Characterizing History Independent Data Structures

Jason D. Hartline* Edwin S. Hong† Alexander E. Mohr‡

William R. Pentney§ Emily C. Rocke§

Abstract

We consider history independent data structures as proposed for study by Naor and Teague [3]. In a history independent data structure, nothing can be learned from the memory representation of the data structure except for what is available from the abstract data structure. We show that for the most part, strong history independent data structures have canonical representations. We provide a natural alternative definition of strong history independence that is less restrictive than [3] and characterize how it restricts allowable representations. We also give a general formula for creating dynamically resizing history independent data structures and give a related impossibility result.

Keywords: *Data structures, history independence, markov chains, algorithms.*

*Microsoft Research, 1065 La Avenida, Mountain View, CA 94043. Email: hartline@microsoft.com. Work done while author was at University of Washington.

†Computing & Software Systems, University of Washington, 1900 Commerce St., Tacoma, WA 98402. Email edhong@u.washington.edu

‡Stony Brook University. Email: amohr@cs.sunysb.edu. Work done while author was at University of Washington.

§Department of Computer Science, University of Washington, Seattle, WA 98195. Email: {bill,ecrocke}@cs.washington.edu

1 Introduction

On April 16, 2000, the New York Times published an article regarding the CIA's role in the overthrow of the Iranian government in 1953. In addition to the article, the Times' website posted a CIA file from 1954 that detailed the actions of various revolutionaries involved in the plot. The Times opted to black out many of the names mentioned in the document; some of the people referred to were still alive and residing in Iran, and could have been put at risk for retribution. The file was published as an Adobe PDF file that contained the original document in its entirety and an overlay covering up parts of the document. Shortly after releasing the document, some Internet users reverse engineered the overlay and made the original document available on the Web. In an environment where information is valuable, private, incriminating, etc., the Times' blunder represents a particularly grievous instance of the problems caused by data structures that retain information about previous operations performed upon them. History independent data structures are designed not to reveal any information beyond that necessarily provided by the contents of the data structure.

The idea of maintaining a data structure so that no extraneous information is available was first explicitly studied by Micciano [2]. This work studied "oblivious trees" where no information about past operations could be deduced from the *pointer structure* of the nodes in the search tree. In [4] Snyder studied bounds on the performance of search, insert, and delete functions on uniquely represented, in terms of the pointer structure, tree based data structures. More stringent history independence requirements were studied by Naor and Teague in [3]. In their model, the entire memory representation of a history independent data structure, not just the pointer structure, must not divulge information about previous states of the data structure. Following [3] we consider two types of history independence: *weak history independence*, in which we assume that a data structure will only be observed once; and *strong history independence*, in which case the data structure may be observed multiple times. A data structure is history independent if nothing can be learned from the data structure's memory representation during these observations except for the current abstract state of the data structure.

In Section 3 we give a simple definition of strong history independence. We show in Section 6 this definition equivalent to that of [3]. Under this definition, strong history independent implementations of many data structures must satisfy a natural canonicality criterion (Sections 4). For example, a strongly history independent implementation of a hash table has the property that up to randomness in the initialization of the hash table, e.g., the choice of hash functions, the hash table's representation in memory must be deterministically given by its contents. This answers an open question posed in [3] about the necessity of canonical representations.

In Section 5 we consider a natural relaxation of strong history independence, where non-canonical representations and randomness can be used. However, we show that even under this less restrictive definition, there are still very stringent limitations on using non-canonical representations.

Finally, in Section 7 we discuss the issue of creating dynamically resizing history independent data structures. In [3], Naor and Teague presented a weakly history independent dynamically resizable hash table. The scheme they employ can be generalized to give a technique for making any history independent data structure dynamically resizable using amortized constant time against an oblivious adversary.[1] We give a general technique for dynamically resizing weak history independent data structures in amortized constant time against a non-oblivious adversary. We prove that no such technique exists for strongly history independent dynamically resizing data structures. This result provides insight into the open problem of whether there is a complexity separation between weak and strong history independence [3].

[1] We use two standard variants for analysis of data structures that use randomness. We consider worst case performance on an sequence of operations created by an oblivious adversary, one that has no knowledge of the random coin flips in our data structure. Similarly, we consider worst case performance on a sequence of operations constructed on the fly by an non-oblivious adversary, one that knows our random coin flips. A non-oblivious adversary, for instance, can cause a hash table with a randomized hash function to perform poorly by causing all inserted elements to hash to the same location.

2 Preliminaries

The results presented in this paper apply to history independent data structures in general. To this end we must have a general understanding of data structures. An *abstract data structure* defines the set of operations for a data structure and its semantics. A data structure's *state* is the current value or contents of the data structure as specified by its abstract data structure. A data structure's *representation* in memory for any given state is the physical contents of memory that represent that state. An *implementation* of a data structure gives a map from any valid representation/operation pair to a new representation (and possible output).

We assume that our abstract data structure is deterministic.[2] That is, each operation takes one state deterministically to another state. Define the *state transition graph* to be the directed graph induced on states (as vertices) of the data structure by the operations (directed labeled edges). It is useful to consider the following trichotomy of data structures according to properties their state transition graphs:

- The graph may be acyclic (a DAG). This is the case if it is not possible to return to a state once visited. Examples are the union-find data structure [5] and hash tables that do not support the delete operation.

- The graph may be strongly connected, i.e., all states are mutually reachable. Hash tables (with delete operation), queues, stacks, etc. are all examples of such data structures. We define these (below) as *reversible* data structures because they do not support irreversible operations.

- Otherwise, the graph is a combination of the above consisting of strongly connected components that are interconnected acyclically.

Definition 1 *A data structure is* reversible *if its state transition graph is strongly connected.*

Let A, B, and C represent states of the data structure (i.e., vertices in the graph). Let X and Y represent sequences of operations on the data structure (i.e., directed paths in the state transition graph) with $[X^k]$ being the sequence of operations of X repeated k times and $[X, Y]$ the sequence of operations consisting of the operations of X followed by the operations of Y. We say that B is *reachable* from A, notated $A \rightarrow B$, if some non-empty sequence of operations takes state A to state B. If X is such a sequence of operations, we say $A \xrightarrow{X} B$. If A is reachable from B and B is reachable from A then A and B are *mutually reachable*, notated $A \rightleftharpoons B$. This is synonymous with saying that states A and B are in the same strongly connected component of the state transition graph. We say $\oslash \xrightarrow{X} A$ if, on data structure initialization, the sequence of operations X produces state A.

Note though that if $A \rightleftharpoons B$ for some B, then $A \rightleftharpoons A$; though in general, a state A is not necessarily mutually reachable with itself (E.g., if the state transition graph is acyclic). We call a state not mutually reachable with itself a *transient* state.

