

Composing Copyless Streaming String Transducers

Rajeev Alur¹, Taylor Dohmen², and Ashutosh Trivedi²

¹University of Pennsylvania
²University of Colorado Boulder

September 13, 2022

Abstract

Streaming string transducers (SSTs) implement string-to-string transformations by reading their input 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 reassigned to concatenations of variables and symbols. An SST is called *copyless* if every update is such that no variable gets copied to more than once. Copyless SSTs capture Courcelle’s monadic second-order definable graph transformations (MSOT) when restricted to string-to-string rewritings. The extension of copyless SSTs with nondeterminism is also known to be equivalent to the nondeterministic variant of MSOTs.

MSOTs, both deterministic and nondeterministic, are closed under composition. Given the equivalence of MSOT and copyless SSTs, it is easy to see that copyless SSTs are also closed under composition. The original proofs of this fact, however, were 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 and is characterized by a property we call “diamond-free”: two copies of the variables are never merged. To recover a modified version of the original construction, we provide a method for producing an equivalent copyless NSST from any diamond-free copyful NSST.

1 Introduction

Deterministic streaming string transducers (DSSTs) of Alur and Cerný [1, 2] are single-pass machines that serve as a computational model characterizing certain functions over finite strings. Nondeterministic SSTs (NSSTs) of Alur and Deshmukh [3] generalize this formalism and are interpreted as defining relations rather than functions. Upon reading a symbol, an SST changes state and updates a finite set of string variables, which are initially set to the empty string ϵ , using concatenation expressions, such as $x = ax_1bx_2c$, comprised of variables and symbols 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 the entire input string. As demonstrated by Alur and Cerný [1], Alur and Deshmukh [3], the families of transductions definable by copyless SSTs coincide with those definable by the monadic second-order logic (MSO) string transducers of Courcelle and Engelfriet [6]. Due to this correspondence, transductions definable by copyless DSSTs and NSSTs are called *regular functions* and *regular relations*.

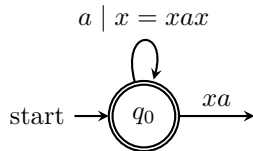


Figure 1: A copyful SST.

A key condition on SSTs that enables their equivalence with MSO transducers is the copylessness of the variable updates. This condition requires each variable to be used at most once across all variable updates occurring on a single transition. For instance, the copyless restriction prohibits the assignment of the form $(x, y) = (aybyc, x)$, where two copies of y are flowing into x , or the assignment $(x, y) = (y, y)$, where variable y is copied into variables x and y both. This ensures that the length of any output string grows linearly in the length of the corresponding input

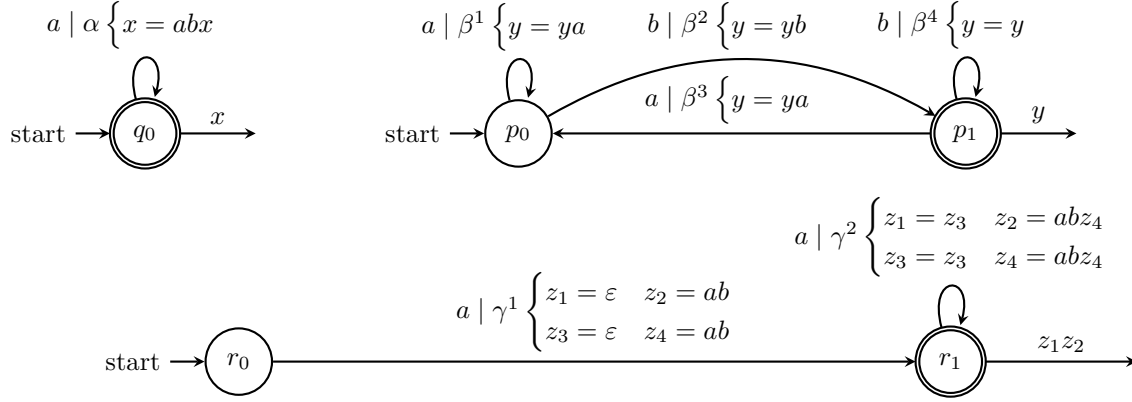


Figure 2: SSTs T_1 (top left), T_2 (top right), and their composition T_3 (bottom).

string. For instance, a copyful SST, such as the one depicted in fig. 1, can implement the mapping $a^k \mapsto a^{2^k}$, which is not definable via an MSO transducer.

The regular functions and regular relations have a number of desirable properties concerning closure and decidability. In particular, as proven in the realm of MSO transducers [6], the regular functions and regular relations are closed under function composition and relation composition, respectively. Effective procedures are given in [1, 3] to construct a copyless SST that realizes the composition of two copyless SSTs given as input. Joost Engelfriet observed [4] that these constructions may result in transducers with copyful assignments. Consider, for instance, the three SSTs shown in fig. 2 where T_3 is the composition of T_1 and T_2 , constructed according to the methods provided by Alur and Cerný [1], Alur and Deshmukh [3]. The transducer T_1 implements the mapping $a^n \mapsto (ab)^n$, while T_2 captures the mapping $a^{n_1}b^{m_1} \dots a^{n_k}b^{m_k} \mapsto a^{n_1}b \dots a^{n_k}b$ for inputs ending in the symbol b . Notice that both input transducers are copyless and deterministic, yet there is a transition in T_3 with a copyful assignment. On the other hand, one may notice that none of these copies are ever combined via concatenation. In the assignment γ^2 , for instance, there are two variables, z_1 and z_3 , both updated using a copy of z_3 , but then z_1 and z_3 are never combined in any later assignment.

The diagram in fig. 3 illustrates how variables flow into other variables during two consecutive applications of assignment γ^2 in T_3 . Despite these assignments 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 on a level further down. 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. We establish that the families of transductions definable by copyless SSTs and diamond-free SSTs coincide. Thus, the diamond-free restriction, while weaker than the copyless restriction, preserves its semantic intent, which is that outputs are linear in length relative to their corresponding inputs.

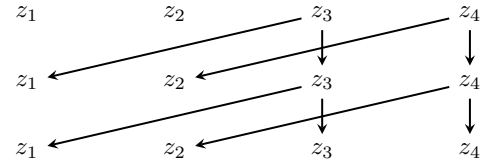


Figure 3: Consecutive applications of γ^2 .

Contributions and Outline. In section 2, we provide preliminary definitions and notations. In section 3, we present the composition construction for both DSSTs and NSSTs, prove their correctness, and establish bounds on the size of the resulting transducers in section 3.1. A key difference between our composition construction and that of [1, 3] is that we carefully index variables of the composite in order to partition them based on their role in the process. In section 3.2, we use this indexing scheme to show that the composite transducer is always diamond-free. Finally, in section 4, we formulate a construction that obtains, from any diamond-free NSST, an equivalent copyless NSST. As a result, the removal of non-copyless assignments can be added as post-processing step to the original composition construction to yield a complete method for combining two copyless SSTs into a third copyless SST realizing their composition. A diagrammatic summary of our results is given in fig. 4. By recovering a composition construction that acts directly on

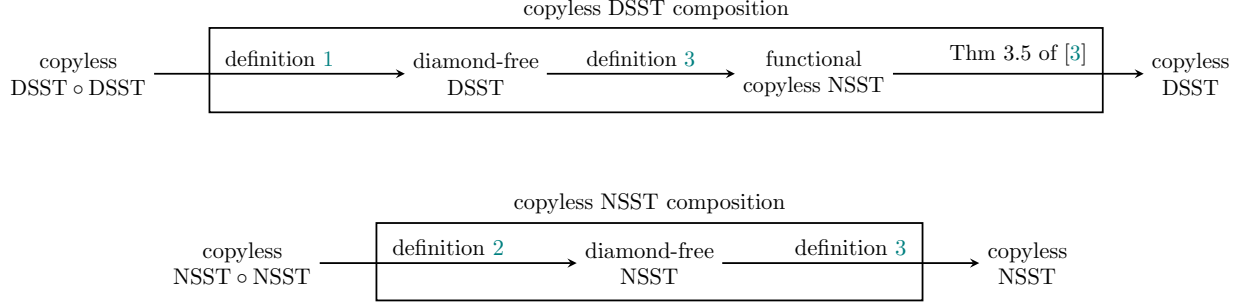


Figure 4: Schematic of the constructions comprising the procedures for composing copyless SSTs.

copyless SSTs, we solidify the foundations of works depending on this operation like those of Thakkar et al. [13], Dave et al. [7]. While our construction yields a large SST, with a state space of triply exponential size in the sizes of the input transducers, it circumvents the process of converting the inputs into MSOTs, composing them, and compiling the result back into a machine, which leads to SSTs of non-elementary size relative to the inputs [11].

