COMPOSING COPYLESS STREAMING STRING TRANSDUCERS

RAJEV ALUR o,a, TAYLOR DOHMEN o,b, AND ASHUTOSH TRIVEDI o,b

a University of Pennsylvania, 3330 Walnut St, Philadelphia, PA 19104 USA
e-mail address: alur@seas.upenn.edu

b University of Colorado, 1111 Engineering Dr, Boulder, CO 80309 USA
e-mail address: taylor.dohmen@colorado.edu, ashutosh.trivedi@colorado.edu

ABSTRACT. Streaming string transducers (SSTs) implement string-to-string transformations by reading each input word in a single left-to-right pass while maintaining fragments of potential outputs in a finite set of string variables. These variables get updated on transitions of the transducer, where they can be assigned new values described by concatenations of variables and output symbols. An SST is called copyless if every update is such that no variable occurs more than once amongst all of the assigned expressions. The transformations realized by copyless SSTs coincide with Courcelle’s monadic second-order logic graph transducers (MSOTs) when restricted to string graphs. Copyless SSTs with nondeterminism are known to be equivalent to nondeterministic MSOTs as well.

MSOTs, both deterministic and nondeterministic, are closed under composition. Given the equivalence of MSOTs and copyless SSTs, it is easy to see that copyless SSTs are also closed under composition. The original proof of this fact, however, was based on a direct construction to produce a composite copyless SST from two given copyless SSTs. A counterexample discovered by Joost Engelfriet showed that this construction may produce copyful transducers. We revisit the original composition constructions for both deterministic and nondeterministic SSTs and show that, although they can introduce copyful updates, the resulting copyful behavior they exhibit is superficial. To characterize this mild copyful behavior, we define a subclass of copyful SSTs, called diamond-free SSTs, in which two copies of a common variable are never combined in any subsequent assignment. In order to recover a modified version of the original construction, we provide a method for producing an equivalent copyless SST from any diamond-free copyful SST.

1. Introduction

Streaming string transducers (SSTs) are computational models for transformations between formal languages. Upon reading a symbol, an SST changes state and updates a finite set of string variables, which are initially set to the empty string \( \varepsilon \), using concatenation expressions, such as \( x := aybzcz \), made up of variables \( y, z \) and symbols \( a, b, c \) from a designated output alphabet. The final output of the transducer is determined by a similar combination of variables and symbols, depending on the final state reached after reading an input word. Deterministic SSTs (DSSTs) [AC10, AC11] model functions over strings, while nondeterministic SSTs (NSSTs) [AD11] are interpreted as relations.

Key words and phrases: Copyless Streaming String Transducers, Sequential Composition, Diamond-Free.
An SST is \emph{copyless} if each variable is used at most once across all variable updates occurring on a single transition. The copyless restriction prohibits updates involving assignments such as
\begin{itemize}
  \item $x := aybyc$ and $y := x$, in which two copies of $y$ flow into $x$, and
  \item $x := y$ and $y := y$, where one copy of $y$ flows into each of the variables $x$ and $y$.
\end{itemize}

An SST that is not copyless is called \emph{copyful} \cite{FR17, FR21}. Figure 1.1 shows an example, $S$, of a copyless SST implementing the function $w \mapsto ww$. Figure 1.2 provides an example, $T$, of a copyful SST that implements the mapping $a^k \mapsto a^{2k+1}$, which is not definable by any copyless SST.

The behavior of SSTs with respect to variable copying is particularly significant when it comes to characterizing the expressive power of the model. Copyful SSTs, as evidenced by Figure 1, can model transductions having output strings of lengths exponential in the lengths of corresponding input strings. Copyless SSTs, on the other hand, have outputs with lengths at most linear in the lengths of the corresponding inputs. As shown by Alur, et al. \cite{AC10, AD11}, the transductions definable by copyless SSTs coincide with those definable by monadic second-order logic graph transducers (MSOTs) \cite{Cou92, Cou94, CE12}. The assumption that variable updates are copyless is crucial for this equivalence, because the linear upper bound on lengths of output strings is a fundamental property of MSOTs.

As MSOTs are closed under composition\footnote{cf. Proposition 3.2 of \cite{Cou92, Cou94} and Theorem 7.14 of \cite{CE12}.}, it follows from their correspondence that the functions and relations definable by SSTs are also closed under composition. Effective procedures are given in \cite{AC10, AD11} to construct a copyless SST that realizes the composition of two copyless SSTs given as input. Joost Engelfriet observed\footnote{via a private correspondence with the authors of \cite{AD11}.}, however, that these constructions can produce transducers with copyful assignments. Consider, for instance, the three deterministic SSTs shown in Figure 2, where $T_3$ is the composition of $T_1$ and $T_2$, constructed according to the methods provided by \cite{AC10, AD11}. The transducer $T_1$ implements the function $a^n \mapsto (ab)^n$, while the transducer $T_2$ models the mapping $a^{n_1}b^{m_1} \ldots a^{n_k}b^{m_k} \mapsto a^{n_1}b \ldots a^{n_k}b$. Notice that both $T_1$ and $T_2$ are copyless, yet there is a transition in $T_3$ with a copyful assignment $\gamma_2$. On the other hand, notice how none of the variables copied by $\gamma_2$ are ever combined in any later assignment.

Directed acyclic graphs are a useful tool for representing the dynamics of variable copying in SSTs, facilitating both analysis and visualization. The graphs depicted in Figure 3, for instance, provide convenient encodings of how information flows between variables of SSTs over the course of a computation. In these illustrations, an edge $x \rightarrow y$ corresponds to a
COMPOSING COPYLESS STREAMING STRING TRANSUDCERS

(2.1) Copyless SST $T_1$.

(2.2) Copyless SST $T_2$.

(2.3) Copyful SST $T_3$.

Figure 2. Two copyless SSTs $T_1$, $T_2$ and a copyless SST $T_3$ implementing their composition. An edge labeled by $a\backslash\alpha$ indicates that the machine transitions from the edge’s source state to its target state upon reading symbol $a$ and applies the assignment $\alpha$ to its set of variables.

Figure 3. A pair of two-step flow graphs from $T$ and $T_3$. Four diamonds are present in (3.1). Zero diamonds are present in (3.2).

transition that involves an assignment to variable $y$ in which $x$ occurs. Figure 3.1 shows how variables flow into other variables during two consecutive applications of the single assignment of $T$, while Figure 3.2 shows this for two consecutive applications of the assignment $\gamma_2$ of $T_3$. Despite $\gamma_2$ being copyful, the diagram makes clear that there is no possibility of two copies of $z_3$ or $z_4$ converging and contributing to a common variable at a later point in time. We call copyful SSTs exhibiting this type of behavior diamond-free. Intuitively, an SST is diamond-free if there is at most one path between any two vertices in the assignment graph induced by any computation. The diamond-free restriction, while weaker than the copyless restriction, preserves its semantic intent: that outputs are linear in the lengths of their corresponding inputs.

Contributions and Outline. In section 2, we introduce the basic definitions and ideas around SSTs. In section 3, we present a slightly modified and more detailed account of the composition constructions of Alur, et al. for both DSSTs and NSSTs. We prove their correctness and establish bounds on the size of the resulting transducers in subsection 3.3.
In subsection 3.4, show that the composite transducers are necessarily diamond-free. Lastly, in section 4, we develop a construction that transforms any diamond-free NSST into an equivalent copyless NSST. As a result, the removal of noncopyless assignments can be added as post-processing step, thereby yielding a complete direct method for SST composition. A diagrammatic summary of our results is given in Figure 4. By recovering a composition construction that acts directly on copyless SSTs, we solidify the foundations of works depending on this operation, such as [TKA13, DDKT21].

Related Work. The diamond-free property has been considered in related literature [LMT19] studying determinization of finitely-ambiguous cost register automata. While our notion of diamond-freeness essentially coincides with the notion considered in that work, our results are independent and apply in a distinct context.

An alternative restriction on copying behavior in SSTs, called bounded-copy, is also considered in related work [AFT12]. An SST is bounded-copy, if in the assignment graph for any computation, there are finitely many paths between any two vertices. For any non-negative integer $k$, a bounded copy SST is called $k$-copy if there are at most $k$ paths between any two vertices in the assignment graph induced by a computation. From this definition, it is clear that copyless SSTs correspond to 0-copy SSTs and diamond-free SSTs correspond to 1-copy SSTs. Note, however, that [AFT12] considers deterministic SSTs while the present work considers nondeterministic SSTs. Furthermore, the methods we use to convert diamond-free NSSTs into copyless NSSTs are simpler and distinct from those employed in [AFT12] to convert bounded-copy DSSTs into copyless DSSTs.