The data structures that we consider may use randomization in choosing which representation to use for any given state. In the context of history independent data structures, randomization over representations is useful both for efficiency and for maintaining history independence, as we will describe below.

Let $\mathbf{a}$ and $\mathbf{b}$ denote representations of states A and B respectively. It will be convenient to view a state as the set of representations that represent that state. Thus we adopt the notation, $\mathbf{a} \in A$, to mean that $\mathbf{a}$ is a representation of A. Let X be such that $A \xrightarrow{X} B$. Let $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right]$ denote the probability that, starting from representation $\mathbf{a}$ of A, the sequence X of operations on the data structure yields representation $\mathbf{b}$ of B. We say $\mathbf{b}$ is *reachable* from $\mathbf{a}$, denoted $\mathbf{a} \rightarrow \mathbf{b}$, if there is some non-empty sequence of operations X such that $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right] > 0$. We define *mutual reachability* for representations as $\mathbf{a} \rightleftharpoons \mathbf{b}$ if $(\mathbf{a} \rightarrow \mathbf{b}) \wedge (\mathbf{b} \rightarrow \mathbf{a})$. We say representation $\mathbf{a}$ of state A is *reachable* if it is reachable from data structure startup, i.e., $\oslash \rightarrow \mathbf{a}$. We henceforth only consider representations that are reachable.

[2] Our results can be extended to randomized abstract data structures by noting that they are equivalent to deterministic abstract data structures when the user picks which operation to apply randomly.

For convenience we will define the notation $\langle \mathbf{a} \rangle$ for a representation $\mathbf{a}$ of state A as the set of representations of A that are mutually reachable by $\mathbf{a}$. If A is transient then we define $\langle \mathbf{a} \rangle = \{\mathbf{a}\}$.

The representation of the data structure encapsulates everything known about it. Therefore $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right]$ must be independent of any states or representations of the data structure prior to it entering representation $\mathbf{a}$ of state A. Thus, a data structure behaves like a Markov chain on its representations except that the transition probabilities are based on which operation is performed. This is formalized in Fact 1.

Fact 1 *If* $A \xrightarrow{X} B$ *and* $B \xrightarrow{Y} C$ *then* $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b} \xrightarrow{Y} \mathbf{c}\right] = \mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right] \cdot \mathbf{Pr}\left[\mathbf{b} \xrightarrow{Y} \mathbf{c}\right]$.

In general, there may be multiple representations of the same state in a data structure. A data structure uses *canonical representations* if it has only one representation per state.

3 History independence

The goal of history independence is to prevent information from being leaked though the representation of a data structure in the case that it is observed by an outside party. As such, the requirements for history independence depend on the nature of the potential observations. Following [3] we define weak history independence for the case where the data structure is only observed once, e.g., when a laptop is lost. Alternatively, a strong history independent data structure allows multiple observations of the data structure to be interspersed throughout a sequence of operations without giving any information about the operations performed between the observations beyond that implied by the states observed.

Definition 2 (Weak History Independence) *A data structure implementation is* weakly history independent *if, for any two sequences of operations X and Y that take the data structure from initialization to state A, the distribution over memory after X is performed is identical to the distribution after Y. That is:*

$$(\varnothing \xrightarrow{X} A) \wedge (\varnothing \xrightarrow{Y} A) \implies \forall\, \mathbf{a} \in A, \quad \mathbf{Pr}\left[\varnothing \xrightarrow{X} \mathbf{a}\right] = \mathbf{Pr}\left[\varnothing \xrightarrow{Y} \mathbf{a}\right].$$

Definition 3 (Strong History Independence (SHI)) *A data structure implementation is strongly history independent if, for any two (possibly empty) sequences of operations X and Y that take a data structure in state A to state B, the distribution over representations of B after X is performed on a representation* $\mathbf{a}$ *is identical to the distribution after Y is performed on* $\mathbf{a}$. *That is:*

$$(A \xrightarrow{X} B) \wedge (A \xrightarrow{Y} B) \implies \forall\, \mathbf{a} \in A,\ \forall\, \mathbf{b} \in B, \quad \mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right] = \mathbf{Pr}\left[\mathbf{a} \xrightarrow{Y} \mathbf{b}\right].$$

Here, A may be the null (pre-initialization) state, $\varnothing$, in which case $\mathbf{a}$ is the empty representation, $\varnothing$. Thus, strongly history independent data structures are a subset of weakly history independent data structures. Although this definition differs from the arguably more complex one given by Naor and Teague [3], we show in Section 6.2 that they are in fact equivalent.

Properties of "$\rightleftharpoons$" and history independence

For strong history independent data structures, because the probability $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right]$ does not depend on the path X; we can introduce the notation $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}]$ to mean $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right]$ for any X taking $A \xrightarrow{X} B$. We now discuss some useful properties of "$\rightleftharpoons$" and "$\to$" on strong history independent data structures.

Transitivity of "$\to$" and "$\rightleftharpoons$": $(\mathbf{a} \to \mathbf{b}) \wedge (\mathbf{b} \to \mathbf{c}) \implies (\mathbf{a} \to \mathbf{c})$, *and similarly,* $(\mathbf{a} \rightleftharpoons \mathbf{b}) \wedge (\mathbf{b} \rightleftharpoons \mathbf{c}) \implies (\mathbf{a} \rightleftharpoons \mathbf{c})$.

Reflexivity: *Under weak history independence,* $\mathbf{a} \rightleftharpoons \mathbf{a}$ *if and only if* $A \rightleftharpoons A$.

Symmetry of "$\rightarrow$": *Under strong history independence, for states A and B with $A \rightleftharpoons B$, $\mathbf{a} \rightarrow \mathbf{b} \Longrightarrow \mathbf{b} \rightarrow \mathbf{a}$.*

Transitivity follows immediately from the definition of reachable. While we only use these properties for our results pertaining to strong history independent data structures, reflexitivity also holds true for weak history independent data structures and transitivity holds true regardless of history independence. See Appendix A for the proofs of reflexivity and symmetry.

Randomization and history independence

Randomization over representations is used in the context of history independent data structures for two purposes: efficiency and maintaining history independence. As an example of how randomness is used to maintain history independence, consider the weakly history independent array based queue implementation proposed in [3]. This is the standard array based queue implementation except that the initial location of the font of the queue is chosen uniformly at random from every array index. If observed once, the queue is equally likely to be located anywhere in the array. Thus, it is weakly history independent.

An example of the use of randomness for efficiency would be the choice of hash functions for a hash table. A good randomized choice of hash functions leads to expected constant time operations in an open addressing hash table [1].