Related Work. The diamond-free property has been considered by Lopez, Monmege, and Talbot [10] to study determinization of finitely-ambiguous cost-register automata. While our notion of diamond-freeness coincides with theirs, our results are independent and apply in distinct contexts.

Another restriction on copying behavior in SSTs, called *bounded-copy*, is introduced by Alur, Filiot, and Trivedi [5]. In that work, 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 at most k paths between any two vertices in the assignment graph induced by any computation. From this definition, it is clear that diamond-free SSTs correspond to 1-copy SST. Note, however, that [5] considers deterministic SSTs while the present work considers nondeterministic SSTs. Furthermore the methods we use to convert diamond-free NSSTs into copyless NSSTs are distinct from those employed in [5] to convert bounded-copy DSSTs into copyless DSSTs.

2 Preliminaries

Let Σ^* be the set of all strings over the alphabet Σ and let ε be the empty string ε . For $s \in \Sigma$ and $w \in \Sigma^*$ we write $s \in w$ if the symbol s occurs in the string w . We use the notation \preceq for the prefix relation: $w \preceq w'$ if some prefix of w' matches w exactly. The symbol \sqsubseteq will denote the suffix relation: $w \sqsubseteq w'$ if some suffix of w' matches w exactly. We write $|w|$ for the length of the string w and $|w|_s$ for the number of occurrences of the symbol s within w . We use product notation to denote iterated concatenation, i.e. $\prod_{k=1}^n w_k = w_1 w_2 \cdots w_n$.

For a relation $R \subseteq X \times Y$, we write $\text{dom } R$ for its domain and $\text{im } R$ for its image. The composition $f_1 \circ f_2 : X \rightarrow Z$ of two functions $f_1 : Y \rightarrow Z$ and $f_2 : X \rightarrow Y$ is defined in the standard fashion as $f_1 \circ f_2 = x \mapsto f_1(f_2(x))$. For a set X , let Id_X be the identity function on X which maps every element x to itself, and let Er_X be the erasing function on X which maps every element of X to the empty string ε .

2.1 Streaming String Transducers

A streaming string transducer is a finite state machine equipped with a set of write-only string variables that are updated at each step of a computation and used to form an output. More formally, a streaming string transducer T is a tuple $(\Sigma, \Omega, Q, q_0, X, A, \Delta, F)$ where Σ and Ω are finite input and output alphabets, Q is a finite set of states, $q_0 \in Q$ is the initial state, X is a finite set of string variables, A is a finite set of type $X \rightarrow (\Omega \cup X)^*$ variable assignment functions, $\Delta \subseteq Q \times \Sigma \times A \times Q$ is a transition relation, $F : Q \rightarrow (\Omega \cup X)^*$ is a partial output function.

An assignment α of type $X \rightarrow (\Omega \cup X)^*$ can naturally be extended to a morphism $(\Omega \cup X)^* \rightarrow (\Omega \cup X)^*$ such that $\alpha(s) = s$, for $s \in \Omega$ and $\alpha(\prod_{k=1}^n w_k) = \prod_{k=1}^n \alpha(w_k)$, for words in $(\Omega \cup X)^*$. When viewed in this

manner as morphisms, assignments can be composed in sequence, and we write $\bigcirc_{k=1}^n \alpha_k = \alpha_1 \circ \alpha_2 \circ \dots \circ \alpha_n$.

Semantics. A run ρ of an SST on an input word $w = w_1 \dots w_n \in \Sigma^*$ is a finite sequence of transitions $q_0 \xrightarrow{w_1, \alpha_1} \dots \xrightarrow{w_n, \alpha_n} q_n$ such that $(q_{k-1}, w_k, \alpha_k, q_k) \in \Delta$ for all $1 \leq k \leq n$. Define the valuation \mathcal{V}_ρ , of ρ , as $\bigcirc_{k=1}^n \alpha_k$. An SST T specifies a relation $\llbracket T \rrbracket \subseteq \Sigma^* \times \Omega^*$ where

$$\llbracket T \rrbracket = \{(w, \text{Er}_X \circ \mathcal{V}_\rho \circ F(q_n)) : \text{exists run } \rho \text{ in } T \text{ on } w \text{ ending in } q_n \text{ and } q_n \in \text{dom } F\}.$$

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

Copying Variables. A string $w \in (\Sigma_1 \cup \Sigma_2)^*$ is copyless in Σ_1 if each element of Σ_1 occurs at most once in w ; in other terms, when $\max_{s \in \Sigma_1} |w|_s \leq 1$. We write $\langle \Sigma_1 \rangle$ for the set of copyless strings over Σ_1^* . We write $\langle \Sigma_1, \Sigma_2 \rangle$ for set of strings over $(\Sigma_1 \cup \Sigma_2)^*$ that are copyless in Σ_1 . An assignment $\alpha \in A$ is *copyless* if the string $\alpha(x_1) \dots \alpha(x_{|X|})$ is copyless in X . It is straightforward to observe that any composition of copyless assignments is also copyless. We say an SST is copyless if, for all transitions $(q, s, \alpha, q') \in \Delta$, the assignment α is copyless. An SST is called *copyful*, as studied by [8, 9], when it does not adhere to the copyless restriction, i.e. 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, and will be used to establish bounds on the size of composite SSTs in section 3.1.

Proposition 1. *If $|X| = n$ such that $2 \leq n$, then $|\langle X \rangle| = \lfloor en! \rfloor$.*

Proof. Strings in $\langle X \rangle$ 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,

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

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

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

Since the map $a \mapsto e^a$ is increasing on $[0, 1]$, the right-most term 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, $\langle X \rangle$ is a finite set and thus has cardinality in the positive integers, so we conclude that $|\langle X \rangle| = \lfloor en! \rfloor$. \square

Flow Graphs and Diamonds. The copying behavior of an SST is characterized by *flow graphs* associated to its assignments. The flow graph, or simply the graph, of an assignment α is a bipartite graph $G(\alpha) = (V(\alpha), E(\alpha))$ such that $V(\alpha) = X^S \cup X^T$, where $X^S = \{x^S : x \in X\}$ and $X^T = \{x^T : x \in X\}$ are distinguished copies of X , and $E(\alpha) = \{(x^S, y^T) : x \in \alpha(y)\}$. The subsets X^S and X^T of $V(\alpha)$ are, respectively, the source and target layers of the graph. Define $G(\alpha\alpha) = (V(\alpha\alpha), E(\alpha\alpha))$ to be the tripartite graph obtained by gluing together two copies of $G(\alpha)$ such that the target layer of the first copy is the source layer of the second. More precisely, $V(\alpha\alpha) = X^S \cup X \cup X^T$ and $E(\alpha\alpha) = \{(x^S, y) : x \in \alpha(y)\} \cup \{(x, y^T) : x \in \alpha(y)\}$.

A *diamond* in $G(\alpha\alpha)$ is a subgraph consisting of four vertices x^S, x, y, x^T and four edges (x^S, x) , (x^S, y) , (x, x^T) , (y, x^T) . As a degenerate case, parallel edges, that is when $x = y$, are considered diamonds as well. An assignment is *diamond-free* if there are no diamonds present in the graph $G(\alpha\alpha)$. Note that copyless assignments are diamond-free, but diamond-free assignments are not necessarily copyless. The

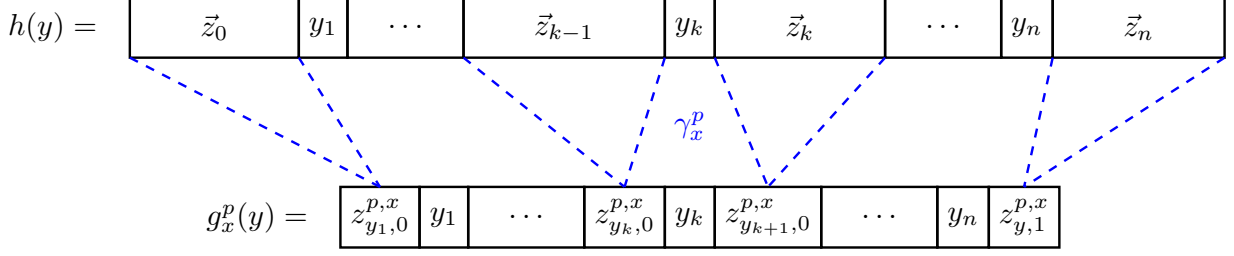


Figure 5: The relationship between an assignment summary $h = \mathcal{G}(p, \alpha(x), f, g)$, and the corresponding shape $g_x^p = \mathcal{S}_x^p(h)$ and assignment $\gamma_x^p = \mathcal{A}_x^p(h)$.

graph $G(\rho) = G(\alpha_1 \cdots \alpha_n)$ of a run $\rho = q_0 \xrightarrow{w_1, \alpha_1} \cdots \xrightarrow{w_n, \alpha_n} q_n$ is obtained by gluing together the graphs $G(\alpha_1), \dots, G(\alpha_n)$ in the same manner used to construct $G(\alpha)$. Since runs can be arbitrary length, we use superscripts matching the subscripts on the states in the run to distinguish the layers of $G(\rho)$. Thus, we have that $V(\rho) = \bigcup_{k=0}^n X^k$ and $E(\rho) = \bigcup_{k=1}^n \{(x^{k-1}, y^k) : x \in \alpha_k(y)\}$. A run is diamond-free iff there is at most one path between any two vertices, i.e. if it does not have a diamond-shaped subgraph. An SST T is diamond-free iff every run in T is diamond-free. Note that an SST may not be diamond-free even if all its assignments are diamond-free.