2. Streaming String Transducers
The natural numbers will be written as $\mathbb{N}$. The cardinality of a set $X$ is denoted by $|X|$, and its powerset by $2^X$.

Let $\Sigma^*$ be the set of all strings over a finite alphabet $\Sigma$ and let $\varepsilon$ be the empty string. For $s \in \Sigma$ and $w \in \Sigma^*$, let $s \in w$ indicate that the symbol $s$ occurs in the string $w$, let $|w|$ denote the length of $w$, and let $|w|_s$ be the number of occurrences of the symbol $s$ within $w$. We use product notation to denote iterated concatenation, so that $\prod_{k=1}^n w_k = w_1w_2\ldots w_n$. 

![Figure 4. Schematic of this paper’s approach to composing copyless SSTs.](image-url)
A morphism, over alphabets $\Sigma$ and $\Lambda$, is a function $h : \Sigma^* \to \Lambda^*$ which respects concatenation such that $h(\varepsilon) = \varepsilon$ and $h(w_1w_2) = h(w_1)h(w_2)$, for any $w_1, w_2 \in \Sigma^*$. Every morphism is completely specified by its restriction to $\Sigma$. For a set $X$, let $\text{id}_X : X \to X$ be the identity function that maps every element $x$ to itself. With respect to a subset $X \subseteq \Sigma$, let $\text{Er}_X : \Sigma^* \to (\Sigma \setminus X)^*$ be the erasing morphism, defined such that $\text{Er}_X(x) = \varepsilon$ if $x \in X$ and $\text{Er}_X(s) = s$ if $s \in \Sigma \setminus X$.

For a function $f : X \to Y$, we write $\text{dom} f$ for its domain and $\text{im} f$ for its image. The composition $g \circ f : X \to Z$ of two functions $f : X \to Y$ and $g : Y \to Z$ is defined in the standard fashion as $x \mapsto g(f(x))$. We adopt the same notation to refer to domains, images, and compositions of relations.

**Definition 2.1** (SST Syntax). A streaming string transducer $T$ is given by a tuple $(\Sigma, \Lambda, Q, q_0, X, A, \Delta, F)$ where $\Sigma$ is a finite input alphabet, $\Lambda$ is a finite output alphabet, $Q$ is a finite set of states, $q_0 \in Q$ is a distinguished initial state, $X$ is a finite set of string variables, $A$ is a finite set of assignment maps with type $X \to (\Lambda \cup X)^*$ for updating variables, $\Delta \subseteq Q \times \Sigma \times A \times Q$ is a transition relation, and $F : Q \to (\Lambda \cup X)^*$ is a partial output function.

An assignment $\alpha$ of type $X \to (\Lambda \cup X)^*$ can naturally be extended to a morphism $(\Lambda \cup X)^* \to (\Lambda \cup X)^*$ such that $\alpha(s) = s$, for $s \in \Lambda$ and $\alpha\left(\prod_{k=1}^{n} w_k\right) = \prod_{k=1}^{n} \alpha(w_k)$, for words in $(\Lambda \cup X)^*$. When viewed in this manner, assignments can be composed in sequence, and we set $\bigcirc_{k=1}^{n} \alpha_k = \alpha_n \circ \alpha_{n-1} \circ \cdots \circ \alpha_1$ as the notation for iterated composition.

The shape of a string $w_0 \prod_{k=1}^{n} x_k w_k \in (X \cup \Lambda)^*$ is a summary $\sqcap x_1 \sqcap x_2 \ldots \sqcap x_n \sqcap$ of the string’s topology, which ignores details of the variable-free substrings occurring between variables. Here, the symbol $\sqcap$ simply indicates the presence of a substring $w_k \in \Lambda^*$. Such shapes are entirely determined by the order of the variables occurring in the original string.

A run of an SST on an input word $w = w_1 \ldots w_n \in \Sigma^*$ is a finite sequence of transitions $q_0 \xrightarrow{w_1} \alpha_1 \ldots w_n \xrightarrow{\alpha_n} q_n$ such that $\langle q_{k-1}, w_k, \alpha_k, q_k \rangle \in \Delta$ for all $1 \leq k \leq n$. For a run $\rho$, define the valuation $V_\rho : (X \cup \Lambda)^* \to \Lambda^*$ as $\bigcirc_{k=1}^{n} \alpha_k$.

**Definition 2.2** (SST Semantics). An SST $T$ specifies a relation $[T] \subseteq \Sigma^* \times \Lambda^*$ where

$$[T] = \{ \langle w, \text{Er}_X \circ V_\rho \circ F(q_n) \rangle : \text{exists run } \rho \text{ in } T \text{ on } w \text{ ending in } q_n \in \text{dom } F \}.$$ 

An SST is deterministic (DSST), if for every pair $\langle q, s \rangle \in Q \times \Sigma$ there is exactly one transition $\langle q, s, \alpha, q' \rangle \in \Delta$, and is nondeterministic (NSST) otherwise. A functional NSST is one for which every element of $\text{dom}[T]$ maps to a unique element of $\text{im}[T]$.

### 2.1. Copying Variables

A string $w \in (\Sigma_1 \cup \Sigma_2)^*$ is copyless in $\Sigma_1$ if each element of $\Sigma_1$ occurs at most once in $w$, i.e. when $\max_{x \in \Sigma_1} \left| w \right|_s \leq 1$. We write $[\Sigma_1]$ for the set of copyless strings over $\Sigma_1$ and $[\Sigma_1, \Sigma_2]$ for set of strings over $(\Sigma_1 \cup \Sigma_2)^*$ that are copyless in $\Sigma_1$.

An assignment $\alpha \in A$ is copyless if the string $\prod_{k=1}^{\left| X \right|} \alpha(x_k)$ is copyless in $X$, making it an element of $[X, \Lambda]$. It is straightforward to observe that any composition of copyless assignments is also copyless. We say an SST is copyless if, for all transitions $\langle q, s, \alpha, q' \rangle \in \Delta$, the assignment $\alpha$ is copyless. An SST is copyful if it has at least one copyful assignment.

The following proposition characterizes the cardinality of the set of all copyless strings over a finite alphabet in terms of the size of that alphabet. It will be used to establish bounds on the size of composite SSTs in subsection 3.3.

**Proposition 2.3.** If $|X| = n$ and $n \geq 2$, then $|[X]| = \lfloor en \rfloor$. 

Proof. Strings in $[X]$ can be any length less than or equal to $n$, so counting the number of elements amounts to counting the number of ways to choose—without repetition—and order $k$ elements from $X$ for $0 \leq k \leq n$. Thus,

$$|[X]| = \sum_{k=0}^{n} \frac{n!}{(n-k)!} = n! \sum_{k=0}^{n} \frac{1}{(n-k)!} = n! \sum_{k=0}^{n} \frac{1}{k!}.$$  

By Taylor’s theorem [Rud76], $e = \sum_{k=0}^{n} \frac{1}{k!} + \frac{e^{a}}{(n+1)!}$, for some $a \in (0, 1)$, and so we obtain the equation

$$|[X]| = n! \left( e - \frac{e^{a}}{(n+1)!} \right) = en! - \frac{e^{a}}{n+1}.$$  

Since the map $a \mapsto e^{a}$ is increasing on the unit interval, the subtrahend of the above difference may be bounded as

$$\frac{1}{n+1} \leq \frac{e^{a}}{n+1} \leq \frac{e}{n+1}.$$  

Furthermore, the assumption that $n \geq 2$ implies that $\frac{e^{a}}{n+1} < 1$. Finally, $[X]$ is a finite set and thus has cardinality in the positive integers, so we conclude that $|[X]| = \lfloor en! \rfloor$. □

2.2. Flow Graphs and Diamonds. The copying behavior of an SST is characterized by flow graphs associated to its assignments.

Definition 2.4 (Flow Graph). Let $\alpha = \alpha_1 \ldots \alpha_n$ be a word over $A^*$. The flow graph of $\alpha$ is a directed acyclic graph $G(\alpha) = (V(\alpha), E(\alpha))$ such that

- the set of vertices $V(\alpha) = \bigcup_{k=1}^{n+1} X_k$ is made up of $n+1$ disjoint copies of $X$ and
- $(x, y) \in E(\alpha)$ if there exists $1 \leq k \leq n$ such that $x \in X_k$, $y \in X_{k+1}$, and $x \in \alpha_k(y)$.

The flow graph $G(\rho)$ of a run $\rho = q_0 \xrightarrow{w_1} \alpha_1 \ldots \xrightarrow{w_n} \alpha_n q_n$ is identified with the flow graph $G(\alpha_1 \ldots \alpha_n)$ of its sequence of assignments.

