Causal Dependencies in Parallel Composition of Stochastic Processes

Ling Cheung* and Martijn Hendriks**

Department of Computer Science, University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{lcheung,martijnh}@cs.ru.nl

Abstract. We present some new perspectives on an old problem: compositionality of trace-style semantics for stochastic processes. The initial step is a more fine-grained analysis of how parallel composition of stochastic processes can be defined in a meaningful way. We make a connection between the notion of adversary models and the formal definitions of stochastic system types and parallel composition operators. In particular, we focus on causal dependencies arising from the so-called strong adversaries and argue that trace-style semantic compositionality cannot be achieved under strong adversaries.
We identify a structural feature of stochastic processes called "invisible probabilistic branching" and illustrate its connection with strong adversaries. Based on these observations, we introduce a new system type together with appropriate notions of observable behavior and parallel composition. We prove a finite approximation theorem for behaviors and use that to prove semantic compositionality. Finally, we present a model of the Chor-Israeli-Li consensus protocol in our new framework.

1 Motivations and Conceptual Analysis

The study of concurrency concerns the development and analysis of system components executing in parallel. Such components may communicate with each other via some well-defined mechanism, for instance, shared variables or asynchronous message passing. Apart from this communication, the evolution of each individual component is independent from the behavior of other components. This independence assumption underlies the so-called *modular* approaches to system development, which are popular in various stages of the development process including design, implementation and analysis.

To put modular methods on firm grounds, it is essential that our formal models come with a convincing notion of parallel composition, one that reflects

* Supported by DFG/NWO bilateral cooperation project Validation of Stochastic Systems (VOSS2)
** Supported by EU IST project IST-2001-35304 Advanced Methods for Timed Systems (AMETIST)

our intuitions about independent evolution of parallel components. This requirement usually takes the form of a semantic compositionality[1] theorem, relating the parallel composition operator to the relevant notion of system behavior.

Compositionality issues have been well-studied for non-deterministic models of distributed computation. For stochastic processes, however, a careful study of parallel composition is somewhat harder to find[2]. In particular, many articles that do present parallel composition operators for stochastic processes fail to provide an explanation of the intuitive *meaning* of these operators.

The present paper aims to provide some initial steps in this direction, with a focus on causal dependencies among behaviors of parallel, stochastic components. Aside from the conceptual analysis, we present a new system type for modeling stochastic processes, together with a parallel composition operator and a notion of trace-style observable behavior. Much attention will be devoted to justifying our view on parallel composition and to arguing that our technical definitions correctly capture those ideas.

We do not expect the reader to agree with every philosophical comment we make in this paper. Our primary goal is to convey the following: (i) there are legitimate foundational issues regarding the use of randomization in our models of parallel computation, and (ii) it is a non-trivial task to fully understand the philosophical implications of one's mathematical definitions.

1.1 Interleaving Semantics and Random Choices

A fundamental idea in concurrency theory is the *interleaving* interpretation of parallel composition. Namely, (i) every atomic step of a composite system is an atomic step of one of its components (or more in case of synchronization); and (ii) the scheduling among components is arbitrary. This view seems perfectly adequate when the basic building blocks of a system are deterministic algorithms, in which case uncertainty arises *only* from choices among distributed components. As it turns out, non-determinism provides a very natural means to modeling such choices, because it captures the idea of complete lack of information (or "ignorance").

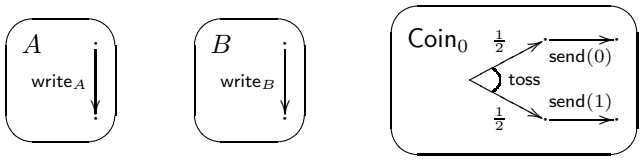
The story is different when our basic algorithms contain stochastic elements. Typically, a component is assumed to have access to some random source which, upon request, draws an element from some pre-specified probability distribution. Under this extension, we are confronted with *two* distinct types of uncertainties: (i) *random* choices within individual components and (ii) *non-deterministic* choices among distributed components. We illustrate this by introducing our running example.

Example 1. Suppose two automata A and B are racing to write their own preference values into a shared register, which can be written to only once, i.e., all

[1] We distinguish "semantic" compositionality from the "syntactic" version, which simply states that a parallel composition operator is definable for a particular formal framework, without reference to the associated notion of observable behavior.

[2] We refer our reader to [SdV04] for a comparative survey on this topic.

subsequent write operations are ignored. Moreover, suppose there is a third automaton Coin_0 which tosses a fair coin and announces the result by sending a one-bit message across a network to an unspecified recipient. ■



One may ask: what possible behavior can/will the composite $A\|B\|\,\mathsf{Coin}_0$ exhibit? There are many possible execution sequences, including, among others, $\mathsf{write}_A\,.\,\mathsf{toss}\,.\,\mathsf{send}(0).\,\mathsf{write}_B$ and $\mathsf{toss}\,.\,\mathsf{write}_B\,.\,\mathsf{send}(1).\,\mathsf{write}_A$. How should one assign probabilities to these sequences? Notice the temporal ordering of the operations write_A and write_B is significant, because it determines the content of the shared register at the end of an execution.

In [DAHK98,Seg95], parallel composition is defined so that the composite of two purely stochastic processes may contain non-determinism, reflecting the lack of information in resolving the choice between these processes. Once the composite process is constructed, one resorts to the so-called *adversaries* (or *schedulers*) in order to remove all remaining non-deterministic choices. We refer to this approach as "compose-and-schedule". Since adversaries are usually history dependent, the outcome of interleaving can become causally connected to previous events. This is the focus of the present paper.

There are of course other approaches to parallel composition of stochastic processes, for example, parameterized composition [JLY01,DAHK98], real-time delay [WSS94] and synchronous execution [dAHJ01,vGSS95]. Due to space constraints, we summarize these approaches in Appendix A and explain why we find them unsatisfactory.

1.2 Adversary Models

In the setting of randomized distributed algorithms, one often speaks of *adversary models* [Asp03,AB04]. An adversary is a function from a finite execution of a system to an available next transition (or a convex combination of such transitions, depending on the particular formalism). Such a function resolves all non-deterministic choices in a system, so that random choices are the only remaining uncertainties. The probability of each execution is then completely determined by the sequence of coin tosses generating that execution.

Adversaries are important for analyzing *worst case* scenarios. That is, one can image that the parties running a protocol are collaborating towards a certain goal (e.g. termination), while the adversary tries to prevent the parties from reaching that goal. Therefore, a stronger adversary model (i.e., adversaries with more knowledge and power) gives a higher degree of confidence in the claim that formal correctness proofs in fact guarantee correct behavior in real life systems.

Figure 1 below depicts an execution of $A\|B\|\,\mathsf{Coin}_0$ from Example 1. This execution is induced by a so-called *strong* adversary, which can observe entire

histories of all system components, including internal states. Indeed, the adversary chooses between write_A and write_B based on the outcome of toss.

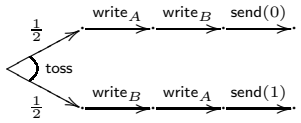


Fig. 1. An execution of $A \| B \| \mathsf{Coin}_0$

Let us consider some implications of the existence of such an adversary, call it $\mathcal{A}$. Let α denote the action sequence $\mathsf{toss} \,.\, \mathsf{write}_A \,.\, \mathsf{write}_B \,.\, \mathsf{send}(0)$ and assume α is observed during a run controlled by $\mathcal{A}$. Then both events $E_H :=$ "coin comes up heads" and $E_A :=$ "write_A precedes write_B" occur and the probability of E_A is 1 at the time of occurrence of E_H. Moreover, if E_H *had not* occurred, then $E_T :=$ "coin comes up tails" *would have* occurred and the probability of E_A *would have* been much lower (in this case, 0). The conjunction of these statements turns out to be exactly how one defines the *causal dependence* between E_H and E_A in the probabilistic counterfactual theory of causation[3].

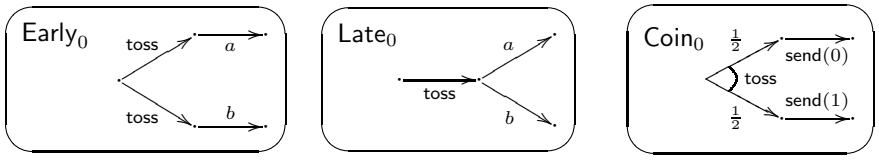
How are we then to provide some justification for this causal dependency? A logical explanation would be that the event E_H somehow produced an effect on the computation environment, so that the system dynamics dictates "A runs relatively faster than B." Notice, this unknown effect must have been produced *prior* to the $\mathsf{send}(0)$ message from Coin_0, because in α the subsequence $\mathsf{write}_A \,.\, \mathsf{write}_B$ precedes $\mathsf{send}(0)$. Moreover, this effect is *not* specified in the formal description of Coin_0.

We now draw some conclusions from our analysis. First, if one insists on the existence of such an adversary $\mathcal{A}$, one must give up the independence assumption we mentioned at the onset of this paper. In particular, the evolution of $A \| B$ is dependent upon the evolution of Coin_0 in some unspecified manner; that is, a manner not described by the action synchronization mechanism of communication. Second, if one chooses to adopt the strong adversary model, then $\mathcal{A}$ is definable and hence one must also give up the aforementioned independence assumption. Finally, since the unknown effect of the previous paragraph is not expressed in terms of actions, it is not captured as part of observable behavior under a trace-style semantics. Consequently, a semantic compositionality theorem would not be possible. Many well-known counterexamples echo this last claim and we provide one of them in Example 2.

Example 2. Consider Early_0 and Late_0 as depicted below, where they synchronize with Coin_0 on the action toss. These two are equivalent in a typical trace-style semantics. Yet, in $\mathsf{Late}_0 \,\|\, \mathsf{Coin}_0$, it is possible to define an execution in which

[3] A detailed discussion on probabilistic causation is clearly beyond the scope of this paper. We refer to [Hit02] for an overview.

the action a is totally correlated with $\mathsf{send}(0)$ and b with $\mathsf{send}(1)$. This is *not* possible for $\mathsf{Early}_0 \parallel \mathsf{Coin}_0$. We refer to [LSV03,CLSV04] for more details. ■



Before moving on, we stress that many weaker forms of adversaries appear in the literature [Asp03,AB04]. Below is a non-exhaustive list.

(1) *Oblivious* adversaries, which can observe only component names.
(2) *Content-oblivious* adversaries, which cannot distinguish operations that differ only in parameter (e.g. $\mathsf{send}(0)$ and $\mathsf{send}(1)$).
(3) *Write-oblivious* adversaries, which cannot observe any value written to a shared memory until that value has been read by some process.
(4) *Local-oblivious* adversaries, which cannot observe any randomly chosen value until it is written to a shared memory.

1.3 Invisible Probabilistic Branching

Now we turn to the main objective of this paper: weakening the strong adversary model just enough to guarantee a semantic compositionality theorem. Clearly, a framework in which semantic compositionality holds is highly desirable. Moreover, our analysis in Section 1.2 shows that the strong adversary model commits one to a strong assumption: the computation environment can detect the outcome of any coin toss *as soon as* the coin lands, even if the outcome is not used in any meaningful way. For example, we can remove the $\mathsf{send}(\cdot)$ transitions from Coin_0 and still enforce the total correlation that A wins if and only if the coin comes up heads. Therefore, we seek a slightly weaker adversary model in order to avoid these irregularities.

We return to Example 1 and try to identify the structural features of Coin_0 that allow the adversary to observe internal random outcomes and to make decisions accordingly. We come with two key observations.

(1) The specification Coin_0 exhibits *invisible probabilistic branching*: there is a random choice over distinct transitions carrying the same label toss;
(2) The adversary $\mathcal{A}$ is obtained in a "compose-and-schedule" fashion (that is, $\mathcal{A}$ uses the joint history of A, B and Coin_0 in making scheduling decisions).

In fact, we have essentially made observation (2) in an earlier paper [CLSV04], where component scheduling is done in a distributed fashion[4]. That is, we assume a token structure on parallel processes in which the (unique) currently active component always chooses the next active component. This approach

[4] Distributed scheduling also appears in [BPW04] in a setting of computational cryptography. We were not aware of the connection at the time we published[CLSV04].

can be characterized as "schedule-and-compose", where local non-deterministic choices are resolved by local schedulers and global choices (i.e., inter-component choices) are resolved using the token structure.

