluated our tool on 16 protocols inspired by the literature; our results confirm that PBKAT can express different types of repeater swap protocols and its semantics is powerful enough to efficiently reason about important properties.

In summary, our contributions are the following:

(1) We design PBKAT, a language with realistic semantics capable of expressing real-world quantum network protocols for Bell pair distribution, reflecting hardware constraints.
(2) We provide a semantics for PBKAT that enables verification of protocols in quantitative terms. Notably, our novel semantics allows for reasoning about the combination of probabilistic and nondeterministic behaviors inherent to quantum networks.
(3) We show that PBKAT semantics is sound and well defined.
(4) We implement a prototype tool to showcase the expressiveness and utility of PBKAT for modeling and quantitative analysis, which we evaluate on 16 repeater swap protocols.

The remainder of the paper is structured as follows. Section 2 introduces the necessary background and provides a literature review of quantum networks and approaches to network specification and verification. Section 3 presents an overview of our formalization. Section 4 formally describes all aspects of PBKAT and its properties. Section 5 demonstrates how PBKAT can be exploited for quantitative analysis, and Section 6 describes our tool, its usage, and the experimental results of our evaluation. Finally, Section 7 presents closing remarks and future work. For brevity, we include complete experimental results, detailed analysis and validation of case studies, and proofs in the Appendices A, B, C and D.

## 2 Background and Related Work

We first introduce the basic concepts surrounding quantum networks and describe the concrete network model proposed by the Quantum Internet Research Group (QIRG) of the Internet Research Task Force (IRTF). We then motivate the need for quantitative reasoning about network properties in practice, and discuss existing approaches for (probabilistic) network specification and verification.

*Bell pairs.* Bell pairs, named after Bell [1964], are the four entangled two-qubit quantum states: $\frac{1}{\sqrt{2}}\left|00\right\rangle \pm \frac{1}{\sqrt{2}}\left|11\right\rangle$ and $\frac{1}{\sqrt{2}}\left|01\right\rangle \pm \frac{1}{\sqrt{2}}\left|10\right\rangle$, where $\left|ij\right\rangle$ represents a pair of qubits in the states $\left|i\right\rangle$ and $\left|j\right\rangle$, and its coefficient squared represents the probability of being read. They are entangled in the sense that reading the value of the 1st qubit determines the value of the 2nd, e.g., in $\frac{1}{\sqrt{2}}\left|01\right\rangle + \frac{1}{\sqrt{2}}\left|10\right\rangle$ the state $\left|01\right\rangle$ denotes that if the 1st qubit is 0 then the 2nd is 1; coefficients $\frac{1}{\sqrt{2}}$ represent a $\frac{1}{2}$ chance of reading either $\left|01\right\rangle$ or $\left|10\right\rangle$. The four Bell states are equivalent, as it is possible to transform any Bell pair into another with single-qubit operations (e.g. a bit-flip) performed locally on only one of the qubits. Since Bell pairs are *maximally entangled* (i.e., they have the strongest non-classical correlations of all possible two-qubit states), they enable remote quantum operations (cf. Figure 2), making them particularly useful as a generic building block for distributed quantum applications.

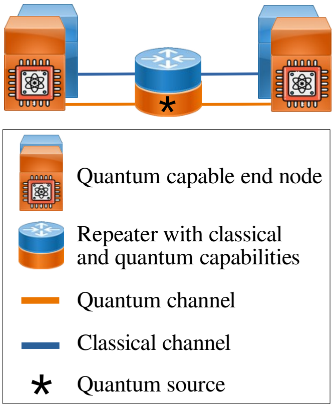
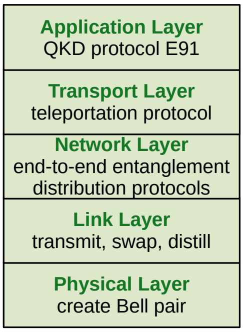


Fig. 1. Illustration of a quantum network for our running example.

*Quantum network hardware and functionality.* The laws of quantum mechanics grant new capabilities to quantum networks beyond their classical counterparts, while imposing constraints on their design. In this paragraph, we provide a high-level overview of quantum network architecture, which is illustrated in Figure 1. The core component are quantum capable **end nodes** that can receive and process entangled qubits and on which quantum applications are run. Each node uses a dedicated subset of qubits, called *communication qubits* [Kozlowski and Wehner 2019], to generate distributed entanglement (Bell pairs); and once a Bell pair is generated, the constituent qubits can be either immediately processed or transferred into memory. At the start of the entanglement distribution process, a **quantum source** creates Bell pairs locally. Once a Bell pair is created, one or both of its entangled qubits are *transmitted* over the **quantum channels**. The quality of entanglement during direct transmission decreases exponentially with the distance. Long distance transmission is thus achieved by using **quantum repeaters**, making them the key building blocks of quantum networks [Briegel et al. 1998; Towsley 2021]. A quantum repeater acts as an intermediary node between two other nodes (as illustrated in the network in Figure 1) by performing *entanglement swapping*. In this process, the repeater consumes the Bell pairs it shares with each of the other two nodes to create a new Bell pair connecting the nodes (directly). Another important physical process in quantum networks, called *entanglement distillation*, addresses decoherence (quantum state degradation over time), by generating a single Bell state from two or more imperfect ones. Distillation is inherently probabilistic, however: when it succeeds, the quality of the state is improved, and when it fails all the Bell pairs are destroyed. This substantially increases resource demands [Pompili et al. 2021]. Current quantum networks benefit from *heralded* schemes [Wehner et al. 2018] to distinguish between successful attempts and failures, meaning that a classical signal announces that a Bell pair is successfully generated. Quantum networks also depend on **classical channels** providing tight synchronization and timely signaling, as required by entanglement distribution schemes.



**Application Layer**
QKD protocol E91

**Transport Layer**
teleportation protocol

**Network Layer**
end-to-end entanglement distribution protocols

**Link Layer**
transmit, swap, distill

**Physical Layer**
create Bell pair

Fig. 2. Platform independent quantum network protocol stack [Abane et al. 2024; Illiano et al. 2022; Li et al. 2024] with protocol examples.

*Quantum network protocols.* Quantum networks, which enable distributed applications, depend on quantum network protocols to establish end-to-end Bell pair distribution [Briegel et al. 1998; Kozlowski et al. 2023]. In this work we focus on protocols in the network layer of the quantum network protocol stack, as illustrated in Figure 2. These protocols rely on quantum and classical networks working together, as outlined in [Kozlowski and Wehner 2019; Li et al. 2023; Rabbie et al. 2022].

Quantum network protocols orchestrate end-to-end Bell pair distribution through four key basic actions. Following the notation of BellKAT [Buckley et al. 2024], these basic actions can create a Bell pair locally at a source, transmit qubits of a Bell pair over a physical quantum channel, swap Bell pairs via repeaters, and distill Bell pairs to improve their quality. We give two running examples of protocols for the network in Figure 1 that use these basic actions to establish Bell pairs between the nodes $A$ and $B$, which we denote as $A \sim B$. Both protocols in Figure 3 have three rounds. Each round contains multiple actions executed concurrently and passes Bell pairs to the subsequent round. At the start, both protocols act in the same manner, *creating* two Bell pairs at the source $C$. Different transmission capabilities at the node $C$ necessitate different actions in subsequent rounds. Protocol (a) *transmits* half of each Bell pair created at $C$ to the neighbors $A$ and $B$ and keeps the other halves in $C$'s memory to obtain $A{\sim}C$ and $B{\sim}C$ respectively, then performs a *swap* at $C$, resulting in $A \sim B$. The swap protocol consumes two Bell pairs with a common endpoint to produce a single Bell pair with the opposite endpoints. Protocol (b) *transmits* both qubits of each Bell pair $C{\sim}C$ to the opposite neighbors, leading to two copies of $A{\sim}B$, and then *distills* them into a fresh $A{\sim}B$ of a better quality.

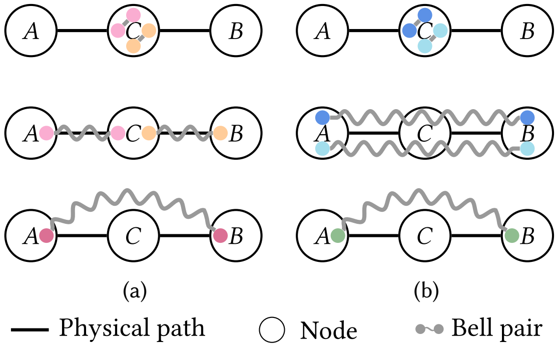


Fig. 3. Entanglement generating protocols (progressing top-to-bottom) on the 3-node network from Figure 1, establishing Bell pairs between nodes $A$ and $B$. Protocol (a) specifies *entanglement swapping* realized by Pompili et al. [2021] and (b) describes *distillation* as simulated in [Coopmans et al. 2021].

Contention between (sub)protocols running in parallel can lead to nondeterministic behavior, for instance due to resource underprovisioning or failures, as illustrated with the next example. Assume that, with some probability, the first round of protocol (a) only succeeds in creating one copy of $C{\sim}C$ instead of two. The missing $C{\sim}C$ copy will lead to resource contention (race condition): when the second round performs two transmits in parallel (one that requires $C \sim C$ to produce $A \sim C$ and the other that requires $C \sim C$ to produce $B \sim C$), only one of the two transmits will execute, i.e., either $A \sim C$ or $B \sim C$ will be produced *nondeterministically* (as shown in Figure 4). Such competition for available Bell pairs within a round between protocols running in parallel is the only source of nondeterminism in PBKAT. On the other hand, if the second round performs the two transmits in a prioritized manner on the same single input Bell pair $C{\sim}C$, e.g., if the priority is given to transmitting to node $A$ (the transmit that aims to produce $B{\sim}C$ is attempted after the transmit to $A{\sim}C$), only the transmit to $A{\sim}C$ is deterministically executed (shown on the left in Figure 4). In Section 4.1, we capture these two ways of combining actions within a round, namely, in parallel or in a prioritized manner, with PBKAT's parallel and ordered composition.

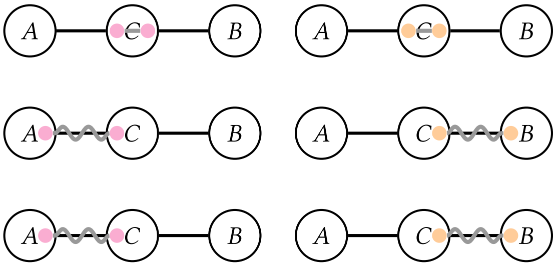


Fig. 4. Entanglement swap protocol (a) in Figure 3 when only one local Bell pair $C{\sim}C$ (instead of two) is successfully created in the first round. In the second round $C{\sim}C$ is nondeterministically transmitted to either $A{\sim}C$ (left) or $B{\sim}C$ (right). In the third round swap cannot execute as it lacks the required $A{\sim}C$ and $B{\sim}C$, so the Bell pair available after round two remains untouched.

Given the intricacies above, the following questions naturally arise:

- What is the probability that protocol (a) produces Bell pair $A{\sim}B$?
- What are minimum and maximum probabilities that protocol (b) produces Bell pair $A{\sim}B$?
- What is the expected number of iterations required for (b) to generate $A{\sim}B$?

- How can we improve the rate at which a protocol generates $A{\sim}B$?
- Is one protocol an optimized version of another protocol considering network constraints?

*Related work on algebraic approaches.* When designing language models, in order to benefit from strong mathematical foundations, it is natural to opt for a (co)algebraic approach. As discussed in the previous paragraph, reasoning about real-world quantum network protocols requires handling the combination of probabilistic behaviors and nondeterminism. To verify even simple properties in quantum networks, concurrency must be considered: to produce one end-to-end Bell pair, many entangled pairs must be created and distributed over intermediate nodes, multiple nodes simultaneously compete for same pairs, and network actions take several pairs as inputs. Thus, a suitable algebraic model must handle concurrent probabilistic and possibilistic behaviors synchronously.

In what follows, we overview the related lines of work that address some of these aspects (in specific domains). Algebraic reasoning about the packet forwarding behavior in classical networks originated with the seminal work on NetKAT [Anderson et al. 2014]. Probabilistic behavior in classical networks was considered in ProbNetKAT [Foster et al. 2016], yielding a scalable tool for verifying packet forwarding protocols with McNetKAT [Smolka et al. 2019b]. McNetKAT's scalability is attributable to its foundation in Guarded KAT (GKAT) [Smolka et al. 2019a], whose equational theory is (almost) linear time and is particularly applicable since it allows for a probabilistic interpretation. However, these works are not directly applicable to quantum networks, as the distribution of Bell pairs (network resources) has no counterpart in classical networks, which instead handle the forwarding of packets (information carriers). Furthermore, none of these works address the combination of probabilistic and nondeterministic behaviors. To handle such systems, Jacobs [2008] proposes the monad of nonempty convex sets of probability distributions, which combines the powerset monad (that models nondeterministic choice) and the probability distribution monad (that models probabilistic choice). Subsequently, Bonchi et al. [2021; 2022] and Mio et al. [2021] present these monads via algebraic theories for nondeterminism and probability. To the best of our knowledge, these theories have never been implemented. In the quantum context, Buckley et al. [2024] propose the BellKAT language to model concurrent behavior in quantum networks, inspired by synchronous Kleene algebra (SKA) [Prisacariu 2010], which allows for an alternative way of handling concurrency by layering synchronous actions into rounds. Conveniently, SKA was proven to have a sound and complete semantics for an analogue of Salomaa's axiomatization of Kleene algebra [Wagemaker et al. 2019]. However, BellKAT (like SKA) lacks probabilities and is thus incapable of modeling the innate randomness and uncertainty in quantum systems, making it unsuitable for quantitative analysis of protocols that arise in practice. And, due to known fundamental limitations on combining standard Kleene algebra techniques with probabilities (see related discussions in GKAT, ProbNetKAT and in the earlier work of Mislove [2006] and Varacca and Winskel [2006]), BellKAT semantics cannot be easily extended to accommodate probabilities.

*Related work on reasoning about quantum systems.* While quantum network testbeds are starting to emerge and move from the lab [Pompili et al. 2021] to metropolitan scale [Knaut et al. 2024; Liu et al. 2024; Stolk et al. 2024], supported by many publicly available quantum network simulators (e.g., NetSquid [Coopmans et al. 2021], SeQUeNCe [Wu et al. 2021], QuNetSim [DiAdamo et al. 2021], QuISP [Satoh et al. 2022], SimulaQron [Dahlberg and Wehner 2018]), and verification of classical network protocols has been intensively investigated, BellKAT is the only prior work we know that considers formal verification of quantum networks. It is worth noting that the properties of quantum network protocols which we strive to verify in this work are different from the properties at the application layer (cf. Figure 2), as considered in quantum program verification, where the actual quantum states are of prime importance. Our focus on Bell pair distribution abstracts away the details of quantum computation that would be necessary to verify quantum algorithms. Therefore, our approach departs from the broader landscape of formal quantum program verification, which features a variety of Hoare-style logics [Unruh 2019; Ying 2012; Zhou et al. 2019] and verification tools [Chareton et al. 2021; Hietala et al. 2021; Zhou et al. 2023]. Lewis et al. [2023] and Chareton et al. [2022] provide in-depth surveys of quantum program verification.

The next section introduces our network abstractions (based on the literature on quantum networks [Kozlowski et al. 2023; Pant et al. 2019; Van Meter and Touch 2013] and their specifications [Buckley et al. 2024]), made to capture typical quantum network behaviors.

## 3 PBKAT Overview

We introduce the key features of our PBKAT language and the main challenges in designing its probabilistic semantics for quantitative reasoning about real-world protocols. The main concepts will be explained through protocols from our *running examples* in Figures 3 and 4. We first specify the protocols in PBKAT and illustrate how specifications capture the possibility of actions failing. Next, we rewrite the protocols, showcasing how to control nondeterminism to better handle failures. Finally, we show how to analytically compute the expected outputs of protocols.

*Network behavior.* To tame parallelism, we divide entanglement generating protocols into *rounds* in the manner of Van Meter and Touch [2013]. Each round represents a time window containing synchronously executed *basic actions* (see Figure 3 above). Basic actions within one round can only act on the set of Bell pairs present in the network at the start of the said round, with race conditions emerging if resources are insufficient, i.e., if less Bell pairs are available than required by the actions (as exemplified in Figure 4). In order for a basic action to execute, it must thus first *acquire* a specific set of Bell pairs from those available in the corresponding round, and then use these Bell pairs to generate new entangled pairs. In realistic networks, it is likely that the action fails to generate new pairs, in which case the acquired Bell pairs are destroyed and no new Bell pairs are produced. If the required set cannot be acquired, the action is not executed and no Bell pairs are consumed. After execution, a heralding signal acknowledges the success or failure of each action. The next round then acts on the set of Bell pairs either produced or not consumed by the prior round. We introduce the notion of a *scheduler* that has a full view of the Bell pairs in the network. If at the start of a round there are not enough Bell pairs for all the actions, the scheduler chooses nondeterministically which actions are executed. Leaving nondeterminism unresolved thus covers all the choices (different strategies) of a scheduler.

*Network constraints.* A quantum network must keep track of the Bell pairs it contains. To that end, we assume unique node identifiers that are tracked across the network, along with the Bell pairs, via classical channels. Due to hardware constraints, a concrete quantum network is capable of handling only a limited number of Bell pairs in each round. In a network with nodes $A_1, \dots, A_l$ we denote the maximal number of Bell pairs possible between $A_i$ and $A_j$ by $m_{ij}$. For example, for protocol (b) in Figure 3 the network must be capable of handling at least two copies of $C{\sim}C$ locally at node $C$ and two copies of $A{\sim}B$ between nodes $A$ and $B$. Formally we define *network state* $a \in \mathcal{M}(\mathrm{BP})$ to be a multiset of Bell pairs present in the network; we represent it by $\{\!\{ n_{ij} \times A_i{\sim}A_j \}\!\}_{1 \le i \le j \le l}$, where $0 \le n_{ij} \le m_{ij}$ is the number of Bell pairs of type $A_i{\sim}A_j$, called the *multiplicity* of $A_i{\sim}A_j$ in $a$. (Here Bell pairs are labeled by the location of their qubits, so $2 \times A_i{\sim}A_j$ are indistinguishable, but could carry additional classical data like time stamp or memory location. This representation captures Bell pairs at the appropriate level of abstraction for quantum network protocols in PBKAT.)

*Actions. Basic actions* are the primitives for manipulating Bell pairs. A basic action has the form $r \triangleright (\Sigma\, p \cdot o)$, whose effect entails consuming a multiset of required Bell pairs $r$ and producing from it a multiset of (output) Bell pairs $o$ with probability $p$; the sum $\Sigma$ ranges over possible outputs as a *probability distribution* $\Sigma\, p \cdot o$. The probabilistic nature of quantum actions manifests itself in two ways, either operations are inherently probabilistic (e.g., distillation) or may fail with some probability due to decoherence, qubit loss, or other hardware limitations. We model such failures as $r \triangleright (p \cdot o + (1-p) \cdot \emptyset)$, or $r \triangleright_p o$ for short, where $p$ is the probability that the action succeeds. For example, a swap of $A{\sim}C$ and $B{\sim}C$ at node $C$ in Figure 3(a), which consumes $A{\sim}C$ and $B{\sim}C$ to output $A{\sim}B$ with probability $p$ (or outputs $\emptyset$ with probability $1-p$), denoted $\mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle_p$, is represented as $\{\!\{A{\sim}C, B{\sim}C\}\!\} \triangleright_p \{\!\{A{\sim}B\}\!\}$. A local creation at node $C$ succeeding with probability $p$ is denoted $\mathsf{cr}\langle C\rangle_p$ and represented as $\emptyset \triangleright_p \{\!\{C{\sim}C\}\!\}$. Similarly, $\mathsf{tr}\langle C \rightarrow A{\sim}B\rangle_p$ represents the action that physically transmits one qubit of the Bell pair $C{\sim}C$ to node $A$ and the other qubit to node $B$, but can drop all qubits with probability $1-p$. Distillation $\mathsf{di}\langle A{\sim}B\rangle_p$ requires two copies of $A{\sim}B$ to output a fresh $A{\sim}B$ and inherently fails with probability $1-p$ at least $\frac{1}{2}$ [Coopmans et al. 2021]. When it is clear from the context or not relevant, we omit writing the index $p$.

We provide shorthand notations for common actions, with deterministic ($p = 1$) wait and drop:

| | | | |
|---|---|---|---|
| swap | $\mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle_p \triangleq \{\!\{A{\sim}C, B{\sim}C\}\!\} \triangleright_p \{\!\{A{\sim}B\}\!\}$ | create | $\mathsf{cr}\langle C\rangle_p \triangleq \emptyset \triangleright_p \{\!\{C{\sim}C\}\!\}$ |
| transmit | $\mathsf{tr}\langle C \rightarrow A{\sim}B\rangle_p \triangleq \{\!\{C{\sim}C\}\!\} \triangleright_p \{\!\{A{\sim}B\}\!\}$ | wait | $\mathsf{wait}\langle r\rangle \triangleq r \triangleright r$ |
| distill | $\mathsf{di}\langle A{\sim}B\rangle_p \triangleq \{\!\{A{\sim}B, A{\sim}B\}\!\} \triangleright_p \{\!\{A{\sim}B\}\!\}$ | drop | $\mathsf{drop}\langle r\rangle \triangleq r \triangleright \emptyset$ |

With general basic actions users can specify other quantum operations, for instance, create a Bell pair between neighboring nodes directly, produce multiple Bell pairs, or output different Bell pairs with different probabilities. In practice, statistical probabilities for each quantum operation can be estimated from experiments and simulations as will be done in Section 5.3.