Definition 2.5 (Diamond). Let $\alpha = \alpha_1 \ldots \alpha_n$ be a word over $A^*$, and let $G(\alpha)$ be its flow graph. A diamond is a subgraph of $G(\alpha)$, consisting of two distinct paths from a source vertex $s \in X_i$ to a target vertex $t \in X_j$, with $i < j$.

Diamond-Freeness. An assignment $\alpha$ is diamond-free if $|\alpha(x)|_y \leq 1$ holds for all $x, y \in X$ or, equivalently, if $G(\alpha)$ has no double edges. Copyless assignments are diamond-free, but diamond-free assignments are not necessarily copyless. A run $\rho$ is diamond-free if there is at most one path between any two vertices in $G(\rho)$. An SST is diamond-free if all of its runs are diamond-free. Note that runs and SSTs may fail to be diamond-free even if all assignments in $A$ are diamond-free.
3. Composition Construction

This section presents a modified and elaborated treatment of the composition constructions of Alur, et al. [AC10, AD11]. Our departure from the original versions is centered around two aspects:

1. we impose a strict indexing of the variables of the composite SST that then to be partitioned based on their roles in the process, and
2. we provide explicit detailed definitions, in the form of recursive higher-order functions, of the operators used to produce the composite SST.

The indexing scheme in combination with the recursively defined operators allow us to establish the diamond-free property in an almost type-directed manner.

We begin by presenting the construction under the assumption that the given SSTs are deterministic in subsection 3.1. Then, in subsection 3.2, we describe the necessary adaptations to the procedure for NSSTs. In subsection 3.3, we prove correctness of the modified constructions and establish bounds on the size of the composite SSTs. Lastly, in subsection 3.4, we prove that the composite transducers must be diamond-free.

3.1. DSST Composition. Fix copyless DSSTs $T_1^1 = \langle \Sigma_1, \Sigma_2, Q, q_0, X, A, \Delta_1^1, F_1^1 \rangle$ and $T_2^2 = \langle \Sigma_2, \Sigma_3, P, p_0, Y, B, \Delta_2^2, F_2^1 \rangle$ as inputs. The DSST $T_3^3 = \langle \Sigma_1, \Sigma_3, R, r_0, Z, C, \Delta_3^1, F_3^1 \rangle$ will be the composite transducer. The set of variables of $T_3^3$ is written as $Z$ to allow the use of uppercase $Z$ for individual variables, which makes our index notation more readable.

The main idea underlying the composition constructions is to simulate the state transitions of $T_1^1$ while keeping a record of what $T_2^2$ does when reading each possible variable in $X$ from each possible state in $P$. Since $T_1^1$ builds its output piecewise, these records are critical. By storing the actions of $T_3^3$ for each possible eventuality, it is possible to then chain together those actions which correspond to the run of $T_2^2$ on the output of $T_1^1$. In particular, the composite transducer incorporates two types of summaries as finite maps in the states of $T_3^3$.

1. State summaries are maps $f: (P \times X) \rightarrow P$ that record the state in $T_2^2$ that is reached after reading the contents of a variable $x$, for every possible combination of starting states in $p \in P$ of $T_2^2$ and variables $x \in X$ of $T_1^1$.

2. Shape summaries are maps $g: (P \times X \times Y) \rightarrow [Y \cup Z]$ that store the shape of each variable $y \in Y$ after reading a variable $x \in X$ from a starting point $p \in P$, for every possible combination of parameters.

Since their domains and ranges are finite, each summary is finitely representable and there are finitely many summaries of each type. We write $F$ for the set of all functions with type $(P \times X) \rightarrow P$ and $G$ for the set of all functions of type $(P \times X \times Y) \rightarrow [Y \cup Z]$. The variables $Z = \{ Z_{y,0}, Z_{y,1} : (p, x, y) \in P \times X \times Y \}$ of $T_3^3$ are used by the shape summary maps to store variable-free substrings of shapes.

We now define the higher-order operators that will be used to explicitly define the composition constructions. For the remainder of this section, we set the following conventions.

- $x$ represents strings from $[X, \Sigma_2]$.
- $y$ denotes strings from $[Y \cup Z, \Sigma_3]$ and $[Y, \Sigma_3]$.
- $z$ represents strings from $[Z, \Sigma_3]$. 
• \( g^p_x \) is shorthand for the map \( y \mapsto g(p, x, y) \), where \( g \in \mathbb{G} \).

**State Transition Summarizer.** The map \( F : P \times [X, \Sigma_2] \times \mathbb{F} \to P \) is responsible for updating destination states in \( T_2^3 \) based on updated variables in \( T_2^1 \) and is defined as

\[
F(p, x, f) = \begin{cases} 
  p & \text{if } x = \varepsilon \\
  F(p', x', f) & \text{if } x = s \cdot x' \text{ and } \langle p, s, p' \rangle \in \Delta_3^2 \\
  F(f(p, x), x', f) & \text{if } x = x'.
\end{cases}
\]

(3.1)

**Assignment Summarizer.** The processes that create shape summaries and assignments share a dependency on the map \( G : P \times [X, \Sigma_2] \times \mathbb{F} \times \mathbb{G} \to (Y \to [Y \cup Z, \Sigma_3]) \) defined as

\[
G(p, x, f, g) = \begin{cases} 
  \text{Id}_Y & \text{if } x = \varepsilon \\
  \beta \circ G(p', x', f, g) & \text{if } x = s \cdot x' \text{ and } \langle p, s, \beta, p' \rangle \in \Delta_3^2 \\
  g^p_x \circ G(f(p, x), x', f, g) & \text{if } x = x'.
\end{cases}
\]

(3.2)

**Shape Generator.** The map \( S : P \times X \times (Y \to [Y \cup Z, \Sigma_3]) \to (Y \to [Y \cup Z]) \) computes shape summaries and is given as

\[
S(p, x, h) = y \mapsto \left( \prod_{k=1}^n Z_{y_k, 0}^p Z_{y_k, 1}^p \right) Z_{y, 1}^p \quad \text{if } h(y) = z_0 \prod_{k=1}^n y_k z_k.
\]

(3.3)

**Assignment Generator.** The map \( A : P \times X \times (Y \to [Y \cup Z, \Sigma_3]) \to (Z \to [Z, \Sigma_3]) \) computes assignments and is given as

\[
A(p, x, h) = \begin{cases} 
  Z_{y, 0}^p \to \varepsilon & \text{if } y \notin h(y'), \text{ for all } y' \in Y \\
  Z_{y, 0}^p \to z & \text{if } h(y') = z_0 y y' \text{ or } y'' z y \in h(y') \text{ for some } y', y'' \in Y \\
  Z_{y, 1}^p \to z & \text{if } h(y) = z_0 \text{ or } h(y) = y y' z_0, \text{ for some } y' \in Y.
\end{cases}
\]

(3.4)

We write \( S^p_x \) and \( A^p_x \), respectively, for the mappings \( h \mapsto S(p, x, h) \) and \( h \mapsto A(p, x, h) \). Letting

\[
g^p_x = S^p_x \circ G(p, \alpha(x), f, g) \quad \text{and} \quad \gamma^p_x = A^p_x \circ G(p, \alpha(x), f, g),
\]

the next equation holds for all \( y \in Y \):

\[
G(p, \alpha(x), f, g)(y) = \gamma^p_x \circ g^p_x(y).
\]

(3.5)

Thus, the following diagram commutes, where \( \pi_k \) and \( \iota_k \) are the \( k \)-th canonical projection and injection. This relationship is further illustrated in Figure 5.
The following definition formalizes the construction of $T_3$ from DSSTs $T_1$ and $T_2$.

**Definition 3.1 (DSST Composition).** Suppose that $T_1 = \langle \Sigma_1, Q, q_0, X, A, F_1 \rangle$ and $T_2 = \langle \Sigma_2, P, p_0, Y, B, F_2 \rangle$ are an arbitrary pair of copyless DSSTs. Their composition is a DSST $T_3 = \langle \Sigma_3, R, r_0, Z, C, F_3 \rangle$ where

- $R = Q \times F \times G$ such that $r_0 = \langle q_0, f_0, g_0 \rangle$ with $f_0(p, x) = p$ and $g_0(p, x, y) = y$,
- $Z = \{ Z_{y_0, x} : (p, x, y) \in P \times X \times Y \}$,
- $C = \{ \gamma : (r, s, \gamma, r') \in \Delta_1 \}$,
- $\langle (q, f', g') , s, \gamma, (q', f', g') \rangle \in \Delta_1$ if, there exists $(q, s, \alpha, q') \in \Delta_2$ such that
  1. $f'(p, x) = F(p, \alpha(x), f)$,
  2. $g'(p, x, y) = S(p, \alpha(x), f, g)(y)$, and
  3. $\gamma \left( Z_{y, k} \right) = A(p, \alpha(x), f, g) \left( Z_{y, k} \right)$,
- $F_3(q, f, g) = E_Y \circ G(p, F_1(q), f) \circ F_2 \circ F(p, F_2(q), f)$.