Consider the case that an adversary (not the observer) is determining a sequence of operations to effect on the data structure to make it perform inefficiently. We will consider worst case bounds on efficiency against *oblivious* adversaries and *non-oblivious* adversaries. An oblivious adversary must specify the sequence of operations before the random choices are made by our algorithm. We aim to show that given any sequence of operations, the expected amortized runtime of each operation is low. Again, the hash table is an example of a data structure that performs well against an oblivious adversary.

A non-oblivious adversary, on the other hand, is allowed access to the random bits flipped during the running of the algorithm. Such an adversary can adapt the sequence of operations to the algorithm as it is running in order to make it perform poorly. For example, a hash table with randomized choice of hash functions does not perform well against a non-oblivious adversary because the adversary, knowing the choice of hash function, can choose to only insert elements that hash to the same location.

4 Canonical representations and SHI

Note that if a data structure has canonical representations for each state, it is necessarily SHI. An open question from [3] is whether the converse is true: does strong history independence necessarily imply that the data structure has canonical representations? In this section we answer the question by showing that up to randomization on transition between strongly connected components of the state transition graph the representations are canonical. For example, any strongly history independent implementation of a reversible data structure, e.g., a hash table, has canonical representations up to initial randomness, e.g., choice of hash functions.

Lemma 1 *Under SHI, if $\mathbf{a}$ and $\mathbf{a}'$ are both representations of state A with $\mathbf{a} \rightarrow \mathbf{a}'$, then $\mathbf{a} = \mathbf{a}'$.*

Proof: We show the contrapositive. Let E be the empty sequence of operations. For $\mathbf{a} \neq \mathbf{a}'$, we have $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{E} \mathbf{a}'\right] = 0$ because the data structure is not allowed to change unless an operation is performed. Thus, by SHI, $\mathbf{Pr}[\mathbf{a} \rightarrow \mathbf{a}'] = 0$ so $\mathbf{a} \not\rightarrow \mathbf{a}'$. □

Intuitively, this requires A to have a canonical representation after it is visited for the first time. Before the first visit it may have a distribution of possible representations. Now we extend the requirement for a canonical representation of A to the point where the data structure first hits a state reachable from A. By Lemma 1 and the transitivity of "$\rightarrow$" we have:

Corollary 1 *Under SHI, let* $\mathbf{a}$ *and* $\mathbf{b}$ *be representations of states* A *and* B *such that* $\mathbf{a} \rightleftharpoons \mathbf{b}$. *For all representations* $\mathbf{b}'$ *of* B, $\mathbf{a} \rightleftharpoons \mathbf{b}'$ *if and only if* $\mathbf{b}' = \mathbf{b}$.

Corollary 1 shows that after reaching the first state of a strongly connected component of the state transition graph, there is a chosen canonical representation for every other state in the component. This is a complete answer to the question of whether canonical representations are necessary for strong history independence. In particular we have the following specification of the corollary:

Theorem 1 *For a reversible data structure to be SHI, a canonical representation for each state must be determined during the data structure's initialization.*

Again, examples of reversible data structures, ones in which all states are mutually reachable, are hash tables (that include the delete operation), queues, stacks, etc.

5 Less restrictive SHI

The definition of strong history independence above is highly restrictive, requiring, as just seen, canonical representations of states (after some initial random choices) to qualify. As evident by the proof of Lemma 1, a key contributer to its restrictiveness is that we require that the observer not be able to distinguish a nonempty sequence of operations from an empty one. A natural question would be whether anything can be gained by relaxing this requirement. In the this section, we discuss a relaxation of strong history independence which allows the empty sequence to be distinguished from any other sequence of operations. We show that while this does allow for randomization over representations, the randomization allowed is still very restricted.

5.1 Definition

The following definition introduces a form of strong history independence more permissive than the SHI definition.

Definition 4 (SHI*) *A data structure implementation is strongly history independent if, for any two* nonempty *sequences of operations* X *and* Y,

$$(A \xrightarrow{X} B) \wedge (A \xrightarrow{Y} B) \implies \forall\, \mathbf{a} \in A,\ \forall\, \mathbf{b} \in B,\ \mathbf{Pr}\Big[\mathbf{a} \xrightarrow{X} \mathbf{b}\Big] = \mathbf{Pr}\Big[\mathbf{a} \xrightarrow{Y} \mathbf{b}\Big].$$

SHI* permits an observer to distinguish an empty operation sequence from a nonempty one without considering it to be a violation of history independence. However, given that the operation sequence is nonempty, the observer may still not glean information about how many operations were performed, or what the operations were, besides what is inherent in the observed state of the abstract data structure.

Arguably, SHI* is more practically useful than SHI. In essence, it is nearly as strong as SHI if operations on the data structure are expected to be considerably more frequent than observations of it. If the opposite were true, that observations are more frequent than operations, then the situation approaches constant surveillance, in which case the observer can simply record each state as it occurs, making history independence useless. From a theoretical standpoint, we have found this definition interesting to work with, since it permits data structures that can choose between several representations after each operation (in contrast to Theorem 1).

In the remainder of this section we show that, despite this added flexibility, there are still very strict requirements to which a SHI* data structure must adhere. For example, each state in a reversible SHI* data structure had a "canonical distribution" over representations that is chosen during the initialization process; each operation on the data structure must result in the representation being explicitly resampled from the state's canonical distribution. As an example, this precludes a hash table implementation from using a randomized method for resolving conflicts, as all such conflicts would have to be re-resolved after every operation.

5.2 Canonical Representation Distributions

Unlike in the case of SHI, under SHI* a single state A may have distinct representations $\mathbf{a}$ and $\mathbf{a}'$ with $\mathbf{a} \rightleftharpoons \mathbf{a}'$ (contrast with Lemma 1).

Lemma 2 *Under SHI*, if $\mathbf{a}$ and $\mathbf{a}'$ are both representations of state A with $\mathbf{a} \rightleftharpoons \mathbf{a}'$, then for any representation $\mathbf{b}$ of state B with $A \to B$, $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{a}' \to \mathbf{b}]$.*

Proof: For the case that $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{a}' \to \mathbf{b}] = 0$ the lemma is true. Thus assume without loss of generality that $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] > 0$.