*Guarded protocols.* To specify the control flow of PBKAT protocols, we introduce *guards* that act as additional conditions at the start and end of each round, checking which Bell pairs are present in the network. A guard $\alpha$ is a predicate over a multiset of Bell pairs $\mathcal{M}(\mathrm{BP})$, thus it can be represented by the set of all multisets of Bell pairs for which the test succeeds. The primitives 1 (skip) and 0 (abort) behave like guards that always succeed and abort, respectively. We combine actions and guards into expressions to build protocols. Protocols $e$ and $f$ can be composed in the following ways: sequentially by combining subsequent rounds ($e \mathbin{;} f$), concurrently by combining actions within each round in parallel ($e \parallel f$) or in prioritized manner ($e \circ f$) (as in the motivating example on page 5), conditionally by using branching statements (**if** $\alpha$ **then** $e$ **else** $f$ which we denote as $e +_\alpha f$), and iteratively through while loops (**while** $\alpha$ **do** $e$ which we denote as $e^{(\alpha)}$). Importantly, single rounds have no iterative behavior, there is no interleaving between different rounds, and multiround iteration is guarded (i.e., we have while loops but no Kleene star).

As a first attempt, protocols (a) and (b) in Figure 3 can be expressed without using guards:

$$\text{Protocol (a)} \qquad (\mathsf{cr}\langle C\rangle \circ \mathsf{cr}\langle C\rangle) \mathbin{;} (\mathsf{tr}\langle C \rightarrow A{\sim}C\rangle \parallel \mathsf{tr}\langle C \rightarrow B{\sim}C\rangle) \mathbin{;} \mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle$$
$$\text{Protocol (b)} \qquad (\mathsf{cr}\langle C\rangle \circ \mathsf{cr}\langle C\rangle) \mathbin{;} (\mathsf{tr}\langle C \rightarrow A{\sim}B\rangle \circ \mathsf{tr}\langle C \rightarrow A{\sim}B\rangle) \mathbin{;} \mathsf{di}\langle A{\sim}B\rangle$$

Protocol (a) creates two Bell pairs $C{\sim}C$ locally at node $C$ in round one, transmits one copy to $A{\sim}C$ and the other to $B{\sim}C$ in round two, which are swapped at node $C$ to generate Bell pair $A{\sim}B$ in round three. Protocol (b) has the same first round, then transmits both Bell pairs $C{\sim}C$ to $A{\sim}B$ in round two, which are distilled in round three. We note that the swap action in protocol (a) attempts execution regardless of whether the required Bell pairs $A{\sim}C$ and $B{\sim}C$ have been successfully generated; if they are not available the protocol idly waits for one round (as illustrated for round three in Figure 4). Thus, it is sensible to condition the swap action on guard $\alpha$, which attempts its execution only if there are both $A{\sim}C$ and $B{\sim}C$ available, otherwise the present Bell pair is dropped:

$$\begin{aligned} \text{Protocol } (\mathrm{a}_1) \qquad & (\mathsf{cr}\langle C\rangle \circ \mathsf{cr}\langle C\rangle) \mathbin{;} (\mathsf{tr}\langle C \rightarrow A{\sim}C\rangle \parallel \mathsf{tr}\langle C \rightarrow B{\sim}C\rangle); \\ & (\textbf{if } \alpha \textbf{ then } \mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle \textbf{ else } (\mathsf{drop}\langle A{\sim}C\rangle \parallel \mathsf{drop}\langle B{\sim}C\rangle)) \end{aligned}$$

Similarly, the unguarded version of protocol (b) cannot distinguish between output $A{\sim}B$ that is produced by distillation in round three or by transmission in round two. So, we modify the protocol to repeatedly transmit qubits to nodes $A$ and $B$ until two Bell pairs $A{\sim}B$ are generated, as required for distillation. The guard $\beta$ below checks for the absence of two copies of $A{\sim}B$:

$$\text{Protocol } (\mathrm{b}_1) \qquad (\textbf{while } \beta \textbf{ do } (\mathsf{cr}\langle C\rangle \circ \mathsf{cr}\langle C\rangle) \,;\, (\mathsf{tr}\langle C \rightarrow A{\sim}B\rangle \circ \mathsf{tr}\langle C \rightarrow A{\sim}B\rangle)) \,;\, \mathsf{di}\langle A{\sim}B\rangle$$

For the inherently probabilistic distillation to succeed, multiple iterations of the entire protocol ($\mathrm{b}_1$) are likely to be needed, which can be expressed with another while loop: **while** $\gamma$ **do** protocol ($\mathrm{b}_1$), where the guard $\gamma$ checks for the absence of the (one distilled) Bell pair $A{\sim}B$. Such strategies of repeated attempts until success are universally employed in real-world quantum network protocols [Pompili et al. 2021; Van Meter et al. 2011].

*Probabilistic reasoning.* The innately probabilistic nature of quantum operations and thus quantum networks requires probabilistic reasoning. Moreover, the semantics have to be probabilistic too, as the end result of protocol execution is a *probability distribution* over the possible network states (resulting Bell pairs). Formally, the input to a given protocol is a multiset of Bell pairs, but the output is in $\mathcal{D}(\mathcal{M}(\mathsf{BP}))$ – the set of probability (sub)distributions over $\mathcal{M}(\mathsf{BP})$ representing possible execution outcomes. (In the next paragraph we will show that this set is convex.) Thus, a PBKAT protocol can be thought of as a function that takes a network state as input and returns, with a range of probabilities, Bell pairs that represent the network state after protocol execution.

To enable quantitative reasoning about protocols specified in PBKAT, we assign concrete probabilities of failures to basic actions. Consider our running example protocols ($\mathrm{a}_1$) and ($\mathrm{b}_1$):

$$\begin{aligned}
&\text{Protocol } (\mathrm{a}_1) \qquad && (\mathsf{cr}\langle C\rangle_{0.9} \circ \mathsf{cr}\langle C\rangle_{0.9}) \,;\, (\mathsf{tr}\langle C \rightarrow A{\sim}C\rangle_{0.8} \parallel \mathsf{tr}\langle C \rightarrow B{\sim}C\rangle_{0.7});\\
& && (\mathsf{sw}\langle A{\sim}B \,@\, C\rangle_{0.6} +_{\alpha} (\mathsf{drop}\langle A{\sim}C\rangle \parallel \mathsf{drop}\langle B{\sim}C\rangle))\\
&\text{Protocol } (\mathrm{b}_1) && ((\mathsf{cr}\langle C\rangle_{0.9} \circ \mathsf{cr}\langle C\rangle_{0.9}) \,;\, (\mathsf{tr}\langle C \rightarrow A{\sim}B\rangle_{0.3} \circ \mathsf{tr}\langle C \rightarrow A{\sim}B\rangle_{0.3}))^{(\beta)} \,;\, \mathsf{di}\langle A{\sim}B\rangle_{0.5}
\end{aligned}$$

Now, we can verify that (when executed) protocols behave as expected. For protocol ($\mathrm{a}_1$) to successfully generate the Bell pair $A{\sim}B$, all basic actions must succeed, which happens with probability $0.9 \times 0.9 \times 0.8 \times 0.7 \times 0.6 = 0.27216$. On the other hand, if any basic action fails, the unused Bell pairs are dropped (in round three). Thus, protocol ($\mathrm{a}_1$) outputs distribution $0.27216 \cdot \{\!\{A{\sim}B\}\!\} + 0.72784 \cdot \emptyset$. Similarly, protocol ($\mathrm{b}_1$) succeeds with probability 0.5, because the while loop iterates until both Bell pairs $A{\sim}B$ are available for distillation, with the expected number of iterations $\frac{1}{0.9^2 0.3^2} \approx 14$.

*Nondeterminism.* Nondeterminism in PBKAT models resource contention which is essential for quantum network protocols. Nondeterminism arises exclusively from parallel composition, where both parallel parts execute concurrently, and if there are not enough Bell pairs for both, there are different nondeterministic choices in how Bell pairs are allocated to each part.

For example, assuming the same success probabilities of basic actions as above, consider again possible execution outcomes of protocol (a): Figure 3 illustrates its execution when all basic actions succeed, and Figure 4 shows the two nondeterministic transmissions in which either $\mathsf{tr}\langle C \rightarrow B{\sim}C\rangle_{0.7}$ or $\mathsf{tr}\langle C \rightarrow A{\sim}C\rangle_{0.8}$ succeeds (after only one Bell pair $C{\sim}C$ is successfully created in the first round). Protocol (a) thus yields two different output distributions, which we compute by tracing all possible executions of basic actions and compounding their probabilities, as shown in Figure 5,

$$\begin{aligned}
\mu &= 0.27216 \cdot \{\!\{A{\sim}B\}\!\} + 0.1944 \cdot \{\!\{A{\sim}C\}\!\} + 0.2394 \cdot \{\!\{B{\sim}C\}\!\} + 0.29404 \cdot \emptyset\\
\mu' &= 0.27216 \cdot \{\!\{A{\sim}B\}\!\} + 0.3384 \cdot \{\!\{A{\sim}C\}\!\} + 0.1134 \cdot \{\!\{B{\sim}C\}\!\} + 0.27604 \cdot \emptyset
\end{aligned}$$

where $\mu$ and $\mu'$ are obtained from cyan and brown branches, respectively. All possible output distributions thus range between $\mu$ and $\mu'$ and are captured in the following convex set of distributions:

$$\{\, q \cdot \mu + (1-q) \cdot \mu' \mid q \in [0,1] \,\}$$

In comparison, any execution of the guarded protocol (a$_1$), also shown in Figure 5, results in the output distribution $0.27216 \cdot \{\!\{A{\sim}B\}\!\} + 0.72784 \cdot \emptyset$, irrespective of the nondeterministic choice.

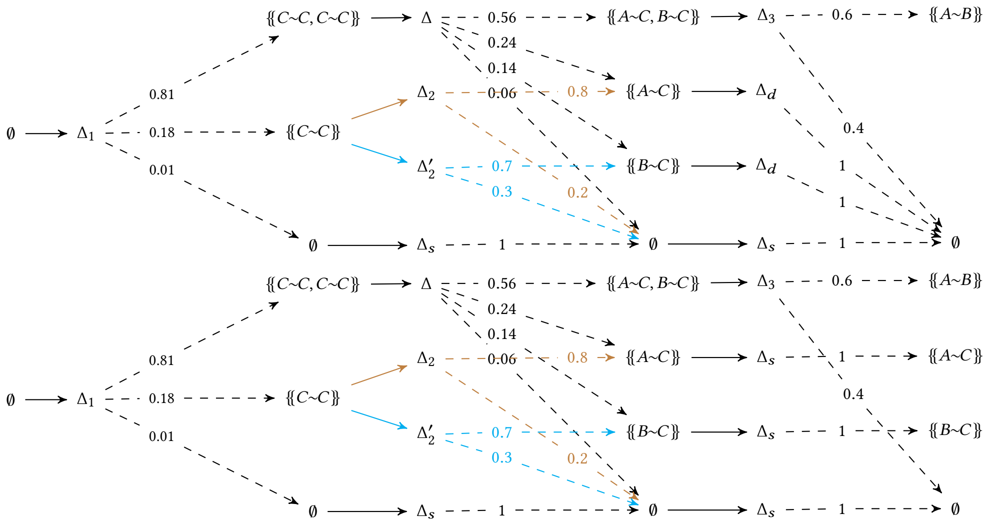


Fig. 5. The executions of Protocols (a$_1$) and (a), drawn in the top and bottom figures, showing all possible executions of actions through three rounds (progressing left to right). Dashed branches denote probabilistic choice, whereas full branches denote (non)deterministic choice. The nondeterministic choices of executing either $\mathrm{tr}\langle C \to B{\sim}C\rangle_{0.7}$ or $\mathrm{tr}\langle C \to A{\sim}C\rangle_{0.8}$ are shown as cyan or brown branches, respectively.

The above examples serve as the intuition to formally model nondeterminism in the following way. We capture all possible execution outcomes of a given protocol in a (convex) set of probability (sub)distributions over multisets of Bell pairs in $\mathcal{M}(\mathrm{BP})$, i.e., as an element in $C(\mathcal{M}(\mathrm{BP}))$ for the monad $C$ of convex sets of probability subdistributions over $\mathcal{M}(\mathrm{BP})$ (cf. definition in Section 4.2).

*Combining probabilities with nondeterminism.* To formally address the fundamental challenges of combining probabilities with nondeterminism we constrain our language model in three ways: (i) with conditionals and while loops we constrain nondeterminism to arise only from parallel composition, (ii) we confine this nondeterminism to single rounds by modeling synchronous concurrency (similar to SKA [Prisacariu 2010]), and (iii) we exclude the probabilistic choice operator. Our language design choices do not restrict quantum network protocol behavior: for (i), practical protocols are guarded and can thus be expressed with conditionals and while loops; for (ii), PBKAT rounds correspond to time synchronization points, thus the only nondeterminism arises from interactions within rounds; and for (iii), the control flow of practical protocols is not probabilistic. These design choices offer two main benefits. First, the PBKAT language is simple and yet expressive enough to faithfully model real-world protocols for end-to-end Bell pair distribution from literature (cf. Section 5.3) including all protocols specifiable in BellKAT. Second, PBKAT can handle probabilistic protocols in the presence of such constrained nondeterminism.

| | | | |
|---|---|---|---|
| Nodes | $\mathsf{N} ::=$ | $A, B, C, \ldots$ | |
| Bell pairs | $\mathsf{BP} \ni bp ::=$ | $\mathsf{N}{\sim}\mathsf{N}$ | |
| Multisets | $\mathcal{M}(\mathsf{BP}) \ni a, b, r, o ::=$ | $\{\!\{bp_1, \ldots, bp_k\}\!\}$ | |
| | $\mid$ | $a \uplus b$ | *multiset union* |
| Guards | $\mathsf{BExp} \ni \alpha, \beta ::=$ | $0$ | *false (abort)* |
| | $\mid$ | $1$ | *true (skip)* |
| | $\mid$ | $a$ | *atomic guard* |
| | $\mid$ | $\alpha \wedge \beta$ | *$\alpha$ and $\beta$* |
| | $\mid$ | $\alpha \vee \beta$ | *$\alpha$ or $\beta$* |
| | $\mid$ | $\bar{\alpha}$ | *not $\alpha$* |
| Protocol expressions | $\mathsf{Exp} \ni e, f ::=$ | $r \triangleright (\Sigma\, p{\cdot}o)$ | *basic action* |
| | $\mid$ | $\alpha$ | *guard* |
| | $\mid$ | $e \; ; \; f$ | *sequential composition* |
| | $\mid$ | $e \parallel f$ | *parallel composition* |
| | $\mid$ | $e \circ f$ | *ordered composition* |
| | $\mid$ | $e +_\alpha f$ | *if $\alpha$ then $e$ else $f$* |
| | $\mid$ | $e^{(\alpha)}$ | *while $\alpha$ do $e$* |

Fig. 6. PBKAT syntax. Basic action $r \triangleright (\Sigma\, p{\cdot}o)$ has two parameters, $r \in \mathcal{M}(\mathsf{BP})$ is a multiset containing the required Bell pairs and $\Sigma\, p{\cdot}o \in \mathcal{D}(\mathcal{M}(\mathsf{BP}))$ is a probability distribution over output multisets of Bell pairs.

## 4 PBKAT Language

This section presents syntax (Section 4.1), semantics (Section 4.2), and properties (Section 4.3).

### 4.1 Syntax

The complete PBKAT syntax is given in Figure 6. A Bell pair $bp$ is represented by an unordered pair of nodes. For multisets of Bell pairs $a, a' \in \mathcal{M}(\mathsf{BP})$ we write $a \uplus a'$ for additive multiset union and $a \backslash a'$ for multiset difference. We assume a finite number of network nodes and limited numbers of Bell pairs at each node, resulting in a finite number of network states. Therefore, all probability (sub)distributions over the states are finitely supported. Guards are predicates over multisets of Bell pairs. (We use "multiset of Bell pairs" and "network state" interchangeably.) Guard 0 acts as *abort*, and guard 1 represents the absence of guards (and actions), which we refer to as *skip*. Language users specify protocols as expressions composed of basic actions and guards that act as additional controls on which actions are executed, and are combined by operators for sequential, parallel, and ordered composition, as well as conditionals and while loops. The sequential composition ($e \,;\, f$) models sequential transitioning through rounds by first applying $e$ to the input multiset and then applying $f$ to each multiset produced by $e$. Parallel and ordered composition govern the execution of actions that occur synchronously within a single round. Here, parallel composition ($e \parallel f$) allows running $e$ and $f$ in parallel, thus allowing for resource competition (in a nondeterministic manner), while ordered composition ($e \circ f$) imposes that $e$ has preference over $f$ in accessing the available Bell pairs in each round. When a guard fails, it aborts without further progress, resulting in no output. Importantly, abort must be distinguished from a basic action that consumes the required Bell pairs and fails and thus outputs no Bell pairs. The *if-then-else* and *while* loops are guarded variants of nondeterministic choice and Kleene star (typically denoted as $+$ and $^\star$ in Kleene algebras, which are notably not part of PBKAT's syntax). By excluding $+$ and $^\star$ from PBKAT's syntax we constrain nondeterminism to the one arising only from parallel composition, making our language simpler. Yet, conditionals and while loops together with other

PBKAT constructs for sequential, parallel and ordered composition are expressive enough to encode typical quantum network protocols (where the control flow is completely deterministic), including all protocols considered in BellKAT. Importantly, all protocols for end-to-end Bell pair distribution can be specified as PBKAT expressions.

## 4.2 Semantics

$$\begin{tikzcd}
\mathrm{Exp} \arrow[rr, "I(-)"] \arrow[drr, "[\![-]\!]"'] & & \mathcal{P}(\mathrm{GSS}) \arrow[d, "[\![-]\!]_{\mathrm{ex}}"] \\
& & \mathcal{M}(\mathrm{BP}) \to C(\mathcal{M}(\mathrm{BP}))
\end{tikzcd}$$

Fig. 7. PBKAT's semantics.

The diagram in Figure 7 overviews the key components in PBKAT's semantics. The semantics $[\![-]\!]$ takes a protocol expression in Exp and returns a function from input $a \in \mathcal{M}(\mathrm{BP})$ to probability subdistributions over network states. It is defined in two steps. The motivation for the two-step construction of semantics is a way to have an abstract language model which permits concrete interpretations, as commonly done in the literature (cf. [Smolka et al. 2019a] and [Buckley et al. 2024]). Our language model captures the control flow (first step), and $[\![-]\!]_{\mathrm{ex}}$ gives the execution semantics on concrete input network states (second step). Concretely, the abstract semantics $I(-)$ first transforms an expression into a set of guarded strings in $\mathrm{GSS} = \mathcal{P}(\mathcal{M}(\mathrm{BP})) ; (\mathcal{P}(\Pi) ; \mathcal{P}(\mathcal{M}(\mathrm{BP})))^*$, which combine sets of network states and (uninterpreted) actions. Then, $[\![-]\!]_{\mathrm{ex}}$ converts each element in GSS to the sequential compositions of guards and actions, and gives them probabilistic interpretation. A formal definition of the monad $C$ for combining probabilities with nondeterminism is given below.

*A primer on combining nondeterminism with probability.* This paragraph introduces the background mathematics necessary to understand the semantics of PBKAT. The following definitions are inspired by the monad $C$ of convex sets of probability distributions [Jacobs 2008]. In particular, the presentation of $C$ by the equational theory of convex semilattices [Bonchi et al. 2021, 2022] is well suited for modeling and reasoning about systems combining nondeterminism and probability, as is the case of quantum networks.

A *subdistribution* over multiset of Bell pairs is a probability assignment $\mu : \mathcal{M}(\mathrm{BP}) \to [0, 1]$ summing up to at most 1, i.e., $\sum \mu(a) \le 1$, where $\mu(a) \ne 0$ for finitely many multisets $a \in \mathcal{M}(\mathrm{BP})$. In particular, the *Dirac distribution* (or *point mass*) on $a \in \mathcal{M}(\mathrm{BP})$ is defined to be:

$$\delta_a(b) = \{\ 1 \quad \text{if } a = b \qquad 0 \quad \text{otherwise}$$

Following the standard identification, we write $\mu$ as $\sum_{a \in \mathcal{M}(\mathrm{BP})} \mu(a) \cdot a$, by summing over nonzero $\mu(a)$ and identifying $\delta_a$ with $a \in \mathcal{M}(\mathrm{BP})$. The set of all subdistributions is thus defined as:

$$\mathcal{D}(\mathcal{M}(\mathrm{BP})) = \{\, \Sigma\, p \cdot a \mid p \in [0, 1],\ \Sigma\, p \le 1,\ a \in \mathcal{M}(\mathrm{BP}) \,\}$$

For a set $S \subseteq \mathcal{D}(\mathcal{M}(\mathrm{BP}))$, its *convex closure* $\mathrm{conv}(S)$ is the smallest convex set that contains $S$:

$$\mathrm{conv}(S) = \{\, \Sigma\, q \cdot \mu \mid q \in [0, 1],\ \Sigma q = 1,\ \mu \in S \,\}$$

We say that a convex set $S$ is (*finitely*) *generated* by its (finite) subset $T$ if $S = \mathrm{conv}(T)$. This leads to the definition of the set of finitely generated convex sets of subdistributions over $\mathcal{M}(\mathrm{BP})$:

$$C(\mathcal{M}(\mathrm{BP})) = \{\, S \subseteq \mathcal{D}(\mathcal{M}(\mathrm{BP})) \mid S \text{ finitely generated and } S = \mathrm{conv}(S) \,\}$$

From here on we use shorthand notation for monad compositions, e.g., the above sets are $\mathcal{M}\mathrm{BP}$, $\mathcal{D}\mathcal{M}\mathrm{BP}$ and $C\mathcal{M}\mathrm{BP}$. Monad composition is done through *flattening* (or *multiplication*) $m \colon CC \Rightarrow C$ that can be expressed in concrete terms as follows. Let $U = \{\Phi\} \subseteq CC\mathcal{M}\mathrm{BP}$ be a finitely generated convex set of subdistributions over $\mathcal{CM}\mathrm{BP}$ with generators $\Phi \in \mathcal{DCM}\mathrm{BP}$. For $\Phi\colon \mathcal{CM}\mathrm{BP} \to [0,1]$ represented as probability assignments with $\mathrm{supp}(\Phi) = \{\, C \mid \Phi(C) \neq 0 \,\}$, $m(U) \in \mathcal{CM}\mathrm{BP}$ is defined:

$$m(U) = \bigcup_{\Phi \in U} \{ \sum_{C \in \mathrm{supp}\,\Phi} \Phi(C) \cdot \mu \ \mid \ \mu \in C \}$$

More concretely, $m(U) = \cup_{\Phi \in U} \mathrm{WMS}(\Phi)$, where the *weighted Minkowski sum* of $\Phi = \sum_{i=1}^{n} p_i \cdot C_i$ is:

$$\mathrm{WMS}(\sum_{i=1}^{n} p_i \cdot C_i) = \{ \sum_{i=1}^{n} p_i \cdot \mu_i \ \mid \ \mu_i \in C_i \text{ for all } 1 \leq i \leq n \}$$

The following example illustrates how convex sets of subdistributions are fitting for quantitative reasoning about quantum network protocols, in particular to analyze protocol executions.