As far as we know, observation (1) is original. In the literature (cf. overview in [SdV04]), most authors specify probabilistic transitions using discrete distributions on the Cartesian product $Act \times S$, where Act is the action alphabet and S is the state space. Take for example the *general probabilistic automata* model of [Seg95], with transition function $\Omega : S \longrightarrow \mathcal{P}(\mathsf{Disc}(Act \times S))$.

In other words, each state $s \in S$ enables a set $\Omega(s)$ of probabilistic transitions, where each transition is given by a discrete distribution on pairs of the form $\langle a, s' \rangle$. In such a setting, it is possible to specify a transition that assigns non-zero probability to some $\langle a, s_1 \rangle$ and $\langle a, s_2 \rangle$, with $s_1 \neq s_2$. This situation is precisely what we called invisible probabilistic branching in observation (1).

Guided by this insight, we introduce a variant of Ω:

$$\Delta : S \longrightarrow \mathcal{P}(Act \to ([0,1] \times S)),$$

with the additional requirement that $\pi_1 \circ f$ is a discrete distribution on Act for all $\mu \in S$ and $f \in \Delta(\mu)$. Notice we use variables μ, ν, etc. for states[5]. Such a function $f \in \Delta(\mu)$ is called a *transition bundle* from μ, representing a one-step evolution from a state specified by μ. Along this step, an action a is observed with probability $\pi_1(f(a))$ and, in that case, the resulting state is $\pi_2(f(a))$.

Clearly, given a bundle $f \in \Delta(\mu)$, we obtain a distribution on $Act \times S$ by assigning probability $\pi_1(f(a))$ to the pair $\langle a, \pi_2(f(a)) \rangle$. Thus our systems are special cases of general probabilistic automata of [Seg95]. Note that the inclusion is strict: because our transition bundles are functions, invisible probabilistic branching cannot be expressed in our system type. This is precisely our goal.

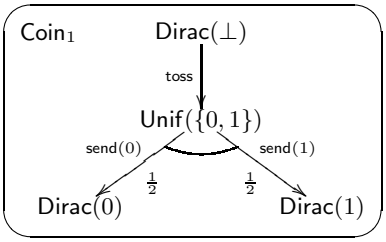


Fig. 2. Coin_1: removing invisible probabilistic branching from Coin_0

Figure 2 illustrates a modification of Coin_0 according to this new system type. The states of Coin_1 are discrete distributions on the set of all possible value assignments to state variables. Specifically, there is a boolean state variable bit, which is initially $\perp$. After the toss action, bit is assigned either 0 or 1, each

[5] We think of a state in our model as a discrete distribution on concrete states of an underlying system. This will become clearer as we discuss Coin_1 of Figure 2.

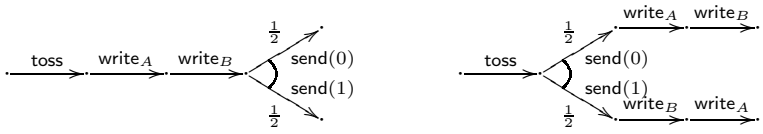
with probability $\frac{1}{2}$; hence the state is represented by the uniform distribution on $\{0,1\}$, denoted $\mathsf{Unif}(\{0,1\})$. Here $\mathsf{Dirac}(s)$ denotes the discrete distribution assigning probability 1 to the value assignment s.

1.4 Parallel Composition and Compositionality

In some literature [DAHK98,Seg95], non-deterministic choices between parallel components are resolved by randomized adversaries (i.e., they return convex combinations of available transitions). We have chosen to work with deterministic adversaries. To justify this design decision, we give some intuition in terms of possible worlds.

The underlying idea is: the outcome of interleaving among distributed components is uniquely determined in each individual world. Whenever some component performs a visible action, we transit from one world to another and hence the outcome of interleaving may differ. Notice such an action need not always affect the world in such a way that relative speeds of processes change. Yet we allow the adversary this freedom in order to strengthen our "worst case" guarantees (i.e., we quantify over more possible cases). An important, but subtle point to note: if we start from the same world and observe the same visible action, then the resulting world should be the same. This is in fact another way of understanding our exclusion of invisible probabilistic branching.

For a more concrete illustration, we consider Example 1 with Coin_0 replaced by Coin_1 of Figure 2. When the coin is tossed, the external world observes the same action toss regardless of the outcome, therefore the target world is the same. Of course we do *not* know whether A wins in this target world or B does, so in our semantics both options are considered. The left execution below depicts one of them. It may also happen that both write operations take place after Coin_1 sends an announcement. In that case, the adversary *can* use the content of the message to decide whether A or B wins. This is depicted below on the right.



As it turns out, the adversary model associated with our framework is very similar to the local-oblivious adversary model of [CIL94], in that scheduling among components is done (implicitly) by a centralized adversary which has no knowledge of "unannounced" random outcomes. The difference is, in a typical setting of randomized algorithms, each component is a purely stochastic process, whereas in our setting components may themselves contain non-determinism.

Much of our technical efforts goes to achieving a clear separation between *local* and *global* non-deterministic choices, so that all local choices are resolved using strictly local information. Thus our notion of parallel composition is also along the lines of "schedule-and-compose" of [CLSV04].

These considerations turn out to be sufficient to guarantee semantic compositionality of our trace-style behavioral preorder (Theorem 4). To our best knowledge, we are the first to achieve this under a centralized scheduling scheme.

1.5 Overview

Sections 2 and 3 introduce notational preliminaries and basic notions of probabilistic input/output automata. In Sections 4 and 5, we define the notion of execution trees and their behavioral abstraction. Section 6 describes our approximation techniques for reasoning about infinite behaviors, where a detailed development can be found in Appendix C. In Section 7 and Appendix D, we define our notion of parallel composition and prove semantic compositionality. Section 8 and Appendix E describe some preliminary modeling work on the Chor-Israeli-Li consensus protocol. Finally, concluding remarks are in Section 9.

2 Preliminaries

Given set X, a function $\mu : X \longrightarrow [0,1]$ is called a *discrete (probability) distribution* on X if $\sum_{x \in X} \mu(x) = 1$. The *support* of μ, denoted $\mathsf{Supp}(\mu)$, is the set $\{x \in X \mid \mu(x) > 0\}$. We write $\mathsf{Disc}(X)$ for the set of all discrete distributions on X. Given $x \in X$, the distribution $\mathsf{Dirac}(x)$ assigns probability 1 to x. If X is finite, then the distribution $\mathsf{Unif}(X)$ assigns probability $\frac{1}{|X|}$ to every x in X.

Given two sets X and Y, we write $X + Y$ for the *disjoint union* of X and Y and $X \times Y$ for the *Cartesian product*, where the projection maps are denoted π_i. Given an equivalence relation $\equiv$ on X, the *quotient* of X under $\equiv$ (written $X/\equiv$) is the partition of X induced by $\equiv$. The *coset* $[x]$ in this quotient is the equivalence class containing x.

The set of all *partial functions* from X to Y is denoted $X \rightharpoonup Y$. For each $f \in X \rightharpoonup Y$, we write $\mathsf{dom}(f)$ for the *domain* of f and $f(x) = \bot$ whenever $x \notin \mathsf{dom}(f)$. Given $f, g \in X \rightharpoonup Y$, we write $f \subseteq g$ whenever the graph of f is a subset of that of g.

3 Probabilistic Input/Output Automata

Throughout this paper, we fix a countable set Act of action symbols.

Definition 1. *A* probabilistic automaton (PA) *A is a triple $\langle S_A, \mu_A^0, \Delta_A \rangle$ where*

- *S_A is the set of states with the initial state $\mu_A^0 \in S_A$;*
- *$\Delta_A : S_A \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times S_A))$ is the transition function satisfying: $\pi_1 \circ f$ is a discrete distribution on Act for all $\mu \in S_A$ and $f \in \Delta_A(\mu)$.*

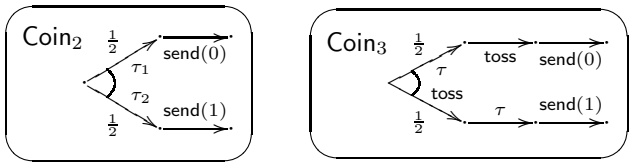
We say that f is a *transition bundle* from μ if $f \in \Delta_A(\mu)$. The set of all transition bundles in A is $\mathsf{Bun}(A) := \bigcup_{\mu \in S_A} \Delta_A(\mu)$. In this paper, we focus on *countably branching* systems: S_A and $\mathsf{Bun}(A)$ are both countable. We refer to Figure 2 for an example of a PA in our sense. Next, we introduce input/output (i/o) distinction. For that we associate an alphabet $Act_A \subseteq Act$ to each PA A.

Definition 2. *A* probabilistic i/o automaton (PIOA) *A is a PA $\langle S_A, \mu_A^0, \Delta_A\rangle$ where Act_A is partitioned into $\{I_A, O_A, H_A\}$ (*input*, *output*, and* hidden *actions, respectively) and Δ_A satisfies the two axioms below.*

1. ***I/O**. For all $\mu \in S_A$ and $f \in \Delta_A(\mu)$, one of the following holds:*
   (a) $\pi_1 \circ f = \mathsf{Dirac}(a)$ for some $a \in I_A$ (f is an input *bundle with label a),*
   (b) $\mathsf{Supp}(\pi_1 \circ f) \subseteq O_A$ (f is an output *bundle), or*
   (c) $\pi_1 \circ f = \mathsf{Dirac}(a)$ for some $a \in H_A$ (f is a hidden *bundle with label a).*
2. ***Input enabling**. For all $\mu \in S_A$ and $a \in I_A$, there exists an input bundle $f \in \Delta_A(\mu)$ with label a.*

We say that A is *closed* if I_A is empty and *open* otherwise. A few points of clarification are in order. The i/o axiom divides transition bundles into three classes: input, output and hidden. We write respectively $\mathsf{Bun}_I(A)$, $\mathsf{Bun}_O(A)$ and $\mathsf{Bun}_H(A)$. As usual, the first two are *visible* and the last two are *locally controlled*. Similar to Input/Output Automata (IOA) of Lynch and Tuttle [LT89], we have an input enabling axiom requiring acceptance of all inputs at every state. Moreover, we avoid synchronization deadlocks by requiring that every input is received with probability 1 (Clause (1a)). This can be viewed as a probabilistic version of input enabling. As a result, our parallel composition operator, unlike most others in the literature, does not involve a mechanism for normalizing probabilities (i.e., collecting and redistributing deadlock probabilities). This greatly simplifies our technical development, especially in proving compositionality (Theorem 4).

Clauses (1b) and (1c) are perhaps a bit more surprising. We explain using two variants of Coin_0. Suppose a hidden bundle may assign non-zero probability to two *distinct* hidden actions, thus leading to two *distinct* end states, as shown in Coin_2. Then Coin_2 also exhibits a form of invisible probabilistic branching and can be used to distinguish Early_0 and Late_0 of Example 2. A similar analysis applies to Coin_3, therefore we must also rule out bundles that assign non-zero probability to both hidden and output actions.



4 Executions

In a stochastic setting, a run of a process corresponds to a local scheduler that resolves all local non-deterministic choices. This is sometimes called a "probabilistic execution" [Seg95]. Such a run can be seen as a purely stochastic tree in which every node is a finite execution sequence of the underlying PA. In our case, the tree-like structure results from the fact that each transition bundle f may lead to multiple end states, one for each a in $\mathsf{Supp}(\pi_1 \circ f)$. When we introduce i/o distinction, the situation is complicated by non-deterministic choices involving inputs. Thus, we use the term "execution tree", rather than probabilistic

execution, to emphasize the fact that execution trees are not purely stochastic. As shown in [CLSV04], execution trees in a closed PIOA are precisely the probabilistic executions in the sense of [Seg95][6].

We begin by defining the notion of branches in execution trees.

