On the Construction of Convergent Transfer Subgraphs in General Labeled Directed Graphs

Mohammed Ghriga Christopher League

Long Island University · Computer Science
1 University Plaza, LLC 206, Brooklyn, NY 11201
mohammed.ghriga@liu.edu christopher.league@liu.edu

Abstract

Let L and L' be finite input and output alphabet sets, respectively. In a graph whose edges are labeled with input/output symbols, the transfer decision problem [3] is to determine whether there are sequences over $L \times L'$ that guarantee transfer from a known vertex v to a destination vertex w. Convergent transfer subgraphs (CTS) are graphical representations of such sequences. This is an abstraction of the transfer of a communication protocol to a specific state for testing purposes.

Li et al. [6] presented a polynomial time algorithm to solve the CTS problem for labeled directed *acyclic* graphs. This was reduced to a linear-time algorithm by Li [5]. However, there are no efficient algorithms for general directed graphs. In this paper, we present an algebraic framework for the incremental construction of convergent transfer subgraphs. We shall prove composition theorems that form the basis for the practical construction of convergent transfer subgraphs over general directed graphs.

1 Motivation

Conformance testing is about determining whether (or to what extent) a particular implementation under test conforms to its formal specification. Generally it refers to *black box* testing, which does not involve analysis of the source code. Rather, we are to determine the level of conformance merely by providing some stimuli to the implementation and observing its responses. The technique is often applied to communication protocols, though this is not essential. In communicating systems, typically a formal specification governs a variety of distinct implementations, and concurrency and non-determinism make verification extremely challenging. The technique, however, can be applied to any system whose behavior can be represented as sequences of inputs and outputs.

The core of any kind of testing procedure is test-case selection. Our goal is to enumerate a set of cases that sufficiently exercise the various behaviors of the system so that the likelihood of faults during production use is reduced. For non-deterministic systems, a static set of test cases is necessarily insufficient, so testing must be *adaptive* [1, 4, 7, 9]. This means that the tester can change its test plan based on observed responses from the implementation under test.

There are many languages and frameworks available for specifying the behavior of concurrent systems. For our purposes, it is sufficient to treat a specification as a *non-deterministic finite state machine.* This is a directed graph where nodes represent system states and each edge is labeled with an input/output pair. If the system is provided with the specified input, we can expect the specified output, and the system will transition to the new state. There are several sources of non-determinism: from a given state and for a given input, there could be multiple edges with same or different outputs, and the same or different destination states. After just a few non-deterministic transitions, it can become extremely difficult to know for certain which state(s) we are in.

Kloosterman [4] identifies three useful kinds of traces to be found in non-deterministic specifications: a *synchronizing sequence* takes the machine from any state back to a start state. It can function essentially as a reset button that the tester can use to guarantee transfer back to a known state. A *transfer sequence* moves from one given state to another (such as from the start state to the source of the next transition to test). Finally, a *unique input/output sequence* [8] serves as a signature to distinguish a given set of states, and thus validate the outcome of a transition under test.

In deterministic automata, these can all be represented by sequences of input symbols, but in the non-deterministic case, they are trees or directed acyclic graphs. A *convergent transfer subgraph* (CTS) is a subgraph of the specification that represents a transfer sequence, moving control from one given state to another. We define it formally in the next section.

Discovering and building convergent transfer subgraphs in a complex specification can be extremely difficult. Ghriga and Kabore [3] presented an algorithm that was polynomial, but only if the specification is sparsely non-deterministic (roughly, a logarithmic amount of non-determinism), and becomes exponential as the number of non-deterministic edges approaches n, the number of nodes. They collaborated with Li [6] to develop an $O(n^2)$ algorithm. Later, this was improved to a linear algorithm by Li [5]. However, all of these results are for *acyclic* specification graphs. For example, the linear algorithm begins with a topological sort, which fails if there are cycles. Iterative behavior is, of course, important in many protocols, and so handling cycles in protocol specifications is critical.