### 3.2. NSST Composition

Definition 3.1 requires some adaptation for NSSTs. One approach to this adaptation might be to generalize the maps $F$ and $G$ independently to accommodate nondeterminism. This would involve defining variations of $F$ and $G$ having codomains $2^P$ and $2^{Y \cup Z, \Sigma_3}$, respectively. Then, one could add transitions in $T_3$ whenever the summaries in the destination state could be obtained via these maps from the summaries in the source state. Doing this, however, could lead to grouping a state summary based...
on one nondeterministic run in $T_3^2$ together with a shape summary based on a different nondeterministic run. To see this, imagine that the NSST in Figure 7 is $T_3^2$. Notice that the string $aa$ leads to four distinct states and induces four distinct assignments, but to faithfully determine the assignments, each destination state should be paired to a unique assignment. For instance, if the state summary points to $p_4$, then the only shape summary that correctly pairs with this state summary represents the effects of applying the assignment $\beta_3 \circ \beta_1$. Proceeding as we have just described would lead to 3 additional incorrect pairings, and this would result in an incorrect composite transducer.

To avoid such situations, it is necessary to synchronize the computation of state summaries with the computation of shape summaries and assignments. This synchronization is achieved by the map $\mathcal{H} : (P \times [X, \Sigma_2] \times F \times G) \rightarrow 2^{P \times [Y \cup Z, \Sigma_1]}$ which combines $F$ and $G$ in more structured way and is defined as

$$\mathcal{H}(p, x, f, g) = \begin{cases} \{\langle p, \text{id}_Y \rangle\} & \text{if } x = \varepsilon, \\ \bigcup \{\langle p'', \alpha \circ h \rangle : \langle p'', h \rangle \in \mathcal{H}(p', x', f, g)\} & \text{if } x = sx', \\ \{\langle p', g^p \circ h \rangle : \langle p', h \rangle \in \mathcal{H}(f(p, x), x', f, g)\} & \text{if } x = xx'. \end{cases}$$

**Definition 3.2** (NSST Composition). Suppose that $T_2^1 = \langle \Sigma_1, \Sigma_2, Q, q_0, X, A, \Delta_2^1, F_2^1 \rangle$ and $T_3^2 = \langle \Sigma_2, \Sigma_3, P, p_0, Y, B, \Delta_3^2, F_3^2 \rangle$ are an arbitrary pair of copyless NSSTs. Their composition is an NSST $T_3^1 = \langle \Sigma_1, \Sigma_3, R, r_0, Z, C, \Delta_3^1, F_3^1 \rangle$ where

- $R = Q \times F \times G$ such that $r_0 = \langle q_0, f_0, g_0 \rangle$ with $f_0(p, x) = p$ and $g_0(p, x, y) = y$,

- $Z = \{Z_{p,x}^{y,0}, Z_{p,x}^{y,1} : \langle p, x, y \rangle \in P \times X \times Y\}$,

- $C = \{\gamma : \langle r, s, \gamma, r' \rangle \in \Delta_3^1\}$,

- $\langle\langle q, f, g\rangle, s, \gamma, \langle q', f', g'\rangle\rangle \in \Delta_3^1$ if there exist $\langle q, s, \alpha, q' \rangle \in \Delta_2^1$ such that
  1. $\langle p', h \rangle \in \mathcal{H}(p, \alpha(x), f, g)$,
  2. $f'(p, x) = p'$,
  3. $g'(p, x, y) = S_{q'}^p(h)(y)$, and
  4. $\gamma\left(Z_{p,x}^{y,k}\right) = A_{q'}^p(h)\left(Z_{p,x}^{y,k}\right)$,

- $F_3^1(q, f, g) = \text{Err}_Y \circ h \circ F_3^2(p)$, with $\langle p, h \rangle \in \mathcal{H}(p_0, F_2^1(q), f, g)$ chosen nondeterministically.
Besides accommodating nondeterminism, the use of $\mathcal{H}$ in Definition 3.2 does not change the essential nature of the composition construction. In particular, the methods by which summaries and corresponding assignments are computed are identical in both Definition 3.1 and Definition 3.2. Only the way in which the computed summaries are grouped to form transitions in $\Delta_3^1$ differs.

**Remark 3.3.** In order to minimize bookkeeping and notational complexity, the analyses in subsection 3.3 and subsection 3.4 are carried out for NSSTs using the maps $\mathcal{F}$ and $\mathcal{G}$ rather than $\mathcal{H}$, but with the assumption that the synchronization enforced by $\mathcal{H}$ holds implicitly.

### 3.3. Correctness and Size Bounds

Suppose $\langle s, t \rangle \in [T_2^1]$ and $\langle t, u \rangle \in [T_3^2]$ such that there are corresponding runs $\rho = q_0 \at \alpha_1 \cdots \alpha_n \at q_n$ and $\sigma = p_0 \at \beta_1 \cdots \beta_m \at p_m$ in $T_2^1$ and $T_3^2$, respectively. By Definition 3.2, for every transition $\langle q_k-1, s_k, \alpha_k, q_k \rangle \in \Delta_3^1$ occurring in $\rho$, there exists a transition $\langle \langle q_k-1, f_k-1, g_k-1 \rangle, s_k, \gamma_k, \langle q_k, f_k, g_k \rangle \rangle \in \Delta_3^1$. Let $\tau$ be the corresponding run $\langle q_0, f_0, g_0 \rangle \at \gamma_1 \cdots \at \gamma_n \at \langle p_n, f_n, g_n \rangle$ in $T_3^3$. Also, let $\alpha = \bigcirc_{k=1}^{n} \alpha_k$ and $\gamma = \bigcirc_{k=1}^{n} \gamma_k$.

**Lemma 3.4.** Suppose that $x \in X$ and $p \in P$. If $\sigma(p, x)$ is a run in $T_3^2$ on input $V_\rho(x)$, starting from state $p$ and ending in state $p'$, then

$$f_n(p, x) = p'.$$

**Proof.** We proceed by induction on $n$, the length of $\rho$ and $\tau$.

**Base case:** Equation 3.6 holds when $n = 0$.

If $n = 0$, then $\rho = q_0$ and $\tau = \langle q_0, f_0, g_0 \rangle$. Therefore, $V_\rho(x) = \varepsilon$, for all $x \in X$, and so for all $p \in P$ and $x \in X$, we have $\sigma(p, x) = p = p'$. By Definition 3.2, it holds that $f_0(p, x) = p$.

**Inductive case:** If Equation 3.6 holds for $n - 1$, then it also holds for $n$.

By Definition 3.2, $f_n(p, x) = \mathcal{F}(p, \alpha_n(x), f_{n-1})$, for all $\langle p, x \rangle \in P \times X$. Observe that $\mathcal{F}(p, \alpha_n(x), f_{n-1})$ partially simulates the transitions of $T_3^2$, by reading $\alpha_n(x)$ in sequence and keeping track of the current state. This simulation operates according to $\Delta_3^2$ when the next symbol of $\alpha_n(x)$ comes from $\Sigma_2$ and according to $f_{n-1}$ when the next symbol comes from $X$. In other words, for every state $p$, $\alpha_n(x)$ induces a sequence of states $p_1, p_2, \ldots, p_{|\alpha_n(x)|}$, via $\mathcal{F}$, such that $p_1 = p$ and either

1. $\langle p_k, s, \beta, p_{k+1} \rangle \in \Delta_3^2$ for some $\beta \in B$, if the $k$th symbol of $\alpha_n(x)$ is $s \in \Sigma_2$, or
2. $f_{n-1}(p_k, x') = p_{k+1}$, if the $k$th symbol of $\alpha_n(x)$ is $x' \in X$. 

**Figure 7.** An NSST illustrating the need for $\mathcal{H}$. 

![NSST Illustration](image-url)
The former case is correct by definition, while the latter follows from the inductive hypothesis. Consequently, we arrive at the equation
\[ f_n(p, x) = \mathcal{F}(p, \alpha_n(x), f_{n-1}) = p_{|\alpha_n(x)|} = p', \]
and we conclude that correctness of \( f_{n-1} \) implies correctness of \( f_n \).

**Lemma 3.5.** Suppose that \( \gamma : Z \rightarrow [Z, \Sigma_3], h : Y \rightarrow [Y \cup Z, \Sigma_3] \), and \( \phi(a) = a \) for any function \( \phi \) and \( a \notin \text{dom} \phi \). For all \( Z \in \mathcal{Z} \), the equation \( \gamma(Z) = h \circ \gamma(Z) = \gamma \circ h(Z) \) holds.