Since $A \rightleftharpoons A$, let W be any nonempty series of operations taking $A \xrightarrow{W} A$. Let X be any nonempty series of operations taking $A \xrightarrow{X} B$. Consider the series of operations $Q_N = [(W)^N, X]$ as performed on representation $\mathbf{a}$. Let a_i be a random variable for the representation of state A after i performances of W and let $\mathcal{E}_N$ be the random event that $a_i = \mathbf{a}'$ for some $i \in \{1, \ldots, N\}$. First, by Fact 1 and SHI*, we have:

$$\mathbf{Pr}\left[\mathbf{a} \xrightarrow{Q_N} \mathbf{b} \;\mid\; \mathcal{E}_N\right] = \mathbf{Pr}[\mathbf{a}' \to \mathbf{b}] \,.$$

Now we show that $\lim_{N\to\infty} \mathbf{Pr}[\mathcal{E}_N] = 1$. As observed in Fact 1, once conditioned on the representation at a_{i-1} the value of a_i can not depend on anything but a_{i-1}. The series of representations $\{a_1, a_2, \ldots\}$ can thus be viewed as a first-order Markov chain, where the states of the Markov chain are the representations of state A reachable from $\mathbf{a}$. Since all such representations are mutually reachable, the Markov chain is irreducible.

We now employ the following basic property of irreducible Markov chains: *In the limit, if the expected number of visits to a state is unbounded then the probability that the state is visited approaches one.* Let $\alpha = \mathbf{Pr}[a_i = \mathbf{a}'] = \mathbf{Pr}[\mathbf{a} \to \mathbf{a}']$. The expected number of occurrences of $\mathbf{a}'$ after N steps is $\sum_{i=1}^{N} \mathbf{Pr}[a_i = \mathbf{a}'] = \alpha N$. Since α is constant, this is unbounded as N increases. Thus, $\lim_{N\to\infty} \mathbf{Pr}[\mathcal{E}_N] = 1$.

To complete the proof, by SHI*,

$$\begin{aligned}
\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] &= \mathbf{Pr}\left[\mathbf{a} \xrightarrow{Q_N} \mathbf{b}\right] \\
&= \mathbf{Pr}\left[\mathbf{a} \xrightarrow{Q_N} \mathbf{b} \;\mid\; \mathcal{E}_N\right] \mathbf{Pr}[\mathcal{E}_N] + \mathbf{Pr}\left[\mathbf{a} \xrightarrow{Q_N} \mathbf{b} \;\mid\; \neg\mathcal{E}_N\right] (1 - \mathbf{Pr}[\mathcal{E}_N]) \\
&= \mathbf{Pr}[\mathbf{a}' \to \mathbf{b}]\, \mathbf{Pr}[\mathcal{E}_N] + \mathbf{Pr}\left[\mathbf{a} \xrightarrow{Q_N} \mathbf{b} \;\mid\; \neg\mathcal{E}_N\right] (1 - \mathbf{Pr}[\mathcal{E}_N]) \,.
\end{aligned}$$

In the limit as N increases, this quantity approaches $\mathbf{Pr}[\mathbf{a}' \to \mathbf{b}]$. However, since SHI* guarantees that it is constant as N changes, it must be that $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{a}' \to \mathbf{b}]$. $\square$

We now give the main theorem of this section which shows that, among other things, reversible data structures have canonical distributions.

Theorem 2 (Canonical Distributions) *Under SHI*, for any $\mathbf{a}, \mathbf{b}, \mathbf{c}$ with $\mathbf{a} \rightleftharpoons \mathbf{b}$ and $\mathbf{a} \to \mathbf{c}$, $\mathbf{Pr}[\mathbf{a} \to \mathbf{c}] = \mathbf{Pr}[\mathbf{b} \to \mathbf{c}]$.*

Proof: First note that $\mathbf{b} \rightleftharpoons \mathbf{a} \to \mathbf{c}$ gives $\mathbf{b} \to \mathbf{c}$. By definition of "$\to$", there exists sequences of operations X and Y with $A \xrightarrow{X} B$ and $B \xrightarrow{Y} C$, respectively. For any such X and Y, $\mathbf{a} \xrightarrow{[X,Y]} \mathbf{c}$ must go through some representation $\mathbf{b}'$ of B that is in $\langle \mathbf{b} \rangle$ (By symmetry and transitivity of "$\rightleftharpoons$"). Thus,

$$\begin{aligned}
\mathbf{Pr}[\mathbf{a} \to \mathbf{c}] = \mathbf{Pr}\left[\mathbf{a} \xrightarrow{[X,Y]} \mathbf{c}\right] &= \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}' \xrightarrow{Y} \mathbf{c}\right] && \text{(By SHI}^*\text{)} \\
&= \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}'\right] \cdot \mathbf{Pr}\left[\mathbf{b}' \xrightarrow{Y} \mathbf{c}\right] && \text{(Fact 1)} \\
&= \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}'\right] \cdot \mathbf{Pr}\left[\mathbf{b} \xrightarrow{Y} \mathbf{c}\right] && \text{(Lemma 2)} \\
&= \mathbf{Pr}\left[\mathbf{b} \xrightarrow{Y} \mathbf{c}\right] \cdot \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}'\right] .
\end{aligned}$$

But X will always take $\mathbf{a}$ to some $\mathbf{b}' \in \langle \mathbf{b} \rangle$, so

$$\mathbf{Pr}[\mathbf{a} \to \mathbf{c}] = \mathbf{Pr}\left[\mathbf{b} \xrightarrow{Y} \mathbf{c}\right] \cdot 1 = \mathbf{Pr}[\mathbf{b} \to \mathbf{c}]\,.$$

$\square$

Definition 5 *For any representation* $\mathbf{b}$ *of state* B *with* $B \rightleftharpoons B$*, denote by* $\mathbf{Pr}[\mathbf{b}]$ *the probability* $\mathbf{Pr}[\mathbf{b} \to \mathbf{b}]$.[3] *When* $B \not\rightleftharpoons B$*, we define* $\mathbf{Pr}[\mathbf{b}] = 1$.

This definition is meaningful because Theorem 2 shows that $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{b}]$ for all $\mathbf{a}$ such that $\mathbf{a} \rightleftharpoons \mathbf{b}$.

5.3 Discussion

One interesting question we do not directly answer in this paper is that of whether there is a complexity separation between SHI and SHI*. The former requires a canonical representation, while the latter only requires that the canonical distribution of the representation be explicitly resampled after every operation. Were there no complexity separation between SHI and SHI*, then when looking for strong history independent data structures we could restrict our attention to data structures with canonical representations, i.e., SHI data structures.

A cursory exploration of this question shows that any SHI* implementation of a reversible data structure that uses no more than a constant number of random bits per operation can be transformed into a SHI implementation which has worst case complexity of the same order as the expected complexity of the SHI* data structure. This is apparent through the observation that simulating all possible outcomes of an operation with expected runtime t on a data structure with at most a constant c coin flips per operation has worst case runtime is $t2^c$ which is $O(t)$.