*Example 4.1.* We consider concrete distributions $\mu_0 = \delta_\emptyset$, $\mu_1 = \delta_{\{\!\{A\sim B\}\!\}}$, $\mu_2 = \delta_{\{\!\{A\sim C, B\sim C\}\!\}}$ and subdistributions $\mu' = 0.5 \cdot \mu_0 + 0.1 \cdot \mu_1$, $\mu'' = 0.5 \cdot \mu_0 + 0.2 \cdot \mu_1 + 0.2 \cdot \mu_2$ in $\mathcal{DM}\mathrm{BP}$ to illustrate the above definitions. Figure 8 represents the convex sets $C_1 = \mathrm{conv}(\{\mu_0, \mu'\})$, $C_2 = \mathrm{conv}(\{\mu_0, \mu_2, \mu''\})$ and their Minkowski sum $\mathrm{WMS}(0.7 \cdot C_1 + 0.3 \cdot C_2)$ (in orange, blue and green, respectively). If we interpret the subdistributions in the Minkowski sum as possible protocol outputs, then the probability of outputting $\emptyset$ is on the interval $[0.35, 1]$, and similarly $[0, 0.13]$ and $[0, 0.3]$ for the other two multisets of Bell pairs $\{\!\{A\sim B\}\!\}$ and $\{\!\{A\sim C, B\sim C\}\!\}$.

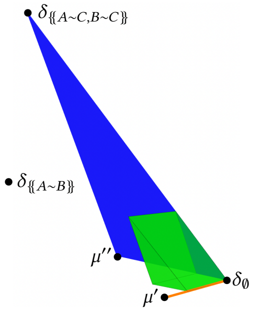


Fig. 8. Minkowski sum.

$$C_1 =$$
$$\left\{q_1 \cdot \mu_0 + (1-q_1) \cdot \mu' \mid q_1 \in [0,1]\right\} =$$
$$\left\{(0.5+0.5q_1) \cdot \emptyset + (0.1-0.1q_1) \cdot \{\!\{A\sim B\}\!\} \mid q_1 \in [0,1]\right\}$$

$$C_2 =$$
$$\left\{q_2 \cdot \mu_0 + q_2' \cdot \mu_2 + (1-q_2-q_2') \cdot \mu'' \mid q_2, q_2', 1-q_2-q_2' \in [0,1]\right\} =$$
$$\left\{ \begin{matrix} 0.5(1-q_2+q_2') \cdot \emptyset + \\ 0.2(1-q_2-q_2') \cdot \{\!\{A\sim B\}\!\} + \\ (0.2-0.2q_2+0.8q_2') \cdot \{\!\{A\sim C, B\sim C\}\!\} \end{matrix} \ \middle| \ q_2, q_2', 1-q_2-q_2' \in [0,1] \right\}$$

$$\mathrm{WMS}(0.7 \cdot C_1 + 0.3 \cdot C_2) =$$
$$\left\{ \begin{matrix} 0.7(q_1 \cdot \mu_0 + (1-q_1) \cdot \mu') + \\ 0.3(q_2 \cdot \mu_0 + q_2' \cdot \mu_2 + (1-q_2-q_2') \cdot \mu'') \end{matrix} \ \middle| \ \begin{matrix} q_1, q_2, q_2', \\ 1-q_2-q_2' \end{matrix} \in [0,1] \right\}$$

The convex set $\mathrm{WMS}(0.7 \cdot C_1 + 0.3 \cdot C_2)$ has six generators, from which we can now read the possible output multisets and with what probabilities they occur:

$$\begin{aligned} &\mathrm{conv}\,(\{1 \cdot \mu_0,\ 0.7 \cdot \mu_0 + 0.3 \cdot \mu_2,\ 0.7 \cdot \mu_0 + 0.3 \cdot \mu'',\ 0.7 \cdot \mu' + 0.3 \cdot \mu_0,\ 0.7 \cdot \mu' + 0.3 \cdot \mu_2,\ 0.7 \cdot \mu' + 0.3 \cdot \mu''\}) = \\ &\mathrm{conv}\,(\{1 \cdot \emptyset,\ 0.7 \cdot \emptyset + 0.3 \cdot \{\!\{A\sim C, B\sim C\}\!\},\ 0.85 \cdot \emptyset + 0.06 \cdot \{\!\{A\sim B\}\!\} + 0.06 \cdot \{\!\{A\sim C, B\sim C\}\!\}, \\ &\qquad 0.65 \cdot \emptyset + 0.07 \cdot \{\!\{A\sim B\}\!\},\ 0.35 \cdot \emptyset + 0.07 \cdot \{\!\{A\sim B\}\!\} + 0.3 \cdot \{\!\{A\sim C, B\sim C\}\!\}, \\ &\qquad 0.5 \cdot \emptyset + 0.13 \cdot \{\!\{A\sim B\}\!\} + 0.06 \cdot \{\!\{A\sim C, B\sim C\}\!\}\}) \end{aligned}$$

*Preliminaries.* When executed, basic actions behave as functions $\mathcal{M}\mathrm{BP} \to \mathcal{DM}\mathrm{BP}$ that take network states as inputs and return distributions:

$$r \triangleright (\Sigma\, p \cdot o) \ : \ a \mapsto \left\{ \ \Sigma\, p \cdot (o \uplus a \backslash r) \quad \text{if } r \subseteq a \qquad a \quad \text{otherwise} \right.$$

Observe that action behavior is conditioned by the test $r \subseteq a$ checking for the presence (or absence) of Bell pairs required by the action. When $r \subseteq a$ the action outputs $o \uplus a \backslash r$ with probability $p$, and when $r \nsubseteq a$ it passes on input $a$. Thus, we break basic actions into two mutually exclusive atomic components which we write as $[\mathbb{1}]r \blacktriangleright (\Sigma\, p \cdot o)$ and $[r]\emptyset \blacktriangleright \emptyset$, where tests $[\mathbb{1}]$ and $[r]$ are atomically tied to actions, and respectively represent *no test* and the *absence* of Bell pairs (i.e., on input $a$ check

**Atomic actions**

$$\begin{array}{rrcll}
\text{Tests} & T \ni t, t' & ::= & \mathbb{1} & \textit{no test} \\
 & & \mid & b & \textit{multiset absence} \\
 & & \mid & t \wedge t' & \textit{conjunction} \\
 & & \mid & t \vee t' & \textit{disjunction} \\
 & & \mid & t \uplus b & \textit{multiset union} \\
\text{Atomic actions} & \Pi \ni \pi & ::= & [t]r \blacktriangleright (\Sigma\, p{\cdot}o) &
\end{array}$$

**Test semantics**

$$\langle\!|t|\!\rangle \quad \in \quad \mathcal{M}(\mathrm{BP}) \longrightarrow \{\top, \bot\}$$

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\langle\!|\mathbb{1}|\!\rangle a & \triangleq & \top & \langle\!|t \uplus b|\!\rangle a & \triangleq & (\langle\!|t|\!\rangle a \setminus b \wedge b \subseteq a) \vee \langle\!|b|\!\rangle a \\
\langle\!|b|\!\rangle a & \triangleq & b \not\subseteq a & \langle\!|t \,\square\, t'|\!\rangle a & \triangleq & \langle\!|t|\!\rangle a \,\square\, \langle\!|t'|\!\rangle a, \text{ with } \square \text{ is either } \wedge \text{ or } \vee
\end{array}$$

**Composition rules**

Atomic actions $\pi = [t]r \blacktriangleright (\Sigma\, p{\cdot}o)$ and $\pi' = [t']r' \blacktriangleright (\Sigma\, p'{\cdot}o')$ are composed by the following rules:

$$\begin{array}{rll}
\pi \circ \pi' \equiv [t \wedge (t' \uplus r)]\hat{r} \blacktriangleright (\Sigma\, pp'{\cdot}\hat{o}) & \text{if } \hat{r} = r \uplus r' \text{ and } \hat{o} = o \uplus o' & \textsc{PNet-Ord} \\
\pi \parallel \pi' \equiv [(t \uplus r') \wedge (t' \uplus r)]\hat{r} \blacktriangleright (\Sigma\, pp'{\cdot}\hat{o}) & \text{if } \hat{r} = r \uplus r' \text{ and } \hat{o} = o \uplus o' & \textsc{PNet-Prl}
\end{array}$$

Fig. 9. Syntax and semantics of PBKAT tests used in the composition of atomic actions. Parallel and ordered composition of $\pi, \pi' \in \Pi$ yields new atomic actions $\pi \parallel \pi'$ and $\pi \circ \pi'$.

for $r \not\subseteq a$, the absence of Bell pairs $r$ in $a$). In general, we define atomic action $[t]r \blacktriangleright (\Sigma\, p{\cdot}o)$ as a map that on input $a$ outputs distribution $\Sigma\, p \cdot (o \uplus a \backslash r)$ if both, $r \subseteq a$ and test $t$ (checking absence of Bell pairs in $a$) succeed, otherwise it aborts. Figure 9 shows the syntax and semantics of tests, and the rules PNet-Ord and PNet-Prl for composing atomic actions. A simple calculation shows that for $\pi, \pi' \in \Pi$ the outputs of parallel and ordered composition $\pi \parallel \pi'$ and $\pi \circ \pi'$ are distributions, thus they are indeed atomic actions. In the special case when $p = 1$ we obtain (deterministic) atomic actions equivalent to those in BellKAT.

*Abstract language model.* A *guarded string* $a_0\pi_1 a_1 \cdots \pi_n a_n$ is an element of the regular set $\mathcal{M}\mathrm{BP}$ ; $(\Pi \,;\, \mathcal{M}\mathrm{BP})^*$. Intuitively, a non-empty string is a trace of a program, where $a_i \in \mathcal{M}\mathrm{BP}$ describe the network's state at the end of round $i$, starting with the initial state $a_0$, and $\pi_i \in \Pi$ represent the transitions triggered between the states. Guarded strings compose via *fusion product* $\diamond$ defined as,

$$a_0\pi_1 a_1 \ldots \pi_n a_n \diamond a'_0\pi'_1 a'_1 \ldots \pi'_{n'} a'_{n'} = \begin{cases} a_0\pi_1 a_1 \ldots \pi_n a_n \pi'_1 a'_1 \ldots \pi'_{n'} a'_{n'} & \text{if } a_n = a'_0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

and *layer-by-layer ordered composition* and *layer-by-layer parallel composition*, defined respectively by the rules (where we assume without loss of generality $n \leq n'$):

$$a_0\pi_1 a_1 \ldots \pi_n a_n \circ a'_0\pi'_1 a'_1 \ldots \pi'_{n'} a'_{n'} = \begin{cases} a_0(\pi_1 \circ \pi'_1)a_1 \ldots (\pi_n \circ \pi'_n)a_n \pi'_{n+1} a'_{n+1} \ldots \pi'_{n'} a'_{n'} & \substack{\text{if } a_i = a'_i \\ \forall\, 0 \leq i \leq n} \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$a_0\pi_1 a_1 \ldots \pi_n a_n \parallel a'_0\pi'_1 a'_1 \ldots \pi'_{n'} a'_{n'} = \begin{cases} a_0(\pi_1 \parallel \pi'_1)a_1 \ldots (\pi_n \parallel \pi'_n)a_n \pi'_{n+1} a'_{n+1} \ldots \pi'_{n'} a'_{n'} & \substack{\text{if } a_i = a'_i \\ \forall\, 0 \leq i \leq n} \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $\pi_i \circ \pi'_i$ and $\pi_i \parallel \pi'_i$ are atomic actions obtained by the rules PNet-Ord and PNet-Prl above.

In order to account for nondeterminism that may arise from combining protocols in parallel, we need to "enlarge" the strings to appropriately capture all nondeterministic choices; we achieve this by using the powerset monad $\mathcal{P}$. The advantage of dividing PBKAT protocols into rounds is that all nondeterminism is confined within single rounds.

A *guarded string of sets* is an element of the set $\mathcal{P}\mathcal{M}\mathsf{BP}\,;(\mathcal{P}\Pi\,;\mathcal{P}\mathcal{M}\mathsf{BP})^*$, denoted GSS. Elements $w = S_0\Omega_1S_1\cdots\Omega_nS_n \in \mathrm{GSS}$ and $w' = S'_0\Omega'_1S'_1\ldots\Omega'_{n'}S'_{n'}$ in GSS compose via *fusion product*,

$$w \diamond w' = \begin{cases} S_0\Omega_1S_1\ldots\Omega_n(S_n{\cap}S'_0)\Omega'_1S'_1\ldots\Omega'_{n'}S'_{n'} & \text{if } S_n{\cap}S'_0 \neq \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

and *layer-by-layer ordered composition* and *layer-by-layer parallel composition* (assuming $n \leq n'$),

$$w \circ w' = \begin{cases} (S_0{\cap}S'_0)(\Omega_1\circ\Omega'_1)(S_1{\cap}S'_1)\ldots(\Omega_n\circ\Omega'_n)(S_n{\cap}S'_n)\Omega'_{n+1}S'_{n+1}\ldots\Omega'_{n'}S'_{n'} & \text{if } S_i{\cap}S'_i{\neq}\emptyset \ \forall\, 0{\leq}i{\leq}n \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$w \parallel w' = \begin{cases} (S_0{\cap}S'_0)(\Omega_1\parallel\Omega'_1)(S_1{\cap}S'_1)\ldots(\Omega_n\parallel\Omega'_n)(S_n{\cap}S'_n)\Omega'_{n+1}S'_{n+1}\ldots\Omega'_{n'}S'_{n'} & \text{if } S_i{\cap}S'_i{\neq}\emptyset \ \forall\, 0{\leq}i{\leq}n \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $\Omega_i \circ \Omega'_i = \{\, \pi\circ\pi' \mid \pi \in \Omega,\ \pi' \in \Omega' \,\}$ and $\Omega_i \parallel \Omega'_i = \{\, \pi\parallel\pi' \mid \pi \in \Omega,\ \pi' \in \Omega' \,\}$ in which atomic actions $\pi\circ\pi'$ and $\pi\parallel\pi'$ are composed by the rules PNet-Ord and PNet-Prl.

We can lift the fusion product of guarded strings of sets to languages. For $L, L' \subseteq \mathrm{GSS}$ we set:

$$L \diamond L' = \{w \diamond w' \mid w \in L, w' \in L'\}$$

We define the $n$-th power of $L$ inductively, as $L^0 = \mathcal{M}\mathsf{BP}$ and $L^{n+1} = L^n \diamond L$. Moreover, for $S \subseteq \mathcal{M}\mathsf{BP}$ we introduce shorthand notation for $\bar{S} = \mathcal{M}(\mathsf{BP})\backslash S$ and $S \diamond L = \{\, S \,\} \diamond L$, and define:

$$L +_S L' = (S \diamond L) \cup (\bar{S} \diamond L') \quad \text{and} \quad L^{(S)} = \bigcup_{n\geq 0}(S \diamond L)^n \diamond \bar{S}$$

Similarly, we lift layer-by-layer ordered and parallel composition to languages:

$$L \circ L' = \{w \circ w' \mid w \in L, w' \in L'\} \quad \text{and} \quad L \parallel L' = \{w \parallel w' \mid w \in L, w' \in L'\}$$

Using the notation introduced above, we next define how PBKAT expressions are interpreted as languages of guarded strings of sets via the semantic map $I(-)\colon \mathsf{Exp} \longrightarrow 2^{\mathcal{P}\mathcal{M}\mathsf{BP};(\mathcal{P}\Pi;\mathcal{P}\mathcal{M}\mathsf{BP})^*}$. Recall that every basic action $r \triangleright (\Sigma\, p{\cdot}o)$ is composed of two atomic actions $[\mathbb{1}]r \blacktriangleright (\Sigma\, p{\cdot}o)$ and $[r]\emptyset \blacktriangleright \emptyset$ conditioned on the presence and absence of required Bell pairs, respectively. A guard $\alpha \in \mathsf{BExp}$ is by definition a predicate over $\mathcal{M}\mathsf{BP}$, which can be thought of as a set of multisets that satisfy the expression. Formally, we define $\mathsf{sat}(\alpha) \subseteq \mathcal{M}\mathsf{BP}$ and the semantic map $I$ in the following way:

$$\begin{aligned} & & I(\pi) &= \{\, \mathcal{M}\mathsf{BP}\,\{\, \pi \,\}\,\mathcal{M}\mathsf{BP} \,\} \\ \mathsf{sat}(0) &= \emptyset & I(\alpha) &= \{\, \mathsf{sat}(\alpha) \,\} \\ \mathsf{sat}(1) &= \mathcal{M}\mathsf{BP} & I(r \triangleright (\Sigma\, p{\cdot}o)) &= \{\, \mathcal{M}\mathsf{BP}\,\{\, [\mathbb{1}]r \blacktriangleright (\Sigma\, p{\cdot}o),\ [r]\emptyset \blacktriangleright \emptyset \,\}\,\mathcal{M}\mathsf{BP} \,\} \\ \mathsf{sat}(a) &= \{\, a \,\} \text{ for } a \in \mathcal{M}\mathsf{BP} & I(e\,;f) &= I(e) \diamond I(f) \\ \mathsf{sat}(\alpha \vee \beta) &= \mathsf{sat}(\alpha) \cup \mathsf{sat}(\beta) & I(e +_\alpha f) &= I(e) +_{\mathsf{sat}(\alpha)} I(f) \\ \mathsf{sat}(\alpha \wedge \beta) &= \mathsf{sat}(\alpha) \cap \mathsf{sat}(\beta) & I(e^{(\alpha)}) &= I(e)^{(\mathsf{sat}(\alpha))} \\ \mathsf{sat}(\bar{\alpha}) &= \mathcal{M}\mathsf{BP} \setminus \mathsf{sat}(\alpha) & I(e \circ f) &= I(e) \circ I(f) \\ & & I(e \parallel f) &= I(e) \parallel I(f) \end{aligned}$$

*Remark 4.1.* Despite having similar name, the *abstract semantics* $I(-)$ of PBKAT is not closely related to the abstract interpretation of probabilistic semantics by Monniaux [2000]. The key difference with that line of work (where a state is a probability distribution) is that PBKAT's abstract semantics deals with uninterpreted (abstract) actions, which are not probabilistic.

*Probabilistic interpretation of executions.* Parallel composition of expressions induces nondeterminism even for the guarded fragment of PBKAT. A realistic interpretation modeling protocol execution thus needs to combine probabilistic and nondeterministic (or possibilistic) choice. Consider the following scenario: Given an input network state, the scheduler nondeterministically chooses which of the actions that require the same Bell pairs will execute. We wish to keep track of all scheduler's possibilities of how to map inputs to distributions over outputs.

To make definitions compatible with concrete network constraints, consider a network on nodes $A_1, \ldots, A_l$ with the upper bound $m_{ij}$ on the number of Bell pairs $A_i{\sim}A_j$. For an $a \in \mathcal{M}\mathrm{BP}$ containing $n_{ij}$ Bell pairs $A_i{\sim}A_j$ we define, $\lfloor a \rfloor = \{\!\{\min(n_{ij}, m_{ij}) \times A_i{\sim}A_j\}\!\}$, that can be thought of as replacing Bell pairs $A_i{\sim}A_j$ with refreshed ones of the same kind once the storage capacity $m_{ij}$ is reached.

For an atomic action $\pi = [t]r \blacktriangleright (\Sigma\, p{\cdot}o)$ we define the execution map $\mathrm{exe}(\pi)\colon \mathcal{M}\mathrm{BP} \to \mathcal{D}\mathcal{M}\mathrm{BP}$:

$$\mathrm{exe}(\pi)(a) = \begin{cases} \Sigma\, p \cdot \lfloor o \uplus a \backslash r \rfloor & \text{if } r \subseteq a \text{ and } \langle\!| t |\!\rangle a = \top \\ 0 & \text{else} \end{cases}$$

The above definition formalizes the behavior of an atomic action that, given that its tests $r \subseteq a$ and $t$ succeed on input $a$, it outputs multiset $\lfloor o \uplus a \backslash r \rfloor$ with probability $p$, i.e., $\mathrm{exe}(\pi)(a)(\lfloor o \uplus a \backslash r \rfloor) = p$.

For a PBKAT expression $e \in \mathrm{Exp}$, the elements of its abstract semantics $I(e) \subseteq \mathrm{GSS}$ are enlarged guarded strings, as defined in the previous paragraph.