**Proof.** Follows from the fact that \( \text{dom} h = Y \) is disjoint from \( \text{im} \gamma = [Z, \Sigma_3] \).

**Lemma 3.6.** Suppose that \( x \in X, p \in P, \) and \( y \in Y \). If \( \sigma(p, x) \) is a run in \( T_3^2 \) on input \( \nu_\rho(x) \), starting from state \( p \), then
\[ \nu_{\sigma(p,x)}(y) = \text{Er}_Z \circ \nu_\tau \circ g_n(p, x, y). \] 

**Proof.** We proceed by induction on \( n \).

**Base case:** Equation 3.7 holds when \( n = 0 \).

Because \( \sigma(p, x) = p \), we have that \( \nu_{\sigma(p,x)}(y) = y \), for all \( y \in Y \). By Definition 3.2 \( g_0(p, x, y) = y \), and so \( \nu_\tau \circ g_0(p, x, y) = y \), since \( \nu_\tau \) is of type \([Z, \Sigma_3] \rightarrow [Z, \Sigma_3] \). Likewise, \( \text{Er}_Z \) does not erase variables from \( Y \), so \( \text{Er}_Z \circ \nu_\tau \circ g_0(p, x, y) = \nu_{\sigma(p,x)}(y) = y \).

**Inductive case:** If Equation 3.7 holds for \( n - 1 \), then it also holds for \( n \).

Suppose that \( \rho' = q_0 \xrightarrow{s_1} \cdots \xrightarrow{s_{n-1}} q_{n-1} \), where \( \rho = \rho' \xrightarrow{s_n} q_n \), and that \( \tau' = \langle q_0, f_0, g_0 \rangle \xrightarrow{s_1} \cdots \xrightarrow{s_{n-1}} \langle q_{n-1}, f_{n-1}, g_{n-1} \rangle \) where \( \tau = \tau' \xrightarrow{s_n} \langle q_n, f_n, g_n \rangle \). Further, suppose that \( \sigma''(p, x) \) is a run in \( T_3^2 \) on \( \nu_{\rho'}(x) \), starting from state \( p \). Let the last assignment of \( \rho \) be \( \alpha_n(x) = w_0 x_1 \cdots x_\ell w_\ell \), and let \( w_i^j \) denote the \( j \)th symbol in \( w_i \). The run \( \sigma(p, x) \) in \( T_3^2 \) can be written as
\[ \sigma(p, x) = p \xrightarrow{w_1^1} \cdots p|_{w_0 x_1 \cdots x_\ell w_\ell} \xrightarrow{\nu_{\sigma''(p|_{w_0 x_1 \cdots x_\ell w_\ell})|_{x_i}}} p|_{w_0 x_1 \cdots x_\ell w_\ell} \xrightarrow{w_\ell^j} p|_{\alpha_n(x)} \].

Consequently, we obtain the following derivation, which completes the proof.

\[ \nu_{\sigma(p,x)} = \bigcirc_{k=1}^{n} \beta_k \circ \bigcirc_{i=1}^{\ell} \nu_{\sigma''(p|_{w_0 x_1 \cdots x_\ell w_\ell})|_{x_i}} \circ \bigcirc_{j=1}^{w_\ell^j} \beta_j^i \] 

(3.8)

\[ = \bigcirc_{k=1}^{n} \beta_k \circ \bigcirc_{i=1}^{\ell} \text{Er}_Z \circ \nu_\tau \circ g_{n-1}(p|_{w_0 x_1 \cdots x_\ell w_\ell}, x_i, \cdot) \circ \bigcirc_{j=1}^{w_\ell^j} \beta_j^i \] 

(3.9)

\[ = \text{Er}_Z \circ \nu_\tau \circ \bigcirc_{k=1}^{n} \beta_k \circ \bigcirc_{i=1}^{\ell} g_{n-1}(p|_{w_0 x_1 \cdots x_\ell w_\ell}, x_i, \cdot) \circ \bigcirc_{j=1}^{w_\ell^j} \beta_j^i \] 

(3.10)

\[ = \text{Er}_Z \circ \nu_\tau \circ \gamma_n \circ g_n(p, x, \cdot) \] 

(3.11)

\[ = \text{Er}_Z \circ \nu_\tau \circ g_n(p, x, \cdot) \] 

(3.12)

\[ = \text{Er}_Z \circ \nu_\tau \circ g_n(p, x, \cdot) \] 

(3.13)

From Equation 3.8, we obtain Equation 3.9 from the inductive hypothesis. We get Equation 3.10 by applying Lemma 3.5. From here, Equation 3.11 is obtained from the definition of \( \mathcal{G} \) in Equation 3.2. Next, Equation 3.12 follows from Equation 3.5 and the definitions of \( S \) and \( A \). Finally, Equation 3.13 is implied by the definition of valuation.
Definition 3.2, then $T_3^1$ holds that $T_3^1$. Suppose that Theorem 3.8. (Correctness) Theorem 3.7

As a result, we get that $F$.

It is clear from Definition 3.1 that Proof. Lemma 3.9. For any state $q, f, g$ equal, in [MP19]. Letting $h = G(p_0, F_2^1(q_n), f_n, g_n)$, we can rewrite Equation 3.16 as

Since $q, f, g$ is comprised by the product $Q \times F \times G$ where $F = (P \times X) \rightarrow P$ and $G = (P \times X \times Y) \rightarrow [Y \cup Z]$. Therefore, we have

$$|F| = |P|^{P \times X} = n^m$$ and $$|G| = |[Y \cup Z]|^{P \times X \times Y} = |[Y \cup Z]|^{\ell nm}.$$ It is clear from Definition 3.1 that $|Z| = 2\ell nm$. Applying Proposition 2.3 yields

As a result, we get that

$$(|Y \cup Z|) = [e(m + 2\ell nm)!].$$

3.4. Establishing the Diamond-Free Property. We define two binary relations over the $Z$ variables of $T_3^1$:

$$Z^{p,x}_{g,b} \sim Z^{p',x'}_{g',b'} \iff x = x' \quad \text{and} \quad Z^{p,x}_{g,b} \equiv Z^{p',x'}_{g',b'} \iff x = x' \land p = p'.$$

It is easy to see that $\sim$ and $\equiv$ are equivalence relations. The relation $\sim$ partitions $Z$ into $|X|$ classes, each of size $2|Y \times P|$. Similarly, the relation $\equiv$ partitions $Z$ into $|X \times P|$ classes each of size $2|Y|$. We write $Z|_x = x$ to denote that $Z$ belongs to the class indexed by $x$ and $Z|_x = (p, x, y)$ to denote that the equivalence class of $Z$ is indexed by $(p, x)$. We next utilize these relations and the structure they impose on $Z$ to show that the composite transducer is bounded-copy. In the following, $\alpha$ is always an assignment in $T_3^1$, and $Z, Z', Z_1$, etc. will refer to generic variables from $Z$.

Lemma 3.9. For any state $(q, f, g) \in R$, variable $Z \in Z$, and $(p, x, y) \in P \times X \times Y$, it holds that $|g(p, x, y)|_Z \leq 1$ and if $Z \in g(p, x, y)$, then $Z|_\equiv = (p, x)$. 

Proof. By definition of the initial shape summary $g_0$ it is clear that $|g_0(p, x, y)|_Z = 0$, regardless of $p, x, y, Z$. Any other shape summary occurring in $T_2^1$ is constructed by the map $S$ which is defined, in Equation 3.3, such that no variable from $Z$ occurs twice in the image of a single element of the domain. Additionally, it is easy to see from the same definition in Equation 3.3 that any $Z \in S(p, x, h)(y)$ must be indexed by the pair $(p, x)$, invariant of $h$ and $y$, thus implying that $Z|_x = (p, x)$.

Lemma 3.10. Let $\alpha : X \to [X, \Sigma_2]$ be arbitrary and suppose that $Z_1 \in \mathcal{G}(p_1, \alpha(x_1), f, g)(y_1)$ and $Z_2 \in \mathcal{G}(p_2, \alpha(x_2), f, g)(y_2)$. If $x_1 \neq x_2$, then $Z_1 \not\sim Z_2$.

Proof. If $Z_1 \sim Z_2$, then there must be a unique variable $x$ such that $\mathcal{G}(p_1, \alpha(x_1), f, g)$ uses a shape summary $g_{x}^{p_1}$ and $\mathcal{G}(p_2, \alpha(x_2), f, g)$ uses a shape summary $g_{x}^{p_2}$ for some states $p_1, p_2$. This implies that $x \in \alpha(x_1)$ and $x \in \alpha(x_2)$ which contradicts our assumption that $\alpha$ is copyless. Consequently, we conclude that the pair of inclusions $Z_1 \in \mathcal{G}(p_1, \alpha(x_1), f, g)(y_1)$ and $Z_2 \in \mathcal{G}(p_2, \alpha(x_2), f, g)(y_2)$ imply that $Z_1 \not\sim Z_2$.