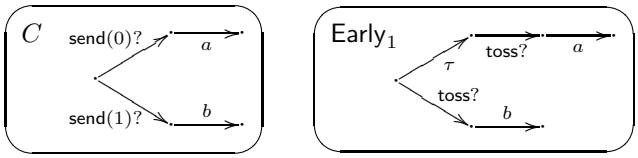
Definition 3. *Let A be a PIOA and let $\mu \in S_A$ be given. We use joint recursion to define the set of* (execution) branches *from μ, denoted $\mathsf{Bran}(\mu)$, together with the function $\mathsf{last} : \mathsf{Bran}(\mu) \longrightarrow S_A$.*

- *The length-one sequence containing μ (written $\underline{\mu}$) is in $\mathsf{Bran}(\mu)$ and is called the* empty *branch, where $\mathsf{last}(\underline{\mu}) := \mu$.*
- *If r is in $\mathsf{Bran}(\mu)$, then so is $r.f.a.\nu$, provided: (i) f is a transition bundle from $\mathsf{last}(r)$; (ii) $a \in Act_A$ and $\pi_1(f(a)) > 0$; (iii) $\nu = \pi_2(f(a))$. Moreover, $\mathsf{last}(r.f.a.\nu) := \nu$.*

We write $\mathsf{Bran}(A)$ for $\mathsf{Bran}(\mu^0)$. The likelihood *of $r \in \mathsf{Bran}(\mu)$ is the value $\Pi[r]$ defined as follows: (i) $\Pi[\underline{\mu}] := 1$, and (ii) $\Pi[r.f.a.\nu] := \Pi[r] \cdot \pi_1(f(a))$. The* trace *of r is the action sequence $\mathsf{tr}(r)$ given by: (i) $\mathsf{tr}(\underline{\mu}) := \epsilon$, and (ii) $\mathsf{tr}(r.f.a.\nu)$ equals $\mathsf{tr}(r).a$ if $a \notin H_A$ and $\mathsf{tr}(r)$ otherwise.*

Essentially, the likelihood of r is obtained by multiplying the probabilities of the sequence of events generating r. Notice we opt for the term "likelihood", rather than "probability". This is because the probabilities involved in $\Pi[r]$ are conditional upon the choices of transition bundles along r, and therefore Π is typically *not* a probability distribution over $\mathsf{Bran}(\mu)$.

As hinted above, some subtle issues complicate the definition of execution trees for open PIOAs. First we give an example of input non-determinism. Consider automaton C below, synchronizing with Coin_1 in Figure 2. (We use ? to indicate input actions.) During a run, C receives a one-bit message from Coin_1 and moves to two different states accordingly. From the perspective of C, the choice between $\mathsf{send}(0)?$ and $\mathsf{send}(1)?$ is non-deterministic, yet a reasonable notion of execution tree should include both branches in the same run.



Next, we consider automaton Early_1. The question is whether an execution tree can contain both branches of Early_1. Here the choice between $\mathsf{toss}?$ and τ can be seen as *both* (i) global, between Early_1 and its environment, and (ii) local, between the two branches within D. Since we take the "schedule-and-compose" approach of parallel composition (Section 1.4), we insist on resolving this choice locally. In other words, each run of Early_1 involves at most one branch. This is essentially an assumption that local schedulers are insensitive to the timing

[6] Our execution trees are essentially the same as pseudo probabilistic executions of [CLSV04].

of inputs relative to internal computations. This often holds in practice, where components use e.g. FIFO queues to store interrupting inputs and handle them only when internal computations are finished.

Thus we arrive at the following definition of execution trees. The symbol $\Box$ indicates complete termination of a system, whereas $\triangle$ indicates that the system is in a waiting state and may be later activated by an input. Moreover, let $\mathcal{I}$ denote the function space $I_A \to \mathsf{Bun}_I(A)$ and let $\mathcal{O}$ denote $\mathsf{Bun}_O(A) \cup \{\triangle\}$.

Definition 4. *Let A be a PIOA and let $\mu \in S_A$ be given. An* (execution) tree *from μ is a partial function $Q : \mathsf{Bran}(\mu) \rightharpoonup (\{\Box\} + \mathsf{Bun}_H(A) + (\mathcal{I} \times \mathcal{O}))$ satisfying the conditions below.*

1. *For all $r \in \mathsf{dom}(Q)$,*
   (a) *$Q(r) \in \mathsf{Bun}_H(A)$ implies $Q(r) \in \Delta_A(\mathsf{last}(r))$;*
   (b) *$Q(r) \in \mathcal{I} \times \mathcal{O}$ implies, for all $a \in I_A$, $(\pi_1(Q(r)))(a)$ is in $\Delta_A(\mathsf{last}(r))$ and has label a;*
   (c) *if $Q(r) \in \mathcal{I} \times \mathcal{O}$ and $\pi_2(Q(r)) \neq \triangle$, then $\pi_2(Q(r)) \in \Delta_A(\mathsf{last}(r))$.*
2. *The domain of Q is generated by the following closure rules:*
   (a) *$\underline{\mu} \in \mathsf{dom}(Q)$;*
   (b) *for all $r \in \mathsf{dom}(Q)$, the one-step extension $r' = r.f(a).a.\pi_2(f(a))$ is in $\mathsf{dom}(Q)$, provided one of the following holds:*
      - *$f = Q(r) \in \mathsf{Bun}_H(A)$ and $a \in \mathsf{Supp}(\pi_1 \circ f)$;*
      - *$Q(r) \in \mathcal{I} \times \mathcal{O}$, $a \in I_A$, and $f = (\pi_1(Q(r)))(a)$;*
      - *$Q(r) \in \mathcal{I} \times \mathcal{O}$, $f = \pi_2(Q(r)) \neq \triangle$, and $a \in \mathsf{Supp}(\pi_1 \circ f)$.*

   (Notice, by Condition (1), r' is in fact a well-defined branch.)

We write $\mathsf{Tree}(\mu)$ for the set of all execution trees from μ and $\mathsf{Tree}(A)$ for $\mathsf{Tree}(\mu^0)$. Intuitively, each such partial function Q corresponds to a local scheduler and $\mathsf{dom}(Q)$ is the set of branches *reachable* under that scheduler. The closure rules defining $\mathsf{dom}(Q)$ capture the idea of "one-step" reachability.

5 Observable Behavior

As given in Definition 3, the behavioral abstraction of a branch r is the sequence $\mathsf{tr}(r)$ of visible action symbols along r. To obtain the abstraction of an execution tree, we first note that the trace function $\mathsf{tr} : \mathsf{Bran}(\mu) \longrightarrow Act^{<\omega}$ induces an equivalence relation on $\mathsf{Bran}(\mu)$ in the obvious way: for $r, r' \in \mathsf{dom}(Q)$, $r \equiv_{\mathsf{tr}} r'$ if and only if $\mathsf{tr}(r) = \mathsf{tr}(r')$. Moreover, the following image mapping $\mathsf{tr} : (\mathsf{Bran}(\mu)/\equiv_{\mathsf{tr}}) \longrightarrow Act^{<\omega}$ is well-defined: $\mathsf{tr}([r]) := \mathsf{tr}(r)$.

Given any $Q \in \mathsf{Tree}(\mu)$, $\mathsf{dom}(Q)$ is a subset of $\mathsf{Bran}(\mu)$, therefore we may restrict $\equiv_{\mathsf{tr}}$ to $\mathsf{dom}(Q)$. Below we state an interesting fact about the quotient $\mathsf{dom}(Q)/\equiv_{\mathsf{tr}}$. Here $\sqsubseteq$ denotes the prefix ordering on sequences.

Theorem 1. *Let $Q \in \mathsf{Tree}(\mu)$ be given. For all $r \in \mathsf{dom}(Q)$, the coset $[r] \in \mathsf{dom}(Q)/\equiv_{\mathsf{tr}}$ is linearly ordered by $\sqsubseteq$ and there exists a unique minimal element. If $[r]$ is finite, then there is also a unique maximal element.*

Theorem 1 is a strong (and perhaps surprising) result. It says, given any tree Q and finite trace α, there is "essentially" at most one branch r in $\mathsf{dom}(Q)$ with trace α. We say "essentially" because there may in fact be more than one, but they differ in a very limited way: if r and r' in $\mathsf{dom}(Q)$ have the same trace, then one must be a prefix of the other. This gives us a convenient way to define observable behavior by grouping together branches with the same trace.

Definition 5. *Let $Q \in \mathsf{Tree}(\mu)$ be given. The* likelihood assignment *induced by Q, denoted Π_Q, is the function from $(I_A \cup O_A)^{<\omega}$ to $[0,1]$ defined by:*

- *$\Pi_Q[\alpha] := 0$ if $\mathsf{tr}([r]) \neq \alpha$ for all $[r] \in \mathsf{dom}(Q)/\equiv_{\mathsf{tr}}$;*
- *otherwise, $\Pi_Q[\alpha] := \Pi[r]$, where r is the unique minimal element of the (unique) coset in $\mathsf{dom}(Q)/\equiv_{\mathsf{tr}}$ with trace α.*

Again, we avoid the terminology "trace distribution" of [Seg95], because our likelihood assignments do not necessarily induce probability measures (due to the presence of input non-determinism). However, the two notions do coincide for closed PIOAs.

Next we introduce the notion of probabilistic systems. This is simply an underlying PIOA together with a set of "admissible" executions. As we shall see in Section 7, some execution trees of a composite automaton are inadmissible because they are not "generated" by execution trees of the components.

Definition 6. *A* probabilistic system (PS) *$\mathcal{P}$ is a pair $\langle A, \mathcal{L}\rangle$, where A is a PIOA and $\mathcal{L} \subseteq \mathsf{Tree}(A)$. We sometimes write $\mathcal{P}_A$ when $\mathcal{L}$ is clear from context. Such a system is* full *if $\mathcal{L} = \mathsf{Tree}(A)$. We write $\Pi(\mathcal{P})$ for the image of $\mathcal{L}$ under Π. We define* behavioral inclusion *($\leq_\Pi$) by: $\mathcal{P}_1 \leq_\Pi \mathcal{P}_2$ if and only if $\Pi(\mathcal{P}_1) \subseteq \Pi(\mathcal{P}_2)$.*

6 Order Structures and Finite Approximation

In this section, we outline our development leading to a finite approximation theorem (Theorem 3). Details can be found in Appendix C. Throughout the section, we fix a PIOA A and $\mu \in S_A$. First, we define a simple order structure on $\mathsf{Tree}(\mu)$, very much analogous to the prefix ordering on sequences. This is easily shown to be a poset. Theorem 2 then shows that every increasing sequence in this poset has a least upperbound.

Definition 7. *Given $Q, Q' \in \mathsf{Tree}(\mu)$, we define $Q \leq Q'$ by: $\mathsf{dom}(Q) \subseteq \mathsf{dom}(Q')$.*

Theorem 2. *The poset $\langle \mathsf{Tree}(\mu), \leq\rangle$ is closed under limits of ω-chains.*

Theorem 2 provides a definition principle for execution trees: every ω-chain $\{Q_k \mid k \in \mathbb{N}\}$ in $\langle \mathsf{Tree}(\mu), \leq\rangle$ has a uniquely determined limit (denoted $\bigvee_{k\in\mathbb{N}} Q_k$). In this way, we can specify an infinite element Q in $\mathsf{Tree}(\mu)$ by constructing an increasing sequence converging to Q.

Next, given an execution tree $Q \in \mathsf{Tree}(\mu)$, we define $Q \restriction_n$ for each $n \in \mathbb{N}$ using an appropriate enumeration $\{r_n \mid n \in \mathbb{N}\}$ of $\mathsf{Bran}(\mu)$. The idea is: (i) $Q \restriction_n$

agrees with Q up to r_n, and (ii) for all $m > n$, $Q \restriction_n$ terminates on r_m. This sequence $\{Q \restriction_n \mid n \in \mathbb{N}\}$ is shown to converge to Q (Lemma 9).

For the final ingredient, we define an order structure on the function space $\mathcal{F} := (I_A \cup O_A)^{<\omega} \to [0,1]$. (Recall from Definition 5 that likelihood assignments are functions in this space.) This structure is analogous to the subset/information ordering on partial functions.

Definition 8. *Let $D_1, D_2 \in \mathcal{F}$ be given. Define $D_1 \leq D_2$ by: for all $\alpha \in (I_A \cup O_A)^{<\omega}$, $D_1(\alpha) \neq 0$ implies $D_1(\alpha) = D_2(\alpha)$.*

Clearly, the poset $\langle \mathcal{F}, \leq \rangle$ is closed under ω-limits (by taking pointwise limits). Moreover, it is immediate that the operator $\Pi : \mathsf{Tree}(\mu) \longrightarrow \mathcal{F}$ is monotone: for all Q, Q' in $\mathsf{Tree}(\mu)$, $Q \leq Q'$ (in the sense of Definition 7) implies $\Pi_Q \leq \Pi_{Q'}$. It is in fact continuous. This is precisely our finite approximation theorem.