The map $\llbracket - \rrbracket_{\mathrm{ex}}\colon \mathrm{GSS} \to \mathcal{M}\mathrm{BP} \to \mathcal{C}\mathcal{M}\mathrm{BP}$ is defined recursively for $a_0 \in \mathcal{M}\mathrm{BP}$ and $w \in \mathrm{GSS}$:

$$\llbracket S \rrbracket_{\mathrm{ex}}(a) = \begin{cases} \{\,\delta_a\,\} & \text{if } a \in S \\ \emptyset & \text{else} \end{cases}$$

$$\llbracket w \rrbracket_{\mathrm{ex}}(a_0) = \begin{cases} \bigcup_{\pi_1 \in \Omega_1} \mathrm{WMS}\left(\sum_{a_1 \in S_1} \mathrm{exe}(\pi_1)(a_0)(a_1) \cdot \llbracket S_1 \Omega_2 S_2 \cdots \Omega_n S_n \rrbracket_{\mathrm{ex}}(a_1)\right) & \text{if } a_0 \in S_0 \\ \emptyset & \text{else} \end{cases}$$

where the weighted Minkowski sum of a sub-distribution $\sum_{i=1}^{k} p_i{\cdot}C_i \in \mathcal{D}\mathcal{C}\mathcal{M}\mathrm{BP}$ is defined as:

$$\mathrm{WMS}\,(\sum_{i=1}^{k} p_i{\cdot}C_i) = \begin{cases} \emptyset & \text{if all } C_i = \emptyset \\ \{\, \sum_{i=1}^{k} p_i{\cdot}\mu_i \;\mid\; \mu_i \in C_i \neq \emptyset \text{ for all } 1 \leq i \leq k\,\} & \text{else} \end{cases}$$

*Remark 4.2.* In the above definitions all sets are considered as convex closures of its generators. These definitions prevent sets of sub-distributions that contain the zero sub-distribution, e.g., $\mathrm{conv}\{0, \delta_1, \delta_2\}$. Alternatively, we could define the convex closure of a set of subdistributions by only taking into account subdistributions with nonzero support.

Next, we lift $\llbracket - \rrbracket_{\mathrm{ex}}$ to languages via another Minkowski summation. For $L \subseteq \mathrm{GSS}$ we define:

$$\llbracket L \rrbracket_{\mathrm{ex}}(a) = \mathrm{WMS}\,(\sum_{w \in L} 1{\cdot}\llbracket w \rrbracket_{\mathrm{ex}}(a)) = \{\, \sum_{w \in L} \delta \;\mid\; \delta \in \llbracket w \rrbracket_{\mathrm{ex}}(a) \neq 0\,\}$$

Finally, we define $\llbracket - \rrbracket\colon \mathrm{Exp} \to (\mathcal{M}\mathrm{BP} \to \mathcal{C}\mathcal{M}\mathrm{BP})$ as composition $\llbracket e \rrbracket = \llbracket I(e) \rrbracket_{\mathrm{ex}}$.

The next example shows how PBKAT semantics faithfully models end-to-end behaviors of protocol executions while taking network constrains into consideration. We will compute the $\llbracket - \rrbracket$ semantics of a variant of protocol (a) by unfolding the above definitions to illustrate the inner workings of our approach (whose implementation will be described in Section 6).

*Example 4.2.* Assume that the network in Figure 1 has the capacity of handling at most two Bell pairs $C{\sim}C$ and one of each $A{\sim}C$ and $B{\sim}C$. Recall the first two rounds of protocol (a) from our running example, which attempt to create two Bell pairs $C{\sim}C$ and transmit them to $A{\sim}C$ and $B{\sim}C$. Since basic actions are likely to fail (and therefore output insufficiently Bell pairs required by the the actions in subsequent rounds), it is sensible to introduce guards as additional checks ensuring that actions are not unnecessarily enabled unless the required Bell pairs are available. To this end we consider protocol ($\mathrm{a}_2$), specified with conditionals as $(e \parallel f)$ ; $\mathsf{sw}\langle A{\sim}B \,@\, C\rangle_{3/5}$ for,

$$e = \big(\mathsf{cr}\langle C\rangle_{2/3} +_\alpha \mathsf{tr}\langle C \to A{\sim}C\rangle_{4/5}\big) ; \mathsf{tr}\langle C \to A{\sim}C\rangle_{4/5} \text{ and } f = \mathsf{cr}\langle C\rangle_{2/3} ; \big(\mathsf{cr}\langle C\rangle_{2/3} +_\alpha \mathsf{tr}\langle C \to B{\sim}C\rangle_{1/2}\big)$$

where guard $\alpha$ checks if there is no $C{\sim}C$ in the network. We can think of $e$ and $f$ as being specified by two practitioners who use slightly different strategies to create Bell pair $C{\sim}C$ in order to transmit it to $A{\sim}C$ and $B{\sim}C$, respectively: $e$ in the first round creates a Bell pair $C{\sim}C$ if there is no $C{\sim}C$ yet available, otherwise it immediately transmits it to $A{\sim}C$, and in the second round $e$ simply calls $\mathsf{tr}\langle C \to A{\sim}C\rangle$. By contrast, $f$ always creates a Bell pair $C{\sim}C$ in the first round, and in the second round $f$ either creates $C{\sim}C$ or transmits it to $B{\sim}C$, depending on the availability of $C{\sim}C$ from the first round. Equivalently, protocols $e$ and $f$ are specified as:

$$\begin{aligned} e &= \big(\emptyset \triangleright_{2/3} \{\!\{C{\sim}C\}\!\} +_\alpha \{\!\{C{\sim}C\}\!\} \triangleright_{4/5} \{\!\{A{\sim}C\}\!\}\big) ; \{\!\{C{\sim}C\}\!\} \triangleright_{4/5} \{\!\{A{\sim}C\}\!\} \\ f &= \emptyset \triangleright_{2/3} \{\!\{C{\sim}C\}\!\} ; \big(\emptyset \triangleright_{2/3} \{\!\{C{\sim}C\}\!\} +_\alpha \{\!\{C{\sim}C\}\!\} \triangleright_{1/2} \{\!\{B{\sim}C\}\!\}\big) \end{aligned}$$

We abbreviate multisets $\{\!\{i \times C{\sim}C,\ j \times A{\sim}C,\ k \times B{\sim}C\}\!\}$ by $a_{ijk}$, and the atomic actions in $e, f$ by,

$$\begin{aligned} \pi_A &= [\mathbb{1}]\{\!\{C{\sim}C\}\!\} \blacktriangleright \big(\tfrac{4}{5}\cdot\{\!\{A{\sim}C\}\!\} + \tfrac{1}{5}\cdot\emptyset\big) & \pi_C &= [\mathbb{1}]\emptyset \blacktriangleright \big(\tfrac{2}{3}\cdot\{\!\{C{\sim}C\}\!\} + \tfrac{1}{3}\cdot\emptyset\big) \\ \pi_B &= [\mathbb{1}]\{\!\{C{\sim}C\}\!\} \blacktriangleright \big(\tfrac{1}{2}\cdot\{\!\{B{\sim}C\}\!\} + \tfrac{1}{2}\cdot\emptyset\big) & \pi_S &= [C{\sim}C]\emptyset \blacktriangleright \emptyset \end{aligned}$$

and use shorthand $\alpha, \bar{\alpha}, 1$ for $\mathsf{sat}(\alpha) = \{\, a_{0jk} \mid j,k \in \{0,1\} \,\}$, $\mathsf{sat}(\bar{\alpha}) = \{\, a_{ijk} \mid i \in \{1,2\},\ j,k \in \{0,1\} \,\}$, $\mathsf{sat}(1) = \mathcal{M}\mathrm{BP} = \{\, a_{ijk} \mid i \in \{0,1,2\},\ j,k \in \{0,1\} \,\}$, respectively. Then, abstract semantics are:

$$I(e) = \left\{ \begin{array}{c} \alpha\,\{\,\pi_C\,\}\,1\,\{\,\pi_A, \pi_S\,\}\,1, \\ \bar{\alpha}\,\{\,\pi_A, \pi_S\,\}\,1\,\{\,\pi_A, \pi_S\,\}\,1 \end{array} \right\} \qquad I(f) = \left\{ \begin{array}{c} 1\,\{\,\pi_C\,\}\,\alpha\,\{\,\pi_C\,\}\,1, \\ 1\,\{\,\pi_C\,\}\,\bar{\alpha}\,\{\,\pi_B, \pi_S\,\}\,1 \end{array} \right\}$$

$$I(e \parallel f) = \left\{ \begin{array}{c} \alpha\,\{\,\pi_C \parallel \pi_C\,\}\,\alpha\,\{\,\pi_A \parallel \pi_C,\ \pi_S \parallel \pi_C\,\}\,1 \\ \alpha\,\{\,\pi_C \parallel \pi_C\,\}\,\bar{\alpha}\,\{\,\pi_A \parallel \pi_B,\ \pi_A \parallel \pi_S,\ \pi_B \parallel \pi_S,\ \pi_S \parallel \pi_S\,\}\,1, \\ \bar{\alpha}\,\{\,\pi_A \parallel \pi_C, \pi_S \parallel \pi_C\,\}\,\alpha\,\{\,\pi_A \parallel \pi_C, \pi_S \parallel \pi_C\,\}\,1, \\ \bar{\alpha}\,\{\,\pi_A \parallel \pi_C, \pi_S \parallel \pi_C\,\}\,\bar{\alpha}\,\{\,\pi_A \parallel \pi_B,\ \pi_A \parallel \pi_S,\ \pi_B \parallel \pi_S,\ \pi_S \parallel \pi_S\,\}\,1 \end{array} \right\}$$

where, by rule PNet-Prl, parallel composition of two atomic actions yields a new atomic action:

$$\begin{aligned} \pi_A \parallel \pi_C &= [\mathbb{1}]\{\!\{C{\sim}C\}\!\} \blacktriangleright \big(\tfrac{8}{15}\cdot\{\!\{A{\sim}C, C{\sim}C\}\!\} + \tfrac{4}{15}\cdot\{\!\{A{\sim}C\}\!\} + \tfrac{2}{15}\cdot\{\!\{C{\sim}C\}\!\} + \tfrac{1}{15}\cdot\emptyset\big) \\ \pi_S \parallel \pi_C &= [C{\sim}C]\emptyset \blacktriangleright \big(\tfrac{2}{3}\cdot\{\!\{C{\sim}C\}\!\} + \tfrac{1}{3}\cdot\emptyset\big) \\ \pi_A \parallel \pi_B &= [\mathbb{1}]\{\!\{C{\sim}C, C{\sim}C\}\!\} \blacktriangleright \big(\tfrac{4}{10}\cdot\{\!\{A{\sim}C, B{\sim}C\}\!\} + \tfrac{4}{10}\cdot\{\!\{A{\sim}C\}\!\} + \tfrac{1}{10}\cdot\{\!\{B{\sim}C\}\!\} + \tfrac{1}{10}\cdot\emptyset\big) \\ \pi_A \parallel \pi_S &= [C{\sim}C, C{\sim}C]\{\!\{C{\sim}C\}\!\} \blacktriangleright \big(\tfrac{4}{5}\cdot\{\!\{A{\sim}C\}\!\} + \tfrac{1}{5}\cdot\emptyset\big) \\ \pi_B \parallel \pi_S &= [C{\sim}C, C{\sim}C]\{\!\{C{\sim}C\}\!\} \blacktriangleright \big(\tfrac{1}{2}\cdot\{\!\{B{\sim}C\}\!\} + \tfrac{1}{2}\cdot\emptyset\big) \\ \pi_S \parallel \pi_S &= [C{\sim}C]\emptyset \blacktriangleright \emptyset \\ \pi_C \parallel \pi_C &= [\mathbb{1}]\emptyset \blacktriangleright \big(\tfrac{4}{9}\cdot\{\!\{C{\sim}C, C{\sim}C\}\!\} + \tfrac{4}{9}\cdot\{\!\{C{\sim}C\}\!\} + \tfrac{1}{9}\cdot\emptyset\big) \end{aligned}$$

Below, we visualize the execution traces of $e \parallel f$, given that at the start there are no Bell pairs in the network, i.e., we take $\emptyset = a_{000} \in \mathcal{M}\mathrm{BP}$ as the input. Note that two strings from $I(e \parallel f)$ execute,

$$a_{000} \xrightarrow{\pi_C \parallel \pi_C} \Big\{\ \tfrac{1}{9}\cdot a_{000} \xrightarrow{\quad\pi_S \parallel \pi_C\quad} \Big\{\ \tfrac{2}{3}\cdot a_{100} + \tfrac{1}{3}\cdot a_{000}$$

$$a_{000} \xrightarrow{\pi_C \parallel \pi_C} \left\{ \begin{array}{l} \tfrac{4}{9}\cdot a_{200} \xrightarrow{\quad\pi_A \parallel \pi_B\quad} \Big\{\ \tfrac{4}{10}\cdot a_{011} + \tfrac{4}{10}\cdot a_{010} + \tfrac{1}{10}\cdot a_{001} + \tfrac{1}{10}\cdot a_{000} \\ \tfrac{4}{9}\cdot a_{100} \Big\langle \begin{array}{l} \xrightarrow{\quad\pi_A \parallel \pi_S\quad} \Big\{\ \tfrac{4}{5}\cdot a_{010} + \tfrac{1}{5}\cdot a_{000} \\ \xrightarrow[\quad\pi_B \parallel \pi_S\quad]{} \Big\{\ \tfrac{1}{2}\cdot a_{001} + \tfrac{1}{2}\cdot a_{000} \end{array} \end{array} \right.$$

$$\begin{array}{crcl@{\qquad}crcl}
\text{skip} & 1 \mathbin{\square} e & \equiv & e \mathbin{\square} 1 \equiv e & \text{commut.} & e \parallel f & \equiv & f \parallel e \\
\text{abort} & 0 \mathbin{\square} e & \equiv & e \mathbin{\square} 0 \equiv 0 & \text{synchrony} & (h \mathbin{;} e) \parallel (k \mathbin{;} f) & \equiv & (h \parallel k) \mathbin{;} (e \parallel f) \\
\text{assoc.} & (e \mathbin{\square} f) \mathbin{\square} g & \equiv & e \mathbin{\square} (f \mathbin{\square} g) & & (h \mathbin{;} e) \circ (k \mathbin{;} f) & \equiv & (h \circ k) \mathbin{;} (e \circ f) \\
\text{guards} & e +_\beta e & \equiv & e & \text{right distrib.} & (e +_\beta f) \mathbin{\square} g & \equiv & (e \mathbin{\square} g) +_\beta (f \mathbin{\square} g) \\
 & e +_\beta f & \equiv & f +_{\bar{\beta}} e & \text{loops} & e^{(\beta)} & \equiv & e \mathbin{;} e^{(\beta)} +_\beta 1 \\
 & e +_\beta f & \equiv & \beta \mathbin{;} e +_\beta f & & (e +_\alpha 1)^{(\beta)} & \equiv & (\alpha \mathbin{;} e)^{(\beta)} \\
 & (e +_\alpha f) +_\beta g & \equiv & e +_{\alpha \wedge \beta} (f +_\beta g) & & & &
\end{array}$$

Fig. 10. PBKAT rules. Symbol $\square$ stands for any operator in $\{\circ, \parallel, ;\}$, where $h$ and $k$ combine basic actions without the ; operator. For ; only the right distributivity rule holds, whereas $\circ$ and $\parallel$ are distributive from both sides, since guards attach to the lowest round. We also include PNet-Ord and PNet-Prl rules for combining atomic actions as part of PBKAT rules.

where different colors indicate the nondeterministic choice between $\pi_A \parallel \pi_S$ and $\pi_B \parallel \pi_S$ that require the same Bell pairs. By definition of $[\![-]\!]_{\text{ex}}$, we sum the subdistributions for each choice, yielding:

$$[\![e \parallel f]\!](a_{000}) = \operatorname{conv}\left(\left\{\begin{array}{l} \frac{23}{135} \cdot a_{000} + \frac{6}{135} \cdot a_{001} + \frac{72}{135} \cdot a_{010} + \frac{24}{135} \cdot a_{011} + \frac{10}{135} \cdot a_{100}, \\ \frac{41}{135} \cdot a_{000} + \frac{36}{135} \cdot a_{001} + \frac{24}{135} \cdot a_{010} + \frac{24}{135} \cdot a_{011} + \frac{10}{135} \cdot a_{100} \end{array}\right\}\right)$$

From this we can, for example, read that protocol $e \parallel f$ on input $\emptyset$ always outputs $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ with probability $\frac{24}{135}$, and outputs $\emptyset$ with probability between $\frac{23}{135}$ and $\frac{41}{135}$. Thus, after the swapping $\mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle_{3/5}$, protocol ($\text{a}_2$) produces Bell pair $A{\sim}B$ with probability $\frac{8}{75}$.

### 4.3 Properties

In this section we present properties and prove correctness of PBKAT semantics, which will be useful for reasoning about and modifying protocols with the purpose of optimization in Section 5.

*Concurrent execution of basic actions.* We prove that the semantics of basic actions is correct with respect to the outcome distributions that we obtain by explicit combinatorial reasoning. In Section 4.2, we showed that the semantics of basic actions correctly describe their execution. Now consider concurrent composition of basic actions $f = r \triangleright (\Sigma\, p \cdot o)$ and $f' = r' \triangleright (\Sigma\, p' \cdot o')$. In the ordered composition $f \circ f'$ the priority is given to the first action, which is captured in,

$$I(f \circ f') = \{\, [\mathbb{1}] r \uplus r' \blacktriangleright (\Sigma\, pp' \cdot o \uplus o'),\ [r \uplus r'] r \blacktriangleright (\Sigma\, p \cdot o),\ [r] r' \blacktriangleright (\Sigma\, p' \cdot o'),\ [r \wedge r'] \emptyset \blacktriangleright \emptyset \,\}$$

obtained by the rule PNet-Ord. Given an input $a \in \mathcal{M}\mathrm{BP}$, exactly one of the following atomic actions in $I(f \circ f')$ is executed (and the others abort), depending on which test succeeds: $r \uplus r' \subseteq a$ when there are sufficient Bell pairs for both actions, $(r \uplus r' \nsubseteq a) \wedge (r \subseteq a)$ if only $f$ gets enough Bell pairs, $(r \nsubseteq a) \wedge (r' \subseteq a)$ when $f'$ uses Bell pairs that $f$ does not require, or $(r \nsubseteq a) \wedge (r' \nsubseteq a)$ when neither action can be executed. On the other hand, parallel execution of basic actions yields:

$$I(f \parallel f') = \{[\mathbb{1}] r \uplus r' \blacktriangleright (\Sigma\, pp' \cdot o \uplus o'),\ [r \uplus r'] r \blacktriangleright (\Sigma\, p \cdot o),\ [r \uplus r'] r' \blacktriangleright (\Sigma\, p' \cdot o'),\ [r \wedge r'] \emptyset \blacktriangleright \emptyset\}$$

evaluated by the rule PNet-Prl. For some inputs $a \in \mathcal{M}\mathrm{BP}$, given that both actions require same Bell pairs but there are only enough for one of them, both tests $(r \uplus r' \nsubseteq a) \wedge (r \subseteq a)$ and $(r \uplus r' \nsubseteq a) \wedge (r' \subseteq a)$ succeed. Then one of the two corresponding actions is chosen nondeterministically – we can think of this as the scheduler nondetermistically choosing an action.

*Algebraic properties.* The rules in Figure 10 for combining PBKAT expressions can be thought of as guarded variants of SKA axioms. The theorems below show that the PBKAT rules (including the PNet-Ord and PNet-Prl rules) are sound with respect to the PBKAT semantics and that the semantics is well defined. A proof sketch of the theorems is in Appendix A.

Formally, the next Theorem 4.1 states that every equivalence provable using the PBKAT rules also holds in the denotational model, resulting in the same end-to-end behavior. This is, $\vdash e \equiv f \Rightarrow [\![e]\!] = [\![f]\!]$, where $\vdash$ denotes provability in PBKAT and $[\![-]\!]$ is defined in Section 4.2. The key step is to prove $\vdash e \equiv f \Rightarrow I(e) = I(f)$, i.e., the soundness of PBKAT rules with respect to the abstract semantics (that builds a language model of guarded strings). Then, the theorem follows since $[\![-]\!]$ is defined by composing $I$ with the execution semantics $[\![-]\!]_{\mathsf{ex}}$.

Theorem 4.1. *If protocol expressions $e, f \in Exp$ are equivalent under PBKAT rules, then their denotational semantics coincide. This is,* $\vdash e \equiv f \Rightarrow [\![e]\!] = [\![f]\!]$.

Furthermore, Theorem 4.2 shows that $[\![-]\!]$ gives probabilistic interpretation to the expressions.

Theorem 4.2. *PBKAT semantics produces sets of well-defined subprobability kernels. This means that if $e \in Exp$ and $a \in \mathcal{M}BP$, then $[\![e]\!](a) \in \mathcal{CM}BP$, i.e., $[\![e]\!](a)$ is a set of subdistributions.*

Theorem 4.2 is due to an important property of the abstract semantics, which we define in the next lemma (cf. the set of guarded strings in Example 4.2):

Lemma 4.1 (Determinacy property). *The language $I(e) \subseteq GSS$ corresponding to the PBKAT expression $e$ satisfies the determinacy property, i.e., whenever strings $w, w' \in I(e)$ agree on their first $n$ sets of states $S_0, \ldots, S_{n-1}$, then they agree on their first $n$ sets of actions $\Omega_1, \ldots, \Omega_n$ and $S_n \cap S'_n = \emptyset$.*

For protocols with no (unbounded) while loops the set of subdistributions is finitely generated. As a consequence, we obtain a tool to reason about the equality of protocols w.r.t. $[\![-]\!]$ in Lemma 4.2. For while-free protocols, the semantic equivalence of two protocols can always be decided by (i) computing the (unique and finite) sets of extreme points (i.e., generators) for the respective convex sets and (ii) checking that the computed sets of extreme points are equal.