3 Composition Construction

This section presents a modified and elaborated presentation of the composition construction of [1, 3]. The departure from [1, 3] is in a detailed indexing of the string variables of the composite that allows a clean partition based on their role in the process, and allows us to establish diamond-freeness in an almost type-directed manner. For this section, fix copyless SSTs $T_{12} = (\Sigma_1, \Sigma_2, Q, q_0, X, A, \Delta_{12}, F_{12})$ and $T_{23} = (\Sigma_2, \Sigma_3, P, p_0, Y, B, \Delta_{23}, F_{23})$ as inputs and $T_{13} = (\Sigma_1, \Sigma_3, R, r_0, Z, C, \Delta_{13}, F_{13})$ as their composite.

The procedure relies on storing two types of *summaries* as finite maps in the states of the composite.

1. The first map is the *state summary* map $f : (P \times X) \rightarrow P$ that records the state in T_{23} that is reached when reading the contents of a variable x , for every possible combination of starting states in $p \in P$ of T_{23} and variables $x \in X$ of T_{12} .
2. The second map is the *shape summary* map $g : (P \times X \times Y) \rightarrow \langle Y \cup Z \rangle$ that stores 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 inputs.

Since their domains and ranges are finite, these maps are finitely representable and furthermore there are finitely many maps of each given type. We write \mathbb{F} for the space of all functions with type $(P \times X) \rightarrow P$ and \mathbb{G} for the space of all functions of type $(P \times X \times Y) \rightarrow \langle Y \cup Z \rangle$.

The main idea of the construction is to simulate the state transitions of T_{12} while keeping a record of what T_{23} does when reading any possible register in X from any possible state in P . Since T_{12} builds its output in a piecewise manner, and not linearly as ordinary one-way finite state transducers do, these records are essential. By storing the actions of T_{23} for any possible eventuality, it is possible to then string together those actions which correspond to the run of T_{23} on the output of T_{12} . The f map allows us to store a destination state based on the contents of the X variables, while the g map tracks the shapes of variables in T_{23} resulting from applying sequences of assignments induced by the contents of variables in T_{12} . The Z variables are used by the shape summary maps to store non-variable substrings of shapes and thereby ensure the finitude of shape summaries.

Below are defined a number of functions that will facilitate the construction of T_{13} . We use \vec{x} to represent strings from $\langle X, \Sigma_2 \rangle$, \vec{y} to denote strings from $\langle Y \cup Z, \Sigma_3 \rangle$ or $\langle Y, \Sigma_3 \rangle$, and \vec{z} to denote strings from $\langle Z, \Sigma_3 \rangle$. We write g_x^p for the map $y \mapsto g(p, x, y)$, where $g \in \mathbb{G}$.

State Transition Summarizer. The operator $\mathcal{F} : (P \times \langle X, \Sigma_2 \rangle \times \mathbb{F}) \rightarrow P$ is defined as

$$\mathcal{F}(p, \vec{x}, f) = \begin{cases} p & \text{if } \vec{x} = \varepsilon \\ \mathcal{F}(p', \vec{x}', f) & \text{if } \vec{x} = s\vec{x}' \text{ and } (p, s, \alpha, p') \in \Delta_{23} \\ \mathcal{F}(f(p, x), \vec{x}', f) & \text{if } \vec{x} = x\vec{x}'. \end{cases} \quad (1)$$

Assignment Summarizer. The processes that create shape summaries and assignments share a dependency on operator $\mathcal{G} : (P \times \langle X, \Sigma_2 \rangle \times \mathbb{F} \times \mathbb{G}) \rightarrow (Y \rightarrow \langle Y \cup Z, \Sigma_3 \rangle)$ defined as

$$\mathcal{G}(p, \vec{x}, f, g) = \begin{cases} \text{Id}_Y & \text{if } \vec{x} = \varepsilon \\ \beta \circ \mathcal{G}(p', \vec{x}', f, g) & \text{if } \vec{x} = s\vec{x}' \text{ and } (p, s, \beta, p') \in \Delta_{23} \\ g_x^p \circ \mathcal{G}(f(p, x), \vec{x}', f, g) & \text{if } \vec{x} = x\vec{x}'. \end{cases} \quad (2)$$

Shape Generator. The operator $\mathcal{S} : (P \times X \times (Y \rightarrow \langle Y \cup Z, \Sigma_3 \rangle)) \rightarrow (Y \rightarrow \langle Y \cup Z \rangle)$ computes shape summaries and is given as

$$\mathcal{S}(p, x, h) = y \mapsto \left(\prod_{k=1}^n z_{y_k, 0}^{p, x} y_k \right) z_{y, 1}^{p, x} \quad \text{if } h(y) = \vec{z}_0 \prod_{k=1}^n y_k \vec{z}_k. \quad (3)$$

Assignment Generator. The operator $\mathcal{A} : (P \times X \times (Y \rightarrow \langle Y \cup Z, \Sigma_3 \rangle)) \rightarrow (Z \rightarrow \langle Z, \Sigma_3 \rangle)$ computes assignments and is given as

$$\mathcal{A}(p, x, h) = \begin{cases} z_{y, 0}^{p, x} \mapsto \varepsilon & \text{if } y \notin h(y'), \text{ for all } y' \in Y \\ z_{y, 0}^{p, x} \mapsto \vec{z} & \text{if } \vec{z}y \preceq h(y') \text{ or } y''\vec{z}y \in h(y') \text{ for some } y', y'' \in Y \\ z_{y, 1}^{p, x} \mapsto \vec{z} & \text{if } h(y) = \vec{z} \text{ or } y'\vec{z} \sqsubseteq h(y), \text{ for some } y' \in Y. \end{cases} \quad (4)$$

We write \mathcal{S}_x^p and \mathcal{A}_x^p , respectively, for the mappings $h \mapsto \mathcal{S}(p, x, h)$ and $h \mapsto \mathcal{A}(p, x, h)$. Alternatively stated, \mathcal{S}_x^p and \mathcal{A}_x^p are defined such that the next equation holds for all $y \in Y$.

$$\mathcal{G}(p, \alpha(x), f, g)(y) = \gamma_x^p \circ g_x^p(y). \quad (5)$$

Thus, the following diagram, in which π_k and ι_k are the k^{th} canonical projection and injection, commutes. This relationship is further illustrated in fig. 5.

$$\begin{array}{ccccc} & & \mathcal{G}(p, \alpha(x), f, g) & & \\ & \swarrow \mathcal{A}_x^p & \updownarrow \circ \begin{smallmatrix} \mathcal{A}_x^p, \mathcal{S}_x^p \end{smallmatrix} & \searrow \mathcal{S}_x^p & \\ \gamma_x^p & \xrightleftharpoons[\pi_1]{\iota_1} & (\gamma_x^p, g_x^p) & \xrightleftharpoons[\pi_2]{\iota_2} & g_x^p \end{array}$$

The following definition formalizes the construction of T_3 from DSSTs T_1 and T_2 using the operators we have just defined.