Theorem 3. *For any ω-chain $\{Q_k \mid k \in \mathbb{N}\}$ in $\mathsf{Tree}(\mu)$, $\Pi_{\bigvee_{k\in\mathbb{N}} Q_k} = \bigvee_{k\in\mathbb{N}} \Pi_{Q_k}$.*

7 Parallel Composition and Compositionality

Intuitively, the basic building blocks of a distributed system are purely stochastic processes. These are modeled by (trivially) full probabilistic systems (cf. Definition 6). In a parallel composition of these basic elements, we consider the underlying composite PIOA together with execution trees that are generated in some appropriate sense (Definition 9).

To carry out this development, we start with the notion of compatibility: PIOAs A and B are said to be *compatible* if $O_A \cap O_B = Act_A \cap H_B = Act_B \cap H_A = \emptyset$. The composite $A \| B$ is defined as usual with action synchronization. In particular, every input bundle of the composite carrying label a involves an input bundle of every component with a in its signature. Similarly, every locally controlled bundle of the composite is generated by a locally controlled bundle of (exactly) one of its components, synchronizing with input bundles of other components whenever appropriate. The formal definition is standard but tedious, therefore it is presented in Appendix D (Definition 13).

Moreover, given a transition bundle f in a composite $\|_{1\leq i\leq n} A_i$, one can define the *ith-projection* of f, denoted $\pi_{A_i}(f)$, in the obvious way. Similarly, one can also define the *ith-projection* of a branch r in the composite, denoted $\pi_{A_i}(r)$. (Again, details are in Appendix B, Definitions 14 and 15.) This allows us to define "generated" execution trees and following that parallel composition for probabilistic systems. Then we are ready to present our main theorem.

Definition 9. *Let A, B be compatible PIOAs and let $\langle \mu_A, \mu_B \rangle \in S_{A\|B}$ be given. An execution tree $Q \in \mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ is said to be* generated *by $Q_A \in \mathsf{Tree}(\mu_A)$ and $Q_B \in \mathsf{Tree}(\mu_B)$ if $\pi_A(\mathsf{dom}(Q)) \subseteq \mathsf{dom}(Q_A)$ and $\pi_B(\mathsf{dom}(Q)) \subseteq \mathsf{dom}(Q_B)$.*

Definition 10. *Let $\mathcal{P}_A = \langle A, \mathcal{L}_A \rangle$ and $\mathcal{P}_B = \langle B, \mathcal{L}_B \rangle$ be probabilistic systems with A and B compatible. The* parallel composite *of $\mathcal{P}_A$ and $\mathcal{P}_B$, denoted $\mathcal{P}_A \| \mathcal{P}_B$, is given by the underlying PIOA $A\|B$ together with the set $\mathcal{L}_{A\|B} := \{Q \in \mathsf{Tree}(A\|B) \mid Q$ is generated by some $Q_A \in \mathcal{L}_A$ and $Q_B \in \mathcal{L}_B\}$.*

Theorem 4 (Compositionality). *Let $\mathcal{P}_A, \mathcal{P}_B$ and $\mathcal{P}_C$ be probabilistic systems such that A and B have the same signature and are both compatible with C. Assume that $\mathcal{P}_A \leq_{\Pi} \mathcal{P}_B$. Then $\mathcal{P}_A \| \mathcal{P}_C \leq_{\Pi} \mathcal{P}_B \| \mathcal{P}_C$.*

8 The Chor-Israeli-Li Protocol

Distributed consensus (or *agreement*) describes a class of problems in which a set of parallel processes try to agree on a single value from a pre-specified domain by exchanging messages. The famous impossibility result of [FLP85] shows that *no* deterministic consensus algorithm can guarantee correct termination in an asynchronous setting with undetected single process failure. This prompted the move to randomized algorithms for consensus (cf. overview in [Asp03]) and the termination condition is weakened to a probabilistic statement: the set of all non-terminating runs has probability 0. Chor, Israeli and Li (CIL) were the first to provide such a solution, which guarantees wait-free termination (i.e., all but one process may fail) in a shared-memory setting [CIL87,CIL94].

The CIL protocol is of special interest to us, because its correctness relies on the assumption that adversaries cannot observe internal probabilistic branching of any process until the random outcome has been written to the shared memory. We present in Appendix E some preliminary modeling efforts. In particular, we provide PIOA codes for both the shared memory and a typical protocol party.

We also argue that the algorithm of [CIL94] and a simplified version in [Asp03] both contain a minor initialization error (which, to our best knowledge, has not been reported elsewhere). Although the error is easily remediable, it serves as a fresh reminder that formal methods are in fact valuable in detecting human errors.

9 Conclusions and Future Work

Compositionality is important for practical reasons, because it justifies modular reasoning about large systems. In our view, it also serves as a criterion in evaluating whether a formal model bears sufficient resemblance to reality (or at least the abstract view of reality one has chosen for the formal analysis). Therefore, we take a more philosophical approach to the problem of compositionality.

We start with some vague intuitions about parallel processes and try to understand what happens when parallel components can toss coins. This exercise leads to several useful conceptual discoveries. In Section 1 and Appendix A, we argue that the more traditional definitions of parallel composition for stochastic processes (e.g. bias factors and "compose-and-schedule") do not fit very well with our intuitive pictures. Instead, the notion of adversary models from the setting of randomized consensus algorithms provides a lot more insight. As we modify our definitions accordingly, compositionality indeed falls into place.

The most significant difficulty we encountered is finding a consistent way to assign probabilities to various interleavings in a parallel composition (cf. Exam-

ple 1). In our view, this is due largely to the fact that interleaving is inherently timing related, and yet our basic framework is untimed.

In other words, a trace-style semantics does not always capture properties such as "some internal computations require more time than others." For example, when division operations are performed on randomly chosen values, processes may exhibit delays of various lengths. In this way, internal random choices can realistically affect the outcome of interleaving. (This, however, does not provide a direct explanation for the execution in Figure 1, because the specification of Coin_0 does not involve any computations on the randomly chosen bit.) Therefore, we believe that our analysis will benefit from a move to a timed setting, where timing properties can be treated explicitly.

Finally, we see much potential in formal verification of weak adversary protocols such as the CIL protocol mentioned in Section 8. Typically, weak adversary algorithms are much simpler and more efficient compared to their strong counterparts (cf. [AB04]). In our view, the foundational difficulties associated with the strong adversary model (cf. Section 1) increase the appeal of weak adversary algorithms. We believe it will be fruitful to study weak adversary models in a more systematic manner, providing suitable stochastic system types as well as verification techniques.

10 Acknowledgment

We are greatly indebted to Nancy Lynch and Frits Vaandrager, with whom we corresponded extensively during the writing of this paper. We also thank those on whom we have imposed our conversations, including Luca de Alfaro, James Aspnes, Christel Baier, Thomas Henzinger, Jesse Hughes, Wolter Pieters and Roberto Segala.

References

[dAHJ01] L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In K.G. Larsen and M. Nielsen, editors, *Proceedings CONCUR 01,* Aalborg, Denmark, August 20-25, 2001, volume 2154 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2001.

[DAHK98] P. D'Argenio, H. Hermanns, and J.-P. Katoen. On generative parallel composition. In *Proceedings PROBMIV'98*, *ENTCS* 22:105–122, 1998.

[Asp03] J. Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2-3):165–175, 2003.

[AB04] Y. Aumann and M.A. Bender. Efficient low-contention asynchronous consensus with the value-oblivious adversary scheduler. Accepted to *Distributed Computing*, 2004.

[AH90] J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 11(3):441-461, 1990.

[BPW04] M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems. *Cryptology ePrint Archive*, Report 2004/082, 2004.

[CLSV04] L. Cheung, N.A. Lynch, R. Segala, and F.W. Vaandrager. Switched probabilistic i/o automata. In *Proceedings First International Colloquium on Theoretical Aspects of Computing (ICTAC2004),* Guiyang, China, 20-24 September 2004, *Lecture Notes in Computer Science*. Springer-Verlag, 2004. To appear.

[CIL87] B. Chor, A. Israeli, and M. Li. On processor coordination using asynchronous hardware. In *Proceedings PODC'87*, pages 86–97, ACM Press New York, 1987

[CIL94] B. Chor, A. Israeli, and M. Li. Wait-free consensus Using asynchronous hardware. *SIAM Journal on Computing*, 23(4):701–712, 1994.

[FLP85] M. Fischer, N.A. Lynch, and M.S. Paterson Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985

[vGSS95] R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121:59–80, 1995.

[Hit02] C. Hitchcock. Probabilistic causation. In E.N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Fall 2002. Available at `http://plato.stanford.edu/archives/fall2002/entries/causation-probabilistic/`.

[JLY01] B .Josson, K.G. Larsen, and W. Yi. Probabilistic extensions of process algebras. In *Handbook of Process Algebras*, Elsevier, North Holland, 2001.

[KNS01] M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In *Proceedings CAV'01*, LNCS 2102, pages 194-206, 2001.

[LSV03] N.A. Lynch, R. Segala, and F.W. Vaandrager. Compositionality for probabilistic automata. In R. Amadio and D. Lugiez, editors, *Proceedings 14th International Conference on Concurrency Theory (CONCUR 2003),* Marseille, France, volume 2761 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, September 2003.

[LT89] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[SdV04] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier et al., editor, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer-Verlag, 2004.

[WSS94] S.-H. Wu, S. A. Smolka, and E. W. Stark. Composition and behaviors of probabilistic i/o automata. In B. Jonsson and J. Parrow, editors, *Proceedings CONCUR 94,* Uppsala, Sweden, volume 836 of *Lecture Notes in Computer Science*, pages 513–528. Springer-Verlag, 1994.

A Other Approaches to Parallel Composition

Here we give a more detailed account on the various approaches to parallel composition of stochastic processes.

– *Parameterized composition* [JLY01,DAHK98]. Instead of a single (binary) composition operator, one considers a family $\{\|_p\}_{p\in[0,1]}$ of operators. The parameter p indicates the bias towards the left process and $1-p$ towards

the right process. In Example 1, $A\|_p B$ yields: with probability p, write_A is performed before write_B and the opposite situation occurs with probability $1-p$.

There are two main disadvantages associated with this approach. First, there is no obvious reason that one should privilege a single number $p \in [0,1]$ as a faithful representative of the underlying situation between two processes. Except for degenerate cases in which p equals 0 or 1, the operator $\|_p$ is not associative. And, unless we adopt the uniform distribution (i.e., $p = \frac{1}{2}$), the operator $\|_p$ is not commutative. These properties are quite contrary to our intuitions about parallel processes. Second, the parameter p is static, in the sense that the left process will always receive bias p regardless of activities both in and around the process. This is again unsatisfactory.

– *Real-time delay* [WSS94]. To each state of a process, one associates a delay parameter δ. Upon entering a state, every process draws a real-time delay from an exponential distribution with the parameter δ for that state. Among a group of parallel processes, the process with the shortest delay performs the next move. Using specific properties of exponential distributions, one can calculate the bias towards each component.

  Here we find it questionable that the delay patterns of processes can be universally characterized by exponential distributions. Even though this assumption may be reasonable in certain applications (say, in a setting of hybrid systems), it is only natural that we seek alternatives that are suitable for a broader class of problems.

– *Synchronous execution* [dAHJ01,vGSS95]. In this model, components make simultaneous moves, even if they are not involved in action synchronization. In Example 1, $\langle \mathsf{write}_A, \mathsf{write}_B, \mathsf{toss} \rangle$ would be a composite move. Assuming independence of the random choices, the probability of a composite move is simply the product of the probabilities of the atomic moves involved.

  This approach takes us away from the realm of interleaving semantics. It sometimes leads to technical difficulties in formal verification. For example, one must distinguish between atomic and composite actions, while the complexity of composite actions grows with the number of system components being analyzed. Moreover, during a synchronous execution, components evolve in a lock-step fashion, which is typically not the case in a truly distributed setting. Therefore, asynchronous models have been more traditional for analyzing distributed algorithms.