Lemma 4.2. *If protocol expressions $e$ and $f$ contain no while loops, then $[\![e]\!] = [\![f]\!]$ is decidable.*

*Relation to BellKAT and (probabilistic) GKAT.* Besides for allowing probabilistic actions, PBKAT's semantics differs in two crucial aspects from BellKAT, both aiming to address nondeterminism. The first aspect is that PBKAT's syntax has no operator for expressing nondeterministic choice. The second aspect is that PBKAT's strings of actions are guarded, where a guard filters the multisets of Bell pairs and retains only those that satisfy the test. Conditionals and while loops in PBKAT use guards to resolve the inherent nondeterminism in the union and iteration operators (denoted as + and $\star$ in BellKAT). This way, the only nondeterminism in PBKAT arises from parallel composition of actions within single rounds (see the discussion of nondeterminism in Section 3).

PBKAT rules in Figure 10 are closely related to the axiomatization of GKAT [Smolka et al. 2019a]. In addition to GKAT's axioms, PBKAT has synchrony axioms describing the interaction between the synchronous (parallel and ordered) compositions and the sequential composition, capturing the intended lock-step behaviour. The only GKAT axiom not included in PBKAT is the fixpoint axiom. (As we are not aiming for completeness, we decided to not include the fixpoint axiom, which would require introducing the concept of the "empty word property". This is in line with works BellKAT, ProbNetKAT and McNetKAT, that all have sound but not complete axiomatization; cf. Section 2.) Despite having similar name, there is a key difference with Probabilistic GKAT [Różowski et al. 2023], namely PBKAT does not have probabilistic choice (for probabilistic branching) – in this sense, PBKAT's semantics is closer to a specific probabilistic interpretation of the original GKAT.

## 5 Quantitative Analysis

This section makes use of the properties of PBKAT semantics and its explicit representation. Besides enabling practical *verification* by checking semantic equivalences of protocols, PBKAT properties enable quantitative analysis of protocol executions and consequently their *optimization*.

### 5.1 Protocol Optimization

Quantitative reasoning is particularly useful to analyze the effects of quantum network protocol executions through different branches of the execution traces, some of which may be quite counterintuitive. Concretely, such analysis can be used for optimizing the scheduling of actions within protocol rounds (e.g., as a way to control the power of an adversarial scheduler [Rand and Zdancewic 2016]), or for resolving nondeterminism in the style of pGCL [McIver and Morgan 2005, Chapter 1.8] or demonic outcome logic [Zilberstein et al. 2024].

To showcase protocol optimization, we incrementally build optimized versions of the entanglement swap protocol from our running example.

*Example 5.1.* Consider the repeater swap protocol (a), running on the three-node network in Figure 1, with the goal to generate end-to-end Bell pair $A{\sim}B$. The protocol first creates two Bell pairs $C{\sim}C$ locally at node $C$ (acting as a source), then transmits one copy to $A{\sim}C$ and the other to $B{\sim}C$, which are subsequently swapped at node $C$ (now acting as a repeater) to generate Bell pair $A{\sim}B$.

For simplicity, this example focuses on the transmit subprotocol and assumes the network is constrained to handle at most two Bell pairs $C{\sim}C$ and one of each $A{\sim}C$ and $B{\sim}C$. Below, we fix the success probabilities of basic actions to $\mathsf{cr}\langle C\rangle_{0.9}$, $\mathsf{tr}\langle C \to A{\sim}C\rangle_{0.8}$ and $\mathsf{tr}\langle C \to B{\sim}C\rangle_{0.5}$, and build and compare different protocols, aiming to generate $A{\sim}C$ and $B{\sim}C$ with the highest probabilities.

**I.** Consider protocols $e$ and $f$ that create $C{\sim}C$ and transmit it to $A{\sim}C$ and $B{\sim}C$, respectively:

$$e = \mathsf{cr}\langle C\rangle_{0.9} \,;\, \mathsf{tr}\langle C \to A{\sim}C\rangle_{0.8} \triangleq \emptyset \triangleright_{0.9} \{\!\{C{\sim}C\}\!\} \,;\, \{\!\{C{\sim}C\}\!\} \triangleright_{0.8} \{\!\{A{\sim}C\}\!\}$$
$$f = \mathsf{cr}\langle C\rangle_{0.9} \,;\, \mathsf{tr}\langle C \to B{\sim}C\rangle_{0.5} \triangleq \emptyset \triangleright_{0.9} \{\!\{C{\sim}C\}\!\} \,;\, \{\!\{C{\sim}C\}\!\} \triangleright_{0.5} \{\!\{B{\sim}C\}\!\}$$

We wish to compare the semantic of protocols $e$ and $f$ running concurrently, namely $[\![ e \parallel f ]\!]$ and $[\![ e \circ f ]\!]$ – the difference between them is that, when there is only one $C{\sim}C$ available, in $e \parallel f$ there are two possibilities where $C{\sim}C$ can be transmitted, either to $A{\sim}C$ or $B{\sim}C$, and in $e \circ f$ protocol $e$ has priority over $f$, thus $C{\sim}C$ is transmitted to $A{\sim}C$. Protocols $e \parallel f$ and $e \circ f$, given an empty input, return distributions $\mathrm{conv}(\{\mu_1, \mu_2\})$ and $\{\mu_1\}$ respectively, where $\mu_1$ and $\mu_2$ correspond to whether $\mathsf{tr}\langle C \to A{\sim}C\rangle_{0.8}$ or $\mathsf{tr}\langle C \to B{\sim}C\rangle_{0.5}$ is selected. Note that the probability of outcome $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ is independent of whether priority is given to $e$ or $f$:

$$\mu_1 = 0.324 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.468 \cdot \{\!\{A{\sim}C\}\!\} + 0.081 \cdot \{\!\{B{\sim}C\}\!\} + 0.127 \cdot \emptyset$$
$$\mu_2 = 0.324 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.324 \cdot \{\!\{A{\sim}C\}\!\} + 0.171 \cdot \{\!\{B{\sim}C\}\!\} + 0.181 \cdot \emptyset$$

**II.** Next we increase the probability of generating $A{\sim}C$ and $B{\sim}C$ by rerunning protocols $e \parallel f$ and $e \circ f$. The evaluation of the semantics of $(e \parallel f) \,;\, (e \parallel f)$ and $(e \circ f) \,;\, (e \circ f)$ on the empty input is,

$$[\![ (e \parallel f) \,;\, (e \parallel f) ]\!](\emptyset) = \mathrm{conv}(\{\nu_1, \nu_2, \dots \nu_{14}\}) \quad \text{and} \quad [\![ (e \circ f) \,;\, (e \circ f) ]\!](\emptyset) = \{\nu_1\}$$

where in the convex set with 14 generators, e.g., respective distributions $\nu_2, \nu_3$ have max, min probabilities of generating both Bell pairs, and $\nu_4$ corresponds to the semantics of $(f \circ e) \,;\, (f \circ e)$:

$$\nu_1 = 0.61884 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.337896 \cdot \{\!\{A{\sim}C\}\!\} + 0.027135 \cdot \{\!\{B{\sim}C\}\!\} + 0.016129 \cdot \emptyset$$
$$\nu_2 = 0.678456 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.248328 \cdot \{\!\{A{\sim}C\}\!\} + 0.050229 \cdot \{\!\{B{\sim}C\}\!\} + 0.022987 \cdot \emptyset$$
$$\nu_3 = 0.607176 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.319608 \cdot \{\!\{A{\sim}C\}\!\} + 0.050229 \cdot \{\!\{B{\sim}C\}\!\} + 0.022987 \cdot \emptyset$$
$$\nu_4 = 0.653832 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.222264 \cdot \{\!\{A{\sim}C\}\!\} + 0.091143 \cdot \{\!\{B{\sim}C\}\!\} + 0.032761 \cdot \emptyset$$

The first protocol with unresolved nondeterminism may succeed with higher probability than the second deterministic one. Surprisingly, the probability of the outcome $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ is lower when priority is given to $e$ as opposed to $f$ in both runs. This goes against the expectation that giving priority to actions with the highest probability of success ($e$ contains $\mathsf{tr}\langle C \to A{\sim}C\rangle_{0.8}$) should lead to the most likely outcome. (Compare this with angelic choice as a way of resolving nondeterminism in [Morgan et al. 1996].) In our case, giving priority to actions with the lowest probability of success ($f$ with $\mathsf{tr}\langle C \to B{\sim}C\rangle_{0.5}$), proves to be more beneficial.

**III.** When protocols run in parallel, we use guards to control both protocols in a synchronized manner. Protocol II can be improved by guarding the fourth round, so that priority is given to the transmit that hasn't yet succeeded. Let $e'$ and $f'$ concurrently create and transmit Bell pairs,

$$e'{=}\mathsf{cr}\langle C\rangle_{0.9};(\mathsf{tr}\langle C \to A{\sim}C\rangle_{0.8} {+_\alpha} \mathsf{tr}\langle C \to B{\sim}C\rangle_{0.5}) \quad f'{=}\mathsf{cr}\langle C\rangle_{0.9};\big(\mathsf{tr}\langle C \to B{\sim}C\rangle_{0.5} {+_\beta} \mathsf{tr}\langle C \to A{\sim}C\rangle_{0.8}\big)$$

where $\alpha$ and $\beta$ check for the absence of $A{\sim}C$ and $B{\sim}C$, respectively. This prevents $C{\sim}C$ from being transmitted twice to the same neighbor. Figure 11 shows the execution semantics of $e' \parallel f'$ for all the inputs that are relevant for iteration. By the same methods as before, we evaluate that from input $\emptyset$, protocol $e' \parallel f'$ returns $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ with probabilities 0.324 after one iteration and 0.766228 after two; this outcome occurs regardless of which action is selected when only one $C{\sim}C$ is available and both transmits require it. Concretely, the output of two iterations is $\mathrm{conv}(\{v_1', v_2'\})$:

$$v_1' = 0.766228 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.201006 \cdot \{\!\{A{\sim}C\}\!\} + 0.0166374 \cdot \{\!\{B{\sim}C\}\!\} + 0.016129 \cdot \emptyset$$
$$v_2' = 0.766228 \cdot \{\!\{A{\sim}C, B{\sim}C\}\!\} + 0.156654 \cdot \{\!\{A{\sim}C\}\!\} + 0.0443574 \cdot \{\!\{B{\sim}C\}\!\} + 0.032761 \cdot \emptyset$$

**IV.** In practice, most protocols are expressed as guarded iterations of subprotocols until the requested Bell pairs are successfully generated. In what follows, we first specify a protocol that repeatedly creates fresh copies of $C{\sim}C$ and transmits them to $A{\sim}C$ and $B{\sim}C$ until both transmits succeed and then we evaluate its expected outputs. We define the protocol as **while** $\tau$ **do** $(e' \parallel f') \triangleq (e' \parallel f')^{(\tau)}$, where $e'$ and $f'$ are as in III, and the guard $\tau = (0 \times A{\sim}C) \vee (0 \times B{\sim}C)$ checks for the absence of either $A{\sim}C$ or $B{\sim}C$. For quantitative analysis of this protocol, it is most convenient to represent the semantics of $e' \parallel f'$ with stochastic matrices (as will be elaborated on in the next section). Each row in a stochastic matrix represents the output distribution that a protocol returns from a given input. The semantics of $e' \parallel f'$ is represented by two matrices,

$$\begin{bmatrix} .181 & .171 & .324 & .324 \\ 0 & 0.0784 & 0 & .9216 \\ 0 & 0 & .3025 & .6975 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} .127 & .081 & .468 & .324 \\ 0 & 0.0784 & 0 & .9216 \\ 0 & 0 & .3025 & .6975 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

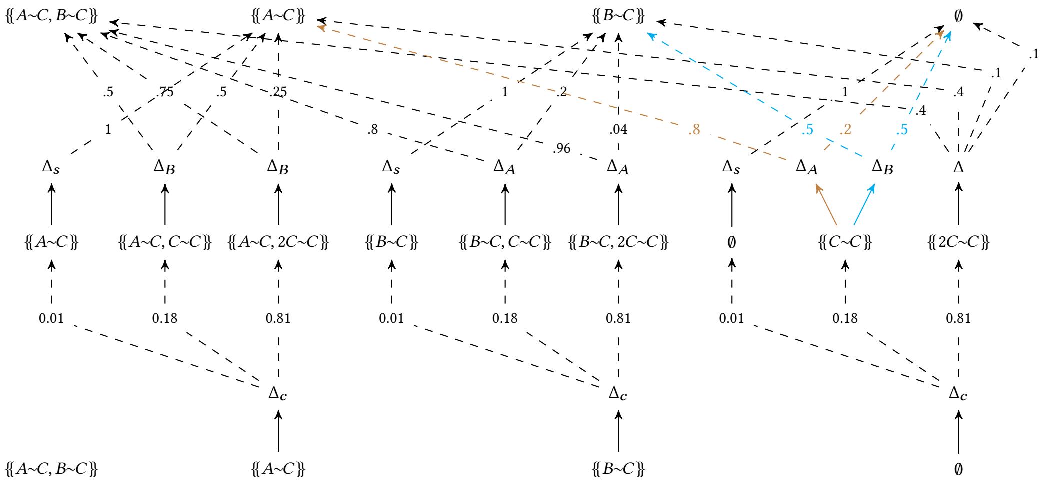


Fig. 11. Distributions in one iteration of protocol $(e' \parallel f')^{(\tau)}$ in Example 5.1 (progressing bottom to top).

(capturing nondeterministic choice depicted in Figure 11), with the entries labeled top-down and left-right by $\emptyset$, $\{\!\!\{B{\sim}C\}\!\!\}$, $\{\!\!\{A{\sim}C\}\!\!\}$, $\{\!\!\{A{\sim}C, B{\sim}C\}\!\!\}$. Protocol iterations are then modeled with matrix multiplications. Observe that to evaluate the probability of generating both $A{\sim}C$ and $B{\sim}C$ it is irrelevant which of the two matrices is used. For example, the probability of producing $\{\!\!\{A{\sim}C, B{\sim}C\}\!\!\}$ from $\emptyset$ after three iterations is 0.926988 and 0.999382 after seven. Furthermore, the expected number of needed iterations is approximately 2.

## 5.2 Stochastic Matrices

In this section we give another representation of PBKAT semantics, with stochastic matrices.

For brevity, first assume a protocol expression $e \in \mathsf{Exp}$ that contains conditionals together with sequential and ordered composition, but no while loops and furthermore, has no parallel composition that could lead to nondeterministic choice when executed. Then, since the set of network states is finite and can thus be ordered, $[\![e]\!]$ maps an input $a \in \mathcal{M}\mathsf{BP}$ to a unique distribution over outputs, therefore $[\![e]\!](a)$ can be represented by a stochastic vector. This is equivalent to characterizing the semantics of $e$ as a probability assignment $[\![e]\!](a) \colon \mathcal{M}\mathsf{BP} \to [0,1]$, such that $\sum_{b \in \mathcal{M}\mathsf{BP}} [\![e]\!](a)(b) = 1$. Hence, $[\![e]\!]$ can be further represented by a square matrix $\mathcal{B}[\![e]\!] \in [0,1]^{\mathcal{M}\mathsf{BP} \times \mathcal{M}\mathsf{BP}}$ indexed by$\mathcal{M}\mathsf{BP}$, in which the stochastic vector corresponding to input $a$ appears as the $a$-th row. Concretely, the matrix entry $\mathcal{B}[\![e]\!]_{ab} = [\![e]\!](a)(b)$ gives the probability that the execution of protocol $e$ produces output $b \in \mathcal{M}\mathsf{BP}$ on input $a \in \mathcal{M}\mathsf{BP}$. Representing PBKAT semantics with stochastic matrices is useful for modeling the behaviors of protocol executions as Markov chains (i.e., probability transition systems with state space $\mathcal{M}\mathsf{BP}$), ensuring that the sequential composition operator behaves as expected. This makes them particularly suitable for the analysis of iterative behavior of protocols, like the one analyzed in Example 5.1. The limiting behavior of finite state Markov chains has been well studied in the literature (e.g., see [Privault 2018]), and for so-called *absorbing* Markov chains the limit distribution can be computed exactly.

In what follows, we deduce the limiting behavior of the while loop $[\![e^{(\beta)}]\!]$ from the matrix representation of $[\![e]\!]$. We say that a state $a \in \mathcal{M}\mathsf{BP}$ is absorbing if it transitions to itself with probability 1. For example, for $e +_\beta 1$ and input $a$ that passes the guard $\bar{\beta}$ (i.e., $a \in \mathsf{sat}(\bar{\beta})$), $a$ is an absorbing state. It can be shown that, when all input states can reach an absorbing state, the limit distribution of the while loop is computable and equal to $\mathcal{B}[\![e^{(\beta)}]\!] = \lim_{n\to\infty} \mathcal{B}[\![(e +_\beta 1)^{(n)}]\!]$.

We remark that, by the same methods, stochastic representation of a protocol containing nondeterministic choice must be represented by a set of stochastic matrices, as illustrated in Example 5.1. The representation with sets of stochastic matrices contains equivalent information to the one with convex sets, but it's less precise as it contains much redundancy. For example, we need two matrices even if only one input (matrix row) leads to nondeterminism. Thus, we lose information on different inputs yielding different amounts of nondeterminism, while PBKAT's semantics produces a separate convex hull for each input. As multiplication of stochastic matrices represents sequential composition, it is semantically equivalent to taking Minkowski sums and then a convex hull.

## 5.3 Case Studies

In this section we express in PBKAT language two quantum network protocols reported in the literature: the repeater swap experiment realized by Pompili et al. [2021] and the simulation of a repeater swap protocol with distillation in NetSquid [Coopmans et al. 2021].

*Repeater swap protocol realized by Pompili et al. [2021].* The first demonstration of entanglement swapping from Bell pairs stored on remote nodes was experimentally achieved by Pompili et al. [2021]. The network consists of three nodes, $A$, $B$ and $C$, connected as in our Figure 1. Each node has nitrogen vacancy center electronic spin as a communication qubit. In addition, the middle node $C$ employs a Carbon-13 nuclear spin as a memory qubit.

In what follows, we express their protocol in PBKAT language with real-life specifications. The network can handle at most one each of the Bell pairs $A{\sim}C$, $B{\sim}C$ and $A{\sim}B$. First, the protocol needs to distribute entanglement between neighboring nodes, connected by physical quantum channels, to generate $A{\sim}C$ and $B{\sim}C$. To this end, we combine creation and transmission into single basic actions, which we specify as $\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\}$ and $\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\}$. Node $C$ stores the qubit of whichever link is generated first in the memory. Once entanglement is established on both links as described above, the entanglement swapping is executed that consumes $A \sim C$ and $B \sim C$ to produce $A \sim B$. In the reported experiment, Bell pairs $A{\sim}C$ and $B{\sim}C$ are established at the rates 9 Hz and 7 Hz, and the average rate of the combined protocol is 1/(40 s). It uses a timeout of 450 attempts before the sequence is restarted, as a balance between optimizing the entanglement generation rate and quality of the stored state. We specify Pompili et al.'s repeater swap protocol as,

$$\left(\left((\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\} +_\alpha 1)^{(450)} \parallel (\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\} +_\beta 1)^{(450)}\right) ; \mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle_{p_C}\right)^{(\gamma)}$$

where $(e +_\kappa 1)^{(n)}$ stands for the $n$-th unrolling of the while loop $e^{(\kappa)}$, and guards $\alpha$, $\beta$ and $\gamma$ check for the absence of $A{\sim}C$, $B{\sim}C$ and $A{\sim}B$, respectively. We deduced the success probabilities of quantum actions from the statistics reported on the physical experiments: $p_A = 0.0036$, $p_B = 0.0028$ and $p_C = 0.0071$, assuming that a round corresponds to 0.4 ms time window.

*Repeater swap protocol with distillation of Coopmans et al. [2021].* To specify the repeater swap protocol with distillation of Coopmans et al. [2021] on our network in Figure 1, we will combine the running example protocols (a) and (b). We combine creation and transmission into single basic actions, which we specify as $\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\}$ and $\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\}$ with success probabilities as in the previous example. The goal is to distribute entanglement between end nodes $A$ and $B$, by swapping Bell pairs $A{\sim}C$ and $B{\sim}C$ at node $C$, as in (a). In this scenario, however, before performing the swap, we improve the quality (also called fidelity) of entangled states $A{\sim}C$ and $B{\sim}C$ with a round of distillation. (A practitioner writing a protocol knows what is the appropriate number of distillation rounds required to improve state's quality above the required threshold.) To this end we require a network that is capable of handling at least one end-to-end Bell pair $A{\sim}B$, and two of each $A{\sim}C$ and $B{\sim}C$. For distillation of $A{\sim}C$, we repeatedly attempt to create two copies of $A{\sim}C$, and then distill them into a fresh $A{\sim}C$. To ensure that the subprotocol indeed returns an improved $A{\sim}C$, the distill action is called only after two Bell pairs were successfully created, which we specify as,

$$e_d^{(\beta)} \quad \text{where} \quad e_d = \left((\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\}) \circ (\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\})\right)^{(\beta_2)} ; \mathsf{di}\langle A{\sim}C\rangle_{0.5}$$

where guard $\beta_2$ checks for the absence of two copies of $A{\sim}C$, and $\beta$ checks for the absence of $A{\sim}C$.