Definition 1 (DSST Composition). *For a pair of copyless DSSTs $T_{12} = (\Sigma_1, \Sigma_2, Q, q_0, X, A, \Delta_{12}, F_{12})$ and $T_{23} = (\Sigma_2, \Sigma_3, P, p_0, Y, B, \Delta_{23}, F_{23})$, their composition is a DSST $T_{13} = (\Sigma_1, \Sigma_3, R, r_0, Z, C, \Delta_{13}, F_{13})$ where*

- $R = Q \times \mathbb{F} \times \mathbb{G}$ such that $r_0 = (q_0, f_0, g_0)$ with $f_0(p, x) = p$ and $g_0(p, x, y) = y$,
- $Z = \{z_{y, 0}^{p, x}, z_{y, 1}^{p, x} : (p, x, y) \in P \times X \times Y\}$,
- $C = \{\gamma : (r, s, \gamma, r') \in \Delta_{13}\}$,
- $((q, f, g), s, \gamma, (q', f', g')) \in \Delta_{13}$ iff, there exists $(q, s, \alpha, q') \in \Delta_{12}$ such that
 - $f'(p, x) = \mathcal{F}(p, \alpha(x), f)$,

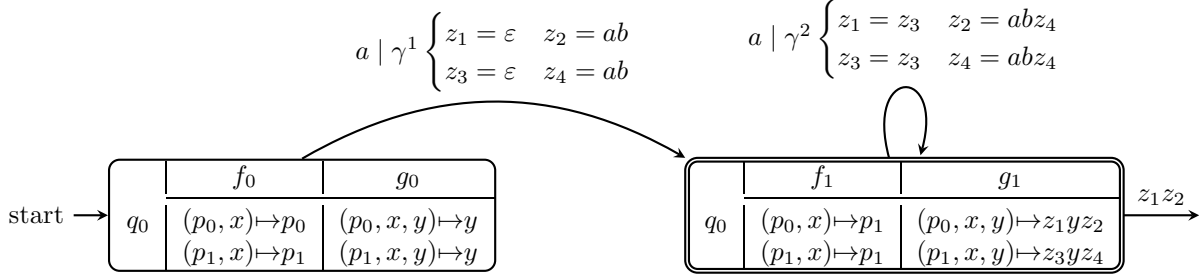


Figure 6: A detailed view of internal maps contained in the states of T_3 from Figure 2, when constructed according to Definition 1. Note that we omit unreachable states.

- $g'(p, x, y) = \mathcal{S}_x^p(\mathcal{G}(p, \alpha(x), f, g))(y)$,
- $\gamma(z) = \mathcal{A}_x^p(\mathcal{G}(p, \alpha(x), f, g))(z)$ for $z \in \{z_{y,0}^{p,x}, z_{y,1}^{p,x} : y \in Y\}$,
- $F_{13}(q, f, g) = \text{Er}_Y \circ \mathcal{G}(p_0, F_{12}(q), f, g) \circ F_{23}(\mathcal{F}(p_0, F_{12}(q), f))$.

Definition 1 requires minimal adaptation for NSSTs. In particular, it becomes necessary to synchronize the computation of state summaries with the computation of shape summaries and assignments. To see why, consider the NSST in fig. 7. If we use \mathcal{F} and \mathcal{G} independently, we end up with pairs like $(p_4, \beta_2 \circ \beta_6)$ which summarize runs that are not possible in the machine. This synchronization is achieved by combining \mathcal{F} and \mathcal{G} into another operator $\mathcal{H} : (P \times \langle X, \Sigma_2 \rangle \times \mathbb{F} \times \mathbb{G}) \rightarrow 2^{P \times (Y \rightarrow \langle Y \cup Z, \Sigma_3 \rangle)}$ which is given as follows.

$$\mathcal{H}(p, \vec{x}, f, g) = \begin{cases} \{(p, \text{Id}_Y)\} & \text{if } \vec{x} = \varepsilon \\ \bigcup_{(p, s, \alpha, p') \in \Delta_{23}} \{(p'', \alpha \circ h) : (p'', h) \in \mathcal{H}(p', \vec{x}', f, g)\} & \text{if } \vec{x} = s\vec{x}' \\ \{(p', g_x^p \circ h) : (p', h) \in \mathcal{H}(f(p, x), \vec{x}', f, g)\} & \text{if } \vec{x} = x\vec{x}' \end{cases}$$

The following definition formalizes the construction of T_3 from NSSTs T_1 and T_2 .

Definition 2 (NSST Composition). *For a pair of copyless NSSTs $T_{12} = (\Sigma_1, \Sigma_2, Q, q_0, X, A, \Delta_{12}, F_{12})$ and $T_{23} = (\Sigma_2, \Sigma_3, P, p_0, Y, B, \Delta_{23}, F_{23})$, their composite is a NSST $T_{13} = (\Sigma_1, \Sigma_3, R, r_0, Z, C, \Delta_{13}, F_{13})$ where*

- $R = Q \times \mathbb{F} \times \mathbb{G}$ such that $r_0 = (q_0, f_0, g_0)$ with $f_0(p, x) = p$ and $g_0(p, x, y) = y$,
- $Z = \{z_{y,0}^{p,x}, z_{y,1}^{p,x} : (p, x, y) \in P \times X \times Y\}$,
- $C = \{\gamma : (r, s, \gamma, r') \in \Delta_{13}\}$,
- $((q, f, g), s, \gamma, (q', f', g')) \in \Delta_{13}$ iff there exist $(q, s, \alpha, q') \in \Delta_{12}$ such that
 - $(p', h) \in \mathcal{H}(p, \alpha(x), f, g)$,
 - $f'(p, x) = p'$,
 - $g'(p, x, y) = \mathcal{S}_x^p(h)(y)$,
 - $\gamma(z) = \mathcal{A}_x^p(h)(z)$, for $z \in \{z_{y,0}^{p,x}, z_{y,1}^{p,x} : y \in Y\}$,
- $F_{13}(q, f, g) = \text{Er}_Y \circ h \circ F_{23}(p)$ with $(p, h) \in \mathcal{H}(p_0, F_{12}(q), f, g)$ chosen nondeterministically.

Note that, besides accommodating nondeterminism in the inputs, the use of \mathcal{H} in definition 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 definitions 1 and 2. The way in which these elements are combined to form transitions in Δ_{13} differs, but this does not influence the manner in which variables are copied by assignments in C . Fortunately, our results are largely independent of this distinction. In order to minimize the book-keeping and notation involved in our arguments, the analyses in sections 3.1 and 3.2 are given in terms of DSSTs and definition 1. Theorems 1 and 2, however, apply both for DSSTs and for NSSTs.

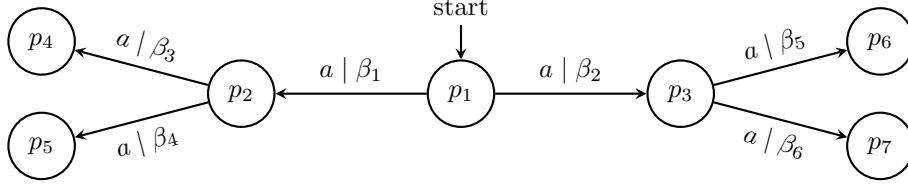


Figure 7: An NSST illustrating the need for \mathcal{H} .

3.1 Correctness and Size Bounds

Suppose $(s, t) \in \llbracket T_{12} \rrbracket$ and $(t, u) \in \llbracket T_{23} \rrbracket$ such that there are corresponding runs $\rho = q_0 \xrightarrow{s_1, \alpha_1} \dots \xrightarrow{s_n, \alpha_n} q_n$ and $\sigma = p_0 \xrightarrow{t_1, \beta_1} \dots \xrightarrow{t_m, \beta_m} p_m$. By definition 1, for every transition $(q_{k-1}, s_k, \alpha_k, q_k) \in \Delta_{12}$ occurring in ρ , there exists a transition $((q_{k-1}, f_{k-1}, g_{k-1}), s_k, \gamma_k, (q_k, f_k, g_k)) \in \Delta_{13}$. Let τ be the corresponding run $(q_0, f_0, g_0) \xrightarrow{s_1, \gamma_1} \dots \xrightarrow{s_n, \gamma_n} (q_n, f_n, g_n)$ in T_{13} . Also, let $\alpha = \bigcirc_{k=1}^n \alpha_k$ and $\gamma = \bigcirc_{k=1}^n \gamma_k$.

Lemma 1. *For any $x \in X$ and $p \in P$, if $\sigma(p, x)$ is the run in T_{12} on input $\mathcal{V}_\rho(x)$, starting from state p and ending in state p' , then the following equation holds for all $n \in \mathbb{N}$.*

$$f_n(p, x) = p' \quad (6)$$

Proof. We proceed by induction on n , the length of ρ and τ .

Base case. eq. (6) holds when $n = 0$.

If $n = 0$, then $\rho = q_0$ and $\tau = (q_0, f_0, g_0)$. Therefore, $\mathcal{V}_\rho(x) = \varepsilon$, 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 1, it holds that $f_0(p, x) = p$.

Inductive case. If eq. (6) holds for $n - 1$, then it also holds for n .