Lemma 3.11. For any state $\langle q, f, g \rangle \in R$, copyless assignment $\alpha : X \to [X, \Sigma_2]$, variable $Z \in Z$, and $\langle p, x, y \rangle \in P \times X \times Y$ it holds that $|\mathcal{G}(p, \alpha(x), f, g)(y)|_Z \leq 1$.

Proof. We proceed by induction on $\alpha(x)$.

Base case: $|\mathcal{G}(p, \varepsilon, f, g)|_Z \leq 1$.

By definition of $\mathcal{G}$, in Equation 3.2, it follows that $\mathcal{G}(p, \varepsilon, f, g)(y) = y$ when $\alpha(x) = \varepsilon$. Since no variables from $Z$ ever occur in such a string, $|\mathcal{G}(p, \varepsilon, f, g)(y)|_Z = 0 \leq 1$.

Inductive case: If $|\mathcal{G}(p, x, f, g)(y)|_Z \leq 1$ and either $\alpha(x) = xs$ or $\alpha(x) = xx'$, then $|\mathcal{G}(p, \alpha(x), f, g)(y)|_Z \leq 1$.

The disjunction contained in the inductive hypothesis leads us to consider two cases.

(1) Suppose that $\alpha(x) = xs$. By applying the definition of $\mathcal{G}$ we obtain the equation

$$\mathcal{G}(p, \alpha(x), f, g) = \mathcal{G}(p, x, f, g) \circ \beta,$$

where $(\mathcal{F}(p, x, f), s, \beta, p') \in \Delta_2^2$. Since the type of $\beta$ is $Y \rightarrow [Y, \Sigma_3]$, composing this assignment with $\mathcal{G}(p, x, f, g)$ does not introduce or remove any $Z$ variables from the resulting strings, we get

$$|\mathcal{G}(p, xs, f, g)(y)|_Z = |\mathcal{G}(p, x, f, g) \circ \beta(y)|_Z = |\mathcal{G}(p, x, f, g)(y)|_Z.$$

Assuming the inductive hypothesis, we have $|\mathcal{G}(p, x, f, g)(y)|_Z \leq 1$, which, in combination with Equation 3.18, implies the inequality $|\mathcal{G}(p, xs, f, g)(y)|_Z \leq 1$.

(2) Otherwise, suppose that $\alpha(x) = xx'$. Then, applying the definition of $\mathcal{G}$ implies that

$$\mathcal{G}(p, \beta(x), f, g)(y) = \mathcal{G}(p, x, f, g) \circ g(\mathcal{F}(p, x, f), x', y)$$

holds for every $y \in Y$. By assumption, we know that $|\mathcal{G}(p, x, f, g)(y)|_Z \leq 1$, and, applying Lemma 3.9, we obtain the inequality $|g(\mathcal{F}(p, x, f), x', y)|_Z \leq 1$. However, $Z \in \mathcal{G}(p, x, f, g)(y)$ and $Z \in g(\mathcal{F}(p, x, f), x', y)$ cannot hold simultaneously, since this would imply that $|\alpha(x)|_{x'} = 2$ thereby contradicting the assumption that $\alpha$ is copyless. This mutual exclusivity may be restated quantitatively as

$$|\mathcal{G}(p, x, f, g)(y)|_Z + |g(\mathcal{F}(p, x, f), x', y)|_Z \leq 1,$$

and this implies $|\mathcal{G}(p, xx', f, g)(y)|_Z \leq 1$. 

\qed
Lemma 3.12. For any state \( \langle q, f, g \rangle \in R \), copyless assignment \( \alpha : X \to [X, \Sigma_2] \), variable \( Z \in Z \), and \( \langle p, x \rangle \in P \times X \), it holds that \( \sum_Y |G(p, \alpha(x), f, g)(y)|_Z \leq 1 \).

Proof. Suppose that \( \langle q, s, \alpha, g' \rangle \in \Delta_1 \) is the corresponding transition in \( T_2^1 \). If both the inclusions \( Z \in G(p, \alpha(x), f, g)(y_1) \) and \( Z \in G(p, \alpha(x), f, g)(y_2) \) hold, then there must exist \( x, x' \in (X \cup \Sigma_2)^* \) and \( x' \in X \) such that \( xx' = \alpha(x) \) and \( Z \in g(F(p, x, f), x', y_3) \), for some variable \( y_3 \), occurring both \( G(p, x, f, g)(y_1) \) and \( G(p, x, f, g)(y_2) \). However, we know that compositions of copyless assignments, such as \( G(p, x, f, g) \), remain copyless. Thus, if \( y_3 \in G(p, x, f, g)(y_1) \) and \( y_3 \in G(p, x, f, g)(y_2) \) both hold, there must exist a copyful assignment in \( T_2^2 \). This contradicts our initial assumption that \( T_2^2 \) is copyless, so we infer that a single variable \( Z \) cannot occur in more than one element of the set \( \{ G(p, \alpha(x), f, g)(y) : y \in Y \} \). This fact, in combination with Lemma 3.11, which asserts that \( Z \) can occur at most once in any element of this set, allows us to conclude that \( \sum_{y \in Y} |G(p, \alpha(x), f, g)(y)|_Z \leq 1 \).

Lemma 3.13. Consider \( Z \in Z \) and \( \alpha : X \to [X, \Sigma_2] \). If \( Z \in G(p_1, \alpha(x_1), f, g)(y_1) \) and \( Z \in G(p_2, \alpha(x_2), f, g)(y_2) \), then \( x_1 = x_2 \) and either \( (i) p_1 = p_2 \) and \( y_1 = y_2 \) or \( (ii) p_1 \neq p_2 \).

Proof. Since \( Z \overset{X}{\sim} Z \), Lemma 3.10 implies that \( x_1 = x_2 \), we are left with two cases.

1. First, suppose that \( p_1 = p_2 \). If \( y_1 \neq y_2 \), then we have that \( Z \in G(p, \alpha(x), f, g)(y_1) \) and \( Z \in G(p, \alpha(x), f, g)(y_2) \). It implies that
   \[ |G(p, \alpha(x), f, g)(y_1)|_Z + |G(p, \alpha(x), f, g)(y_2)|_Z = 2, \]
   and thus contradicts Lemma 3.12. Therefore, it follows that \( p_1 = p_2 \) implies \( y_1 = y_2 \).

2. Otherwise, suppose that \( p_1 \neq p_2 \). If \( y_1 = y_2 \), then \( Z \in G(p_1, \alpha(x), f, g)(y) \) and \( Z \in G(p_2, \alpha(x), f, g)(y) \). Conversely, if \( y_1 \neq y_2 \), then \( Z \in G(p_1, \alpha(x), f, g)(y_2) \) and \( Z \in G(p_2, \alpha(x), f, g)(y_1) \). Neither possibility raises a contradiction, so the inequality \( p_1 \neq p_2 \) is independent of the relationship between \( y_1 \) and \( y_2 \).

Lemma 3.14. Suppose that \( \gamma : Z \to [Z, \Sigma_3] \) and \( Z, Z_1, Z_2 \in Z \). If \( Z \in \gamma(Z_1) \) and \( Z \in \gamma(Z_2) \), then \( Z_1 \overset{X}{\sim} Z_2 \) and either \( Z_1 = Z_2 \) or \( Z_1 \neq Z_2 \).

Proof. If \( Z \in \gamma(Z_1) \) and \( Z \in \gamma(Z_2) \), then there exist pairs \( \langle p_1, x_1 \rangle \) and \( \langle p_2, x_2 \rangle \) such that we have the inclusions \( Z \in \mathcal{A}_1^p(G(p_1, \alpha(x_1), f, g))(Z_1) \) and \( Z \in \mathcal{A}_2^p(G(p_2, \alpha(x_2), f, g))(Z_2) \). From Equation 3.4, it follows that if these memberships hold, then the pair of memberships \( Z \in G(p_1, \alpha(x_1), f, g)(y_1) \) and \( Z \in G(p_2, \alpha(x_2), f, g)(y_2) \) also hold, for some \( y_1, y_2 \in Y \). Now, Lemma 3.13 entails that either \( x_1 = x_2, p_1 = p_2, y_1 = y_2, \) or \( x_1 = x_2 \) and \( p_1 \neq p_2 \). In either case, \( Z_1 \overset{X}{\sim} Z_2 \). For the former case, we have equality of the states and \( Y \) variables, which, in combination with Lemma 3.12, implies \( Z_1 = Z_2 \). In the latter case, \( Z_1 \) and \( Z_2 \) are not indexed by the same state, so \( Z_1 \neq Z_2 \).