Analogously, an improved Bell state $B{\sim}C$ is generated by the protocol with two while loops,

$$e_d'^{(\beta')} \quad \text{where} \quad e_d' = \left((\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\}) \circ (\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\})\right)^{(\beta_2')} ; \mathsf{di}\langle B{\sim}C\rangle_{0.5}$$

where guards $\beta'$ and $\beta_2'$ tests for the absences of $B \sim C$. Then, the repeater swap protocol with distillation by Coopmans et al. [2021] is specified below as:

$$\left(e_d^{(\beta)} \parallel e_d'^{(\beta')}\right) ; \mathsf{sw}\langle A{\sim}B \mathbin{@} C\rangle_{p_C}$$

## 6 Evaluation

In this section we evaluate the PBKAT reasoning capabilities using the examples presented earlier. We also compare our tool with the state-of-the-art framework BellKAT. Our results are presented in Table 1. We are primarily interested in success probabilities (probabilities of generating the

Table 1. Performance measurements for PBKAT and BellKAT tools using the examples from the earlier sections. We report memory, execution time, and the size of the output $O = [\![p]\!](\emptyset)$ (note, $O \in \mathcal{C}(\mathcal{M}(\mathrm{BP}))$ for PBKAT and $O \in \mathcal{P}(\mathcal{M}(\mathrm{BP}))$ for BellKAT). For PBKAT we also report the probability (or probability range) of successful generation of the desired Bell pair. As BellKAT does not support conditionals it cannot express many PBKAT protocols, we mark those with "–".

| | | **PBKAT** | | | | **BellKAT** | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Protocol** | **Goal** | **Memory** | **Time** | $\lvert O\rvert$ | $p$(**Goal**) | **Memory** | **Time** | $\lvert O\rvert$ | **Goal?** |
| §3(a) | $\{\!\{A{\sim}B\}\!\}$ | 6 MiB | <1 s | 2 | 27.2% | 6 MiB | <1 s | 4 | maybe |
| §3(a$_1$) | $\{\!\{A{\sim}B\}\!\}$ | 6 MiB | <1 s | 1 | 27.2% | – | – | – | |
| Ex. 4.2 | $\{\!\{A{\sim}B\}\!\}$ | 6 MiB | <1 s | 2 | 10.7% | – | – | – | |
| Ex. 5.1(I), $\circ$ | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 1 | 32.4% | 6 MiB | <1 s | 4 | maybe |
| Ex. 5.1(I), $\|$ | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 2 | 32.4% | 6 MiB | <1 s | 4 | maybe |
| Ex. 5.1(II), $\circ$, 2 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 1 | 61.9% | 7 MiB | <1 s | 9 | maybe |
| Ex. 5.1(II), $\|$, 2 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 14 | 60.7–67.8% | 7 MiB | <1 s | 9 | maybe |
| Ex. 5.1(II), $\circ$, 3 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 1 | 78.2% | 8 MiB | <1 s | 16 | maybe |
| Ex. 5.1(II), $\|$, 3 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 24 MiB | 26 s | 22 | 77.4–85.2% | 8 MiB | <1 s | 16 | maybe |
| Ex. 5.1(III), 1 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 2 | 32.4% | – | – | – | |
| Ex. 5.1(III), 2 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 2 | 76.6% | – | – | – | |
| Ex. 5.1(IV), 3 iter. | $\{\!\{A{\sim}C, B{\sim}C\}\!\}$ | 6 MiB | <1 s | 2 | 92.7% | – | – | – | |
| §5.3(sw) | $\{\!\{A{\sim}B\}\!\}$ | 736 MiB | 6 s | 1 | 0.41% | – | – | – | |
| §5.3(di), outer | $\{\!\{A{\sim}B\}\!\}$ | 2263 MiB | 15 s | 1 | <0.01% | – | – | – | |
| §5.3(di), inner | $\{\!\{A{\sim}B\}\!\}$ | 1178 MiB | 8 s | 1 | 0.11% | – | – | – | |
| §5.3(di), mixed | $\{\!\{A{\sim}B\}\!\}$ | 8271 MiB | 377 s | 1 | 0.02% | – | – | – | |

"Goal" multiset of Bell pairs) and amount of nondeterminism (reflected in the number of generators $|O|$ in the convex set produced by PBKAT). Additionally, we show memory usage and runtime. All our experiments were carried out on a laptop with an Intel® Core™ i7-10850H CPU 2.70 GHz and 16 GiB of RAM, and had as an initial state the empty multiset $\emptyset$.

*Basic examples and optimization.* For the first three protocols of Table 1, we note that they require negligible computational resources, and the conditional clean-up in protocol (a$_1$) reduces the output nondeterminism ($|O|$), keeping the success probability intact. Next, we showcase the use of the PBKAT tool for protocol optimization in Example 5.1. Our tool does not perform optimizations automatically, but supports an automated quantitative analysis of different protocol versions, verifying that one has better entanglement generation rate than the others. In entries (I) and (II), protocols differ in only two aspects: number of iterations and restrictions put on the scheduler by using ordered ($\circ$) vs. parallel composition ($\|$). Not surprisingly, additional iterations increase the probability of success, while the use of $\|$ adds more nondeterminism. What is notable is the added flexibility of nondeterminism improving probability of success; e.g., compare $\|$ vs $\circ$ for 3 iterations: while the minimum probability is 1% lower, the maximum probability is 7% higher. The above finding highlights the necessity of combining probability and non-determinism. Protocols (III) and (IV) add conditionals to (I) and (II), in order to ensure that only those Bell pairs get created that are still missing. Results show that such a change both improves the probability of success and reduces nondeterminism. These examples reflect the PBKAT's trade-off between more nondeterminism and increased resource requirements, a common phenomenon in verification space.

*Comparison with BellKAT.* The last three columns of Table 1 show the results of running BellKAT on PBKAT protocols that only use syntax supported by BellKAT; in particular, all the protocols that use conditionals are not expressible in BellKAT. Let us consider first, the 3-iteration $\parallel$ protocol (II), BellKAT is using much less computational resources compared to PBKAT, which is expected since BellKAT does not have to track probabilities or compute convex hulls.

BellKAT's limitation is the lack of probabilistic reasoning. All the actions that have non-trivial success probability (neither 0 nor 1) are treated equally as nondeterministically failing ones. Thus, when asked about successful generation of the "Goal" Bell paris, BellKAT can only say "maybe". Note that all the analysis that can be performed by BellKAT can also be performed by PBKAT by simply assuming probability 0.5 for nondeterministically failing actions and collecting the multisets of Bell pairs that have non-zero probability, albeit using more resources for tracking probabilities.

*Case study and scalability.* The final four rows of performance measurements in Table 1 deal with the case study examples from Section 5.3. For the repeater swap protocol of Pompili et al. [2021], PBKAT tool computes the probability of generating $\{\!\{A{\sim}B\}\!\}$ equal to $\approx$ 0.4%. The computation requires memory in the order of hundreds of MiB and takes a few seconds to complete. In the long version of the paper we manually validate the result using probability theory. For the repeater swap protocol with distillation of Coopmans et al. [2021], we consider different variants based on the number of loop iterations. In Section 5.3 we presented the protocol as having nested while loops: the inner loop followed by distillation (exit conditions $\beta_2$ and $\beta_2'$) and the outer loop followed by a swap (exit conditions $\beta$ and $\beta'$). The variants that we evaluated use different values for the number $n_i$ of inner and $n_o$ outer iterations, while keeping the total number of rounds before swap $n_o \cdot (n_i + 1)$ equal to 450, which is consistent with the timeout in the repeater swap experiment. Specifically, for the "inner" variant we choose $n_i = 449$, $n_o = 1$, for the "outer" variant we choose $n_i = 1$, $n_o = 225$, and for the "mixed" variant we use a combination of the two $n_i = 49$, $n_o = 9$. As the last three rows of Table 1 demonstrate, the inner variant results in higher probability of success (0.11%) than outer and mixed (<0.01% and 0.02%, respectively). From the performance and scalability standpoint, we see that PBKAT is able to compute the success probabilities within few minutes on a standard laptop. We note that the current PBKAT implementation is a proof-of-concept, and no attempt has been made in optimizing its performance.

## 7 Conclusion and Future Work

The realization of quantum networks will enable large-scale applications of quantum communication, but significant research and engineering efforts are still required for them to reach full functionality. Our work provides a step in this direction, by enabling specification and reasoning about quantum network protocols. The language we propose, PBKAT, can express entanglement distribution protocols in a realistic way. We tackle the protocol complexity arising from the combination of probabilistic and nondeterministic behaviors, inherent to the way quantum networks operate, with a rigorous semantics designed for quantitative analysis. PBKAT's usefulness is showcased by expressing and evaluating a number of real-world quantum network protocols.

Our work provides many new avenues for future work, including (i) the enhancement of PBKAT to consider other features of quantum communication, and (ii) the generalization of PBKAT to other domains. For (i), enhancements could be adding a notion of time-evolution to the semantics in order to capture decoherence and extending PBKAT to handle quantum states other than Bell pairs, such as single qubits or tripartite W and GHZ states [Dür et al. 2000]. For (ii), generalization can be achieved by extracting a coequationally defined language model for PBKAT and applying them to other systems exhibiting round behavior (e.g., bulk-synchronous parallel model [Valiant 1990] or hardware design [Halbwachs et al. 1991]).

## References

Amar Abane, Michael Cubeddu, Van Sy Mai, and Abdella Battou. 2024. Entanglement Routing in Quantum Networks: A Comprehensive Survey. arXiv:2408.01234 https://arxiv.org/abs/2408.01234

Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. *SIGPLAN Notices* 49, 1 (2014), 113–126. https://doi.org/10.1145/2578855.2535862

John Stewart Bell. 1964. On the Einstein Podolsky Rosen Paradox. *Physics Physique Fizika* 1, 3 (1964), 195–200. https://doi.org/10.1103/PhysicsPhysiqueFizika.1.195

Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. 2021. The Theory of Traces for Systems with Nondeterminism and Probability. In *Proceedings of the 34th ACM/IEEE Symposium on Logic in Computer Science*. 1–14. https://doi.org/10.1109/LICS.2019.8785673

Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. 2022. The Theory of Traces for Systems with Nondeterminism, Probability, and Termination. *Logical Methods in Computer Science* 18 (2022), 1–66. Issue 2. https://doi.org/10.46298/lmcs-18(2:21)2022

Hans Jürgen Briegel, Wolfgang Dür, Juan Ignacio Cirac, and Peter Zoller. 1998. Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication. *Physical Review Letters* 81, 26 (1998), 5932–5935. https://doi.org/10.1103/PhysRevLett.81.5932

Anita Buckley, Pavel Chuprikov, Rodrigo Otoni, Robert Soulé, Robert Rand, and Patrick Eugster. 2024. An Algebraic Language for Specifying Quantum Networks. *Proceedings of the ACM on Programmming Languages* 8, PLDI (2024), 1–23. https://doi.org/10.1145/3656430

Christophe Chareton, Sébastien Bardin, François Bobot, Valentin Perrelle, and Benoît Valiron. 2021. An automated deductive verification framework for circuit-building quantum programs. *Lect. Notes Comput. Sci.* 12648 (2021), 148–177. https://doi.org/10.1007/978-3-030-72019-3_6

Christophe Chareton, Sébastien Bardin, Dongho Lee, Benoît Valiron, Renaud Vilmart, and Zhaowei Xu. 2022. Formal Methods for Quantum Programs: A Survey. arXiv:2109.06493

Tim Coopmans, Robert Knegjens, Axel Dahlberg, David Maier, Loek Nijsten, Julio de Oliveira Filho, Martijn Papendrecht, Julian Rabbie, Filip Rozpędek, Matthew Skrzypczyk, Leon Wubben, Walter de Jong, Damian Podareanu, Ariana Torres-Knoop, David Elkouss, and Stephanie Wehner. 2021. NetSquid, a NETwork Simulator for QUantum Information using Discrete events. *Communications Physics* 4, 164 (2021), 1–15. https://doi.org/10.1038/s42005-021-00647-8

Axel Dahlberg and Stephanie Wehner. 2018. SimulaQron—a simulator for developing quantum internet software. *Quantum Science and Technology* 4, 1 (sep 2018), 015001. https://doi.org/10.1088/2058-9565/aad56e

Stephen DiAdamo, Janis Nötzel, Benjamin Zanger, and Mehmet Mert Beşe. 2021. QuNetSim: A Software Framework for Quantum Networks. *IEEE Transactions on Quantum Engineering* 2 (2021), 1–12. https://doi.org/10.1109/TQE.2021.3092395

C. Delle Donne, M. Iuliano, B. van der Vecht, G. M. Ferreira, H. Jirovská, T. J. W. van der Steenhoven, A. Dahlberg, M. Skrzypczyk, D. Fioretto, M. Teller, P. Filippov, A. R.-P. Montblanch, J. Fischer, H. B. van Ommen, N. Demetriou, D. Leichtle, L. Music, H. Ollivier, I. te Raa, W. Kozlowski, T. H. Taminiau, P. Pawełczak, T. E. Northup, R. Hanson, and S. Wehner. 2025. An Operating System for Executing Applications on Quantum Network Nodes. *Nature* 639, 8054 (2025), 321–328. https://doi.org/10.1038/s41586-025-08704-w

Wolfgang Dür, Guifre Vidal, and Juan Cirac. 2000. Three Qubits can be Entangled in Two Inequivalent Ways. *Physical Review A* 62, 6 (2000), 1–12. https://doi.org/10.1103/PhysRevA.62.062314

Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. 2016. Probabilistic NetKAT. In *Proceedings of the 25th European Symposium on Programming Languages and Systems*. 282–309. https://doi.org/10.1007/978-3-662-49498-1_12

Laszlo Gyongyosi and Sandor Imre. 2022. Advances in the Quantum Internet. *Commun. ACM* 65, 8 (2022), 52–63. https://doi.org/10.1145/3524455

Nicolas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud. 1991. The Synchronous Data Flow Programming Language LUSTRE. *Proc. IEEE* 79, 9 (1991), 1305–1320. https://doi.org/10.1109/5.97300

Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu, and Michael Hicks. 2021. A verified optimizer for Quantum circuits. *Proc. ACM Program. Lang.* 5, POPL, Article 37 (2021), 29 pages. https://doi.org/10.1145/3434318

Jessica Illiano, Marcello Caleffi, Antonio Manzalini, and Angela Sara Cacciapuoti. 2022. Quantum Internet Protocol Stack: A Comprehensive Survey. *Computer Networks* 213, 109092 (2022), 1–26. https://doi.org/10.1016/j.comnet.2022.109092

Bart Jacobs. 2008. Coalgebraic Trace Semantics for Combined Possibilitistic and Probabilistic Systems. *Electronic Notes in Theoretical Computer Science* 203, 5 (2008), 131–152. https://doi.org/10.1016/j.entcs.2008.05.023 Proceedings of the 9th Workshop on Coalgebraic Methods in Computer Science.

Keith Kenemer. 2024. Error Correction in Quantum Networks. Aliro Technologies. https://www.aliroquantum.com/blog/an-overview-of-quantan-overview-of-quantum-error-correction-in-entanglement-based-networksum-error-correction-in-entanglement-based-networks

C. M. Knaut, A. Suleymanzade, Y.-C. Wei, D. R. Assumpcao, P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, M. Sutula, G. Baranes, N. Sinclair, C. De-Eknamkul, D. S. Levonian, M. K. Bhaskar, H. Park, M. Lončar, and M. D. Lukin. 2024. Entanglement of Nanophotonic Quantum Memory Nodes in a Telecom Network. *Nature* 629, 8012 (2024), 573–578. https://doi.org/10.1038/s41586-024-07252-z

Wojciech Kozlowski and Stephanie Wehner. 2019. Towards Large-Scale Quantum Networks. In *Proceedings of the 6th ACM International Conference on Nanoscale Computing and Communication*. 1–7. https://doi.org/10.1145/3345312.3345497

Wojciech Kozlowski, Stephanie Wehner, Rodney Van Meter, Bruno Rijsman, Angela Sara Cacciapuoti, Marcello Caleffi, and Shota Nagayama. 2023. Architectural Principles for a Quantum Internet. RFC 9340 (https://www.rfc-editor.org/info/rfc9340).

Marco Lewis, Sadegh Soudjani, and Paolo Zuliani. 2023. Formal Verification of Quantum Programs: Theory, Tools, and Challenges. *ACM Transactions on Quantum Computing* 5, 1, Article 1 (2023), 35 pages. https://doi.org/10.1145/3624483

Yuan Li, Hao Zhang, Chen Zhang, Tao Huang, and F. Richard Yu. 2024. A Survey of Quantum Internet Protocols From a Layered Perspective. *IEEE Communications Surveys & Tutorials* 26, 3 (2024), 1606–1634. https://doi.org/10.1109/COMST.2024.3361662

Zhonghui Li, Kaiping Xue, Jian Li, Lutong Chen, Ruidong Li, Zhaoying Wang, Nenghai Yu, David S. L. Wei, Qibin Sun, and Jun Lu. 2023. Entanglement-Assisted Quantum Networks: Mechanics, Enabling Technologies, Challenges, and Research Directions. *IEEE Communications Surveys & Tutorials* 25, 4 (2023), 2133–2189. https://doi.org/10.1109/COMST.2023.3294240

Jian-Long Liu, Xi-Yu Luo, Yong Yu, Chao-Yang Wang, Bin Wang, Yi Hu, Jun Li, Ming-Yang Zheng, Bo Yao, Zi Yan, Da Teng, Jin-Wei Jiang, Xiao-Bing Liu, Xiu-Ping Xie, Jun Zhang, Qing-He Mao, Xiao Jiang, Qiang Zhang, Xiao-Hui Bao, and Jian-Wei Pan. 2024. Creation of Memory–Memory Entanglement in a Metropolitan Quantum Network. *Nature* 629, 8012 (2024), 579–585. https://doi.org/10.1038/s41586-024-07308-0

Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer Nature. https://doi.org/10.1007/b138392

Matteo Mio, Ralph Sarkis, and Valeria Vignudelli. 2021. Combining Nondeterminism, Probability, and Termination: Equational and Metric Reasoning. In *Proceedings of the 36th ACM/IEEE Symposium on Logic in Computer Science*. 1–14. https://doi.org/10.1109/LICS52264.2021.9470717

Michael W. Mislove. 2006. On Combining Probability and Nondeterminism. *Electronic Notes in Theoretical Computer Science* 162 (2006), 261–265. https://doi.org/10.1016/j.entcs.2005.12.113 Proceedings of the Workshop "Essays on Algebraic Process Calculi" (APC 25).

David Monniaux. 2000. Abstract Interpretation of Probabilistic Semantics. In *Static Analysis*, Jens Palsberg (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 322–339. https://doi.org/10.1007/978-3-540-45099-3_17

Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic Predicate Transformers. *ACM Transactions on Programming Languages and Systems* 18, 3 (1996), 325–353. https://doi.org/10.1145/229542.229547

Michael A. Nielsen and Isaac L. Chuang. 2011. *Quantum Computation and Quantum Information*. Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

Mihir Pant, Hari Krovi, Don Towsley, Leandros Tassiulas, Liang Jiang, Prithwish Basu, Dirk Englund, and Saikat Guha. 2019. Routing Entanglement in the Quantum Internet. *npj Quantum Information* 5, 25 (2019), 1–9.