By definition 1, $f_n(p, x) = \mathcal{F}(p, \alpha_n(x), f_{n-1})$, for all $(p, x) \in P \times X$. Observe that $\mathcal{F}(p, \alpha_n(x), f_{n-1})$ partially simulates the transitions of T_{23} , by reading $\alpha_n(x)$ in sequence and keeping track of the current state. This simulation operates according to Δ_{23} when the next symbol of $\alpha_n(x)$ comes from Σ_2 and according to f_{n-1} when the next symbol comes from X . In other other words, for every state p , $\alpha_n(x)$ induces a sequence of states $\langle p_k \rangle_{k=0}^{|\alpha_n(x)|}$, via \mathcal{F} , such that $p_0 = p$ and either (i) $(p_k, s, \beta, p_{k+1}) \in \Delta_{23}$ for some $\beta \in B$, if the k^{th} symbol of $\alpha_n(x)$ is $s \in \Sigma_2$, or (ii) $f_{n-1}(p_k, x') = p_{k+1}$, if the k^{th} symbol of $\alpha_n(x)$ is $x' \in X$. The former case is correct by definition, while the latter follows from the inductive hypothesis. Thus $f_n(p, x) = \mathcal{F}(p, \alpha_n(x), f_{n-1}) = p_{|\alpha_n(x)|} = p'$, and correctness of f_{n-1} implies correctness of f_n . \square

Lemma 2. *Consider $\gamma : Z \rightarrow \langle Z, \Sigma_3 \rangle$ and $h : Y \rightarrow \langle Y \cup Z, \Sigma_3 \rangle$, and suppose $\phi(a) = a$ for any function ϕ and $a \notin \text{dom } \phi$. For all $z \in Z$, 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 $\text{im } \gamma = \langle Z, \Sigma_3 \rangle$. \square

Lemma 3. *For any $x \in X$, $p \in P$, and $y \in Y$, if $\sigma(p, x)$ is the run in T_{12} on input $\mathcal{V}_\rho(x)$, starting from state p , then the following equation holds for all $n \in \mathbb{N}$.*

$$\mathcal{V}_{\sigma(p, x)}(y) = \text{Er}_Z \circ \mathcal{V}_\tau \circ g_n(p, x, y) \quad (7)$$

Proof. We proceed by induction on n .

Base case. eq. (7) holds when $n = 0$.

Because $\sigma(p, x) = p$, we have that $\mathcal{V}_{\sigma(p, x)}(y) = y$, for all $y \in Y$. By definition 1 $g_0(p, x, y) = y$, and so $\mathcal{V}_\tau \circ g_0(p, x, y) = y$, since \mathcal{V}_τ is of type $\langle Z, \Sigma_3 \rangle \rightarrow \langle Z, \Sigma_3 \rangle$. Likewise Er_Z does not erase variables from Y , so $\text{Er}_Z \circ \mathcal{V}_\tau \circ g_0(p, x, y) = y = \mathcal{V}_{\sigma(p, x)}(y)$.

Inductive case. If eq. (7) holds for $n - 1$, then it also holds for n .

Suppose that $\rho' = q_0 \xrightarrow{s_1, \alpha_1} \dots \xrightarrow{s_{n-1}, \alpha_{n-1}} q_{n-1}$, where $\rho = \rho' \xrightarrow{s_n, \alpha_n} q_n$, and that $\tau' = (q_0, f_0, g_0) \xrightarrow{s_1, \gamma_1} \dots \xrightarrow{s_{n-1}, \gamma_{n-1}} (q_{n-1}, f_{n-1}, g_{n-1})$ where $\tau = \tau' \xrightarrow{s_n, \gamma_n} (q_n, f_n, g_n)$. Further suppose that $\sigma'(p, x)$ is the run in T_{23} on $\mathcal{V}_{\rho'}(x)$, starting from state p . Let $\alpha_n(x) = w_0 x_1 \dots x_l w_l$, and let w_i^j denote the j^{th} symbol in w_i . A run $\sigma(p, x)$ in T_{23} can be written as

$$\sigma(p, x) = p \xrightarrow{w_0^1, \beta_0^1} \dots p_{|w_1 x_1 \dots w_i|} \xrightarrow{x_i, \mathcal{V}_{\sigma'}(p_{|w_0 x_1 \dots w_{i-1}|}, x_i)} p_{|w_0 x_1 \dots x_i|} \dots \xrightarrow{w_l^{|w_l|}, \beta_l^{|w_l|}} p_{|\alpha_n(x)|}.$$

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

$$\mathcal{V}_{\sigma(p, x)}(y) = \left(\bigcirc_{k=1}^{|w_0|} \beta_0^k \circ \bigcirc_{i=1}^l \mathcal{V}_{\sigma'}(p_{w_0 x_1 \dots w_{i-1}}, x_i) \circ \bigcirc_{j=1}^{|w_i|} \beta_j^i \right) (y) \quad (8)$$

$$= \left(\bigcirc_{k=1}^{|w_0|} \beta_0^k \circ \bigcirc_{i=1}^l \text{Er}_Z \circ \mathcal{V}_{\tau'} \circ g_{n-1}(p_{w_0 x_1 \dots w_{i-1}}, x_i, \cdot) \circ \bigcirc_{j=1}^{|w_i|} \beta_j^i \right) (y) \quad (9)$$

$$= \left(\text{Er}_Z \circ \mathcal{V}_{\tau'} \circ \bigcirc_{k=1}^{|w_0|} \beta_0^k \circ \bigcirc_{i=1}^l g_{n-1}(p_{w_0 x_1 \dots w_{i-1}}, x_i, \cdot) \circ \bigcirc_{j=1}^{|w_i|} \beta_j^i \right) (y) \quad (10)$$

$$= \text{Er}_Z \circ \mathcal{V}_{\tau'} \circ \mathcal{G}(p, \alpha_n(x), f_{n-1}, g_{n-1})(y) \quad (11)$$

$$= \text{Er}_Z \circ \mathcal{V}_{\tau'} \circ \gamma_n \circ g_n(p, x, y) \quad (12)$$

$$= \text{Er}_Z \circ \mathcal{V}_{\tau} \circ g_n(p, x, y) \quad (13)$$

From eq. (8), we obtain eq. (9) from the inductive hypothesis. We get eq. (10) by applying lemma 2. From here, eq. (11) is obtained from the definition of \mathcal{G} in eq. (2). Next, eq. (12) follows from eq. (5) and the definitions of \mathcal{S} and \mathcal{A} . Finally, eq. (13) is implied by the definition of valuation. \square

Theorem 1. If T_{13} is a composition of T_{23} and T_{12} according to the construction detailed in this section, then the following hold.

1. $\llbracket T_{13} \rrbracket = \llbracket T_{23} \rrbracket \circ \llbracket T_{12} \rrbracket$
2. If T_{12} has k states and l variables and T_{23} has n states and m variables, then T_{13} has a state space of size $O(kn^{ln}((2lnm + m)!)^{lnm})$ and $2lnm$ variables.

Proof. The proof is in two parts.

1. Observe the following derivation:

$$\llbracket T_{13} \rrbracket(s) = \text{Er}_Z \circ \mathcal{V}_{\tau} \circ F_{13}((q_n, f_n, g_n)) \quad (14)$$

$$= \text{Er}_Z \circ \mathcal{V}_{\tau} \circ \text{Er}_Y \circ \mathcal{G}(p_0, F_{12}(q_n), f_n, g_n) \circ F_{23}(\mathcal{F}(p_0, F_{12}(q_n), f_n)) \quad (15)$$

$$= \text{Er}_Y \circ \text{Er}_Z \circ \mathcal{V}_{\tau} \circ \mathcal{G}(p_0, F_{12}(q_n), f_n, g_n) \circ F_{23}(p_m). \quad (16)$$

From eq. (14), we obtain eq. (15) by applying definition 1. Equation (16) follows from eq. (15) via application of lemma 1. Considering the output of T_{12} as the value of a variable x_{out} , as in [12], we can rewrite eq. (16) as

$$\begin{aligned} \llbracket T_{13} \rrbracket(s) &= \text{Er}_Y \circ \text{Er}_Z \circ \mathcal{V}_{\tau} \circ \gamma_{n+1} \circ g_{n+1}(p_0, x_{\text{out}}, \cdot) \circ F_{23}(p_m), \\ \text{where } \gamma_{n+1} &= \mathcal{A}(p_0, x_{\text{out}}, \mathcal{G}(p_0, F_{12}(q_n), f_n, g_n)) \\ \text{and } g_{n+1}(p_0, x_{\text{out}}, \cdot) &= \mathcal{S}(p_0, x_{\text{out}}, \mathcal{G}(p_0, F_{12}(q_n), f_n, g_n)). \end{aligned} \quad (17)$$

Since by definition $\llbracket T_{12} \rrbracket(s) = \text{Er}_X \circ \mathcal{V}_{\rho}(x_{\text{out}})$, applying lemma 3 to eq. (17) yields

$$\llbracket T_{13} \rrbracket(s) = \text{Er}_Y \circ \mathcal{V}_{\sigma} \circ F_{23}(p_m) = \llbracket T_{23} \rrbracket(t).$$

2. By definition 1, the state space of R of T_{13} is comprised by the product $Q \times \mathbb{F} \times \mathbb{G}$ where $\mathbb{F} = (P \times X) \rightarrow P$ and $\mathbb{G} = (P \times X \times Y) \rightarrow \langle Y \cup Z \rangle$. Therefore, we have

$$|\mathbb{F}| = |P|^{|P \times X|} = n^{ln} \text{ and } |\mathbb{G}| = |\langle Y \cup Z \rangle|^{|P \times X \times Y|} = |\langle Y \cup Z \rangle|^{lnm}.$$

It is clear from definition 1 that $|Z| = 2lnm$, and applying proposition 1 yields the equality

$$|\langle Y \cup Z \rangle| = \lfloor e(m + 2lnm)! \rfloor.$$

As a result, we get that $|R| = kn^{ln} \lfloor e(m + 2lnm)! \rfloor^{lnm}$ and thus $|R| \in O(kn^{ln}((2lnm + m)!)^{lnm})$. □

3.2 Establishing the Diamond-Free Property

We define two binary relations, $\overset{\sim}{\sim}$ and \equiv , over the Z variables:

1. $z_{y,b}^{p,x} \overset{\sim}{\sim} z_{y',b'}^{x',p'}$ iff $x = x'$,
2. $z_{y,b}^{p,x} \equiv z_{y',b'}^{x',p'}$ iff $x = x'$ and $p = p'$.