We allow randomness in the initialization process and remove the randomness from all subsequent operations as follows. Arbitrarily picking the lexicographically smallest representation as the canonical representation for each state we can get a SHI data structure by, for each operation, simulating all possible random choices and choosing the lexicographically smallest result. In doing this simulation we will need only $O(t)$ additional storage for the simulated changes, i.e., memory locations and new values. These simulated changes can then be directly compared in $O(t)$ time to find the lexicographically smallest one. The total runtime is $O(t)$ per operation.

6 Order of Observations

Now considering both SHI and SHI*, we show that the probability distribution over representations for a set of observed states is independent of the order that the states are observed.[4] Since the states must be observed in an order consistent with the state transition graph, this is only interesting for states that can be observed in a different order, i.e., states that are mutually reachable (in the same connected component of the state transition graph). As an example, for $\mathbf{a}$ and $\mathbf{b}$ such that $A \rightleftharpoons B$, this result implies that $\mathbf{Pr}[\emptyset \to \mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\emptyset \to \mathbf{b} \to \mathbf{a}]$. We use this result to show that our definition of strong history independence is in fact equivalent to that of [3].

6.1 Entrance probabilities

The next few lemmas show that the probability $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}]$ is factorable into a $\mathbf{Pr}[\mathbf{b}]$ and a probability that depends only on the mutually reachable representations of $\mathbf{b}$.

[3] Under the more restrictive SHI, $\mathbf{Pr}[\mathbf{b}] = 1$ for any $\mathbf{b}$.

[4] Since SHI is more restrictive than SHI* it suffices to show this result for SHI*.

Definition 6 *For any representation* $\mathbf{a} \in A$ *and* $\mathbf{b} \in B$ *with* $A \to B$*, define*

$$\mathbf{Pr}[\mathbf{a} \to \langle \mathbf{b} \rangle] = \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}[\mathbf{a} \to \mathbf{b}'] \,.$$

Note that if $\mathbf{a} \rightleftharpoons \mathbf{b}$ then $\mathbf{Pr}[\mathbf{a} \to \langle \mathbf{b} \rangle] = 1$.

Lemma 3 (Factorability of $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}]$) *For any representations* $\mathbf{a}$ *and* $\mathbf{b}$ *with* $A \to B$

$$\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{a} \to \langle \mathbf{b} \rangle] \cdot \mathbf{Pr}[\mathbf{b}] \,.$$

Proof: This is trivial if $B \not\rightleftharpoons B$. For the $B \rightleftharpoons B$ case we have (this first step is identical to the first step of the proof of Theorem 2):

$$\begin{aligned} \mathbf{Pr}[\mathbf{a} \to \mathbf{b}] &= \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}[\mathbf{a} \to \mathbf{b}'] \cdot \mathbf{Pr}[\mathbf{b}' \to \mathbf{b}] \\ &= \mathbf{Pr}[\mathbf{b} \to \mathbf{b}] \cdot \sum_{\mathbf{b}' \in \langle \mathbf{b} \rangle} \mathbf{Pr}[\mathbf{a} \to \mathbf{b}'] \\ &= \mathbf{Pr}[\mathbf{b}] \cdot \mathbf{Pr}[\mathbf{a} \to \langle \mathbf{b} \rangle] \,. \end{aligned}$$

□

Lemma 4 *For any representations* $\mathbf{a}$*,* $\mathbf{b}$*,* $\mathbf{c}$ *with* $A \to B$ *and* $\mathbf{b} \rightleftharpoons \mathbf{c}$*,*

$$\mathbf{Pr}[\mathbf{a} \to \langle \mathbf{b} \rangle] = \mathbf{Pr}[\mathbf{a} \to \langle \mathbf{c} \rangle] \,.$$

Proof: By the same argument as in Lemma 3, summing over all $\mathbf{c}' \in \langle \mathbf{c} \rangle$,

$$\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{b}] \cdot \mathbf{Pr}[\mathbf{a} \to \langle \mathbf{c} \rangle] \,.$$

Setting this equal to $\mathbf{Pr}[\mathbf{b}] \cdot \mathbf{Pr}[\mathbf{a} \to \langle \mathbf{b} \rangle]$ (Lemma 3) gives the proof. □

6.2 Probabilities under permutation

We now show that the joint distribution of the representations any sequence of observed states does not depend on the order that the states are observed in.

Theorem 3 *Under strong history independence (SHI or SHI*), given a sequence of observed states* $S_1, \ldots, S_n$ *and a permutation (given by* π*) that gives a sequence of states* $S_{\pi_1}, \ldots, S_{\pi_n}$ *that is consistent with the state transition graph, we have*

$$\mathbf{Pr}[\oslash \to \mathbf{s}_1 \to \mathbf{s}_2 \to \ldots \to \mathbf{s}_n] = \mathbf{Pr}[\oslash \to \mathbf{s}_{\pi_1} \to \mathbf{s}_{\pi_2} \to \ldots \to \mathbf{s}_{\pi_n}]$$

with $\mathbf{s}_i$ *denoting a representation of state* S_i *for all* i*.*

Proof: From $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ pick a subsequence $\mathbf{r}_1 = \mathbf{s}_{i_1}, \mathbf{r}_2 = \mathbf{s}_{i_2}, \ldots, \mathbf{r}_k = \mathbf{s}_{i_k}$ (with $i_j < i_{j+1}$) that has exactly one representative from each group of mutually reachable representations, as well as all representations of transient states. Note that for all j, $S_j \to S_{j+1}$, because sequence $S_1, \ldots, S_n$ is consistent with the with the state transition graph. Furthermore, because there is only one representative per group of mutually reachable representations, $S_{i_{j+1}} \not\to S_{i_j}$. Since $S_{\pi_1}, \ldots, S_{\pi_n}$ must be consistent with the state transition graph, i.e., $S_{\pi_j} \to S_{\pi_{j+1}}$, it must be that π respects the order of $i_1, \ldots, i_k$ (i.e., for all j, $\pi_{i_j} < \pi_{i_{j+1}}$).

Using Theorem 2 and Lemmas 3 and 4, it is easy to see that $\mathbf{Pr}[\oslash \to \mathbf{s}_1 \to \mathbf{s}_2 \to \ldots \to \mathbf{s}_n]$ is the product of two factors: one that depends only on the transition probabilities between the representative states,

$$\prod_{i=1}^{k} \mathbf{Pr}[\mathbf{r}_i \to \langle \mathbf{r}_{\mathrm{i}+1} \rangle] \,,$$

and one that depends only on the characteristic probabilities of the states themselves,

$$\prod_{i=1}^{n} \mathbf{Pr}[\mathbf{s}_j] \, .$$

Note that the first factor depends on the order of the representative states, which is the same for both permutations as only one order is possible, and the second does not depend on order at all. Therefore $\mathbf{Pr}[\emptyset \to \mathbf{s}_{\pi_1} \to \mathbf{s}_{\pi_2} \to \ldots \to \mathbf{s}_{\pi_n}]$ can be expressed as the product of the same two factors. This proves the theorem. □

Consider the following alternative definition of strong history dependence as given by Naor and Teague [3].