Lemma 3.15. For \( \gamma : Z \to [Z, \Sigma_3] \), if \( Z_1 \in \gamma(Z) \) and \( Z_2 \in \gamma(Z) \), then \( Z_1 = Z_2 \) or \( Z_1 \overset{X}{\sim} Z_2 \).

Proof. If \( Z_1, Z_2 \in \gamma(Z) \), then \( Z_1, Z_2 \in G(p, \alpha(x), f, g)(y) \) for some \( y \). Both variables must be introduced by a shape summary constructed by \( G(p, \alpha(x), f, g) \). This yields two cases: (1) both variables belong to a summary mapping \( g_x^y \) with common parameters from \( P \) and \( X \), or (2) each variable belongs to summary mappings \( g_x^p_1 \) and \( g_x^p_2 \) with distinct parameters. When \( Z_1, Z_2 \) both occur in some element of \( \bigcup_{y \in Y} g(p, x', y) \), then \( Z_1 \equiv Z_2 \) by Lemma 3.9. Otherwise, when \( Z_1 \) occurs in an element of \( \bigcup_{y \in Y} g(p_1, x_1, y) \) and \( Z_2 \) occurs in some element of \( \bigcup_{y \in Y} g(p_2, x_2, y) \), it must hold that \( x_1 \neq x_2 \) because \( \alpha \) is copyless. Since the \( Z \) variables in question cannot be indexed by the same \( X \) variable, we conclude that \( Z_1 \overset{X}{\sim} Z_2 \).
Theorem 3.16. If $T_3^1$ is the composition of $T_3^2$ and $T_3^3$ constructed according to Definition 3.2, then $T_3^1$ is diamond-free.

Proof. Lemma 3.15 and Lemma 3.14 imply that any pair of copies of a single variable never occur in a common variable later in the computation. Therefore, it is impossible for a diamond to occur in any flow graph of a run in $T_3^1$. \hfill $\Box$

4. REMOVING COPIES FROM DIAMOND-FREE NSSTs

In this section, provide a procedure that produces an equivalent copyless NSST when given a diamond-free NSST. The SST $T_3$ from Figure 2, also displayed in Figure 8 along with the flow graphs of its assignments, will be used as a running example to illustrate the argument.

**Assignment Decomposition.** The first portion of the conversion procedure involves decomposing each copyful assignment $\alpha$ in the given diamond-free NSST into a collection of copyless assignments that “cover” $\alpha$ in the sense that the union of these copyless mappings coincides exactly with the mapping $\alpha$. This decomposition subroutine, DECOMPOSE, is given in algorithm 1. Figure 9 provides an illustration of the execution of DECOMPOSE when given the copyful assignment $\gamma_2$ of $T_3$ as input.

**Algorithm 1:** Decomposition of a copyful assignment into copyless assignments.

```
1 function DECOMPOSE($\alpha$)
2     if $\alpha$ is copyless then
3         return \{ $\alpha$ \}
4     $D \leftarrow \emptyset$
5     Choose $x \in X$ with distinct $y, z \in X$ such that $x \in \alpha(y)$ and $x \in \alpha(z)$
6     foreach $y \in X$ such that $x \in \alpha(y)$ do
7         foreach $z \in X \setminus \{y\}$ such that $x \in \alpha(z)$ do
8             Let $\beta$ be a new assignment that is identical to $\alpha$
9             Set $\beta(z) = \varepsilon$
10            if $\beta$ is copyless then
11                $D \leftarrow D \cup \{ \beta \}$
12            else
13                $D \leftarrow D \cup$ DECOMPOSE($\beta$)
14     return $D$
```

**Figure 8.** The SST $T_3$ and flow graphs for each of its assignments.
**Lemma 4.1.** If $\alpha$ is diamond-free, then each $\alpha' \in \text{Decompose}(\alpha)$ is copyless and

$$\alpha = \bigcup_{\alpha' \in \text{Decompose}(\alpha)} \alpha'.$$

**Proof.** Each $\alpha_k$ being copyless follows from the fact that the conditionals at lines 2 and 10 of algorithm 1 ensure only copyless assignments are added to $D$. If the conditional at line 2 is entered then $\{\alpha\}$ is returned and $\alpha = \alpha$, thus satisfying the second assertion of the lemma. Otherwise, a variable $x$ is selected such that $x$ is copied by $\alpha$. The subsequent loop iterates through the possible target variables of for $x$, fixes one such $y$ at each iteration, creates a new assignment $\beta$ as an identical copy of $\alpha$, and then deletes $\beta(z)$ for all other variables $z$ such that $x \in \alpha(z)$. Thus, every mapping in the assignment $\alpha$ is present in $\beta$ for at least one iteration of the loop. Furthermore, this process either adds each $\beta$ into $D$ or recurses with $\beta$ as a parameter. The former guarantees that at least one assignment in $D$ includes any given mapping from $\alpha$. In the latter case, a mapping will not be removed later on, since the variable $x$ will no longer be copied by any assignment $\beta$ passed recursively to Decompose. Therefore, every piece of $\alpha$ occurs in some assignment in $D$ and $\alpha = \bigcup_{\alpha' \in \text{Decompose}(\alpha)} \alpha'$. □

**Definition 4.2** (Diamond-Free $\rightarrow$ Copyless). Suppose that $T = \langle \Sigma, \Lambda, Q, q_0, X, A, \Delta, F \rangle$ is a diamond-free NSST. Let $T' = \langle \Sigma, \Lambda, Q', q'_0, X, A', \Delta', F' \rangle$ be a copyless NSST where

- $Q' = Q \times 2^X$ and $q'_0 = (q_0, \emptyset)$;
- $A' = \bigcup_{\alpha \in A} \text{Decompose}(\alpha)$;
- $\langle \langle p, Y, s, \beta, \langle q, Z \rangle \rangle \rangle \in \Delta'$ if the following conditions both hold:
  1. no pair $x, y \in X$ exists such that $y \in Y$ and $y \in \beta(x)$,
  2. $\langle p, s, \alpha, q \rangle \in \Delta$ such that $\beta \in \text{Decompose}(\alpha)$ and $y \in Z \iff \exists x \in X. x \in \alpha(y) \land x \notin \beta(y);$
- $F'(\langle q, Y \rangle) = F(q)$ if $x \notin F(q)$, for all $x \in Y$. 

![Figure 9. Decompose applied to the copyful assignment $\gamma_2$ of $T_3$.](image-url)
In the sequel, suppose that $T = \langle \Sigma, \Lambda, Q, q_0, X, A, \Delta, F \rangle$ is a diamond-free NSST. Additionally suppose that $T' = \langle \Sigma, \Lambda, Q', q'_0, X, A', \Delta', F' \rangle$ is the copyless NSST constructed according to Definition 4.2.

**Lemma 4.3.** If $\rho$ is an arbitrary run $q_0 \xrightarrow{w_1}{\alpha_1} \ldots \xrightarrow{w_n}{\alpha_n} q_n$ in $T$ and $\mathcal{R}(\rho)$ is the set of all runs of the form $\langle q_0, \emptyset \rangle \xrightarrow{w_1}{\beta_1} \ldots \xrightarrow{w_n}{\beta_n} \langle q_n, Y_n \rangle$ in $T'$, then

$$\mathcal{V}_\rho = \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'}$$

**Proof.** We proceed by induction on the run $\rho$.

**Base case:** $\rho$ is the empty run.

In this case, $\mathcal{R}(\rho)$ is a singleton set containing the empty run and thus $\mathcal{V}_\rho = \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'}$.

**Inductive case:** $\sigma = \rho \xrightarrow{s}{\alpha}$ is a run in $T$ on $w$ and $\mathcal{V}_\rho = \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'}$.