Our contribution in this paper is an algebraic framework for the *incremental* construction and manipulation of convergent transfer subgraphs within general directed graphs. Specifically, we define two operators on graphs and show that the set of convergent transfer subgraphs is closed over these operations. This enables a way to reason about CTS composition, both sequentially and in parallel.

In the next section, we formalize our graph notation and define formally the different kinds of subgraphs and the operators we propose. Section 3 states and proves a number of useful properties about our operators and observations about the subgraphs. Finally, we discuss some consequences and future directions in section 4.

2 Formalization

Our directed graphs are defined by a set of vertices V and a set of edges $E \subseteq V \times L_{in} \times L_{out} \times V$. The sets L_{in} and L_{out} are input and output alphabets, respectively; a pair from $L_{in} \times L_{out}$ comprises an edge label. The notation $V(H)$ fetches the vertex set of a graph H, and $E(H)$ designates the edge set. We let G,H range over graphs; v,w,x,y,z (and variants) range over vertices; e over edges; and a,b,c over input/output symbols.

We use the notation a/b for a label in $L_{in} \times L_{out}$, and the notation $(x,a/b,y)$ for an edge in E. Superscripts are used to refer to the components of an edge:

e^-	is the initial vertex of edge e	x in the example
e^+	is the terminal (destination) vertex	y
e^I	is the input symbol on edge e	a
e^O	is the output symbol on edge e	b

Using these, we define a few important sets and notations relating to graph edges:

- $out_H(v) = \{e^I \mid e \in E(H) \wedge e^- = v\}$ is the set of all input symbols on outgoing edges of v.
- $E_H(v,a) = \{e \in E(H) \mid e^- = v \wedge e^I = a\}$ is the set of all outgoing edges of v that have input symbol a.
- $d^-_H(v,a) = |E_H(v,a)|$ is the out-degree of a vertex v with respect to the input symbol a.

In all cases, the subscript designating the graph of interest may be omitted if it is clear from the context.

The subgraph relation is defined entirely by the subset relation on vertex and edge sets. That is, G is a subgraph of H, written $G \subseteq H$, if and only if $V(G) \subseteq V(H)$ and $E(G) \subseteq E(H)$. Any subset of vertices V' induces a subgraph $H[V']$ by including all edges $\{e \in E(H) \mid e^- \in V' \wedge e^+ \in V'\}$ incident on vertices in V'.

A path is a (possibly empty) sequence of *connected* edges. That is, for any adjacent pairs e_1e_2 in a path, $e_1^+ = e_2^-$. A path $e_1, e_2, \ldots e_n$ is a path from x to y in H, denoted $P_H(x,y)$, if each $e_i \in E(H)$ and $e_1^- = x$ and $e_n^+ = y$. $R_H(v)$ is the set of vertices reachable from v; that is, $R_H(v) = \{v' \in V(H) \mid \exists P_H(v,v')\}$.

With these notations in place, we are now prepared to define formally one of the central concepts of this work: the transfer subgraph. The definition is exactly the one used in [2, 3, 6]; in later work, Li [5] introduced an alternative definition based on edge-coloring instead of input/output alphabets.

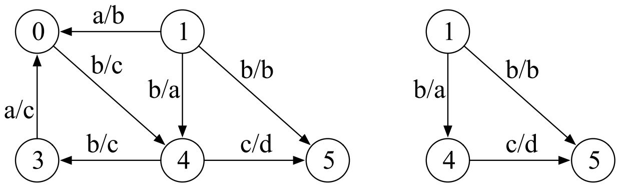

Figure 1: A graph G (left) and a convergent transfer subgraph $CTS(v_1, v_5)$ in G.