Definition 7 (NT-SHI and NT-SHI*) *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be sequences of operations and let $P_1 = \{i_1^1, \ldots, i_\ell^1\}$ and $P_2 = \{i_1^2, \ldots, i_\ell^2\}$ be two lists of observation points such that for all $b \in \{1, 2\}$ and $1 \leq j \leq \ell$ we have $1 \leq i_j^b \leq |\mathcal{S}_b|$ and the state following the i_j^1 prefix of $\mathcal{S}_1$ and the i_j^2 prefix of $\mathcal{S}_2$ are identical (for NT-SHI* we further require that for $j \neq k$ that $i_j^b \neq i_k^b$). A data structure implementation is* strongly history independent *if for any such sequences the joint distribution over representations at the observation points of P_1 and the corresponding points of P_2 are identical.*[5]

Note that this definition allows the observations of the operations to be made out of order, i.e., with $i_j^b > i_{j+1}^b$. It is for this reason that Theorem 3 is required to show that SHI and NT-SHI are equivalent.

Corollary 2 *The definition of SHI (resp. SHI*) is equivalent to the definition of NT-SHI (resp. NT-SHI*), the strong history independence definition of Naor and Teague [3].*

7 Dynamic Resizing Data Structures

Many array based data structures use dynamic resizing techniques of doubling or halving in size as they grow or shrink. The open addressing hash table or the array based queue are classic examples. This technique combined with an amortized analysis yields data structures with amortized constant time operations. A weakly history-independent hash table that dynamically resizes is presented in [3]. The resizing scheme for the hash table generalizes to convert any weakly history independent array-based data structure with constant time operations and linear time resize into a dynamically resizing one with amortized constant time per operation against an oblivious adversary. However, one might wonder whether a strongly history independent data structure can be dynamically resized while maintaining its strongly history independent nature. In addition to providing the minor details necessary to modify Naor and Teague's (oblivious) dynamic resizing method to preserve strong history independence, we provide the following new results for non-oblivious adversaries:

- There is a general method for making any history independent data structure with constant time operations and linear time resize into a weakly history independent dynamically resizing data structure with amortized constant time operations against a non-oblivious adversary.

- In contrast, there is no general method for making a strongly history independent array based data structure with a linear time resize operation into a strongly history independent dynamically resizing data structure with amortized constant time operations against a non-oblivious adversary.

We will tailor our description of these techniques to data structures with explicitly defined size, capacity, and unit insertion and unit deletion operations such as a hash table or an array based queue. Note that if there is no deletion operation then the standard approach of doubling the capacity when the data structure is full, i.e. when the size is equal to the capacity, is efficient for non-oblivious adversaries. If there is no insertion operation then the data structure cannot grow and resizing is irrelevant. Let n denote the size of the data structure and N the capacity.

[5][3] does not define NT-SHI*. We include it here as the natural relaxation to nonempty sequences for comparison to SHI*.

The results given below for strong history independent data structures apply for both SHI and SHI*. For the lower bounds given below we can assume the least restrictive form of strong history independence, i.e., SHI*; likewise, for positive result we consider the most restrictive form of strong history independence, i.e., SHI. Though, in the discussion and proofs this distinction will be irrelevant.

Oblivious Adversary

One way of maintaining the strong history independence of an array-based data structure when making it dynamically resizable is to maintain a canonical size N for the data structure that depends only upon n, the number of items stored. As an example, consider the strategy where we double the size on inserts when $n = N$ (i.e. when it is full), and we halve the size on removes when $n = N/2 + 1$ (i.e. when it is half full). With this strategy, N is always the smallest power of two that is at least n. The existence of this canonical size implies the strong history independence of this scheme. Unfortunately, it is not efficient: when there are $n = N = 2^i$ elements in the data structure, repeated insert and removal operations will force continual linear time resizes.

Naor and Teague show in [3] that we can modify the above strategy to resize at random values that are spaced approximately at power-of-two intervals instead of exactly at powers of two. This gives an efficient method for resizing (against an oblivious adversary) that can maintain the history independence of the underlying data structure.

There are many adequate ways to compute random values that are approximately spaced at power-of-two intervals. The approach we use is to fix a U chosen uniformly at random from $[0, 1]$ and set resize thresholds at $t_i = 2^{i+U}$. We maintain that the size of the data structure, n, and capacity, N, satisfy $N = t_i$ when $n \in (2^{i+U-1}, 2^{i+U}]$. When $N = t_i$, we store the bounds t_{i-1} and t_i in the data structure to simplify the process of deciding when to resize. For weakly history independent data structures, we could compute a different U for each t_i as they were needed only storing t_{i-1} and t_i. For satisfying strong history independence, however, it is important to fix the resize locations in advance so that the canonical size N for every n will always be the same as n grows and shrinks. We do this by fixing a value for U when the data structure is created, and continually reusing U.

Note that U needs to be arbitrarily precise to correctly foil an adversary for arbitrarily large n. If such precision is required, we will need to store arbitrarily many random bits in memory when the data structure is created so that arbitrarily large t_is can be computed as they are needed.

Just like the resizing method in [3], any oblivious adversary will not know the resize locations and cannot make our resizing algorithm take more than amortized constant time per operation. This technique allows us to convert any strongly history independent array-based data structure with linear time resize into a dynamically resizable one that has an amortized time per operation equivalent to the time per operation of the original data structure up to a constant factor.

Non-oblivious Adversary and Weak History Independence

The technique of using canonical sizes against a non-oblivious adversary does not work. The adversary is assumed to know the coin flips of the algorithm and thus knows $T = t_0, t_1, \ldots$ and for any i can pick t_i and perform t_i insert operations putting the data structure at the brink of resizing. It can then repeatedly invoke insert and delete operations, each taking linear time.

We now consider the non-oblivious adversary and weak history independence. We show how to make any fixed capacity (weak or strong) history independent array-based data structure with a linear time resize operation into an amortized constant time, dynamically resizing, weakly history independent data structure. Once again, n and N denote the size and the capacity of the data structure, respectively. The principle behind this method is to make N be a random variable that is uniform on $\{n, \ldots, 2n - 1\}$ and show that is it possible to maintain this distribution while only resizing with probability $O(1/n)$ per operation.