S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden. 2020. Advances in Quantum Cryptography. *Advances in Optics and Photonics* 12, 4 (2020), 1012–1236. https://doi.org/10.1364/AOP.361502

M. Pompili, S. L. N. Hermans, S. Baier, H. K. C. Beukers, P. C. Humphreys, R. N. Schouten, R. F. L. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse, S. Wehner, and R. Hanson. 2021. Realization of a Multinode Quantum Network of Remote Solid-State Qubits. *Science* 372, 6539 (2021), 259–264. https://doi.org/10.1126/science.abg1919

Cristian Prisacariu. 2010. Synchronous Kleene Algebra. *The Journal of Logic and Algebraic Programming* 79, 7 (2010), 608–635. https://doi.org/10.1016/j.jlap.2010.07.009

Nicolas Privault. 2018. *Understanding Markov Chains*. Springer Singapore. https://doi.org/10.1007/978-981-13-0659-4

Julian Rabbie, Kaushik Chakraborty, Guus Avis, and Stephanie Wehner. 2022. Designing Quantum Networks Using Preexisting Infrastructure. *npj Quantum Information* 8, 5 (2022), 1–12. https://doi.org/10.1038/s41534-021-00501-3

Robert Rand and Steve Zdancewic. 2016. Models for Probabilistic Programs with an Adversary. Workshop on Probabilistic Programming Semantics. http://pps2016.soic.indiana.edu/2015/12/16/adversaries

Wojciech Różowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva. 2023. Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 261)*, Kousha Etessami, Uriel Feige, and Gabriele Puppis (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 136:1–136:20. https://doi.org/10.4230/LIPIcs.ICALP.2023.136

Ryosuke Satoh, Michal Hajdušek, Naphan Benchasattabuse, Shota Nagayama, Kentaro Teramoto, Takaaki Matsuo, Sara Ayman Metwalli, Takahiko Satoh, Shigeya Suzuki, and Rodney Van Meter. 2022. QuISP: a Quantum Internet Simulation Package. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 354–365. https://doi.org/10.1109/QCE53715.2022.00048

Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. 2019a. Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time. *Proceedings of the ACM on Programming Languages* 4, POPL (2019), 1–28. https://doi.org/10.1145/3371129

Steffen Smolka, Praveen Kumar, David M. Kahn, Nate Foster, Justin Hsu, Dexter Kozen, and Alexandra Silva. 2019b. Scalable Verification of Probabilistic Networks. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 190–203. https://doi.org/10.1145/3314221.3314639

Arian J. Stolk, Kian L. van der Enden, Marie-Christine Slater, Ingmar te Raa-Derckx, Pieter Botma, Joris van Rantwijk, J. J. Benjamin Biemond, Ronald A. J. Hagen, Rodolf W. Herfst, Wouter D. Koek, Adrianus J. H. Meskers, René Vollmer, Erwin J. van Zwet, Matthew Markham, Andrew M. Edmonds, J. Fabian Geus, Florian Elsen, Bernd Jungbluth, Constantin Haefner, Christoph Tresp, Jürgen Stuhler, Stephan Ritter, and Ronald Hanson. 2024. Metropolitan-Scale Heralded Entanglement of Solid-State Qubits. *Science Advances* 10, 44 (2024). https://doi.org/10.1126/sciadv.adp6442

Don Towsley. 2021. The Quantum Internet: Recent Advances and Challenges. Keynote at the *29th IEEE International Conference on Network Protocols*. https://icnp21.cs.ucr.edu

Dominique Unruh. 2019. Quantum Hoare Logic with Ghost Variables. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '19)*. IEEE Computer Society, Los Alamitos, CA, USA, Article 47, 13 pages. https://doi.org/10.1109/LICS.2019.8785779 arXiv:1902.00325

Leslie G. Valiant. 1990. A Bridging Model for Parallel Computation. *Commun. ACM* 33, 8 (1990), 103–111. https://doi.org/10.1145/79173.79181

Rodney Van Meter and Joe Touch. 2013. Designing Quantum Repeater Networks. *IEEE Communications Magazine* 51, 8 (2013), 64–71. https://doi.org/10.1109/MCOM.2013.6576340

Rodney Van Meter, Joe Touch, and Clare Horsman. 2011. Recursive Quantum Repeater Networks. *Progress in Informatics* 8 (2011), 65–79. https://doi.org/10.2201/NiiPi.2011.8.8

Daniele Varacca and Glynn Winskel. 2006. Distributing Probability over Non-Determinism. *Mathematical Structures in Computer Science* 16, 1 (2006), 87–113. https://doi.org/10.1017/S0960129505005074

Jana Wagemaker, Marcello Bonsangue, Tobias Kappé, Jurriaan Rot, and Alexandra Silva. 2019. Completeness and Incompleteness of Synchronous Kleene Algebra. In *Proceedings of the 13th International Conference on Mathematics of Program Construction*. 385–413. https://doi.org/10.1007/978-3-030-33636-3_14

Chonggang Wang, Akbar Rahman, Ruidong Li, Melchior Aelmans, and Kaushik Chakraborty. 2023. *Application Scenarios for the Quantum Internet*. Technical Report. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-qirg-quantum-internet-use-cases/16

Stephanie Wehner, David Elkouss, and Ronald Hanson. 2018. Quantum Internet: A Vision for the Road Ahead. *Science* 362, 6412 (2018), 1–9. https://doi.org/10.1126/science.aam9288

Xiaoliang Wu, Alexander Kolar, Joaquin Chung, Dong Jin, Tian Zhong, Rajkumar Kettimuthu, and Martin Suchara. 2021. SeQUeNCe: a customizable discrete-event simulator of quantum networks. *Quantum Science and Technology* 6, 4 (2021), 045027. https://doi.org/10.1088/2058-9565/ac22f6

Mingsheng Ying. 2012. Floyd–Hoare Logic for Quantum Programs. *ACM Transactions on Programming Languages and Systems* 33, 6, Article 19 (January 2012), 19 pages. https://doi.org/10.1145/2049706.2049708

Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. 2023. CoqQ: Foundational Verification of Quantum Programs. *Proceedings of the ACM on Programming Languages* 7, POPL, Article 29 (January 2023), 29 pages. https://doi.org/10.1145/3571222 arXiv:2207.11350

Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An Applied Quantum Hoare Logic. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '19)*. Association for Computing Machinery, New York, NY, USA, 1149–1162. https://doi.org/10.1145/3314221.3314584

Noam Zilberstein, Dexter Kozen, Alexandra Silva, and Joseph Tassarotti. 2024. A Demonic Outcome Logic for Randomized Nondeterminism. arXiv:2410.22540 https://arxiv.org/pdf/2410.22540

## A Soundness

This section outlines the proof of Theorems 4.1 and 4.2 stating that PBKAT semantics is correct. The fact that $e \equiv f$ implies $[\![e]\!] = [\![f]\!]$ follows from the soundness of PBKAT rules (listed in Figure 10) with respect to the abstract semantics (i.e., $\vdash e \equiv f \Rightarrow I(e) = I(f)$ proven in Lemma A.1), and the observation that $[\![e]\!](a)$ is a set of subdistributions for any $a \in \mathcal{M}\mathrm{BP}$ (see Lemma A.2).

Lemma A.1. *PBKAT rules are sound with respect to the abstract semantics. This means that the implication* $\vdash e \equiv f \Rightarrow I(e) = I(f)$ *is valid.*

Proof. We prove the soundness of three most interesting rules (as the other are proven analogously or are trivial), by unfolding the definitions of PBKAT's abstract language model in Section 4.2.

- Associativity of the conditional $(e +_\alpha f) +_\beta g \equiv e +_{\alpha\wedge\beta} (f +_\beta g)$ holds as,

$$\begin{aligned} I((e +_\alpha f) +_\beta g) &= I(e +_\alpha f) +_{\mathrm{sat}(\beta)} I(g) = \mathrm{sat}(\beta) \diamond I(e +_\alpha f) \cup \mathrm{sat}(\bar\beta) \diamond I(g) \\ &= \mathrm{sat}(\beta) \diamond (\mathrm{sat}(\alpha) \diamond I(e) \cup \mathrm{sat}(\bar\alpha) \diamond I(f)) \cup \mathrm{sat}(\bar\beta) \diamond I(g) \\ &= \mathrm{sat}(\beta) \cap \mathrm{sat}(\alpha) \diamond I(e) \cup \mathrm{sat}(\beta) \cap \mathrm{sat}(\bar\alpha) \diamond I(f) \cup \mathrm{sat}(\bar\beta) \diamond I(g) \\ I(e +_{\alpha\wedge\beta} (f +_\beta g)) &= I(e) +_{\mathrm{sat}(\alpha\wedge\beta)} I(f +_\beta g) = \mathrm{sat}(\alpha \wedge \beta) \diamond I(e) \cup \mathrm{sat}(\overline{\alpha \wedge \beta}) \diamond I(f +_\beta g) \\ &= \mathrm{sat}(\alpha) \cap \mathrm{sat}(\beta) \diamond I(e) \cup \mathrm{sat}(\bar\alpha) \cup \mathrm{sat}(\bar\beta) \diamond I(f +_\beta g) \\ &= \mathrm{sat}(\alpha) \cap \mathrm{sat}(\beta) \diamond I(e) \cup \mathrm{sat}(\bar\alpha) \cup \mathrm{sat}(\beta) \diamond I(f) \cup \mathrm{sat}(\bar\beta) \diamond I(g) \end{aligned}$$

  since $(\mathrm{sat}(\bar\alpha) \cup \mathrm{sat}(\bar\beta)) \cap \mathrm{sat}(\beta) = \mathrm{sat}(\bar\alpha) \cap \mathrm{sat}(\beta)$ and $(\mathrm{sat}(\bar\alpha) \cup \mathrm{sat}(\bar\beta)) \cap \mathrm{sat}(\bar\beta) = \mathrm{sat}(\bar\beta)$.
- The property of the while loop $(e +_\alpha 1)^{(\beta)} \equiv (\alpha\,;e)^{(\beta)}$ follows from:

$$\begin{aligned} I\big((e +_\alpha 1)^{(\beta)}\big) &= I(e +_\alpha 1)^{(\mathrm{sat}(\beta))} = \cup_{n\geq 0}(\mathrm{sat}(\beta) \diamond I(e +_\alpha 1))^n \diamond \mathrm{sat}(\bar\beta) \\ &= \cup_{n\geq 0}(\mathrm{sat}(\beta) \cap \mathrm{sat}(\alpha) \diamond I(e))^n \diamond \mathrm{sat}(\bar\beta) \\ I\big((\alpha\,;e)^{(\beta)}\big) &= I(\alpha\,;e)^{(\mathrm{sat}(\beta))} = \cup_{n\geq 0}(\mathrm{sat}(\beta) \diamond I(\alpha\,;e))^n \diamond \mathrm{sat}(\bar\beta) \end{aligned}$$

  where equality in row 2 holds as $\mathrm{sat}(\bar\alpha) \cap \mathrm{sat}(\beta) \cap \mathrm{sat}(\alpha) = \emptyset = \mathrm{sat}(\beta) \cap \mathrm{sat}(\bar\alpha) \cap \mathrm{sat}(\bar\beta)$.
- Synchrony property $(h\,;e) \parallel (k\,;f) \equiv (h \parallel k)\,;(e \parallel f)$ holds for single round protocols $h, k$:

$$I((h\,;e) \parallel (k\,;f)) = I(h\,;e) \parallel I(k\,;f) = (I(h) \diamond I(e)) \parallel (I(k) \diamond I(f))$$

  contains words $(S_0 \cap S_0')(\Omega_1 \parallel \Omega_1')(S_1 \cap S_1') \dots (\Omega_n \parallel \Omega_n')(S_n \cap S_n')\Omega_{n+1}'S_{n+1}' \dots \Omega_{n'}'S_{n'}'$ if $S_i \cap S_i' \neq \emptyset$ for all $0 \leq i \leq n$, which are composed from $S_0\Omega_1S_1 \in I(h)$, $S_0'\Omega_1'S_1' \in I(k)$ and $S_1\Omega_2S_2 \dots \in I(e)$, $S_1'\Omega_2'S_2' \dots \in I(f)$. These words are the same as the words in:

$$I((h \parallel k)\,;(e \parallel f)) = I(h \parallel k) \diamond I(e \parallel f) = (I(h) \parallel I(k)) \diamond (I(e) \parallel I(f))$$

□

Lemma A.2. *For a PBKAT expression e, the elements in* $[\![e]\!]$ *are Markov subkernels (in other words, for an input multiset* $a \in \mathcal{M}\mathrm{BP}$*, the elements in* $[\![e]\!](a)$ *are subdistributions). Specifically, for every subdistribution* $\mu \in [\![e^{(\alpha)}]\!](a)$ *and multiset* $a' \in \mathcal{M}\mathrm{BP}$*, the probability* $\mu(a')$ *is a limit of some increasing sequence* $\mu_n(a')$ *for* $\mu_n \in [\![(e +_\alpha 1)^n\,;\bar\alpha]\!](a)$*.*

Proof. The abstract semantics builds from an expression $e$ the language $I(e)$, containing guarded strings of sets of the form $w = \Omega_0 S_1 \Omega_1 \dots S_n \Omega_n \in \mathcal{PM}\mathrm{BP}\,;(\mathcal{P}\Pi\,;\mathcal{PM}\mathrm{BP})^*$. The semantics of $e$ for an input multiset $a \in \mathcal{M}\mathrm{BP}$ is defined in Section 4.2 as follows:

$$[\![e]\!](a) = [\![I(e)]\!]_{\mathrm{ex}}(a) = \mathrm{WMS}\,(\sum_{w\in I(e)} 1\cdot[\![w]\!]_{\mathrm{ex}}(a)) = \{\sum_{w\in I(e)} \delta \mid \delta \in [\![w]\!]_{\mathrm{ex}}(a) \neq 0\}$$

The key observation that we need to prove for the lemma to hold, is that for any $a \in \mathcal{M}\mathrm{BP}$, the elements of $[\![e]\!](a)$ are indeed subdistributions.

First we check that elements of $[\![w]\!]_{\mathrm{ex}}(a)$ are subdistributions. Atomic actions in every set $S_i$ that constitutes the string $w$ come from concurrent basic actions (which we explicitly calculate in Section 4.3 using the rules PNet-Ord and PNet-Prl). The only atomic actions in $S_i$ that are not mutually exclusive under execution are those that capture nondeterministic choice, which we represent with distinct subdistributions in the convex set of possible outputs (cf. Example 4.2).

Next we check that $\sum_{w \in I(e)} \delta$ for $\delta \in [\![w]\!]_{\mathrm{ex}}(a)$ is also a subdistribution. This follows from the *determinancy property* of $I(-)$ defined in Lemma 4.1. For example, $\{S_0\Omega_1S_1,\ S_0\Omega_1S_1',\ S_0'\Omega_1'S_1'\}$ and $\{S_0\Omega_1S_1,\ S_0'\}$ satisfy the determinacy property where $S_0 \cap S_0' = \emptyset$ and $S_1 \cap S_1' = \emptyset$, while $\{S_0\Omega_1S_1,\ S_0\}$ and $\{S_0\Omega_1S_1,\ S_0\Omega_1'S_1'\}$ do not. Availing the inductive definition of the abstract semantics $I(-)$ in Section 4.2, it suffices to check the determinancy property is preserved by the operators that are interpreted as union, i.e., conditionals and while loops. The following language,

$$I(e' +_\alpha e'') = \mathsf{sat}(\alpha) \diamond I(e') \cup \mathsf{sat}(\bar{\alpha}) \diamond I(e'')$$

is indeed a disjoint union since $\mathsf{sat}(\alpha) \cap \mathsf{sat}(\bar{\alpha}) = \emptyset$. Similarly it holds,

$$\begin{aligned} I(e^{(\alpha)}) &= \bigcup_{n \geq 0} (\mathsf{sat}(\alpha) \diamond I(e))^n \diamond \mathsf{sat}(\bar{\alpha}) \\ &= \{\mathsf{sat}(\bar{\alpha})\} \cup \mathsf{sat}(\alpha) \diamond I(e) \diamond \mathsf{sat}(\bar{\alpha}) \cup \mathsf{sat}(\alpha) \diamond I(e) \diamond \mathsf{sat}(\alpha) \diamond I(e) \diamond \mathsf{sat}(\bar{\alpha}) \cup \cdots \end{aligned}$$

which also preserves the determinancy property.

□

## B A Note on Probabilities and the Expected Number of Attempts until Success

In this section $f$ and $g$ denote actions that succeed with probabilities $p$ and $q$.

Lemma B.1. *Assume that action $f$ succeeds with probability $p$. The probability that the action succeeds after $n$ rounds is $1 - (1 - p)^n$. The expected number of attempts until the first success is $1/p$.*

Proof. It is well known that the expectation of a geometric variable is $1/p$, where $p$ is the success factor for the variable, which follows from the sum:

$$\sum_{k=1}^{\infty} kp(1 - p)^{k-1} = 1/p$$

□

Lemma B.2. *Assume that actions $f$ and $g$ succeed with probabilities $p$ and $q$ independently, and in each round (i.e., attempt or trial) both actions are executed concurrently. The probability that both actions succeed (for the first time) after $k$ rounds is:*

$$q(1 - q)^{k-1}(1 - (1 - p)^k) + p(1 - p)^{k-1}(1 - (1 - q)^k) - pq(1 - p)^{k-1}(1 - q)^{k-1}$$

Proof. We need to consider all possibilities where one action succeeds at the $k$-th attempt and the other action succeeds at the $k$-th attempt or earlier. □

Lemma B.3. *Assume that actions $f$ and $g$ succeed with probabilities $p$ and $q$ independently, and in each round both actions are executed concurrently. The probability that both actions succeed after $n$ rounds is:*

$$1 - (1 - p)^n - (1 - q)^n + (1 - p)^n(1 - q)^n = (1 - (1 - p)^n)(1 - (1 - q)^n)$$

Proof. This follows from summing the probabilities in Lemma B.2 for $k$ ranging from 1 to $n$. □

Lemma B.4. *Assume that actions $f$ and $g$ succeed with probabilities $p$ and $q$ independently, and in each round both actions are executed concurrently. The expected number of attempts until the first success of both actions is:*

$$\frac{1}{p} + \frac{1}{q} - \frac{1}{p+q-pq}$$

Proof. We denote by $r(k)$ the probability that both actions succeed for the first time after $k$ rounds, which we calculated in Lemma B.2. Then, $\sum_{k=1}^{\infty} k\, r(k) = \frac{1}{p} + \frac{1}{q} - \frac{1}{p+q-pq}$. □

Lemma B.5. *Assume that an action $f$ succeeds with probability $p$, and in each round 2 attempts are executed concurrently. The probability of 2 successes after $n$ rounds is:*

$$1 - (1-p)^{2n} - 2np(1-p)^{2n-1}$$

Proof. This follows from subtracting from 1 the probabilities of zero and one successful attempts. □

Lemma B.6. *Let actions $f$ and $g$ succeed with probabilities $p$ and $q$.*

- *Assume that $f$ and $g$ are independent, and that the action $f$ is called until it succeeds, followed by the action $g$ being called until it succeeds. The expected number of attempts required for both actions to succeed is $1/p + 1/q$.*
- *Assume that the action $f$ is called until it succeeds followed by the action $g$ called once; and this is repeated until also $g$ succeeds. The expected number of attempts required for both actions to succeed is $1/p \times 1/q$.*

Proof. Define independent random variables $X$ and $Y$ to be the number of attempts to the first success of $f$ and $g$, respectively. Since the expectation of a sum of random variables is the sum of their expectations, it holds $\mathbb{E}(X+Y) = \mathbb{E}(X) + \mathbb{E}(Y)$, and similarly $\mathbb{E}(XY) = \mathbb{E}(X)\,\mathbb{E}(Y)$. □

The stochastic matrix capturing the limiting behavior of the while loop $[\![ e^{(\beta)} ]\!]$ in Section 5.2.

Lemma B.7. *After reordering the states so that absorbing states appear in the first block of the matrix $\mathcal{B}[\![e]\!] = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix}$, the $n$-th unrolling of the while loop $e^{(\beta)}$ is represented by the matrix:*

$$\mathcal{B}[\![(e +_\beta 1)^{(n)}]\!] = \begin{bmatrix} I & 0 \\ A & E_{22}^{n} \end{bmatrix} \quad \textit{for} \quad A = (I + E_{22} + \cdots + E_{22}^{n-1})E_{21}$$

## C Manual Validation of Probabilities for Protocols from Case-study in Section 5.3

We manually validate the results in Section 6 for case-studies using probability theory in Appendix B.

### C.1 Repeater swap experiment realized by Pompili et al. [2021] in Section 5.3

To show that these probabilities indeed match the reported data we calculate the expectations of required iterations using the lemmas in Appendix B: The expected creations of Bell pairs $A{\sim}C$ and $B{\sim}C$ occur after $1/p_A \approx 280$ and $1/p_B \approx 360$ attempts, thus the expected number of rounds required for successful (concurrent) creation of both Bell pairs is 480 by Lemma B.4, which is in line with the experimental timeout of 450 attempts reported in [Pompili et al. 2021]. In fact, the probability of creating both $A{\sim}C$ and $B{\sim}C$ in 450 rounds is approximately 0.58 by Lemma B.3, that – followed by one attempt of swapping with success probability $p_A \approx 1/140 \approx 0.0071$ – yields the probability of creating $\{\!\{A{\sim}B\}\!\}$ to be 0.004, which perfectly matches the result computed by our tool in Table 1 (cf. row corresponding to (sw) protocol). Furthermore, since the expected number of required swapping attempts is $1/p_C \approx 140$, the expected time for $A{\sim}B$ creation – assuming time window 0.4 ms for each round – is $1/0.58 \times 450 \times 140 \times 0.0004$ s $\approx 40$ s (cf. Lemma B.6), as reported by Pompili et al. [2021].

### C.2 Repeater swap protocol with distillation of Coopmans et al. [2021] in Section 5.3

To compare the two protocols in Section 5.3, we specify the repeater swap protocol with distillation using the same actions and iteration numbers as in the repeater swap experiment. For simplicity we assume that distillation succeeds with the maximum possible probability $p_d = 1/2$ (recall that distillation is inherently probabilistic, see Section 2). Next we adjust the specification of the repeater swap protocol with distillation in Section 5.3, by replacing the nested while loops with fixed numbers $n_i$ and $n_o$ of guarded iterations:

$$
\begin{aligned}
e_d &= \left((\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\}) \circ (\emptyset \triangleright_{p_A} \{\!\{A{\sim}C\}\!\}) +_{\beta_2} 1\right)^{(n_i)} ; (\mathsf{di}\langle A{\sim}C\rangle_{p_d} +_{\bar{\beta}_2} \mathsf{drop}\langle A{\sim}C\rangle) \\
e'_d &= \left((\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\}) \circ (\emptyset \triangleright_{p_B} \{\!\{B{\sim}C\}\!\}) +_{\beta'_2} 1\right)^{(n_i)} ; (\mathsf{di}\langle B{\sim}C\rangle_{p_d} +_{\bar{\beta}'_2} \mathsf{drop}\langle B{\sim}C\rangle)
\end{aligned}
$$

Then the repeater swap protocol with $n_o$ distillation attempts is specified below as:

$$
\left(\left(e_d +_{\beta} 1\right)^{(n_o)} \parallel \left(e'_d +_{\beta'} 1\right)^{(n_o)}\right) ; \mathsf{sw}\langle A{\sim}B \,@\, C\rangle_{p_C}
$$

In Section 6 we evaluated three variants of the protocol for different $n_i$ and $n_o$ ("inner" variant $n_i = 449$, $n_o = 1$, "outer" variant $n_i = 1$, $n_o = 225$, and "mixed" variant $n_i = 49$, $n_o = 9$), computed by our tool (cf. the rows corresponding to (di) protocol in Table 1).

We apply Lemma B.5 to calculate that after 449 rounds the probabilities of creating $\{\!\{A{\sim}C, A{\sim}C\}\!\}$ and $\{\!\{B{\sim}C, B{\sim}C\}\!\}$ are 0.8335 and 0.7161, respectively, which followed by a distillation attempt (with ideal $p_d = 1/2$) yield 0.4168 and 0.358 probability of success for each subprotocol $e_d$ and $e'_d$. Then, after performing the swapping at $C$, the probability of creating end-to-end $A{\sim}B$ is $0.4168 \times 0.358 \times 0.0071$ in the order of 0.1%. We thus expect the protocol to succeed after $1/0.001 \times 449 \times 0.0004$ seconds which is $\approx 3$ minutes (cf. Lemma B.6).

Similarly, by Lemma B.5 the probabilities of creating $\{\!\{A{\sim}C, A{\sim}C\}\!\}$ and $\{\!\{B{\sim}C, B{\sim}C\}\!\}$ after 49 inner rounds are 0.0491 and 0.0312, and after distillation, $e_d$ and $e'_d$ succeed with probabilities 0.0245 and 0.0156, respectively. After 9 concurrent iterations of $e_d$ and $e'_d$ we succeed with probability $0.2004 \times 0.1320$ by Lemma B.3, which after performing the swap yields 0.0002 success probability. We therefore expect the protocol to succeed after $1/0.0002 \times 49 \times 9 \times 0.0004$ seconds or $\approx 15$ minutes.

