Characterizing History Independent Data Structures

Jason D. Hartline[1], Edwin S. Hong[1], Alexander E. Mohr[1], William R. Pentney[1], and Emily C. Rocke[1]

Department of Computer Science, University of Washington, Seattle, WA 98195.
{hartline,edhong,amohr,bill,ecrocke}@cs.washington.edu

Abstract. We consider history independent data structures as proposed for study by Teague and Naor [2]. In a history independent data structure, nothing can be learned from the 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 also provide a natural less restrictive definition of strong history independence 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.

The full version of this extended abstract is available from:
`http://www.cs.washington.edu/research/computation/theory-night/papers/hist-indep.ps`

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 which 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., data structures that retain information about previous operations performed upon them can cause considerable problems; the Times' blunder represents a particularly grievous instance of this. 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 [1]. 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 [3], Snyder studied bounds on the performance of search, insert and delete functions on uniquely represented data structures, which employ a canonical representation for each possible state of the data structure. More stringent history independence requirements were studied by Naor and Teague in [2]. 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 [2] 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. In the full version of this paper we show that this definition equivalent to that of [2]. Under this definition, any strong history independent implementation of a data structure must satisfy a natural canonicality criterion (Section 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 is deterministically given by its contents. This answers an open question posed in [2] 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 6 we discuss the issue of creating dynamically resizing history independent data structures. 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 [2].

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.[1] 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 edges). It is useful to consider the following trichotomy of state transition graphs according to standard graph properties:

- The graph may be acyclic (a DAG). This is the case if it is not possible to return to a previously visited state. Examples are the union-find data structure [4] 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 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 in 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.

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 $\mathbf{a} \in A$. 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

[1] 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.

$\mathbf{a} \to \mathbf{b}$, if there is some sequence of operations X such that $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right] > 0$. We say $\mathbf{a} \rightleftharpoons \mathbf{b}$ if $(\mathbf{a} \to \mathbf{b}) \wedge (\mathbf{b} \to \mathbf{a})$. We say representation $\mathbf{a}$ of state A is *reachable* if it is reachable from data structure startup, i.e., $\oslash \to \mathbf{a}$. We henceforth only consider representations that are reachable. We say $\oslash \to \mathbf{a}$ if there exists X such that $(\oslash \xrightarrow{X} A) \wedge \mathbf{Pr}\left[\oslash \xrightarrow{X} \mathbf{a}\right] > 0$.

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 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 Note 1.

Note 1. If $A \xrightarrow{X} B$ and $B \xrightarrow{Y} C$ then,

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

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 [2] 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 for multiple observations of the data structure without giving any information about the operations 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:*

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

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 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, $\oslash$, in which case $\mathbf{a}$ is the empty representation, $\oslash$. 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 [2], we show in the full paper that they are in fact equivalent.

For strong history independent data structures, because the $\mathbf{Pr}\left[\mathbf{a} \xrightarrow{X} \mathbf{b}\right]$ 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 history independent data structures.

Transitivity of "$\to$" and "$\rightleftharpoons$": *Under* SHI*,* $(\mathbf{a} \to \mathbf{b}) \wedge (\mathbf{b} \to \mathbf{c}) \Longrightarrow (\mathbf{a} \to \mathbf{c})$*, and similarly,* $(\mathbf{a} \rightleftharpoons \mathbf{b}) \wedge (\mathbf{b} \rightleftharpoons \mathbf{c}) \Longrightarrow (\mathbf{a} \rightleftharpoons \mathbf{c})$.
Reflexivity: $\mathbf{a} \rightleftharpoons \mathbf{a}$ *if and only if* $A \rightleftharpoons A$.
Symmetry of "$\to$": *Under* SHI*, for states A and B with* $A \rightleftharpoons B$*,* $\mathbf{a} \to \mathbf{b} \Longrightarrow \mathbf{b} \to \mathbf{a}$.

Transitivity follows immediately from the definition of reachable. See the full version of the paper for the proof of reflexivity and symmetry.

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 [2] 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* $\mathbf{a} \to \mathbf{a}'$*, 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*, $\mathbf{Pr}[\mathbf{a} \to \mathbf{a}'] = 0$ so $\mathbf{a} \not\to \mathbf{a}'$. □

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. From Lemma 1 by transitivity of "$\to$" 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,* $\mathbf{a} \rightleftharpoons \mathbf{b}'$ *iff* $\mathbf{b}' = \mathbf{b}$.