We show below how to modify the insert and delete function of any data structure to resize when necessary to retain the uniform distribution on $\{n, \ldots, 2n - 1\}$ as n changes. Although N is a random variable the actual value of N is known when working with the data structure.

Insert:

1. if $N = n$
   - Resize data structure to size uniform on $\{n+1, \ldots, 2(n+1)-1\}$.
2. Otherwise (i.e., $N > n$)
   - With probability $2/(n+1)$ resize to $N = 2n$ or $N = 2n+1$, that is:
   $$N \leftarrow \begin{cases} 2n & \text{with probability } 1/(n+1) \\ 2n+1 & \text{with probability } 1/(n+1) \\ \text{no change} & \text{otherwise.} \end{cases}$$
3. Insert new item.

To show correctness we must show that given a call to insert with N uniform on $\{n, \ldots, 2n-1\}$, after insert the new capacity is uniform $\{n+1, \ldots, 2(n+1)-1\}$. Clearly if step 1 occurs, the new capacity is uniform as desired. On the other hand, if step 2 occurs the probability that N is unchanged is $1-2/(n+1) = (n-1)/(n+1)$. Since each of these $n-1$ possible values for the old N is equally likely, the probability that N is any one of them is $1/(n+1)$. Clearly $\Pr[N = 2n] = \Pr[N = 2n+1] = 1/(n+1)$. Thus, the new N is uniformly distributed over $\{n+1, \ldots, 2n+1\}$.

We now show that the runtime of our insert (due to resizing) is expected constant time. Assuming that a resize takes linear time, we need the probability of resize of be $O(1/n)$. Step 1, which always incurs a resize, occurs with probability $1/n$. Step 2 incurs a resize with probability $2/(n+1)$. Thus the total probability of resize is less than $3/n$; since the resize operation is linear in n, the expected time spent in resizing is O(1).

Given a data structure of size $n+1$ with N uniformly distributed over $\{n+1, \ldots, 2n+1\}$. We give the following delete operation which yields a data structure with N uniform over $\{n, \ldots, 2n-1\}$

Delete:

1. Delete the item.
2. if $N \geq 2n$
   - Resize data structure to size uniform $\{n, \ldots, 2n-1\}$.
3. otherwise (i.e., $N < 2n$)
   - With probability $1/n$ resize to $N = n$, that is:
   $$N \leftarrow \begin{cases} n & \text{with probability } 1/n \\ \text{no change} & \text{otherwise.} \end{cases}$$

The correctness and complexity of this delete with resizing is similar to the insert, and we omit the details.

Non-oblivious Adversary and Strong History Independence

The technique employed in the previous section for making amortized dynamically resizing weak history independent data structures fails when strong history independence is required. The technique described maintains a canonical distribution of possible capacities for each n such that the probability is $O(1/n)$ that the capacity needs to be changed on an insert or delete (to maintain the correct distribution on capacities). However, strongly history independent data structures cannot use such a technique because in states that are mutually reachable, randomness over choice of representation must be completely regenerated during each operation (Theorem 2).

We will show any strongly history independent data structure that has

- non-constant unamortized time resizes, and
- insert operations that can be undone in constant number of operations (i.e., delete),

has amortized non-constant time operations. Thus, there is no general technique for taking a constant time strongly history independent data structure with inserts and deletes that has a linear time resize operation for changing the capacity and making it resizable.

Consider, for example, the *deque* (double-ended queue) that supports operations *inject* and *eject* from the front of the deque and *push* and *pop* from the rear of the deque. The insert operations *inject* and *push* have corresponding delete operations *eject* and *pop*. A weakly history independent deque can be implemented in an array in the same way a *queue* is implemented in an array [3]. The implementation of a strongly history independent deque is an open question, as is the implementation of a queue. However, the result we now prove tells us that either there is a strongly history independent deque with constant time resize or there is no non-oblivious, amortized constant time resizing, strongly history independent deque. This would provide a separation result between strong history independence and weak history independence because amortized constant time resizable, weak history independent deques exist.

Theorem 4 *Any strongly history independent data structure that dynamically resizes (in non-constant unamortized time) and has a sequence of undoable insert operations has non-constant amortized resizes.*

Proof: Let N' be a large value (we will take it in the limit for this result). Consider any sequence of insert operations $X_1, \ldots, X_{N'}$ that we will perform in order on the data structure, taking it from state S_0 to state $S_{N'}$. Let $\bar{X}_i$ be the operation(s) that undo(es) X_i.

Note that any data structure in which operations are undoable is reversible. Thus, all states are mutually reachable, and once we initialize the data structure and end up in some representation $\mathbf{s}_0$ of state S_0, the reachable representations of state S_i are limited to those that are mutually reachable from $\mathbf{s}_0$, i.e., $\mathbf{s}_i$ such that $\mathbf{s}_0 \rightleftharpoons \mathbf{s}_i$. Furthermore, the probability that s_i is the representation of state S_i is exactly $\mathbf{Pr}[\mathbf{s}_0 \to \mathbf{s}_i]$ for either the case that we arrive in S_i from applying operation X_i from state S_{i-1} or from applying $\bar{X}_{i+1}$ from state S_{i+1}. Conditioned on $\mathbf{s}_0$ being the initial representation, let $p_i^{N'}$ be the probability that the representation of state S_i has capacity at least N' when performing X_i from S_{i-1} or $\bar{X}_{i+1}$ from S_{i+1}. By SHI*, $\mathbf{Pr}[\mathbf{s}_{i-1} \to \mathbf{s}_i] = \mathbf{Pr}[\mathbf{s}_0 \to \mathbf{s}_i]$. Thus,

$$p_i^{N'} = \sum_{\mathbf{s}' \in \langle \mathbf{s}_i \rangle} \{\mathbf{Pr}[\mathbf{s}_0 \to \mathbf{s}'] \ : \ \mathbf{s}' \text{ has capacity at least } N'\}.$$

All representations of state $S_{N'}$ have capacity at least N' because they have size N'. Thus, $p_{N'}^{N'} = 1$. Also, for N' suitably large, $p_0^{N'} = 0$. As such, there must be an k such that $p_k^{N'} < 1/2$ and $p_{k+1}^{N'} \geq 1/2$. The probability of resize on transition from S_k to S_{k-1}, given initial representation $\mathbf{s}_0$, is thus at least $(1 - p_k^{N'}) \times p_{k+1}^{N'} \geq 1/4$. As this resize takes (non-amortized) non-constant time, the sequence of operations given by $Y = [X_1, \ldots, X_k, [X_{k+1}, \bar{X}_{k+1}]^k]$ takes at least $O(k)$ times the unamortized cost of resize for $3k$ operations, which yields an amortized cost per operation on the same order as the cost of resize (which we have assumed to be non-constant). □

8 Conclusions

We have shown the relationship between strong history independence and canonical representations. In doing so we have proposed a new definition, SHI*, of strong history independence that allows non-empty sequences of operations to be distinguished from empty ones. We leave as an open question whether there is a complexity separation between the two, i.e., does there exist any interesting data structure that has an efficient SHI* implementation but not an efficient SHI implementation?