B Basic Lemmas and Proofs

Lemma 1. *Let A be a PIOA and let $\mu \in S_A$ be given. For all $r \in \mathsf{Bran}(\mu)$, we have $\Pi[r] > 0$.*

Proof (Lemma 1). Definition 3 requires $\pi_1(f(a)) > 0$ for every well-formed branch $r.f.a.\pi_2(f(a))$. Using a simple inductive argument on the structure of branches, we obtain the desired conclusion. □

Lemma 2. *Let $Q \in \mathsf{Tree}(\mu)$ and $r' \in \mathsf{dom}(Q)$ be given. For all $r \in \mathsf{Bran}(\mu)$ with $r \sqsubseteq r'$, we have $r \in \mathsf{dom}(Q)$.*

Proof (Lemma 2). Define $X := \{r' \in \mathsf{dom}(Q) \mid \forall r \sqsubseteq r', r \in \mathsf{dom}(Q)\} \subseteq \mathsf{dom}(Q)$. It is easy to check that X satisfies the closure rules in Definition 4, therefore it must be the case that $\mathsf{dom}(Q) \subseteq X$. □

Lemma 3. *Let $Q \in \mathsf{Tree}(\mu)$ and $r \in \mathsf{dom}(Q)$ be given. Suppose the branch $r_1 = r.f.a.\pi_2(f(a))$ is also in $\mathsf{dom}(Q)$.*

1. *If f is a hidden bundle, then $Q(r) \in \mathsf{Bun}_H(A)$ and $f = Q(r)$.*
2. *If f is an input bundle, then $Q(r) \in \mathcal{I} \times \mathcal{O}$ and $f = (\pi_1(Q(r)))(a)$.*
3. *If f is an output bundle, then $Q(r) \in \mathcal{I} \times \mathcal{O}$ and $f = \pi_2(Q(r))$.*

Proof (Lemma 3). For Item (1), assume $Q(r) \notin \mathsf{Bun}_H(A)$ or $f \neq Q(r)$. Define $X := \mathsf{dom}(Q) \setminus \{r_1\}$. It is easy to check that X satisfies the closure rules in Definition 4, contradicting minimality of $\mathsf{dom}(Q)$. Similarly for Items (2) and (3). □

Lemma 4. *Let $Q \in \mathsf{Tree}(\mu)$ be given. For all $r \in \mathsf{dom}(Q)$ and $n \in \mathbb{N}$, there is at most one n-step extension r' of r such that $r' \in \mathsf{dom}(Q)$ and $\mathsf{tr}(r') = \mathsf{tr}(r)$.*

Proof (Lemma 4). By the i/o axiom, every hidden transition bundle carries a unique label, thus leads to a unique target state. Hence, for every $r \in \mathsf{dom}(Q)$ with $f = Q(r) \in \mathsf{Bun}_H(A)$, the branch $r' = r.f.a.\pi_2(f(a))$ is the unique one-step extension of r in $\mathsf{dom}(Q)$. With this observation, the desired conclusion follows from a simple inductive argument. □

Proof (Theorem 1). We prove these claims by induction on the length of $\mathsf{tr}([r])$. For the base case, note that ϵ is the unique length-0 trace and $[\underline{\mu}]$ (where $\underline{\mu}$ is the empty branch) is the unique coset with trace ϵ. Let $r_1, r_2 \in \mathsf{dom}(Q)$ be given with $\mathsf{tr}(r_1) = \mathsf{tr}(r_2) = \epsilon$ and $r_1 \neq r_2$. Without loss, we may assume that $|r_1| = n \leq m = |r_2|$. By Lemma 2, the length-n prefix r_0 of r_2 is also in $\mathsf{dom}(Q)$. Then r_0 and r_1 are both length-n extensions of $\underline{\mu}$ in $\mathsf{dom}(Q)$ and $\mathsf{tr}(r_0) = \mathsf{tr}(r_1) = \epsilon$. By Lemma 4, it must be the case that $r_0 = r_1$, hence $r_1 \sqsubseteq r_2$. Clearly, the unique minimal element of $[\underline{\mu}]$ is $\underline{\mu}$. If $[\underline{\mu}]$ is finite, we may choose $r \in [\underline{\mu}]$ with maximal length. Again by Lemma 4, this element is unique.

For the induction step, we consider $[r'] \in \mathsf{dom}(Q)/\equiv_{\mathsf{tr}}$ with trace $\alpha.a$. By Lemma 2, there is r_0 in $\mathsf{dom}(Q)$ with trace α. By the induction hypothesis, we may choose a minimal such r_0. Suppose $[r_0]$ is infinite. By the induction hypothesis, $[r_0]$ is a ω-chain (w.r.t. $\sqsubseteq$) starting from r_0. Moreover, for every $r \in [r_0]$, it must be the case that $f = Q(r)$ is a hidden bundle with label b for some $b \in H_A$ and the one-step extension $r.f.b.(\pi_2(f(b)))$ is again in $[r_0]$. Therefore, it is easy to check that $X := \mathsf{dom}(Q) \setminus \{r'\}$ satisfies the closure rules in Definition 4. This contradicts the minimality of $\mathsf{dom}(Q)$ and hence we conclude that $[r_0]$ is finite.

By the induction hypothesis, we may choose the unique maximal element $r_1 \in [r_0]$. If $Q(r_1) = \square$, then $X := \mathsf{dom}(Q) \setminus \{r'\}$ satisfies the closure rules in Definition 4, contradicting minimality of $\mathsf{dom}(Q)$. Then it must be the case that $Q(r_1) \in \mathcal{I} \times \mathcal{O}$. There are two cases.

- $a \in I_A$. Let $g = (\pi_1(Q(r_1)))(a)$. Then $r_2 := r_1.g.a.\pi_2(g(a))$ is an element of $\mathsf{dom}(Q)$ with trace $\alpha.a$, hence in $[r']$. By minimality of $\mathsf{dom}(Q)$ and uniqueness of r_1, we know that every $r_3 \in [r']$ is an extension of r_2. Now we apply Lemma 4 to r_2 as we did in the base case and we conclude that $[r']$ is linearly ordered by prefix. Obviously, r_2 is the minimal element and, if $[r']$ is finite, there is a unique element with maximal length.
- $a \in O_A$. Since $\mathsf{dom}(Q)$ is $\sqsubseteq$-closed and r_1 is the unique maximal element of $[r_0]$, we know that $r_1 \sqsubseteq r'$. If $\pi_2(Q(r_1)) = \triangle$, then $X := \mathsf{dom}(Q) \setminus \{r'\}$ satisfies the closure rules in Definition 4, contradicting minimality of $\mathsf{dom}(Q)$. Therefore it must be the case that $g = \pi_2(Q(r_1)) \in \mathsf{Bun}_O(A)$. Using a similar minimality argument, we know that $\pi_1(g(a)) > 0$, therefore $r_2 := r_1.g.a.\pi_2(g(a))$ is well-defined. The rest is similar to the input case.

□

Lemma 5. *Let $Q \in \mathsf{Tree}(\mu)$ and $[r] \in \mathsf{dom}(Q)/\equiv_{\mathsf{tr}}$ be given. For all $r' \in [r]$, we have $\Pi[r] = \Pi[r']$.*

Proof (Lemma 5). By Theorem 1, $[r]$ contains a unique minimal element. Without loss, we assume that r is such an element. Again by Theorem 1, every $r' \in [r]$ is an n-step extension of r for some $n \in \mathbb{N}$. Thus we may prove the claim by induction on n.

Clearly, $\Pi[r] = \Pi[r]$. For the induction step, take $r'' = r'.f.a.\pi_2(f(a))$, where f is a hidden bundle. By the i/o axiom, $\pi_1(f(a)) = 1$, therefore $\Pi[r''] = \Pi[r']$, which equals $\Pi[r]$ by the induction hypothesis. □

Lemma 6. *Let A and B be PIOAs and let $\mu_A \in S_A$ and $\mu_B \in S_B$ be given. Suppose we have $Q_A \in \mathsf{Bran}(\mu_A)$ and $Q_B \in \mathsf{Bran}(\mu_B)$ with $\Pi_{Q_A} = \Pi_{Q_B}$. Then there exists a bijection $h : \mathsf{dom}(Q_A)/\equiv_{\mathsf{tr}} \longrightarrow \mathsf{dom}(Q_B)/\equiv_{\mathsf{tr}}$ preserving both trace and likelihood.*

Proof (Lemma 6). Let $[r_A] \in \mathsf{dom}(Q_A)/\equiv_{\mathsf{tr}}$ be given and let α denote the trace of $[r_A]$. By Lemma 1 and Definition 5, we know that $\Pi_{Q_B}[\alpha] = \Pi_{Q_A}[\alpha] \neq 0$. Therefore we may choose $[r_B] \in \mathsf{dom}(Q_A)/\equiv_{\mathsf{tr}}$ such that the trace of $[r_B]$ is α and the likelihood of $[r_B]$ is $\Pi_{Q_B}[\alpha] = \Pi_{Q_A}[\alpha]$, which equals the likelihood of $[r_A]$. Clearly, such $[r_B]$ is unique and we may define $h([r_A]) := [r_B]$. Then h is trace- and likelihood-preserving. Since the domain of h is a partition induced by tr and h is tr-preserving, it follows immediately that h is injective. Moreover, notice that we can define a right inverse h^{-1} of h in exactly the same manner as we did h, therefore h is also surjective. □

C Order Structures and Finite Approximation: Detailed Development

To prove Theorem 2, we introduce an order structure $\preceq$ on

$$\mathcal{D} := \{\bot\} + \{\Box\} + \mathsf{Bun}_H(A) + (\mathcal{I} \times \mathcal{O})$$

and prove that $\leq$ on $\mathsf{Tree}(\mu)$ coincides with the pointwise ordering induced by $\preceq$. Intuitively, we want: "unreachable" $\preceq$ "terminating" $\preceq$ "non-terminating".

Definition 11. *Recall that* $\mathcal{I} = I_A \to \mathsf{Bun}_I(A)$ *and* $\mathcal{O} = \{\triangle\} \cup \mathsf{Bun}_O(A)$. *Define* $\preceq_{\mathcal{O}} := \mathcal{I}d_{\mathcal{O}} \cup \{\langle \triangle, f\rangle \mid f \in \mathcal{O}\}$. *Let* $\preceq_{\mathcal{I}\times\mathcal{O}}$ *by the ordering on* $\mathcal{I}\times\mathcal{O}$ *generated by* $\mathcal{I}d_{\mathcal{I}}$ *and* $\preceq_{\mathcal{O}}$ *component-wise. Then* $\preceq$ *is the transition closure of the following:*

$$\mathcal{I}d_{\mathcal{D}} \cup \{\langle \bot, \Box\rangle\} \cup \{\langle \Box, d\rangle \mid d \in \mathcal{D}, d \neq \bot\} \cup \preceq_{\mathcal{I}\times\mathcal{O}} .$$

Lemma 7. *Let* $Q_1, Q_2 \in \mathsf{Tree}(\mu)$ *be given and assume that* $Q_1 \leq Q_2$. *For all branches* $r \in \mathsf{dom}(Q_1)$, *we have*

1. *if* $Q_1(r) \in \mathsf{Bun}_H(A)$, *then* $Q_1(r) = Q_2(r)$;
2. *if* $Q_1(r) \in \mathcal{I}\times\mathcal{O}$, *then so is* $Q_1(r)$ *and* $\pi_1(Q_1(r)) = \pi_1(Q_2(r))$;
3. *if* $Q_1(r) \in \mathcal{I}\times\mathcal{O}$ *and* $\pi_2(Q_1(r)) \neq \triangle$, *then* $\pi_2(Q_1(r)) = \pi_2(Q_2(r))$;

Proof (Lemma 7). We treat only the second claim, as the other two follow similarly. Let $\langle F, f\rangle$ denote $Q_1(r)$ and choose any $a \in I_A$. Then $\mathsf{dom}(Q_1)$ contains the branch $r.F(a).a.\pi_2(F(a)(a))$, hence so does $\mathsf{dom}(Q_2)$. By Lemma 3, $Q_2(r) \in \mathcal{I}\times\mathcal{O}$ and $(\pi_1(Q_1(r)))(a) = F(a) = (\pi_1(Q_2(r)))(a)$. Since this holds for every $a \in I_A$, we have $\pi_1(Q_1(r)) = \pi_1(\mathbf{Q}_2(r))$. □

Theorem 5. *For all* $Q_1, Q_2 \in \mathsf{Tree}(\mu)$, $Q_1 \leq Q_2$ *if and only if* $Q_1(r) \preceq Q_2(r)$ *for all* $r \in \mathsf{Bran}(\mu)$.

Proof (Theorem 5). Suppose $Q_1 \not\leq Q_2$. Then there exists r such that $Q_1(r) \neq \bot$ but $Q_2(r) = \bot$, which implies $Q_1(r) \not\preceq Q_2(r)$. Therefore the "only if" part holds.