It is easy to see that $\overset{\sim}{\sim}$ and \equiv are equivalence relations. The relation $\overset{\sim}{\sim}$ 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 $2|Y|$. Abusing notation, we sometimes write $[z]_{\overset{\sim}{\sim}} = x$ to denote that z belongs to the class indexed by x . Similarly, we write $[z]_{\equiv} = (p, x)$ to denote that the equivalence class of z is indexed by this pair. We next utilize these relations and the structure they impose on Z to show that the composite transducer is bounded-copy. In the following, α is always an assignment in T_{13} , and z, z', z_1 , etc. will refer to generic variables from Z .

Lemma 4. *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_{13} is constructed by the map \mathcal{S} which is defined, in eq. (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 eq. (3) that any $z \in \mathcal{S}(p, x, h)(y)$ must be indexed by the pair (p, x) , invariant of h and y , thus implying that $[z]_{\equiv} = (p, x)$. □

Lemma 5. *Let $\alpha : X \rightarrow \langle X, \Sigma_2 \rangle$ 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 $z_1 \not\overset{\sim}{\sim} z_2$.*

Proof. If $z_1 \overset{\sim}{\sim} z_2$, 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 α 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 $z_1 \not\overset{\sim}{\sim} z_2$. □

Lemma 6. *For any state $(q, f, g) \in R$, copyless assignment $\alpha : X \rightarrow \langle X, \Sigma_2 \rangle$, variable $z \in Z$, and $(p, x, y) \in P \times X \times Y$ 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 eq. (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 $|\mathcal{G}(p, \vec{x}, f, g)(y)|_z \leq 1$ and either $\alpha(x) = \vec{x}s$ or $\alpha(x) = \vec{x}x'$, then $|\mathcal{G}(p, \alpha(x), f, g)(y)|_z \leq 1$.

In the case that $\alpha(x) = \vec{x}s$, by applying the definition of \mathcal{G} we obtain the equation $\mathcal{G}(p, \alpha(x), f, g) = \mathcal{G}(p, \vec{x}, f, g) \circ \beta$, where $(\mathcal{F}(p, \vec{x}, f), s, \beta, p') \in \Delta_{23}$. Since the type of β is $Y \rightarrow \langle Y, \Sigma_3 \rangle$, composing this assignment with $\mathcal{G}(p, \vec{x}, f, g)$ does not introduce or remove any Z variables from the resulting strings, we get

$$|\mathcal{G}(p, \vec{x}s, f, g)(y)|_z = |\mathcal{G}(p, \vec{x}, f, g) \circ \beta(y)|_z = |\mathcal{G}(p, \vec{x}, f, g)(y)|_z. \quad (18)$$

Assuming the inductive hypothesis, we have $|\mathcal{G}(p, \vec{x}, f, g)(y)|_z \leq 1$, which, in combination with eq. (18), implies the desired inequality: $|\mathcal{G}(p, \vec{x}s, f, g)(y)|_z \leq 1$.

Otherwise, $\alpha(x) = \vec{x}x'$ and applying the definition of \mathcal{G} yields the functional equation:

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

By assumption, we know that $|\mathcal{G}(p, \vec{x}, f, g)(y)|_z \leq 1$, and, applying lemma 4, we obtain the inequality $|g(\mathcal{F}(p, \vec{x}, f), x', y)|_z \leq 1$. However, $z \in \mathcal{G}(p, \vec{x}, f, g)(y)$ and $z \in g(\mathcal{F}(p, \vec{x}, f), x')(y)$ cannot hold simultaneously, since this would imply that $|\alpha(x)|_{x'} = 2$ thereby contradicting the assumption that α is copyless. This mutual exclusivity may be restated quantitatively as $|\mathcal{G}(p, \vec{x}, f, g)(y)|_z + |g(\mathcal{F}(p, \vec{x}, f), x', y)|_z \leq 1$, implying $|\mathcal{G}(p, \vec{x}x', f, g)(y)|_z \leq 1$. □

Lemma 7. For any state $(q, f, g) \in R$, copyless assignment $\alpha : X \rightarrow \langle X, \Sigma_2 \rangle$, variable $z \in Z$, and $(p, x) \in P \times X$, it holds that $\sum_Y |\mathcal{G}(p, \alpha(x), f, g)(y)|_z \leq 1$.

Proof. Suppose that $(q, s, \alpha, q') \in \Delta_{12}$ is the corresponding transition in T_{12} . If both the inclusions $z \in \mathcal{G}(p, \alpha(x), f, g)(y_1)$ and $z \in \mathcal{G}(p, \alpha(x), f, g)(y_2)$ hold, then there must exist $\vec{x}, \vec{x}' \in (X \cup \Sigma_2)^*$ and $x' \in X$ such that $\vec{x}x'\vec{x}' = \alpha(x)$ and $z \in g(\mathcal{F}(p, \vec{x}, f), x', y_3)$, for some variable y_3 , occurring both $\mathcal{G}(p, \vec{x}, f, g)(y_1)$ and $\mathcal{G}(p, \vec{x}, f, g)(y_2)$. However, we know that compositions of copyless assignments, such as $\mathcal{G}(p, \vec{x}, f, g)$, remain copyless. Thus, if $y_3 \in \mathcal{G}(p, \vec{x}, f, g)(y_1)$ and $y_3 \in \mathcal{G}(p, \vec{x}, f, g)(y_2)$ both hold, there must exist a copyful assignment in T_{23} . This contradicts our initial assumption that T_{23} is copyless, so we infer that a single variable z cannot occur in more than one element of the set $\{\mathcal{G}(p, \alpha(x), f, g)(y) : y \in Y\}$. This fact, in combination with lemma 6, which asserts that z can occur at most once in any element of this set, allows us to conclude that $\sum_{y \in Y} |\mathcal{G}(p, \alpha(x), f, g)(y)|_z \leq 1$. □

Lemma 8. Consider $z \in Z$ and $\alpha : X \rightarrow \langle X, \Sigma_2 \rangle$. If $z \in \mathcal{G}(p_1, \alpha(x_1), f, g)(y_1)$ and $z \in \mathcal{G}(p_2, \alpha(x_2), f, g)(y_2)$, 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 $z \stackrel{x}{\sim} z$, lemma 5 implies that $x_1 = x_2$. This leaves two cases.

1. First, suppose that $p_1 = p_2$. If $y_1 \neq y_2$, then we have that $z \in \mathcal{G}(p, \alpha(x), f, g)(y_1)$ and $z \in \mathcal{G}(p, \alpha(x), f, g)(y_2)$. It implies that $|\mathcal{G}(p, \alpha(x), f, g)(y_1)|_z + |\mathcal{G}(p, \alpha(x), f, g)(y_2)|_z = 2$ and therefore contradicts lemma 7, so $p_1 = p_2$ implies $y_1 = y_2$.
2. Otherwise $p_1 \neq p_2$. If $y_1 = y_2$, then $z \in \mathcal{G}(p_1, \alpha(x), f, g)(y)$ and $z \in \mathcal{G}(p_2, \alpha(x), f, g)(y)$. If $y_1 \neq y_2$, then $z \in \mathcal{G}(p_1, \alpha(x), f, g)(y_2)$ and $z \in \mathcal{G}(p_2, \alpha(x), f, g)(y_1)$.

Neither possibility raises a contradiction, so either may hold when $p_1 \neq p_2$. □

Lemma 9. For any $\gamma : Z \rightarrow \langle Z, \Sigma_3 \rangle$ and $z, z_1, z_2 \in Z$, if $z \in \gamma(z_1)$ and $z \in \gamma(z_2)$, then $z_1 \stackrel{x}{\sim} z_2$ and either (i) $z_1 = z_2$ or (ii) $z_1 \neq z_2$.