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 restricted *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 contributor 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 randomness over representations, the randomness 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.

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 must have a "canonical distribution" over representations that is chosen during the initialization process. Moreover, 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 conflicts would have to be remembered and 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 Note 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 Note 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 $\mathbf{Pr}[\mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\mathbf{a}' \to \mathbf{b}]$. □

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 strong history independence, 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 at least one sequence of operations X with $A \xrightarrow{X} B$, and at least one Y with $B \xrightarrow{Y} C$. For any such X and Y, $\mathbf{a} \xrightarrow{[X,Y]} \mathbf{c}$ must go through some representation $\mathbf{b}'$ of B. By symmetry and transitivity of "$\rightleftharpoons$", $\mathbf{b}' \rightleftharpoons b$. Let $\langle \mathbf{b} \rangle$ be the set of all such $\mathbf{b}'$. Thus,

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

□

5.3 Order of Observations

In the full paper we continue these arguments by showing that the probability distribution over representations for a set of observed states is independent of the order in which the states are observed.[2] Note, however, that 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 $A \rightleftharpoons B$, the result implies that $\mathbf{Pr}[\emptyset \to \mathbf{a} \to \mathbf{b}] = \mathbf{Pr}[\emptyset \to \mathbf{b} \to \mathbf{a}]$. This order independence implies that our definition of strong history independence is in fact equivalent to that of [2]:

Definition 5 (*NT-SHI* and *NT-SHI).** *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be sequences of operations and let $P_1 = \{i^1_1, \ldots, i^1_\ell\}$ and $P_2 = \{i^2_1, \ldots, i^2_\ell\}$ be two lists of observation points such that for all $b \in \{1, 2\}$ and $1 \le j \le \ell$ we have $1 \le i^b_j \le |\mathcal{S}_b|$ and the state following the i^1_j prefix of $\mathcal{S}_1$ and the i^2_j prefix of $\mathcal{S}_2$ are identical (for*

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

NT-SHI* *we further require that for* $j \neq k$ *that* $i^b_j \neq i^b_k$*). 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.*[3]

Note that this definition allows the observations of the operations to be made out of order, i.e., $i^b_j > i^b_{j+1}$. It is for this reason that the order invariance described above is required to show that *SHI* and *NT-SHI* are equivalent.

6 Dynamic Resizing Data Structures

Many open addressing, i.e., 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 [2]. 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 data structure with amortized constant time per operation against an oblivious adversary. Dynamic resizing for oblivious adversaries are discussed in more detail in the full paper; in this section we will focus on the non-oblivious case.

A non-oblivious adversary is allowed access to the random bits flipped during the running of the algorithm. Such an adversary can adapt a sequence of operations to the data structure 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.

We provide the following results on dynamically resizing history independent data structures 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

[3] [2] does not define *NT-SHI**. We include it here as the natural relaxation to nonempty sequences for comparison to *SHI**.

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.

Non-oblivious Adversary and Weak History Independence
We show how to make any fixed capacity history independent (weakly or strongly) array-based data structure with a linear time resize operation into a constant time amortized dynamically resizing weakly history independent data structure. The principle behind this method is to maintain the invariant that N is random variable that is uniform on $\{n, \ldots, 2n-1\}$ while each insert or delete operation only causes the data structure to resize with probability $O(1/n)$.

We show below how to modify the insert function of any data structure to maintain our invariant on N. The delete function is similar. See the full version of the paper for details. 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 on $\{n+1, \ldots, 2(n+1)-1\}$. 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).

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 against a non-oblivious adversary. Thus, there is no general technique for taking a constant time strongly history 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. 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 weak history independent amortized constant time resizing deques exist.

Theorem 3. *Against a non-oblivious adversary, 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) linear 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). $\square$

7 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 *SHI*?

We have also given a general technique for dynamically resizing weak history independent data structures. We have shown that for a standard efficiency metric this is not possible for strong history independent data structures. We show 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. D. Micciancio. Oblivious data structures: Applications to cryptography. In *Proc. of 29th ACM Symposium on Theory of Computing*, pages 456–464, 1997.
2. M. Naor. and V. Teague. Anti-persistence: History Independent Data Structures. In *Proc. of 33nd Symposium Theory of Computing*, May 2001.
3. L. Synder. On Uniquely Represented Data Structures. In *Proc. of 28th Symposium on Foundations of Computer Science*, 1977.
4. Robert E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.