Now suppose $Q_1 \leq Q_2$. If $Q_1(r) = \bot$, then clearly $Q_1(r) \preceq Q_2(r)$. Otherwise, $r \in \mathsf{dom}(Q_1) \subseteq \mathsf{dom}(Q_2)$. If $Q_1(r) = \Box$, then $Q_1(r) \preceq Q_2(r)$; else we apply Lemma 7 to conclude the same. □

By virtue of Theorem 5, we may use either definition of $\leq$ on $\mathsf{Tree}(\mu)$ as we see fit. Next we prove that $\langle \mathcal{D}, \preceq\rangle$ is closed under ω-limits and hence so is $\langle \mathsf{Tree}(\mu), \leq\rangle$.

Lemma 8. *The poset* $\langle \mathcal{D}, \preceq\rangle$ *is closed under limits of* ω*-chains.*

Proof (Lemma 8). Let $\{d_k \mid k \in \mathbb{N}\}$ be an ω-chain in $\langle \mathcal{D}, \preceq\rangle$. Without loss, we may assume that $d_{K_1} \neq \bot$ for some K_1. If $d_k = \Box$ for all $k \geq K_1$, then the limit is $\Box$. Otherwise, let K_2 be the least k such that $d_k \notin \{\bot, \Box\}$. If $d_{K_2} \in \mathsf{Bun}_H(A)$, then $d_k = d_{K_2}$ for all $k \geq K_2$, hence the limit is d_{K_2}.

Otherwise, $\{d_k \mid k \geq K_2\}$ is an ω-chain in $\langle \mathcal{I}\times\mathcal{O}, \preceq_{\mathcal{I}\times\mathcal{O}}\rangle$, which is closed under limits of ω-chains. Therefore, $\langle \mathcal{D}, \preceq\rangle$ is also closed under limits of ω-chains. □

We are now ready for the proof of Theorem 2.

Proof (Theorem 2). Let $\{Q_k \mid k \in \mathbb{N}\} \subseteq \mathsf{Tree}(\mu)$ be an ω-chain with respect to $\leq$. By Lemma 8, we may define Q by taking pointwise limits. We need to show that $Q \in \mathsf{Tree}(\mu)$.

Condition (1) in Definition 4 is satisfied because for all $r \in \mathsf{dom}(Q)$, there exists k such that $Q(r) = Q_k(r)$. Moreover, since $\bot$ is the bottom element for

$\preceq$, we have $\mathsf{dom}(Q) = \bigcup_{k\in\mathbb{N}} \mathsf{dom}(Q_k)$. Hence $\mathsf{dom}(Q)$ is closed under the rules in Condition (2). We need to show it is the smallest such set.

Let $X \subseteq \mathsf{Bran}(\mu)$ be also closed under those rules. We prove that, for all $r \in \mathsf{Bran}(\mu)$, $r \in \mathsf{dom}(Q)$ implies $r \in X$. This is done by induction on the structure of r.

- Clearly $\underline{\mu} \in X$.
- Consider $r.f.a.\pi_2(f(a)) \in \mathsf{dom}(Q)$. If f is a hidden bundle, then by Lemma 3 we know that $f = Q(r)$. By the definition of branches, we have $\pi_1(f(a)) > 0$. Therefore $r.f.a.\pi_2(f(a))$ must be in X. The two other cases (input and output) follow similarly.

□

Recall our assumption that Act is countable and A is countably branching. Therefore, $\mathsf{Bran}(\mu)$ is countable for all $\mu \in S_A$. Take any enumeration $\{r_0, r_1, r_2, \ldots\}$ that is monotone with respect to the prefix ordering. That is, for all m and n, $m \leq n$ implies $r_m \sqsubseteq r_n$. This is always possible by first obtaining a $\sqsubseteq$-monotone enumeration of the set of all finite sequences over the alphabet $S_A \cup \mathsf{Bun}(A) \cup Act$, and then intersecting that enumeration with $\mathsf{Bran}(\mu)$. Using this enumeration, we define (finite) approximations of execution trees.

Definition 12. *Let $Q \in \mathsf{Tree}(\mu)$ be given. For each $n \in \mathbb{N}$, define*

- $\mathsf{dom}^n_\flat(Q) := \mathsf{dom}(Q) \cap \{r_0, \ldots, r_n\}$,
- $\mathsf{dom}^n_\sharp(Q) := \{r.Q(r).a.\pi_2(Q(r)) \mid r \in \mathsf{dom}^n_\flat(Q) \text{ and } a \in \mathsf{Supp}(\pi_1 \circ (Q(r)))\}$,
- $\mathsf{dom}^n(Q) := \mathsf{dom}^n_\flat(Q) \cup \mathsf{dom}^n_\sharp(Q)$.

Then the n-th approximation *of Q, denoted $Q \upharpoonright_n$, is the partial function with domain $\mathsf{dom}^n(Q)$ such that:*

- *for all $r \in \mathsf{dom}^n_\flat(Q)$, $Q \upharpoonright_n (r) = Q(r)$;*
- *for all $r \in \mathsf{dom}^n_\sharp(Q)$, $Q \upharpoonright_n (r) = \Box$.*

Since the enumeration $\{r_0, r_1, r_2, \ldots\}$ is monotone with respect to prefix, it is easy to check that $Q \upharpoonright_n$ is well-defined for all $n \in \mathbb{N}$.

Lemma 9. *For every $Q \in \mathsf{Tree}(\mu)$, the sequence $\{Q \upharpoonright_n \mid n \in \mathbb{N}\}$ forms a chain with respect to $\leq$. Moreover, $Q = \bigvee_{n\in\mathbb{N}} Q \upharpoonright_n$.*

Proof (Lemma 9). The first claim is trivial. The second follows from Theorem 2. □

This leads to the proof of Theorem 3.

Proof (Theorem 3). By the proof of Theorem 2, we know that $\mathsf{dom}(\bigvee_{k\in\mathbb{N}} Q_k) = \bigcup_{k\in\mathbb{N}} \mathsf{dom}(Q_k)$. Since Π_Q is defined entirely in terms of $\mathsf{dom}(Q)$ for all $Q \in \mathsf{Tree}(\mu)$, the desired equality follows. □

D Parallel Composition and Compositionality: Detailed Development

First we give the formal definition of parallel composition for PIOAs.

Definition 13. *Let $\{A_i \mid 1 \leq i \leq n\}$ be pairwise compatible PIOAs. We define the composite $\|_{1\leq i\leq n} A_i$ to be the following PIOA P.*

1. *$S_P := \prod_{i=1}^n S_i$ and the start state is $\langle \mu_1^0, \ldots, \mu_n^0 \rangle$;*
2. *$I_P := (\bigcup_{i=1}^n I_i) \setminus (\bigcup_{i=1}^n O_i)$, $O_P := (\bigcup_{i=1}^n O_i)$, and $H_P := (\bigcup_{i=1}^n H_i)$;*
3. *given a state $\langle \mu_1, \ldots, \mu_n \rangle$, $\Delta_P(\langle \mu_1, \ldots, \mu_n \rangle)$ contains precisely those functions $f : Act_P \longrightarrow [0,1] \times S_P$ satisfying one of the following conditions.*
   (a) *There exists $1 \leq i \leq n$ and locally controlled transition bundle g_i from μ_i such that:*
      - *for all $a \in O_i \cup H_i$, $f(a) = \langle \pi_1(g_i(a)), \langle \nu_1, \ldots, \nu_n \rangle \rangle$, where*
        - *$\nu_i = \pi_2(g_i(a))$;*
        - *for all $j \neq i$ such that $a \notin I_j$, $\nu_j = \mu_j$;*
        - *for all $j \neq i$ such that $a \in I_j$, $\nu_j = \pi_2(h_j(a))$ for some input transition bundle h_j from μ_j with label a;*
      - *for all other $a \in Act_P$, $f(a) = \langle 0, \langle \mu_1, \ldots, \mu_n \rangle \rangle$.*
   (b) *There exists action $a \in I_P$ such that:*
      - *$f(a) = \langle 1, \langle \nu_1, \ldots, \nu_n \rangle \rangle$, where*
        - *for all i such that $a \in I_i$, $\nu_i = \pi_2(h_i(a))$ for some input transition bundle h_i from μ_i carrying label a;*
        - *for all i such that $a \notin I_i$, $\nu_i = \mu_i$;*
      - *for all other $b \in Act_P$, $f(b) = \langle 0, \langle \mu_1, \ldots, \mu_n \rangle \rangle$.*

Such a composite P is *closed* if $I_P = \emptyset$. Sometimes we write $\|^n \{A_i \mid 1 \leq i \leq n\}$ for $\|_{1\leq i\leq n} A_i$. For $n = 2$, we write $\|$ and use infix notation. It is easy to check that $\|_{1\leq i\leq n}$ is well-defined for all n and that $\|$ is commutative and associative.

Lemma 10. *The automaton $\|_{1\leq i\leq n} A_i$ is in fact a PIOA.*

Proof (Lemma 10). Let $\mu = \langle \mu_1, \ldots, \mu_n \rangle$ and $f \in \Delta_{\|_{1\leq i\leq n} A_i}(\langle \mu_1, \ldots, \mu_n \rangle)$ be given. If f arises from Clause (3b) in Definition 2, then clearly $\pi_1 \circ f$ is the dirac measure on a. Otherwise, assume that f arises from Clause (3a). Choose locally controlled transition bundle g_i from μ_i such that f is induced by g_i. Notice that $\pi_1 \circ f$ coincides with $\pi_1 \circ g_i$ on $O_i \cup H_i$ and is 0 elsewhere. Since $\pi_1 \circ g_i$ is a discrete distribution, so is $\pi_1 \circ f$. Moreover, it follows that $\mathsf{Supp}(\pi_1 \circ f) \subseteq O_i$ or $\mathsf{Supp}(\pi_1 \circ f) = \{a\}$ for some $a \in H_i$.

The argument above shows that f is a PA in the sense of Definition 1 and that f satisfies the i/o axiom in Definition 2. Input enabling follows from Clause (3b) of Definition 2 and the fact that every A_i satisfies input enabling. □

Lemma 11. *The composition operator $\|$ is commutative and associative.*

Proof (Lemma 11). Commutativity is trivial. For associativity, it is easy to see that $(A\|B)\|C$ is isomorphic to $\|^3\{A, B, C\}$. □

Next we consider projection operations on transition bundles and execution branches.

Definition 14. *Let $\{A_i \mid 1 \leq i \leq n\}$ be pairwise compatible PIOAs and let $\langle \mu_1, \ldots, \mu_n \rangle$ be a state in $\|_{1 \leq i \leq n} A_i$. Let f be a transition bundle from the state $\langle \mu_1, \ldots, \mu_n \rangle$. For each $1 \leq i \leq n$, we define the* ith-projection *of f, denoted $\pi_{A_i}(f)$, as follows:*

- *if f arises from a locally controlled bundle g_i from μ_i, then $\pi_{A_i}(f) := g_i$;*
- *if f arises from a locally controlled bundle g_j from μ_j with $i \neq j$, then $\pi_{A_i}(f)$ is the partial function from I_i to $\mathsf{Bun}_I(A_i)$ given by: for all $a \in I_i$,*
  - *if $a \notin \mathsf{Supp}(\pi_1 \circ g_j)$, then $\pi_{A_i}(f)(a) := \perp$ (undefined);*
  - *otherwise, $\pi_{A_i}(f)$ is the unique input bundle h_i from μ_i with $\pi_1(h_i(a)) = 1$ and $\pi_2(h_i(a)) = \pi_{A_i}(\pi_2(f(a)))$;*
- *if f is an input bundle carrying label a and $a \in I_i$, then $\pi_{A_i}(f)$ is the unique input bundle h_i from μ_i such that $\pi_1(h_i(a)) = 1$ and $\pi_2(h_i(a)) = \pi_{A_i}(\pi_2(f(a)))$;*
- *otherwise, $\pi_{A_i}(f) := \perp$ (undefined).*

Using Definition 13, it is easy to check that $\pi_{A_i}(f)$ is well-defined. Notice that π_{A_i} on transition bundles is a partial operation, because it need not be the case that all components are involved in every step of the composite. Next we treat execution branches.