Definition 1 (transfer subgraph) *Let G be a labeled directed graph and $v,v' \in V(G)$. A subgraph G' of G is said to be a transfer subgraph from vertex v to v', written $TSG(v,v')$, if the following properties hold:*

1. $v,v' \in V(G')$
2. $\forall v_i \in V(G'), v_i \in R_{G'}(v) \wedge v' \in R_{G'}(v_i)$
3. $\forall v_i \in V(G'), |out_{G'}(v_i)| \leq 1 \wedge (|out_{G'}(v_i)| = 0 \leftrightarrow v_i = v')$
4. $\forall v_i \in V(G'), \forall a \in L_{in}, a \in out_{G'}(v_i) \rightarrow d^-_{G'}(v_i,a) = d^-_G(v_i,a)$

A more narrative explanation may be helpful. The first condition requires that the source vertex v and the destination (sink) vertex v' are vertices of the subgraph. The second property states that any vertex v_i in the TSG is reachable from v, and in turn, the destination v' is reachable from v_i. The third property expresses the requirement that all outgoing edges at any vertex (with the sole exception of the sink, v') share the same input symbol. The sink is the only vertex with no outgoing edges. The fourth property ensures that all non-deterministic choices in the parent graph G are preserved in the subgraph.

Definition 2 (convergent transfer) *A convergent transfer subgraph $CTS(v,v')$ is a transfer subgraph that is* acyclic.

The convergent transfer subgraph is equivalent to a transfer sequence in protocol conformance testing [2], to effect a transfer from one state (represented by the source vertex) to another state (the sink). An example of a general graph and a particular CTS can be found in figure 1.

We identify one more general kind of graph, useful for reasoning about properties of the transfer subgraphs:

Definition 3 (zero-one) *A graph H is said to be a 0-1 graph (or subgraph) if $|out_H(v)| \in \{0,1\}$ for every $v \in V(H)$.*

That is, all outgoing edges at any vertex share the same input symbol. A 0-1 graph drops all but one requirement of the transfer subgraph, so we have the hierarchy *CTS* $\subseteq$ *TSG* $\subseteq$ *0-1*.

The remaining definitions in this section are about the operators we propose for constructing transfer subgraphs. We begin with the observation that any 0-1 subgraph can be characterized by a simple function over its vertices.

Definition 4 (characterization) *We can represent a 0-1 subgraph $H \subseteq G$ by a characteristic function $\mu_H : V(G) \to (L_{in} \cup \{0, \perp\})$, defined as follows:*

$$\mu_H(v) = \begin{cases} 0 & \textit{if } v \in V(H) \wedge out_H(v) = \emptyset \\ a & \textit{if } v \in V(H) \wedge out_H(v) = \{a\} \\ \perp & \textit{if } v \notin V(H) \end{cases}$$

Obviously, the symbols 0 and $\perp$ are chosen to be distinct from any input symbol. $\mu_H(v) = 0$ indicates that v has no outgoing edges in the 0-1 subgraph H (it says nothing about the edges in the general parent graph G). $\mu_H(v) = \perp$ indicates that the vertex v is not part of the 0-1 subgraph. Otherwise, $\mu_H(v)$ maps the vertex to the unique input symbol on its outgoing edges.

Just as any 0-1 subgraph can be characterized by a function, conversely one can *construct* the 0-1 subgraph H from the original graph G and the function. Therefore, 0-1 subgraphs (including transfer subgraphs) can be interchangeably referred to by their characteristic functions.

Definition 5 (projection) *Let G be a graph, and $v \in V(G)$. The operation $G[v]$ denotes the subgraph induced by all the vertices in G reachable from v. Formally, $G[v] = G[R_G(v)]$.*

Projection is a simple operation, and it is easy to show that the universe of 0-1 subgraphs of a given graph G is closed under it. We state this property here, but more complex properties will be addressed in the next section.

Lemma 1 (closed under projection) *For a given graph G, if $G' \subseteq G$ is a 0-1 subgraph, then $G'[v]$ is also a 0-1 subgraph for any $v \in V(G')$. The same closure property holds for TSG and CTS too.*

Proof is immediate from definitions 1, 2, and 5. □

The composition of 0-1 subgraphs is best described by reasoning in terms of their characteristic functions. First we define an operator over the *range* of these functions: the input alphabet L_{in} with the additional symbols 0 and $\perp$.