For a standard efficiency metric, there is a general technique for dynamically resizing weak history independent data structures. We have shown that efficient dynamic resizing under this metric is not possible for strong history independent data structures, but leave as an open question whether there exist strong history independent data structures that would benefit from such a resizing.

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[2] D. Micciancio. Oblivious data structures: Applications to cryptography. In *Proc. of 29th ACM Symposium on Theory of Computing*, pages 456–464, 1997.

[3] M. Naor. and V. Teague. Anti-persistence: History Independent Data Structures. In *Proc. of 33nd Symposium Theory of Computing*, May 2001.

[4] L. Synder. On Uniquely Represented Data Structures. In *Proc. of 28th Symposium on Foundations of Computer Science*, 1977.

[5] Robert E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.

A Properties of "$\rightarrow$"

Lemma 5 (Reflexivity) *Under weak history independence, for any state A with representation $\mathbf{a}$, $\mathbf{a} \rightleftharpoons \mathbf{a}$ if $A \rightleftharpoons A$.*

Proof: Given state $A \rightleftharpoons A$ with representation $\mathbf{a}$, suppose the contrary: $\mathbf{a} \not\rightleftharpoons \mathbf{a}$. Let W be any operation sequence with $\oslash \xrightarrow{W} A$, and X be any operation sequence with $A \xrightarrow{X} A$. Some such W and X must exist because all states are reachable and $A \rightleftharpoons A$. Consider the family of operation sequences $Q_N = [W, (X)^{N-1}]$ for $N \geq 1$. When sequence Q_N is performed, state A is visited N times. For $i \geq 0$, let a_i be a random variable for the representation of state A after $[W, (X)^i]$.

Because we assume $\mathbf{a} \not\rightleftharpoons \mathbf{a}$, and so $\mathbf{Pr}[\mathbf{a} \rightarrow \mathbf{a}] = 0$, the representation $\mathbf{a}$ can occur at most once in Q_N. Under weak history independence, $\mathbf{Pr}[a_i = \mathbf{a}] = \mathbf{Pr}\left[\oslash \xrightarrow{Q_i} \mathbf{a}\right]$ is the same for all i. Thus, let $\alpha = \mathbf{Pr}[a_i = \mathbf{a}]$. Let C be a random variable for the number of times that $a_i = \mathbf{a}$ during the performance of Q_N. We know that $0 \leq C \leq 1$ because $\mathbf{a}$ may occur at most once.

$$\mathbf{E}[C] = \sum_{i=0}^{N-1} \mathbf{Pr}[a_i = \mathbf{a}] = \alpha N.$$

But as N increases, $\mathbf{E}[C] = \alpha N$ will be larger than 1, which contradicts our assumption that $C \leq 1$. Thus we have $\mathbf{a} \rightarrow \mathbf{a}$. □

Theorem 5 (Symmetry of $\rightarrow$) *Under strong history independence, for states A and B with $A \rightleftharpoons B$, $\mathbf{a} \rightarrow \mathbf{b} \Longrightarrow \mathbf{b} \rightarrow \mathbf{a}$.*

Proof: Given states $A \rightleftharpoons B$, and representations $\mathbf{a}$ of A and $\mathbf{b}$ of B such that $\mathbf{a} \rightarrow \mathbf{b}$, suppose the contrary: $\mathbf{b} \not\rightarrow \mathbf{a}$.

Let X, Y, and Z be such that $\oslash \xrightarrow{W} A \xrightarrow{X} B \xrightarrow{Y} A$ and consider the family of operation sequences $Q_N = [W, (X, Y)^N]$ for $N \geq 1$. When sequence Q_N is performed, states A and B are visited $N + 1$ and N times, respectively. For $i \geq 1$ let b_i (resp. a_i) be a random variable for the representation of state B (resp. A) after $[W, (X, Y)^{i-1}, X]$ (resp. $[W, (X, Y)^i]$).

Let $\alpha = \mathbf{Pr}[\oslash \rightarrow \mathbf{a}] = \mathbf{Pr}[a_i = \mathbf{a}]$ and $\beta = \mathbf{Pr}[\mathbf{a} \rightarrow \mathbf{b}]$. Our assumption $\mathbf{b} \not\rightarrow \mathbf{a}$, i.e., $\mathbf{Pr}[\mathbf{b} \rightarrow \mathbf{a}] = 0$, implies that representation $\mathbf{a}$ can never follow representation $\mathbf{b}$. Since any time $a_i = \mathbf{a}$ we have an independent (by Fact 1) probability β that $b_{i+1} = \mathbf{b}$, this also implies that each occurrence of $\mathbf{a}$ has an independent probability of at least β of being the last such occurrence.

Thus, $\mathbf{Pr}[\mathbf{a}$ occurs $\geq k$ times during $Q_N] \leq (1-\beta)^{k-1}$. Since $\beta > 0$, for any ϵ there is a finite k_ϵ such that $\mathbf{Pr}[\mathbf{a}$ occurs $\geq k_\epsilon$ times during $Q_N] \leq \epsilon$. Note that k_ϵ and ϵ are independent of N.

Now let C be a random variable for the number of times that $a_i = \mathbf{a}$ during the performance of Q_N. The expectation $\mathbf{E}[C] = \alpha N$. However, we also know that with probability $(1-\epsilon)$, $C < k_\epsilon$. Therefore,

$$\mathbf{E}[C] = N\alpha < (1-\epsilon)k_\epsilon + \epsilon N.$$

Thus,

$$\alpha < (1-\epsilon)\tfrac{k_\epsilon}{N} + \epsilon.$$

Assuming $\mathbf{b} \not\rightharpoonup \mathbf{a}$, this inequality should hold for any ϵ, including one strictly smaller than α. But since k_ϵ and ϵ are fixed independently of N, as N increases, $(1-\epsilon)\frac{k_\epsilon}{N}$ becomes arbitrarily small. The right side of the inequality will then be less than α for $\epsilon < \alpha$ and sufficiently large N, forming a contradiction. Therefore the assumption is false, and $\mathbf{a} \rightleftharpoons \mathbf{b}$. $\square$