Definition 15. *For each $1 \leq i \leq n$, we define a projection function π_{A_i} from $\mathsf{Bran}(\langle \mu_1, \ldots, \mu_n \rangle)$ to $\mathsf{Bran}(\mu_i)$ as follows.*

- $\pi_{A_i}(\underline{\langle \mu_1, \ldots, \mu_n \rangle}) := \underline{\mu_i}$.
- *Consider $r' = r.f.a.\pi_2(f(a))$.*
  - *If $\pi_{A_i}(f) = g \in \mathsf{Bun}(A_i)$, then it must be the case that $a \in \mathsf{Supp}(\pi_1 \circ g)$ and we define $\pi_{A_i}(r') := \pi_{A_i}(r).g.a.\pi_2(g(a))$.*
  - *If $\pi_{A_i}(f) = G \in I_i \rightharpoonup \mathsf{Bun}_I(A_i)$ and $a \in \mathsf{dom}(G)$, then $\pi_{A_i}(r') := \pi_{A_i}(r).G(a).a.\pi_2(G(a)(a))$.*
  - *Otherwise $\pi_{A_i}(r') := \pi_{A_i}(r)$.*

Using Definitions 13 and 14, one can easily check that $\pi_{A_i}(r)$ is well-defined for every $r \in \mathsf{Bran}(\langle \mu_1, \ldots, \mu_n \rangle)$. Notice, unlike the case for transition bundles, π_{A_i} is a total operation on execution branches.

For Lemmas 12 through 16, let $Q \in \mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ be generated by $Q_A \in \mathsf{Tree}(\mu_A)$ and $Q_B \in \mathsf{Tree}(\mu_B)$ and let r be a branch in $\mathsf{dom}(Q)$ with $Q(r) = \Box$.

Lemma 12. *Suppose $f = Q_A(\pi_A(r)) \in \mathsf{Bun}_H(A)$. Then there is bundle g from $\mathsf{last}(r)$ such that $\pi_A(g) = f$. Moreover, there is Q' in $\mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ such that Q' is generated by Q_A and Q_B and $\mathsf{dom}(Q') = \mathsf{dom}(Q) \cup \{r.g.a.\pi_2(g(a))\}$, where a is the unique member of $\mathsf{Supp}(\pi_1 \circ f)$.*

Proof (Lemma 12). By the definition of $\Delta_{A\|B}$, there is a bundle g from $\mathsf{last}(r)$ in which component A follows bundle f and component B simply stutters. Then we define Q' as follows:

– $Q'(r) := g$;
– $Q'(r.g.a.\pi_2(g(a))) := \Box$;
– for all other $r' \in \mathsf{Bran}(\langle \mu_A, \mu_B \rangle)$, $Q'(r') := Q(r')$.

It is routine to check that Q' is an execution tree generated by Q_A and Q_B and satisfying the desired condition. □

Lemma 13. *Suppose $Q_A(\pi_A(r)) \in \mathcal{I}_A \times \mathcal{O}_A$ and $I_B \cap I_{A\|B} = \emptyset$. Then for all $a \in I_{A\|B}$, there is $g_a \in \Delta_{A\|B}(\mathsf{last}(r))$ such that $\pi_A(g_a) = (\pi_1(Q_A(\pi_A(r))))(a)$. Moreover, there is Q' in $\mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ such that Q' is generated by Q_A and Q_B and $\mathsf{dom}(Q') = \mathsf{dom}(Q) \cup \{r.g_a.a.\pi_2(g_a(a)) \mid a \in I_{A\|B}\}$.*

Proof (Lemma 13). Let $a \in I_{A\|B}$ be given. By assumption, a is not in the signature of B. By the definition of $\Delta_{A\|B}$, there is a bundle g_a from $\mathsf{last}(r)$ in which component A follows bundle $(\pi_1(Q_A(\pi_A(r))))(a)$ and component B simply stutters. Let G be the function from $I_{A\|B}$ to $\mathsf{Bun}_I(A\|B)$ defined by $G(a) := g_a$. Then we define Q' as follows:

– $Q'(r) := \langle G, \triangle \rangle$;
– for all $a \in I_{A\|B}$, $Q'(r.g_a.a.\pi_2(g_a(a))) := \Box$;
– for all other $r' \in \mathsf{Bran}(\langle \mu_A, \mu_B \rangle)$, $Q'(r') := Q(r')$.

It is routine to check that Q' is an execution tree generated by Q_A and Q_B and satisfying the desired condition. □

Lemma 14. *Suppose $Q_A(\pi_A(r)) \in \mathcal{I}_A \times \mathcal{O}_A$ and $I_B \cap I_{A\|B} = \emptyset$. Suppose in addition $f = \pi_2(Q_A(\pi_A(r))) \neq \triangle$ and $\mathsf{Supp}(\pi_1 \circ f) \cap I_B = \emptyset$. Then there is $g \in \Delta_{A\|B}(\mathsf{last}(r))$ such that $\pi_A(g) = f$. Moreover, there is Q' in $\mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ such that Q' is generated by Q_A and Q_B and $\mathsf{dom}(Q')$ equals*

$$\mathsf{dom}(Q) \cup \{r.g_a.a.\pi_2(g_a(a)) \mid a \in I_{A\|B}\} \cup \{r.g.b.\pi_2(g(b)) \mid b \in \mathsf{Supp}(\pi_1 \circ g)\},$$

where each g_a is as given in Lemma 13.

Proof (Lemma 14). Let $b \in \mathsf{Supp}(\pi_1 \circ f)$ be given. By assumption, b is not in the signature of B. By the definition of $\Delta_{A\|B}$, there is a bundle g from $\mathsf{last}(r)$ in which component A follows bundle f and component B simply stutters. Let G be the function from $I_{A\|B}$ to $\mathsf{Bun}_I(A\|B)$ as defined in the proof of Lemma 13. Then we define Q' as follows:

– $Q'(r) := \langle G, g \rangle$;
– for all $a \in I_{A\|B}$, $Q'(r.g_a.a.\pi_2(g_a(a))) := \Box$;
– for all $b \in \mathsf{Supp}(\pi_1 \circ f)$, $Q'(r.g.b.\pi_2(g(b))) := \Box$;
– for all other $r' \in \mathsf{Bran}(\langle \mu_A, \mu_B \rangle)$, $Q'(r') := Q(r')$.

It is routine to check that Q' is an execution tree generated by Q_A and Q_B and satisfying the desired condition. □

Lemma 15. *Suppose $Q_A(\pi_A(r)) \in \mathcal{I}_A \times \mathcal{O}_A$ and $Q_B(\pi_B(r)) \in \mathcal{I}_B \times \mathcal{O}_B$. Then for all $a \in I_{A\|B}$, there is $g_a \in \Delta_{A\|B}(\mathsf{last}(r))$ such that*

– *if* $a \in I_A$, *then* $\pi_A(g_a) = (\pi_1(Q_A(\pi_A(r))))(a)$ *and*
– *if* $a \in I_B$, *then* $\pi_B(g_a) = (\pi_1(Q_B(\pi_B(r))))(a)$.

Moreover, there is $Q' \in \mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ *such that* Q' *is generated by* Q_A *and* Q_B *and* $\mathsf{dom}(Q')$ *equals* $\mathsf{dom}(Q) \cup \{r.g_a.a.\pi_2(g_a(a)) \mid a \in I_{A\|B}\}$.

Proof (Lemma 15). Let $a \in I_{A\|B}$ be given. Let f_a^A denote $(\pi_1(Q_A(\pi_A(r))))(a)$ if $a \in I_A$ and $\bot$ otherwise. Similarly for f_a^B. By the definition of $\Delta_{A\|B}$, there is unique bundle g_a from $\mathsf{last}(r)$ with $\pi_A(g_a) = f_a^A$ and $\pi_B(g_a) = f_a^B$. Now we may define G and Q' in the same fashion as in the proof of Lemma 13. □

Lemma 16. *Suppose* $Q_A(\pi_A(r)) \in \mathcal{I}_A \times \mathcal{O}_A$ *and* $Q_B(\pi_B(r)) \in \mathcal{I}_B \times \mathcal{O}_B$. *Suppose in addition that* $f = \pi_2(Q_A(\pi_A(r))) \neq \triangle$. *Then there is* $g \in \Delta_{A\|B}(\mathsf{last}(r))$ *such that* $\pi_A(g) = f$ *and* $\pi_B(g)(b) = \pi_2(Q_B(\pi_B(r)))(b)$ *for all* $b \in \mathsf{Supp}(\pi_1 \circ g) \cap I_B$. *Moreover, there is* Q' *in* $\mathsf{Tree}(\langle \mu_A, \mu_B \rangle)$ *such that* Q' *is generated by* Q_A *and* Q_B *and* $\mathsf{dom}(Q')$ *equals*

$$\mathsf{dom}(Q) \cup \{r.g_a.a.\pi_2(g_a(a)) \mid a \in I_{A\|B}\} \cup \{r.g.b.\pi_2(g(b)) \mid b \in \mathsf{Supp}(\pi_1 \circ g)\},$$

where each g_a *is as given in Lemma 15.*

Proof (Lemma 16). Let $b \in \mathsf{Supp}(\pi_1 \circ f)$ be given. Let f_b^B denote the bundle $(\pi_2(Q_B(\pi_B(r))))(b)$ if $b \in I_B$ and $\bot$ otherwise. Again using the definition of $\Delta_{A\|B}$, there is a unique bundle g from $\mathsf{last}(r)$ with $\pi_A(g) = f$ and $\pi_B(g)(b) = f_b^B$ for all $b \in \mathsf{Supp}(\pi_1 \circ f)$. Now we may define G and Q' as we did in Lemma 14. □

We are now ready to prove Theorem 4.

Proof (Theorem 4). Let $Q_{A,C}$ be an execution tree in $A\|C$. Suppose it's generated by Q_A and Q_C. By assumption, choose Q_B so that $\Pi(Q_A) = \Pi(Q_B)$.

For each $n \in \mathbb{N}$, define $Q_{A,C,n} := Q_{A,C} \restriction_n$. From the chain $\{Q_{A,C,n} \mid n \in \mathbb{N}\}$, we construct a chain $\{Q_{B,C,n} \mid n \in \mathbb{N}\}$ of execution trees in $B\|C$ so that, for all n,

(1) $\Pi(Q_{A,C,n}) = \Pi(Q_{B,C,n})$, and
(2) $Q_{B,C,n}$ is generated by Q_B and Q_C,
(3) $Q_{B,C,n}(q) = \square$ for every maximal $q \in \mathsf{dom}(Q_{B,C,n})$ (w.r.t. prefix).

The base case is trivial. For the inductive step, assume we have constructed such $Q_{B,C,n}$. By Lemma 6, there are trace- and likelihood-preserving bijections

– h from $\mathsf{dom}(Q_A)/\equiv_{\mathsf{tr}}$ to $\mathsf{dom}(Q_B)/\equiv_{\mathsf{tr}}$ and
– h_n from $\mathsf{dom}(Q_{A,C,n})/\equiv_{\mathsf{tr}}$ to $\mathsf{dom}(Q_{B,C,n})/\equiv_{\mathsf{tr}}$.

Consider r_{n+1}.If $Q_{A,C}(r_{n+1})$ equals $\square$ or is in $\mathsf{Bun}_H(A\|C)$, then we define $Q_{B,C,n+1} := Q_{B,C,n}$. Clearly Conditions (1), (2), and (3) are met.

Otherwise, let $\langle F, f \rangle$ denote $Q_{A,C}(r_{n+1})$. We say that $\langle F, f \rangle$ is *independent* from component C if $I_C \cap I_{A\|C} = \emptyset$ and

– either $f = \triangle$
– or f is locally controlled by A and $I_C \cap \mathsf{Supp}(\pi_1 \circ f) = \emptyset$.

First suppose $\langle F, f\rangle$ is in fact independent from component C. Let q_0 be the maximal branch in the coset $h_n([r_{n+1}]) \in \mathsf{dom}(Q_{B,C,n})/\equiv_{\mathsf{tr}}$. By the definition of $Q_{A,C,n}$, we know that $Q_{A,C,n}(r_{n+1}) = \Box$. By Theorem 1, it follows that $\mathsf{tr}(r_{n+1})$ is maximal in $\mathsf{tr}(\mathsf{dom}(Q_{A,C,n}))$. By the induction hypothesis, we have that $\mathsf{tr}(q_0) = \mathsf{tr}(r_{n+1})$ is maximal in $\mathsf{tr}(\mathsf{dom}(Q_{B,C,n})) = \mathsf{tr}(\mathsf{dom}(Q_{A,C,n}))$; therefore q_0 must be maximal in $\mathsf{dom}(Q_{B,C,n})$ with respect to the prefix ordering. Again, applying the inductive hypothesis, we have $Q_{B,C,n}(q_0) = \Box$.