The valuation for $\sigma$ in $T$ may be written as $\mathcal{V}_\sigma = \mathcal{V}_\rho \circ \alpha$, and $\mathcal{R}(\sigma)$ may be written as

$$\mathcal{R}(\sigma) = \bigcup_{\beta \in \text{Decompose}(\alpha)} \bigcup_{\rho' \in \mathcal{R}(\rho)} \left\{ \rho' \xrightarrow{\alpha}{\beta} \langle r, Z \rangle : \langle \langle q_n, Y_n \rangle, s, \alpha, \langle r, Z \rangle \rangle \in \Delta' \right\}.$$

As a result, we have that

$$\bigcup_{\sigma' \in \mathcal{R}(\sigma)} \mathcal{V}_{\sigma'} = \bigcup_{\rho' \in \mathcal{R}(\rho)} \left( \bigcup_{\beta \in \text{Decompose}(\alpha)} \mathcal{V}_{\rho'} \circ \beta \right).$$

**Figure 10.** A copyless functional NSST equivalent to $T_3$ from Figure 2 and constructed according to Definition 4.2. Note that we omit disconnected states from the diagram.
Applying the inductive hypothesis and Lemma 4.1 to this equation, we obtain the desired equivalence:

$$\bigcup_{\sigma' \in \mathcal{R}(\sigma)} \mathcal{V}_{\sigma'} = \bigcup_{\beta \in \text{DECOMPOSE}(\alpha)} \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'} \circ \beta = \bigcup_{\beta \in \text{DECOMPOSE}(\alpha)} \mathcal{V}_{\rho} \circ \beta = \mathcal{V}_{\rho} \circ \alpha = \mathcal{V}_{\sigma}.$$  

**Lemma 4.4.** If \(\langle w, w' \rangle \in [T],\) then \(\langle w, w' \rangle \in [T']\).

**Proof.** Assuming \(\langle w, w' \rangle \in [T],\) then there must be a run \(\rho = q_0 \xrightarrow{w_1}{\alpha_1} \ldots \xrightarrow{w_n}{\alpha_n} q_n \) in \(T\) on \(w\) such that \(\text{Er}_X \circ \mathcal{V}_{\rho} \circ F(q_n) = w'.\) By Lemma 4.3, it holds that

$$w' = \text{Er}_X \circ \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'} \circ F(q_n),$$

and, by Definition 4.2, it holds, for any \(Y_n \subseteq X,\) that

$$w' = \text{Er}_X \circ \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'} \circ F'((q_n, Y_n)),$$

such that there is no variable \(x\) occurring both in \(Y_n\) and in \(F(q_n).\) Each run in \(\mathcal{R}(\rho)\) corresponds to a nondeterministic choice resolving which copies induced by assignments in \(\rho\) will be used later in the computation. Since each assignment in \(\rho\) is diamond-free, no two copies can be combined in a later assignment, and thus there at least one run in \(\mathcal{R}(\rho)\) corresponding to the appropriate choice of copies for simulating \(\rho.\) Therefore, \(\langle w, w' \rangle \in [T]\) implies \(\langle w, w' \rangle \in [T']\).  

**Lemma 4.5.** If \(\langle w, w' \rangle \in [T'],\) then \(\langle w, w' \rangle \in [T]\).

**Proof.** Assuming \(\langle w, w' \rangle \in [T'],\) there exists a run \(\rho' = \langle q_0, \emptyset \rangle \xrightarrow{w_1}{\beta_1} \ldots \xrightarrow{w_n}{\beta_n} \langle q_n, Y_n \rangle \) in \(T'\) on \(w\) such that

$$w' = \text{Er}_X \circ \mathcal{V}_{\rho'} \circ F'((q_n, Y_n)).$$

By Definition 4.2, there exists a transition \(\langle q_{k-1}, w_k, \alpha_k, q_k \rangle \in \Delta,\) for each transition \(\langle q_{k-1}, Y_{k-1} \rangle, w_k, \beta_k, \langle q_k, Y_k \rangle \in \Delta' \text{ in } \rho',\) such that the memberships \(\beta_k \in \text{DECOMPOSE}(\alpha_k)\) and \(y \in Y_k\) hold if the memberships \(x \in \alpha_k(y)\) and \(x \notin \beta_k(y)\) hold for some \(x \in X.\) Therefore, we may construct the run \(\rho = q_0 \xrightarrow{w_1}{\alpha_1} \ldots \xrightarrow{w_n}{\alpha_n} q_n\) for which \(\rho' \in \mathcal{R}(\rho)\) by stringing together these transitions from \(\Delta.\) This shows that every accepting run \(\rho'\) in \(T'\) on \(w\) has a corresponding accepting run \(\rho \in T\) on \(w\) such that

$$\mathcal{V}_{\rho} \circ F(q_n) = \mathcal{V}_{\rho'} \circ F'((q_n, Y_n)).$$

Hence, \(\langle w, w' \rangle \in [T']\) implies \(\langle w, w' \rangle \in [T].\)

**Theorem 4.6.** Every diamond-free NSST with \(n\) states and \(m\) variables is equivalent to a copyless NSST with \(n2^m\) states and \(m\) variables.

**Proof.** Equivalence of \([T]\) and \([T']\) follows immediately from the conjunction of Lemma 4.4 and Lemma 4.5. The number of states and variables in \(T'\) is clear from Definition 4.2.
5. Conclusion

This paper revisits the constructions for sequential composition of copyless SSTs developed by Alur, et al. [AC10, AD11]. While the constructions produce copyful transducers in general, we showed that the composite SSTs exhibit only the tamest of copyful behavior. As a characterization, we defined diamond-free SSTs and proved that the composite transducer is always diamond-free. Moreover, we formulated a method for transforming any diamond-free SST into an equivalent copyless SST. Applying the composition construction followed by the conversion from diamond-free to copyless comprises a complete and direct procedure for composing copyless SSTs.

REFERENCES

[AC10] Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2010.1.

[AC11] Rajeev Alur and Pavol Cerný. Streaming transducers for algorithmic verification of single-pass list-processing programs. In Symposium on Principles of Programming Languages (POPL). ACM, 2011. URL: https://doi.org/10.1145/1926385.1926454.

[AD11] Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. In International Colloquium on Automata, Languages and Programming (ICALP (2)), LNCS. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-22012-8_1.

[AFT12] Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In LICS. IEEE Computer Society, 2012. URL: https://doi.org/10.1109/LICS.2012.18.

[CE12] Bruno Courcelle and Joost Engelfriet. Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach, volume 138 of Encyclopedia of mathematics and its applications. Cambridge University Press, 2012. URL: http://www.cambridge.org/fr/knowledge/isbn/item5758776/?sitelocale=fr_FR.

[Con92] Bruno Courcelle. Monadic second-order graph transductions. In CAAP, 17th Colloquium on Trees in Algebra and Programming, Proceedings, volume 581 of Lecture Notes in Computer Science, pages 124–144. Springer, 1992. URL: https://doi.org/10.1007/3-540-55251-0_7.

[Con94] Bruno Courcelle. Monadic second-order definable graph transductions: A survey. Theor. Comput. Sci., 126(1):53–75, 1994. URL: https://doi.org/10.1016/0304-3975(94)00268-2.

[DDKT21] Vrunda Dave, Taylor Dohmen, Shankara Narayanan Krishna, and Ashutosh Trivedi. Regular model checking with regular relations. In Fundamentals of Computation Theory - 23rd International Symposium, FCT, volume 12867 of Lecture Notes in Computer Science, pages 190–203. Springer, 2021. URL: https://doi.org/10.1007/978-3-030-86593-1_13.
[FR17] Emmanuel Filiot and Pierre-Alain Reynier. Copyful streaming string transducers. In \textit{Reachability Problems RP}, volume 10506 of \textit{Lecture Notes in Computer Science}, pages 75–86. Springer, 2017. URL: \url{https://doi.org/10.1007/978-3-319-67089-8_6}.

[FR21] Emmanuel Filiot and Pierre-Alain Reynier. Copyful streaming string transducers. \textit{Fundam. Informaticae}, 178(1-2):59–76, 2021. URL: \url{https://doi.org/10.3233/FI-2021-1998}.

[LMT19] Théodore Lopez, Benjamin Monmege, and Jean-Marc Talbot. Determinisation of finitely-ambiguous copyless cost register automata. In \textit{44th International Symposium on Mathematical Foundations of Computer Science, MFCS}, volume 138 of \textit{LIPIcs}, pages 75:1–75:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: \url{https://doi.org/10.4230/LIPIcs.MFCS.2019.75}.

[MP19] Anca Muscholl and Gabriele Puppis. Equivalence of finite-valued streaming string transducers is decidable. In \textit{46th International Colloquium on Automata, Languages, and Programming, ICALP}, volume 132 of \textit{LIPIcs}, pages 122:1–122:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: \url{https://doi.org/10.4230/LIPIcs.ICALP.2019.122}.

[Rud76] W. Rudin. \textit{Principles of Mathematical Analysis}. International series in pure and applied mathematics. McGraw-Hill, 1976.

[TKA13] Jay Thakkar, Aditya Kanade, and Rajeev Alur. Transducer-based algorithmic verification of retransmission protocols over noisy channels. In \textit{Formal Techniques for Distributed Systems - Joint IFIP WG 6.1 International Conference, FMOODS/FORTE}, volume 7892 of \textit{Lecture Notes in Computer Science}, pages 209–224. Springer, 2013. URL: \url{https://doi.org/10.1007/978-3-642-38592-6_15}.

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit \url{https://creativecommons.org/licenses/by/4.0/} or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777 Berlin, Germany.