Definition 6 (composition) *The binary operator $\oplus$ is a left-associative composition over the elements in $L_{in} \cup \{0, \perp\}$. Let $a, b \in L_{in} \cup \{0, \perp\}$. The operator $\oplus$ is defined as follows:*

$$\begin{aligned} a \oplus 0 &= 0 \\ a \oplus \perp &= a \\ a \oplus b &= b \quad \textit{where } b \in L_{in} \end{aligned}$$

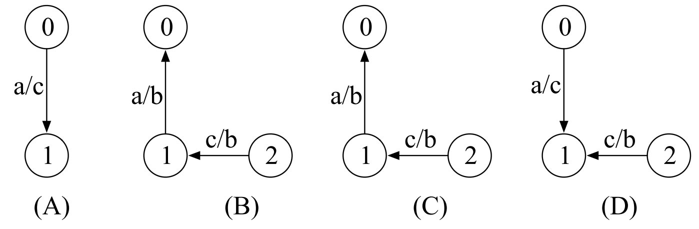

Figure 2: Graph composition example: $C = A \oplus B$ and $D = B \oplus A$.

Intuitively, the composition operator favors its right operand, unless that operand is $\perp$ (undefined); in this case alone, it uses the left operand. In particular, this means that composition is not commutative: for a given $a \in L_{in}$, $a \oplus 0 = 0$ but $0 \oplus a = a$. Now we overload the composition operator on the graphs themselves. Figure 2 contains some sample graphs and their compositions.

Definition 7 (subgraph composition) *Let G be a graph. Let $G_1, G_2 \subseteq G$ be 0-1 subgraphs. The composition of G_1 and G_2, denoted $G_1 \oplus G_2$, is the subgraph of G whose characteristic function is $\mu_{G_1 \oplus G_2}(v) = \mu_{G_1}(v) \oplus \mu_{G_2}(v)$ for all $v \in V(G)$.*

3 Properties

Lemma 2 (characteristic of composition) *Let G be a graph, and $G_1, G_2 \subseteq G$ be 0-1 subgraphs. The characteristic function $\mu_{G_1 \oplus G_2}$ of their composition can be restated as follows:*

$$\mu_{G_1 \oplus G_2}(v) = \begin{cases} \mu_{G_2}(v) & \textit{if } v \in V(G_2) \\ \mu_{G_1}(v) & \textit{if } v \in V(G_1) \backslash V(G_2) \\ \perp & \textit{if } v \notin V(G_1) \cup V(G_2) \end{cases}$$

for all $v \in V(G)$.

Proof. Definition 7 states that $\mu_{G_1 \oplus G_2}(v) = \mu_{G_1}(v) \oplus \mu_{G_2}(v)$. • For $v \in V(G_2)$, $\mu_{G_1}(v) \oplus \mu_{G_2}(v) = \mu_{G_2}(v)$ since $\mu_{G_2}(v) \in L_{in} \cup \{0\}$. • For $v \in V(G_1) \backslash V(G_2)$, $\mu_{G_1}(v) \oplus \mu_{G_2}(v) = \mu_{G_1}(v) \oplus \perp = \mu_{G_1}(v)$. • Finally, when $v \notin V(G_1) \cup V(G_2)$, $\mu_{G_1}(v) \oplus \mu_{G_2}(v) = \perp \oplus \perp = \perp$. □

Lemma 3 (subgraph containment) *Let G be a graph, and $G_1, G_2 \subseteq G$ be 0-1 subgraphs. Then, $G_2 \subseteq (G_1 \oplus G_2) \subseteq (G_1 \cup G_2)$.*