Now let q_B denote the maximal branch in the coset $h([\pi_A(r_{n+1})])$ in the quotient $\mathsf{dom}(Q_B)/\equiv_{\mathsf{tr}}$. Notice that $\mathsf{tr}(q_0) = \mathsf{tr}(r_{n+1})$, therefore $\mathsf{tr}(\pi_B(q_0)) = \mathsf{tr}(\pi_A(r_{n+1})) = \mathsf{tr}(q_B)$. By Theorem 1 and maximality of q_B, we have $\pi_B(q_0) \sqsubseteq q_B$ and hence q_B is $\pi_B(q_0)$ followed by a finite number of hidden bundles. By repeated application of Lemma 12, we obtain $Q_1 \in \mathsf{Tree}(B\|C)$ such that Q_1 is generated by Q_B and Q_C and $\mathsf{dom}(Q_1)$ is $\mathsf{dom}(Q_{B,C,n})$ augmented with hidden bundles in B so that there is $q_1 \in \mathsf{dom}(Q_1)$ with $\pi_B(q_1) = q_B$. From the proof of Lemma 12, we have $Q_1(q_1) = \Box$. Moreover, by maximality of q_B, we know that $Q_B(q_B) \notin \mathsf{Bun}_H(B)$.

Since $\langle F, f\rangle$ is independent from component C, it must be the case that $Q_A(\pi_A(r_{n+1})) \in \mathcal{I}_A \times \mathcal{O}_A$. By Theorem 1, maximality of q_B and the assumption that $\mathit{\Pi}(Q_A) = \mathit{\Pi}(Q_B)$, we have $Q_B(q_B) \in \mathcal{I}_B \times \mathcal{O}_B$. Now we may apply either Lemma 13 or Lemma 14 to obtain $Q_2 \in \mathsf{Tree}(B\|C)$ from Q_1 and $Q_B(q_B)$. Define $Q_{B,C,n+1} := Q_2$. It is routine to check Conditions (1), (2), and (3). This concludes the construction for the case where $\langle F, f\rangle$ is independent from component C. The construction for the case where $\langle F, f\rangle$ is independent from component A is analogous. (In fact, it is much simpler, because we reason only with component C.)

Now we consider the case in which both components A and C are involved in $\langle F, f\rangle$. Construct q_B, q_1 and Q_1 as above. Let q_C denote $\pi_C(r_{n+1})$. Since $\langle F, f\rangle$ is not independent from C, we know that $Q_C(q_C) \in \mathcal{I}_C \times \mathcal{O}_C$. Similarly, it must be the case that $Q_A(\pi_A(r_{n+1})) \in \mathcal{I}_A \times \mathcal{O}_A$. By Theorem 1, maximality of q_B and the assumption that $\mathit{\Pi}(Q_A) = \mathit{\Pi}(Q_B)$, we have $Q_B(q_B) \in \mathcal{I}_B \times \mathcal{O}_B$. Now we may apply either Lemma 15 or Lemma 16 to obtain $Q_3 \in \mathsf{Tree}(B\|C)$ from Q_1, f, $Q_B(q_B)$ and $Q_C(q_C)$. Define $Q_{B,C,n+1} := Q_3$. It is routine to check Conditions (1), (2), and (3). This concludes the construction for the case where both A and C are involved in $\langle F, f\rangle$.

By construction, $\{Q_{B,C,n} \mid n \in \mathbb{N}\}$ forms a chain. By Theorem 2, it has a unique limit in $\mathsf{Tree}(B\|C)$, call it $Q_{B,C}$. We apply Theorem 3 to conclude $\mathit{\Pi}(Q_{A,C}) = \mathit{\Pi}(Q_{B,C})$. □

E Modeling the Chor-Israeli-Li Algorithm

Distributed consensus is a fundamental problem in distributed computing and there exists a large body of research on finding efficient solutions under various

assumptions. In this section, we demonstrate that the Chor-Israeli-Li consensus algorithm can be modeled very naturally in our PIOA framework.

Consider a system of n asynchronous processes, call them $P_1, \ldots, P_n$. These processes communicate via *single-writer multi-reader* registers: every register can be written to by exactly one process and can be read by all processes. Each process is given an initial preference as input and an arbitrary number of processes may fail without being detected. The three correctness requirements are as follows.

(1) *Validity*: the final decision value of any non-faulty process P_i must be the initial input of some process P_j.
(2) *Agreement*: if both processes P_i and P_j terminate successfully, then their decision values must coincide.
(3) *Wait-free (probabilistic) termination*: for each non-faulty process P_i, the probability of P_i terminating successfully after a finite number of steps is 1.

Now we introduce our PIOA model. Let I denote the index set $\{1, 2, \ldots, n\}$ and let V denote the domain of preference values. For each $i \in I$, v_i denotes be the input value of process P_i. Notice, by assumption, $v_i \in V$ for all $i \in I$. Finally, define $\mathbb{N}^\infty := \mathbb{N} \cup \{\infty\}$.

We model all n shared registers using a single PIOA: the *register* automaton. This automaton has two internal variables:

- $\mathit{register} : I \to (V \times \mathbb{N}^\infty) \cup \{\bot\}$ (initially $\mathit{register}(i) = \bot$ for all $i \in I$), and
- $\mathit{request} : I \times I \to \{\mathit{true}, \mathit{false}\}$ (initially $\mathit{request}(i, j) = \mathit{false}$ for all $i, j \in I$).

The action signature of the register automaton is as follows:

- *Input*: $\{\mathit{read}_{i,j}, \mathit{write}_i(v) \mid i, j \in I \wedge v \in V \times \mathbb{N}^\infty\}$, and
- *Output*: $\{\mathit{ack}_{i,j}(v) \mid i, j \in I \wedge v \in (V \times \mathbb{N}^\infty) \cup \{\bot\}\}$.

Notice that there are no internal actions. Figure 3 shows a specification of the register automaton where the transition bundles are written in an adapted precondition-effect style. Recall that each input or hidden bundle must contain exactly one fibre, so we skip the probability clause for those bundles.

Each process P_i is modeled by a single PIOA with 4 state variables:

- $\mathit{pref}_i \in V$ (initially v_i),
- $\mathit{round}_i \in \mathbb{N}^\infty \cup \mathbf{q}(\mathbb{N}^\infty)$ (initially 0),
- $\mathit{status}_i \in \{-1, 0, 1, \ldots, 2n+1\}$ (initially 0), and
- $\mathit{mem}_i : I \to (V \times \mathbb{N}^\infty) \cup \{\bot\}$ (initially $\mathit{mem}_i(j) = \bot$ for all $j \in I$).

Here $\mathbf{q}$ is a constant symbol. Whenever $\mathit{round}_i = \mathbf{q}(r)$, the process P_i is in round r with probability $\frac{n-1}{n}$ and in round $r+1$ with probability $\frac{1}{n}$. The action signature of P_i is as follows.

- *Input*: $\{\mathit{ack}_{i,j}(v) \mid j \in I \wedge v \in (V \times \mathbb{N}^\infty) \cup \{\bot\}\}$.
- *Output*: $\{\mathit{read}_{i,j}, \mathit{write}_i(v) \mid j \in I \wedge v \in V \times \mathbb{N}^\infty\}$.
- *Hidden*: $\{\mathit{crash}_i, \mathit{compute}_i\}$.

bundle (Input)
  fiber $write_i(v)$
    eff: $register(i) := v$

bundle (Input)
  fiber $read_{i,j}$
    eff: $request(i,j) := true$

bundle (Output)
  pre: $request(i,j) = true$
  fiber $ack_{i,j}(register(j))$
    prob: 1
    eff: $request(i,j) := false$

Fig. 3. The bundle specification for the register automaton.

Figure 4 shows the specification of process P_i. Note that P_i terminates with decision value v by writing (v, ∞) to its register. Moreover, for clarity of presentation, the *compute*() procedure in bundle $compute_i$ is specified separately in Figure 5. Finally, we use the following abbreviations:

- $maxround := \max(\{\pi_2(mem_i(j)) \mid j \in I \wedge mem_i(j) \neq \bot\})$,
- $L := \{j \in I \mid \pi_1(mem_i(j)) = maxround\}$, and
- $AL := \{j \in I \mid \pi_1(mem_i(j)) = maxround - 1\}$.

This model is translated almost literally from [CIL94]. Upon our initial study, we discovered that a process P_i may in fact terminate in round 0, before all n processes join the protocol (i.e., some process P_j is still "sleeping"). This can lead to a violation of the agreement condition. We believe this error can be corrected by strengthening the termination condition so that P_i never decides in round 0. That is, we add an extra conjunct $round_i > 0$ to the second **if**-clause in Figure 5.

Our modeling work so far is clearly elementary. We have yet to provide formal correctness proofs. Typically, the two non-probabilistic correctness conditions (namely, validity and agreement) follow easily from invariant-style reasoning. For probabilistic termination, there are two obvious approaches:

- direct manual verification, following the correctness proof given in [CIL94];
- mechanized verification, similar to the approach taken in [KNS01] in verifying the strong adversary consensus protocol of [AH90].

Finally, it may also be possible to reason in a game-theoretic setting, viewing the algorithm as an $n+1$-player game where the n protocol parties collaborate against the last player, the adversary. The potential of this last approach remains to be discovered. The obvious attraction of game theory is the abundance of existing theorems for stochastic games. Moreover, the notion of information sharing found in game theory seems nicely compatible with our philosophical motivations.

bundle (Hidden)
  pre: true
  fiber crash_i
    eff: $\mathit{status}_i := -1$

bundle (Output)
  pre: $\mathit{status}_i = 0 \wedge \mathit{round}_i \in \mathbb{N}^\infty$
  fiber $\mathit{write}_i(\mathit{pref}_i, \mathit{round}_i)$
    prob: 1
    eff: $\mathit{status}_i := 1$

bundle (Output)
  pre: $\mathit{status}_i = 0 \wedge \mathit{round}_i = \mathbf{q}(r)$, where $r \in \mathbb{N}^\infty$
  fiber $\mathit{write}_i(\mathit{pref}_i, r+1)$
    prob: $\frac{1}{2n}$
    eff: $\mathit{status}_i := 1, \mathit{round}_i := r+1$
  fiber $\mathit{write}_i(\mathit{pref}_i, r)$
    prob: $\frac{2n-1}{2n}$
    eff: $\mathit{status}_i := 1, \mathit{round}_i := r$

bundle (Output)
  pre: $\mathit{status}_i \in \{1, 3, 5, \ldots, 2n-1\} \wedge \mathit{round}_i \neq \infty$
  fiber $\mathit{read}_{i,(\mathit{status}_i+1)/2}$
    prob: 1
    eff: $\mathit{status}_i := \mathit{status}_i + 1$

bundle (Input)
  fiber $\mathit{ack}_{i,j}(v)$
    eff: $\mathit{mem}_i(j) := v, \mathit{status}_i := \mathit{status}_i + 1$

bundle (Hidden)
  pre: $\mathit{status}_i = 2n+1$
  fiber $\mathit{compute}_i$
    eff: $\mathit{status}_i := 0, \mathit{compute}()$

Fig. 4. The bundle specification for protocol party P_i.

if $\exists_j \in I\ \mathit{mem}_i(j) = (v, \infty)$ **then**
    $\mathit{pref}_i := v, \mathit{round}_i := \infty$
else if $i \in L \wedge \exists_v \forall_{j \in L \cup AL}\ \pi_1(\mathit{mem}_i(j)) = v$ **then**
    $\mathit{pref}_i := v, \mathit{round}_i := \infty$
else if $i \in L$ **then**
    $\mathit{round}_i := \mathbf{q}(\mathit{round}_i)$
else if $\mathit{maxround} - \mathit{round}_i \leq 2$ **then**
    $\mathit{round}_i := \mathit{round}_i + 1$
    if $\exists_v \forall_{j \in L}\ \pi_1(\mathit{mem}_i(j)) = v$ **then**
        $\mathit{pref}_i := v$
else
    $\mathit{round}_i := \mathit{maxround} - 2$
    $\mathit{pref}_i := \pi_2(\mathit{mem}_i(\min(L)))$

Fig. 5. The $\mathit{compute}()$ procedure.