Proof. If $z \in \gamma(z_1)$ and $z \in \gamma(z_2)$, then there exist pairs (p_1, x_1) and (p_2, x_2) such that we have the inclusions $z \in \mathcal{A}_{x_1}^{p_1}(\mathcal{G}(p_1, \alpha(x_1), f, g))(z_1)$ and $z \in \mathcal{A}_{x_2}^{p_2}(\mathcal{G}(p_2, \alpha(x_2), f, g))(z_2)$. From eq. (4), it follows that if these inclusions hold, then so do the inclusions $z \in \mathcal{G}(p_1, \alpha(x_1), f, g)(y_1)$ and $z \in \mathcal{G}(p_2, \alpha(x_2), f, g)(y_2)$, for some $y_1, y_2 \in Y$. Now, lemma 8 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, $z_1 \stackrel{x}{\sim} z_2$. In the former case, we have equality of the states and Y variables, which, in combination with lemma 7, 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 10. For $\gamma : Z \rightarrow \langle Z, \Sigma_3 \rangle$, if $z_1 \in \gamma(z)$ and $z_2 \in \gamma(z)$, then $z_1 \equiv z_2$ or $z_1 \not\equiv z_2$.

Proof. If $z_1, z_2 \in \gamma(z)$, then $z_1, z_2 \in \mathcal{G}(p, \alpha(x), f, g)(y)$ for some y . Both variables must be introduced by a shape summary constructed by $\mathcal{G}(p, \alpha(x), f, g)$. This yields two cases: (1) both variables belong to a summary mapping $g_{x'}^{p'}$ with common parameters from P and X , or (2) each variable belongs to summary mappings $g_{x_1}^{p_1}$ and $g_{x_2}^{p_2}$ 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 4. 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 α is copyless. Since the Z variables in question cannot be indexed by the same X variable, we conclude that $z_1 \not\equiv z_2$. \square

Lemmas 9 and 10 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 graph of a run in T_{13} , completing our key observation.

Theorem 2. T_{13} is diamond-free.

4 Removing Copies from Diamond-Free NSSTs

In this section, we show that diamond-free NSSTs are equivalently expressive to copyless NSSTs and provide a procedure that produces an equivalent copyless NSST when given a diamond-free NSST. As a corollary, we establish a semantic-preserving transformation between the class of diamond-free DSSTs and copyless functional NSSTs. The SST T_3 from fig. 2, also displayed in fig. 8 along with the graphs of its assignments, will be used as a running example to illustrate the argument.

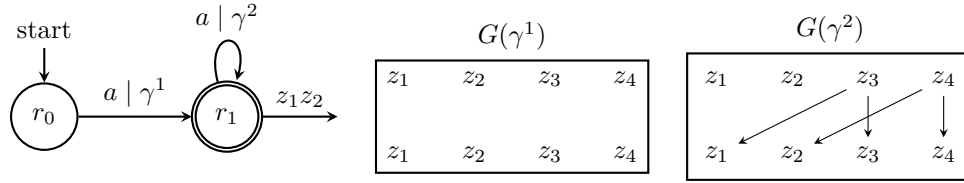


Figure 8: The SST T_3 from fig. 2 (left) and graphs of each assignment (right).

Assignment Decomposition. The first portion of the conversion procedure involves decomposing each copyful assignment α in the given diamond-free NSST into a collection of copyless assignments that “cover” α in the sense that the union of these copyless mappings coincides exactly with the mapping α . This decomposition subroutine—DECOMPOSE—is given in algorithm 1. Figure 9 provides an illustration of the algorithm DECOMPOSE applied to the copyful assignment γ^2 of T_3 . Henceforth, let \mathcal{D} be used to denote the mapping $\alpha \mapsto \text{DECOMPOSE}(\alpha, \emptyset)$.

Lemma 11. If α is diamond-free and $\alpha' \in \mathcal{D}(\alpha)$, then α' is copyless and $\alpha = \bigcup_{\alpha' \in \mathcal{D}(\alpha)} \alpha'$.

Proof. Each α_k being copyless follows from the fact that $\text{DECOMPOSE}(\alpha, D)$ is called with D initially set to \emptyset and the conditionals at lines 2 and 9 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 α . The subsequent loop iterates through the possible target variables of for x , fixes one such y at each iteration, creates a new assignment β as a identical copy of α , and then deletes $\beta(z)$ for all other variables z such that $x \in \alpha(z)$. Thus, every mapping in the assignment α is present in β for at least one iteration of the loop. Furthermore, this process either adds each β into D or recurses with β as a parameter. The former guarantees that at least one assignment in D includes any given mapping from α . In the latter case, a mapping will not be removed later on, since the variable x will no longer be copied in any of the assignments β passed recursively to DECOMPOSE. Therefore, every mapping present in α occurs in some assignment in the decomposition D , so $\alpha = \bigcup_{k=1}^n \alpha_k$. \square

Algorithm 1: Decompose copyful assignment into collection of copyless assignments.

```

1 function DECOMPOSE( $\alpha, D$ )
2   if  $\alpha$  is copyless then
3     return  $D \cup \{\alpha\}$ 
4   Choose  $x \in X$  with distinct  $y, z \in X$  such that  $x \in \alpha(y)$  and  $x \in \alpha(z)$ 
5   foreach  $y \in X$  such that  $x \in \alpha(y)$  do
6     foreach  $z \in X \setminus \{y\}$  such that  $x \in \alpha(z)$  do
7        $\beta \leftarrow \alpha$ 
8        $\beta(z) \leftarrow \varepsilon$ 
9       if  $\beta$  is copyless then
10         $D \leftarrow D \cup \{\beta\}$ 
11      else
12         $D \leftarrow D \cup \text{DECOMPOSE}(\beta, D)$ 
13  return  $D$ 

```

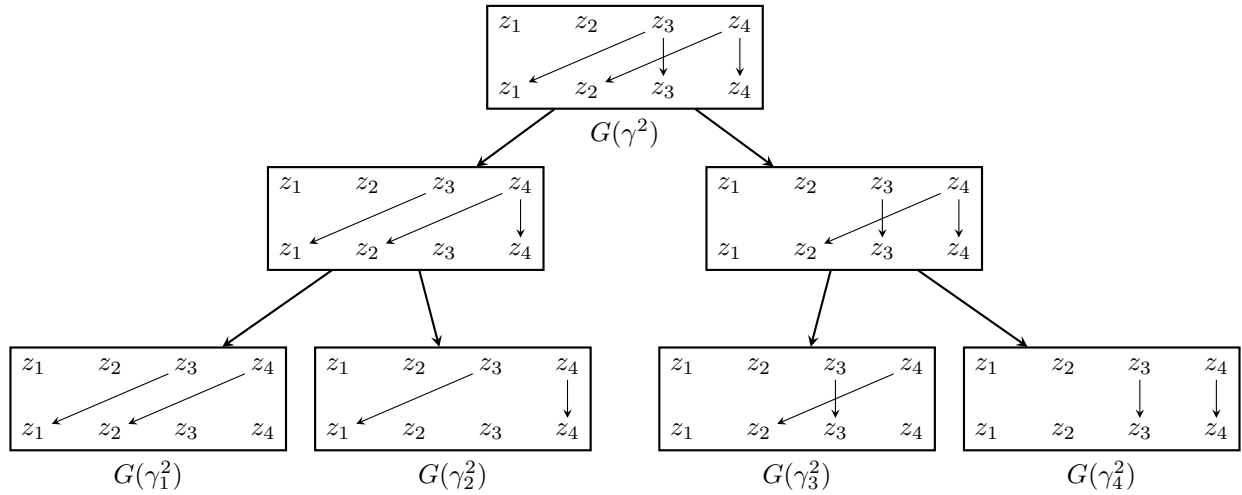


Figure 9: An illustration of DECOMPOSE applied to the copyful assignment γ^2 of T_3 .

Definition 3 (Diamond-Free \rightarrow Copyless). *If $T = (\Sigma, \Omega, Q, q_0, X, A, \Delta, F)$ is a diamond-free NSST, then $T' = (\Sigma, \Omega, Q', q'_0, X, A', \Delta', F')$ is an equivalent copyless NSST where*

- $Q' = Q \times 2^X$ and $q'_0 = (q_0, \emptyset)$,
- $A' = \bigcup_{\alpha \in A} \mathcal{D}(\alpha)$,
- $((p, Y), s, \beta, (q, Z)) \in \Delta'$ iff
 - there does not exist any pair $x, y \in X$ such that $y \in Y$ and $y \in \beta(x)$ and
 - there exists $(p, s, \alpha, q) \in \Delta$ such that (i) $\beta \in \mathcal{D}(\alpha)$ and (ii) $y \in Z$ iff there exists $x \in X$ such that $x \in \alpha(y)$ and $x \notin \beta(y)$,
- $F'((q, Y)) = \begin{cases} \perp & \text{if there exists } x \in Y \text{ such that } x \in F(q) \\ F(q) & \text{otherwise.} \end{cases}$

In the sequel, suppose that $T = (\Sigma, \Omega, Q, q_0, X, A, \Delta, F)$ is a diamond-free NSST. Additionally suppose that $T' = (\Sigma, \Omega, Q', q'_0, X, A', \Delta', F')$ is the copyless NSST constructed according to definition 3. Let ρ be an arbitrary run $q_0 \xrightarrow{w_1, \alpha_1} \dots \xrightarrow{w_n, \alpha_n} q_n$ in T , and let $\mathcal{R}(\rho)$ be the set of all runs of the form $(q_0, \emptyset) \xrightarrow{w_1, \beta_1} \dots \xrightarrow{w_n, \beta_n} (q_n, Y_n)$ in T' where $((q_{k-1}, Y_{k-1}), w_k, \beta_k, (q_k, Y_k)) \in \Delta'$, for all $1 \leq k \leq n$.