Proof. Trivially, $V(G_1 \oplus G_2) = V(G_1 \cup G_2)$, so this property is about the edge sets. It suffices to show that $E(G_1) \subseteq E(G_1 \oplus G_2) \subseteq E(G_1 \cup G_2)$. The composed edge set $E(G_1 \oplus G_2)$ is the union of the following:

$$\bigcup_{v \in V(G_2) \wedge \mu_{G_2}(v) \in L_{in}} E_G(v, \mu_{G_2}(v)) \quad \text{and} \quad \bigcup_{v \in V(G_1) \backslash V(G_2) \wedge \mu_{G_1}(v) \in L_{in}} E_G(v, \mu_{G_1}(v))$$

The former is precisely equal to $E(G_2)$, because G_2 is a 0-1 subgraph. The latter is a *subset* of the edges in $E(G_1)$. Thus, we have $E(G_1) \subseteq E(G_1 \oplus G_2) \subseteq E(G_1 \cup G_2)$ as required. □

Lemma 4 (path containment) *Let G be a graph, and $G_1, G_2 \subseteq G$ be 0-1 subgraphs. If $P = eP'$ is a path in $G_1 \oplus G_2$ and $e \in E(G_2)$, then P is completely contained in G_2.*

Proof by induction on the length of path P'. • Base case: P' is the empty path, so $P = e$. Since $e \in E(G_2)$, the conclusion holds. • Induction: assume the property holds for a path of length k. Once again, let $P = eP'$ in $G_1 \oplus G_2$ where $e \in E(G_2)$, and length of P' is $k+1$. We can write P as $eP''(x, a/b, x')$ where P'' is a path of length k from the destination vertex of e to x. Since $G_1 \oplus G_2$ is a 0-1 subgraph of G, $\mu_{G_1 \oplus G_2} x = a$. Using the induction hypothesis, the path eP'' is completely contained in G_2, so $x \in V(G_2)$ too. Using lemma 2, $\mu_{G_1 \oplus G_2}(x) = \mu_{G_2}(x)$. Hence, $\mu_{G_2}(x) = a$ and $(x, a/b, x') \in E(G_2)$. Therefore, the path $eP''(x, a/b, x')$ is completely contained in G_2. □

Lemma 5 (acyclic) *Let G be a graph, and $G_1, G_2 \subseteq G$ be 0-1 subgraphs. If G_1 and G_2 are acyclic, then so is $G_1 \oplus G_2$.*

Proof by contradiction. Suppose that G_1 and G_2 are acyclic, but $G_1 \oplus G_2$ is not. Hence, there exists a cycle C in $G_1 \oplus G_2$. Since G_1 is acyclic, then cycle C must contain at least one edge from G_2. Let $e = (x, a/b, x')$ be such an edge. We can write $C = eC'$ where C' is a path from x' back to x. Using lemma 4, C is completely contained in G_2. Hence, G' contains a cycle, which contradicts the premise. □

Lemma 6 (compositional closure) *Let G be a graph and $x, y, z \in V(G)$. Let G_1 and G_2 be 0-1 subgraphs of G. If $G_1 = CTS(x, y)$ and $G_2 = CTS(y, z)$ then $(G_1 \oplus G_2)[x]$ is a convergent transfer subgraph from x to z in G.*

Proof. We first consider the boundary cases. • If $x = z$ then x is the sink of a CTS, so $\mu_{G_1 \oplus G_2}(x) = 0$. Specifically, $(G_1 \oplus G_2)[x]$ is the subgraph $(\{x\}, \emptyset)$, which is a CTS from x to $z(= x)$ in G. • If $x = y$ then G_1 is the trivial subgraph and $(G_1 \oplus G_2)[x] = G_2[x] = G_2$, which is a CTS from $x(= y)$ to z in G.

• Now consider the case $x \neq z \wedge x \neq y \wedge y = z$. This time, G_2 is the trivial subgraph and $G_1 \cup G_2 = G_1$ because $z(= y)$ is also in G_1. From lemma 3, we have $G_1 \oplus G_2 \subseteq G_1$. Observe that $\mu_{G_1}(v) = \mu_{G_1 \oplus G_2}(v)$ for $v \in V(G_1)$. This implies $G_1 \subseteq G_1 \oplus G_2$. Hence, $(G_1 \oplus G_2)[x] = G_1[x] = G_1$, which is a CTS from x to $z(= y)$ in G.