It turns out that the strategy of $n_i = 1$ and $n_o = 225$ is the least efficient since the probability of generating end-to-end $A{\sim}B$ is $5 \times 10^{-6}$, which would require two months.

## D Outputs

### D.1 Protocol (a)

*Output.*

$$
\begin{aligned}
\Lbag\, &\{\{\}\} \times \frac{6901}{25000} + \{\{A\sim B\}\} \times \frac{1701}{6250} + \{\{A\sim C\}\} \times \frac{423}{1250} + \{\{B\sim C\}\} \times \frac{567}{5000}, \\
&\{\{\}\} \times \frac{7351}{25000} + \{\{A\sim B\}\} \times \frac{1701}{6250} + \{\{A\sim C\}\} \times \frac{243}{1250} + \{\{B\sim C\}\} \times \frac{1197}{5000} \,\Rbag
\end{aligned}
$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 2 | $[\frac{1701}{6250}, \frac{1701}{6250}]$ | 6 | 0.002 |

*BellKAT Output.*

$$\begin{gathered}\{\{\{\}\},\\ \{\{A \sim B\}\},\\ \{\{A \sim C\}\},\\ \{\{B \sim C\}\}\}\end{gathered}$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 4 | | 6 | 0.002 |

## D.2 Protocol (a1)

*Output.*

$$\llparenthesis \{\{\}\} \times \frac{4549}{6250} + \{\{A \sim B\}\} \times \frac{1701}{6250} \rrparenthesis$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[\frac{1701}{6250}, \frac{1701}{6250}]$ | 6 | 0.002 |

## D.3 Protocol (a2)

*Output.*

$$\begin{gathered}\llparenthesis \{\{\}\} \times \frac{163}{675} + \{\{A \sim B\}\} \times \frac{8}{75} + \{\{A \sim C\}\} \times \frac{8}{15} + \{\{B \sim C\}\} \times \frac{2}{45} + \{\{C \sim C\}\} \times \frac{2}{27},\\ \{\{\}\} \times \frac{253}{675} + \{\{A \sim B\}\} \times \frac{8}{75} + \{\{A \sim C\}\} \times \frac{8}{45} + \{\{B \sim C\}\} \times \frac{4}{15} + \{\{C \sim C\}\} \times \frac{2}{27} \rrparenthesis\end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 2 | $[\frac{8}{75}, \frac{8}{75}]$ | 6 | 0.002 |

## D.4 Protocol 5.1.I (parallel)

*Output.*

$$\begin{gathered}\llparenthesis \{\{\}\} \times \frac{127}{1000} + \{\{A \sim C\}\} \times \frac{117}{250} + \{\{A \sim C, B \sim C\}\} \times \frac{81}{250} + \{\{B \sim C\}\} \times \frac{81}{1000},\\ \{\{\}\} \times \frac{181}{1000} + \{\{A \sim C\}\} \times \frac{81}{250} + \{\{A \sim C, B \sim C\}\} \times \frac{81}{250} + \{\{B \sim C\}\} \times \frac{171}{1000} \rrparenthesis\end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 2 | $[\frac{81}{250}, \frac{81}{250}]$ | 6 | 0.002 |

*BellKAT Output.*

$$\begin{gathered}\{\{\{\}\},\\ \{\{A \sim C\}\},\\ \{\{A \sim C, B \sim C\}\},\\ \{\{B \sim C\}\}\}\end{gathered}$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 4 | | 6 | 0.004 |

## D.5 Protocol 5.1.I (ordered)

*Output.*

$$\Lbag \{\!\{\}\!\} \times \frac{127}{1000} + \{\!\{A \sim C\}\!\} \times \frac{117}{250} + \{\!\{A \sim C, B \sim C\}\!\} \times \frac{81}{250} + \{\!\{B \sim C\}\!\} \times \frac{81}{1000} \Rbag$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[\frac{81}{250}, \frac{81}{250}]$ | 6 | 0.001 |

*BellKAT Output.*

$$\begin{gathered}\{\{\!\{\}\!\},\\ \{\!\{A \sim C\}\!\},\\ \{\!\{A \sim C, B \sim C\}\!\},\\ \{\!\{B \sim C\}\!\}\}\end{gathered}$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 4 | | 6 | 0.004 |

## D.6 Protocol 5.1.II (parallel, two iterations)

*Output.*

$$\begin{gathered}
\llparenthesis \{\{\}\} \times \frac{16129}{1000000} + \{\{A\sim C\}\} \times \frac{9243}{31250} + \{\{A\sim C, B\sim C\}\} \times \frac{40581}{62500} + \{\{B\sim C\}\} \times \frac{38799}{1000000}, \\
\{\{\}\} \times \frac{16129}{1000000} + \{\{A\sim C\}\} \times \frac{9243}{31250} + \{\{A\sim C, B\sim C\}\} \times \frac{4131}{6250} + \{\{B\sim C\}\} \times \frac{5427}{200000}, \\
\{\{\}\} \times \frac{16129}{1000000} + \{\{A\sim C\}\} \times \frac{42237}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{75897}{125000} + \{\{B\sim C\}\} \times \frac{38799}{1000000}, \\
\{\{\}\} \times \frac{16129}{1000000} + \{\{A\sim C\}\} \times \frac{42237}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{15471}{25000} + \{\{B\sim C\}\} \times \frac{5427}{200000}, \\
\{\{\}\} \times \frac{22987}{1000000} + \{\{A\sim C\}\} \times \frac{31041}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{81729}{125000} + \{\{B\sim C\}\} \times \frac{74853}{1000000}, \\
\{\{\}\} \times \frac{22987}{1000000} + \{\{A\sim C\}\} \times \frac{31041}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{84807}{125000} + \{\{B\sim C\}\} \times \frac{50229}{1000000}, \\
\{\{\}\} \times \frac{22987}{1000000} + \{\{A\sim C\}\} \times \frac{17343}{62500} + \{\{A\sim C, B\sim C\}\} \times \frac{19521}{31250} + \{\{B\sim C\}\} \times \frac{74853}{1000000}, \\
\{\{\}\} \times \frac{22987}{1000000} + \{\{A\sim C\}\} \times \frac{17343}{62500} + \{\{A\sim C, B\sim C\}\} \times \frac{4131}{6250} + \{\{B\sim C\}\} \times \frac{7713}{200000}, \\
\{\{\}\} \times \frac{22987}{1000000} + \{\{A\sim C\}\} \times \frac{39951}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{75897}{125000} + \{\{B\sim C\}\} \times \frac{50229}{1000000}, \\
\{\{\}\} \times \frac{22987}{1000000} + \{\{A\sim C\}\} \times \frac{39951}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{15471}{25000} + \{\{B\sim C\}\} \times \frac{7713}{200000}, \\
\{\{\}\} \times \frac{32761}{1000000} + \{\{A\sim C\}\} \times \frac{27783}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{81729}{125000} + \{\{B\sim C\}\} \times \frac{91143}{1000000}, \\
\{\{\}\} \times \frac{32761}{1000000} + \{\{A\sim C\}\} \times \frac{27783}{125000} + \{\{A\sim C, B\sim C\}\} \times \frac{84807}{125000} + \{\{B\sim C\}\} \times \frac{66519}{1000000}, \\
\{\{\}\} \times \frac{32761}{1000000} + \{\{A\sim C\}\} \times \frac{7857}{31250} + \{\{A\sim C, B\sim C\}\} \times \frac{19521}{31250} + \{\{B\sim C\}\} \times \frac{91143}{1000000}, \\
\{\{\}\} \times \frac{32761}{1000000} + \{\{A\sim C\}\} \times \frac{7857}{31250} + \{\{A\sim C, B\sim C\}\} \times \frac{40581}{62500} + \{\{B\sim C\}\} \times \frac{66519}{1000000} \rrparenthesis
\end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 14 | $[\frac{75897}{125000}, \frac{84807}{125000}]$ | 6 | 0.027 |

*BellKAT Output.*

$$\begin{gathered}
\{\{\{\}\}, \\
\{\{A\sim C\}\}, \\
\{\{A\sim C, B\sim C\}\}, \\
\{\{A\sim C, B\sim C, B\sim C\}\}, \\
\{\{A\sim C, A\sim C\}\}, \\
\{\{A\sim C, A\sim C, B\sim C\}\}, \\
\{\{A\sim C, A\sim C, B\sim C, B\sim C\}\}, \\
\{\{B\sim C\}\}, \\
\{\{B\sim C, B\sim C\}\}\}
\end{gathered}$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 9 |  | 7 | 0.008 |

## D.7 Protocol 5.1.II (parallel, three iterations)

*Output.*

$$\begin{gathered}
\llparenthesis\{\{\}\} \times \frac{2048383}{1000000000} + \{\{A \sim C\}\} \times \frac{39228813}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{206518167}{250000000} + \{\{B \sim C\}\} \times \frac{14963697}{1000000000}, \\
\{\{\}\} \times \frac{2048383}{1000000000} + \{\{A \sim C\}\} \times \frac{39228813}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{208521459}{250000000} + \{\{B \sim C\}\} \times \frac{6950529}{1000000000}, \\
\{\{\}\} \times \frac{2048383}{1000000000} + \{\{A \sim C\}\} \times \frac{52149123}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{193597857}{250000000} + \{\{B \sim C\}\} \times \frac{14963697}{1000000000}, \\
\{\{\}\} \times \frac{2048383}{1000000000} + \{\{A \sim C\}\} \times \frac{52149123}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{195601149}{250000000} + \{\{B \sim C\}\} \times \frac{6950529}{1000000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{34040889}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{208176723}{250000000} + \{\{B \sim C\}\} \times \frac{28210203}{1000000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{34040889}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{212151879}{250000000} + \{\{B \sim C\}\} \times \frac{12309579}{1000000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{38648169}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{208521459}{250000000} + \{\{B \sim C\}\} \times \frac{8402139}{1000000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{43965819}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{198251793}{250000000} + \{\{B \sim C\}\} \times \frac{28210203}{1000000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{50231169}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{38830671}{50000000} + \{\{B \sim C\}\} \times \frac{3908511}{200000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{51568479}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{193597857}{250000000} + \{\{B \sim C\}\} \times \frac{16415307}{1000000000}, \\
\{\{\}\} \times \frac{2919349}{1000000000} + \{\{A \sim C\}\} \times \frac{51568479}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{195601149}{250000000} + \{\{B \sim C\}\} \times \frac{8402139}{1000000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{31893867}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{208381977}{250000000} + \{\{B \sim C\}\} \times \frac{34735977}{1000000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{31893867}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{212943573}{250000000} + \{\{B \sim C\}\} \times \frac{16489593}{1000000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{33213357}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{212151879}{250000000} + \{\{B \sim C\}\} \times \frac{14378409}{1000000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{36894807}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{209076957}{250000000} + \{\{B \sim C\}\} \times \frac{11952297}{1000000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{41232357}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{199043487}{250000000} + \{\{B \sim C\}\} \times \frac{34735977}{1000000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{49403637}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{38830671}{50000000} + \{\{B \sim C\}\} \times \frac{4322277}{200000000}, \\
\{\{\}\} \times \frac{4160647}{1000000000} + \{\{A \sim C\}\} \times \frac{49403637}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{196568127}{250000000} + \{\{B \sim C\}\} \times \frac{11952297}{1000000000}, \\
\{\{\}\} \times \frac{5929741}{1000000000} + \{\{A \sim C\}\} \times \frac{30714471}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{208381977}{250000000} + \{\{B \sim C\}\} \times \frac{37684467}{1000000000}, \\
\{\{\}\} \times \frac{5929741}{1000000000} + \{\{A \sim C\}\} \times \frac{30714471}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{212943573}{250000000} + \{\{B \sim C\}\} \times \frac{19438083}{1000000000}, \\
\{\{\}\} \times \frac{5929741}{1000000000} + \{\{A \sim C\}\} \times \frac{40052961}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{199043487}{250000000} + \{\{B \sim C\}\} \times \frac{37684467}{1000000000}, \\
\{\{\}\} \times \frac{5929741}{1000000000} + \{\{A \sim C\}\} \times \frac{40052961}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{203605083}{250000000} + \{\{B \sim C\}\} \times \frac{19438083}{1000000000}\rrparenthesis
\end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 22 | $[\frac{193597857}{250000000}, \frac{212943573}{250000000}]$ | 24 | 26.152 |

*BellKAT Output.*

$$
\begin{gathered}
\{\{\!\{\}\!\}, \\
\{\!\{A \sim C\}\!\}, \\
\{\!\{A \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, B \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, B \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, A \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, A \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, A \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, A \sim C, B \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{B \sim C\}\!\}, \\
\{\!\{B \sim C, B \sim C\}\!\}, \\
\{\!\{B \sim C, B \sim C, B \sim C\}\!\}\}
\end{gathered}
$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 16 | | 8 | 0.015 |

## D.8 Protocol 5.1.II (ordered, two iterations)

*Output.*

$$
⦇\{\!\{\}\!\} \times \frac{16129}{1000000} + \{\!\{A \sim C\}\!\} \times \frac{42237}{125000} + \{\!\{A \sim C, B \sim C\}\!\} \times \frac{15471}{25000} + \{\!\{B \sim C\}\!\} \times \frac{5427}{200000}⦈
$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[\frac{15471}{25000}, \frac{15471}{25000}]$ | 6 | 0.002 |

*BellKAT Output.*

$$
\begin{gathered}
\{\{\!\{\}\!\}, \\
\{\!\{A \sim C\}\!\}, \\
\{\!\{A \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, B \sim C\}\!\}, \\
\{\!\{A \sim C, A \sim C, B \sim C, B \sim C\}\!\}, \\
\{\!\{B \sim C\}\!\}, \\
\{\!\{B \sim C, B \sim C\}\!\}\}
\end{gathered}
$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 9 | | 7 | 0.008 |

## D.9 Protocol 5.1.II (ordered, three iterations)

*Output.*

$$\llparenthesis \{\{\}\} \times \frac{2048383}{1000000000} + \{\{A \sim C\}\} \times \frac{52149123}{250000000} + \{\{A \sim C, B \sim C\}\} \times \frac{195601149}{250000000} + \{\{B \sim C\}\} \times \frac{6950529}{1000000000} \rrparenthesis$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[\frac{195601149}{250000000}, \frac{195601149}{250000000}]$ | 6 | 0.003 |

*BellKAT Output.*

$$\begin{gathered} \{\{\{\}\}, \\ \{\{A \sim C\}\}, \\ \{\{A \sim C, B \sim C\}\}, \\ \{\{A \sim C, B \sim C, B \sim C\}\}, \\ \{\{A \sim C, B \sim C, B \sim C, B \sim C\}\}, \\ \{\{A \sim C, A \sim C\}\}, \\ \{\{A \sim C, A \sim C, B \sim C\}\}, \\ \{\{A \sim C, A \sim C, B \sim C, B \sim C\}\}, \\ \{\{A \sim C, A \sim C, B \sim C, B \sim C, B \sim C\}\}, \\ \{\{A \sim C, A \sim C, A \sim C\}\}, \\ \{\{A \sim C, A \sim C, A \sim C, B \sim C\}\}, \\ \{\{A \sim C, A \sim C, A \sim C, B \sim C, B \sim C\}\}, \\ \{\{A \sim C, A \sim C, A \sim C, B \sim C, B \sim C, B \sim C\}\}, \\ \{\{B \sim C\}\}, \\ \{\{B \sim C, B \sim C\}\}, \\ \{\{B \sim C, B \sim C, B \sim C\}\}\} \end{gathered}$$

*BellKAT Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 16 | | 8 | 0.016 |

## D.10 Protocol 5.1.III (one iteration)

*Output.*

$$\begin{gathered} \llparenthesis \{\{\}\} \times \frac{127}{1000} + \{\{A \sim C\}\} \times \frac{117}{250} + \{\{A \sim C, B \sim C\}\} \times \frac{81}{250} + \{\{B \sim C\}\} \times \frac{81}{1000}, \\ \{\{\}\} \times \frac{181}{1000} + \{\{A \sim C\}\} \times \frac{81}{250} + \{\{A \sim C, B \sim C\}\} \times \frac{81}{250} + \{\{B \sim C\}\} \times \frac{171}{1000} \rrparenthesis \end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 2 | $[\frac{81}{250}, \frac{81}{250}]$ | 6 | 0.002 |

## D.11 Protocol 5.1.III (two iterations)

*Output.*

$$\begin{gathered}\llparenthesis\{\{\}\} \times \frac{16129}{1000000} + \{\{A \sim C\}\} \times \frac{100503}{500000} + \{\{A \sim C, B \sim C\}\} \times \frac{1915569}{2500000} + \{\{B \sim C\}\} \times \frac{83187}{5000000}, \\ \{\{\}\} \times \frac{32761}{1000000} + \{\{A \sim C\}\} \times \frac{78327}{500000} + \{\{A \sim C, B \sim C\}\} \times \frac{1915569}{2500000} + \{\{B \sim C\}\} \times \frac{221787}{5000000}\rrparenthesis\end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 2 | $[\frac{1915569}{2500000}, \frac{1915569}{2500000}]$ | 6 | 0.004 |

## D.12 Protocol 5.1.IV (three iterations)

*Output.*

$$\begin{gathered}\llparenthesis\{\{\}\} \times \frac{2048383}{1000000000} + \{\{A \sim C\}\} \times \frac{68352687}{1000000000} + \{\{A \sim C, B \sim C\}\} \times \frac{23174702721}{25000000000} + \{\{B \sim C\}\} \times \frac{65270529}{25000000000}, \\ \{\{\}\} \times \frac{5929741}{1000000000} + \{\{A \sim C\}\} \times \frac{58002399}{1000000000} + \{\{A \sim C, B \sim C\}\} \times \frac{23174702721}{25000000000} + \{\{B \sim C\}\} \times \frac{226993779}{25000000000}\rrparenthesis\end{gathered}$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 2 | $[\frac{23174702721}{25000000000}, \frac{23174702721}{25000000000}]$ | 6 | 0.007 |

## D.13 Protocol 5.3 (Pompili)

*Output.*

$$\llparenthesis\{\{\}\} \times 0.6271846044 + \{\{A \sim B\}\} \times 0.0040853247 + \{\{A \sim C\}\} \times 0.2272810780 + \{\{B \sim C\}\} \times 0.1414489929\rrparenthesis$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[0.0040853247, 0.0040853247]$ | 736 | 6.828 |

## D.14 Protocol 5.3 (Coopmans, outer)

*Output.*

$$\llparenthesis\{\{\}\} \times 0.9976640046 + \{\{A \sim B\}\} \times 0.0000000091 + \{\{A \sim C\}\} \times 0.0014556579 + \{\{B \sim C\}\} \times 0.0008803284\rrparenthesis$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[0.0000000091, 0.0000000091]$ | 2263 | 14.885 |

## D.15 Protocol 5.3 (Coopmans, inner)

*Output.*

$$\llparenthesis\{\{\}\} \times 0.5225711584 + \{\{A \sim B\}\} \times 0.0010594623 + \{\{A \sim C\}\} \times 0.2675510822 + \{\{B \sim C\}\} \times 0.2088182972\rrparenthesis$$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | $[0.0010594623, 0.0010594623]$ | 1178 | 7.516 |

### D.16 Protocol 5.3 (Coopmans, mixed)

*Output.*

$\llparenthesis \{\!\{\}\!\} \times 0.7203643533 + \{\!\{A \sim B\}\!\} \times 0.0001877432 + \{\!\{A \sim C\}\!\} \times 0.1739095845 + \{\!\{B \sim C\}\!\} \times 0.1055383191 \rrparenthesis$

*Stats.*

| Num Generators | Success probability | Memory (MiB) | Time(s) |
|---|---|---|---|
| 1 | [0.0001877432, 0.0001877432] | 8271 | 376.528 |

## E Examples

PBKAT's semantics faithfully captures the difference, between $e \parallel f$ and $e \circ f$ in Example 5.1, as illustrated in Figure 12.

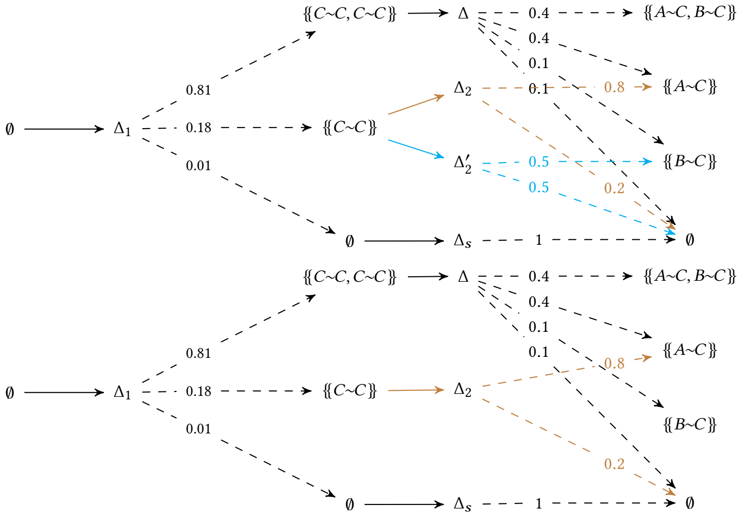


Fig. 12. Execution traces (progressing left to right) of protocols $e \parallel f$ and $e \circ f$ in Example 5.1. The choices between $\mathrm{tr}\langle C \to A{\sim}C\rangle_{0.8}$ or $\mathrm{tr}\langle C \to B{\sim}C\rangle_{0.5}$ are shown as brown or cyan branches, respectively.

Received 01 January 20XX