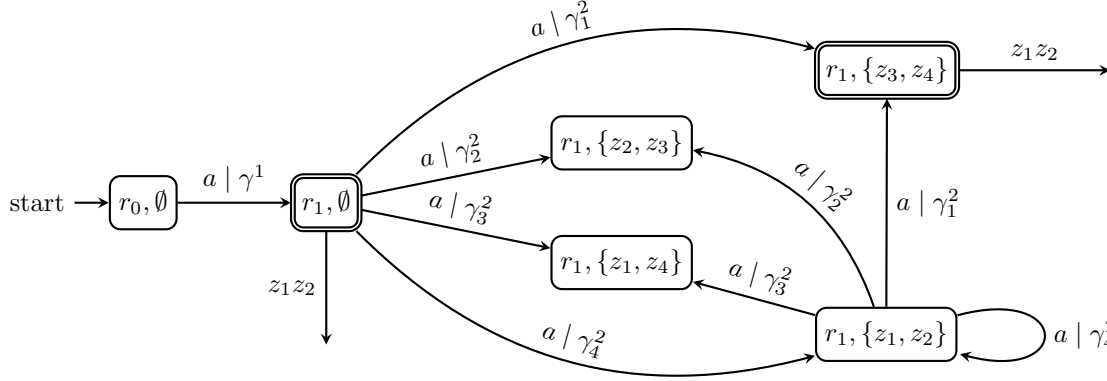


Figure 10: A copyless functional NSST equivalent to T_3 from fig. 2 and constructed according to definition 3. Note that we omit a number of irrelevant states from the diagram.

Lemma 12. $\mathcal{V}_\rho = \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'}$

Proof. We proceed by induction on the run ρ .

Base case. ρ 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. $\sigma = \rho \xrightarrow{s, \alpha} r$ is a run in T on ws and $\mathcal{V}_\rho = \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'}$.

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

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

As a result, we have that $\bigcup_{\sigma' \in \mathcal{R}(\sigma)} \mathcal{V}_{\sigma'} = \bigcup_{\beta \in \mathcal{D}(\alpha)} \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'} \circ \beta$. Applying the inductive hypothesis and lemma 11 to this equation, we obtain the desired equivalence:

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

□

Lemma 13. If $(w, w') \in \llbracket T \rrbracket$, then $(w, w') \in \llbracket T' \rrbracket$.

Proof. Assuming $(w, w') \in \llbracket T \rrbracket$, then there must be a run $\rho = q_0 \xrightarrow{w_1, \alpha_1} \dots \xrightarrow{w_n, \alpha_n} q_n$ in T on w such that $\text{Er}_X \circ \mathcal{V}_\rho \circ F(q_n) = w'$. By lemma 12, it holds that $\text{Er}_X \circ \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'} \circ F(q_n) = w'$, and, by definition 3, it holds that $\text{Er}_X \circ \bigcup_{\rho' \in \mathcal{R}(\rho)} \mathcal{V}_{\rho'} \circ F'((q_n, Y_n)) = w'$ for any $Y_n \subseteq X$ 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 ρ will be used later in the computation. Since each assignment in ρ 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 ρ . Therefore, $(w, w') \in \llbracket T \rrbracket$ implies $(w, w') \in \llbracket T' \rrbracket$. □

Lemma 14. If $(w, w') \in \llbracket T' \rrbracket$, then $(w, w') \in \llbracket T \rrbracket$.

Proof. Assuming $(w, w') \in \llbracket T' \rrbracket$, there exists a run $\rho' = (q_0, \emptyset) \xrightarrow{w_1, \beta_1} \dots \xrightarrow{w_n, \beta_n} (q_n, Y_n)$ in T' on w such that $\text{Er}_X \circ \mathcal{V}_{\rho'} \circ F'((q_n, Y_n)) = w'$. By definition 3, there exists $(q_{k-1}, w_k, \alpha_k, q_k) \in \Delta$, for each transition $((q_{k-1}, Y_{k-1}), w_k, \beta_k, (q_k, Y_k)) \in \Delta'$ occurring in ρ' , such that (i) $\beta_k \in \mathcal{D}(\alpha_k)$ and (ii) $y \in Y_k$ iff there exists $x \in X$ such that $x \in \alpha_k(y)$ and $x \notin \beta_k(y)$. Therefore, we may construct the run $\rho = q_0 \xrightarrow{w_1, \alpha_1} \dots \xrightarrow{w_n, \alpha_n} q_n$ for which $\rho' \in \mathcal{R}(\rho)$ by stringing together these transitions from Δ . This shows that every accepting run ρ' in T' on w has a corresponding accepting run ρ in T on w such that $\mathcal{V}_\rho \circ F(q_n) = \mathcal{V}_{\rho'} \circ F'((q_n, Y_n))$. Hence, $(w, w') \in \llbracket T' \rrbracket$ implies $(w, w') \in \llbracket T \rrbracket$. □

Theorem 3. *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 $\llbracket T \rrbracket$ and $\llbracket T' \rrbracket$ follows immediately from the conjunction of lemmas 13 and 14. The number of states and variables in T' is clear from definition 3. \square

Combining theorem 3 with theorem 3.5 of Alur and Deshmukh [3], which states that every functional NSST is equivalent to a DSST, we obtain the following corollary.

Corollary 1. *Every diamond-free DSST is equivalent to a copyless DSST.*

5 Conclusion

This paper revisits the constructions for the sequential composition of copyless streaming string transducers developed by Alur and Cerný [1], Alur and Deshmukh [3]. While those constructions result in a copyful transducer in general, we showed that these composite transducers exhibit only the tamest of copyful behavior. As a characterization, we defined diamond-free SSTs and proved that the composite transducer is always an element of this subclass of copyful SSTs. Moreover, we showed that diamond-free SSTs are no more powerful than copyless SSTs. To achieve this, we formulated a method for transforming any diamond-free SST into an equivalent copyless SST. The composition construction followed by the conversion from diamond-free to copyless SSTs comprise a complete and direct procedure for composing copyless SSTs.

References

- [1] 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>.
- [2] 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>.
- [3] 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.
- [4] Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. Departmental papers (CIS), Department of Computer & Information Science, University of Pennsylvania, 2011. URL https://repository.upenn.edu/cgi/viewcontent.cgi?article=1607&context=cis_papers.
- [5] 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>.
- [6] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge University Press, New York, USA, 2012.
- [7] 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.
- [8] Emmanuel Filiot and Pierre-Alain Reynier. Copyful streaming string transducers. In *Reachability Problems RP*, volume 10506 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 2017. URL https://doi.org/10.1007/978-3-319-67089-8_6.

- [9] Emmanuel Filiot and Pierre-Alain Reynier. Copyful streaming string transducers. *Fundam. Informaticae*, 178(1-2):59–76, 2021. URL <https://doi.org/10.3233/FI-2021-1998>.
- [10] Théodore Lopez, Benjamin Monmege, and Jean-Marc Talbot. Determinisation of finitely-ambiguous copyless cost register automata. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 138 of *LIPIcs*, pages 75:1–75:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL <https://doi.org/10.4230/LIPIcs.MFCS.2019.75>.
- [11] Albert R Meyer. Weak monadic second order theory of successor is not elementary-recursive. In *Logic colloquium*, pages 132–154. Springer, 1975.
- [12] Anca Muscholl and Gabriele Puppis. Equivalence of finite-valued streaming string transducers is decidable. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 132 of *LIPIcs*, pages 122:1–122:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL <https://doi.org/10.4230/LIPIcs.ICALP.2019.122>.
- [13] Jay Thakkar, Aditya Kanade, and Rajeev Alur. Transducer-based algorithmic verification of retransmission protocols over noisy channels. In *Formal Techniques for Distributed Systems - Joint IFIP WG 6.1 International Conference, FMOODS/FORTE*, volume 7892 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2013. URL https://doi.org/10.1007/978-3-642-38592-6_15.