Next, consider x, y, z to be pairwise distinct. Let $v \in V((G_1 \oplus G_2)[x])$. To show that property 2 of definition 1 is satisfied, it suffices to show that there is a path from v to z in $(G_1 \oplus G_2)[x]$.

• Case 1: suppose $v \in V(G_2)$. Because $G_2 = CTS(y, z)$, there exists a path $P_{G_2}(v, z)$. The same path is in $G_1 \oplus G_2$ because $G_2 \subseteq G_1 \oplus G_2$. Since v is reachable from x in $(G_1 \oplus G_2)[x]$, all vertices in $P_{G_2}(v, z)$ are in $R_{(G_1 \oplus G_2)[x]}(x)$. Hence, $P_{G_2}(v, z)$ is in $(G_1 \oplus G_2)[x]$.

• Case 2: suppose instead that $v \in V(G_1) \backslash V(G_2)$. Because $G_1 = CTS(x, y)$, there is a path $P_{G_1}(v, y)$ of length $k > 0$. We write

$$P_{G_1}(v, y) = (w_0, a_0/b_0, w_1) \ldots (w_{k-1}, a_{k-1}/b_{k-1}, w_k)$$

where $w_0 = v$ and $w_k = y$. Let $S = \{w_1, w_2, \ldots w_{k-1}\} \cap V(G_2)$ represent the set of vertices in the path that are also part of G_2. There are two sub-cases:

• Case 2a ($S = \emptyset$): $\mu_{G_1 \oplus G_2}(w_i) = a_i$ for $0 \leq i < k$, using lemma 2. It follows that $P_{G_1}(v, y)$ is in $G_1 \oplus G_2$. Since $v \in R_{(G_1 \oplus G_2)[x]}(x)$, all vertices in $P_{G_1}(v, y)$ are in $R_{(G_1 \oplus G_2)[x]}(x)$. Hence, the concatenation $P_{G_1}(v, y) P_{(G_1 \oplus G_2)[x]}(y, z)$ is a path from v to z. The existence of $P_{(G_1 \oplus G_2)[x]}(y, z)$ follows immediately from case 1.

• Case 2b ($S \neq \emptyset$): let j be the smallest index such that $w_j \in S$. $j \neq 0$ because $w_j (= v) \notin V(G_2)$ and therefore $w_j \notin S$. However, j exists in the range $0 < j < k$ and the prefix $\{w_1, \ldots w_{j-1}\} \cap S = \emptyset$. From cases 1 and 2.1, it follows that $P_{G_1}(v, w_j) P_{G_1 \oplus G_2}(w_j, z)$ is a path from v to z in $(G_1 \oplus G_2)[x]$, where $P_{G_1}(v, w_j)$ is the path which consists of the first j edges of $P_{G_1}(v, y)$.

Also, it is clear now that $z \in V((G_1 \oplus G_2)[x])$. For property 3 of definition 1, it suffices to observe that $(G_1 \oplus G_2)[x]$ is a 0-1 subgraph of G and z is the only vertex with no outgoing edges. Moreover, property 4 of definition 1 is readily satisfied from our use of characteristic functions. From lemma 5 and definition 5, $(G_1 \oplus G_2)[x]$ is acyclic. Thus, it is a CTS from x to z. □

We are now ready to establish the composition theorems. Let $G_1, G_2, \ldots G_n$ be a sequence of 0-1 subgraphs of a given graph G and $x \in V(G)$. For $n \geq 1$ we define f_n as a function of $n+1$ variables, as follows:

$$\begin{array}{rcll} f_1(G_1, x) & = & G_1[x] & \\ f_k(G_1, G_2 \ldots G_n, x) & = & (f_{n-1}(G_1, G_2 \ldots G_{n-1}, x) \oplus G_n)[x] & \text{if } n \geq 2 \end{array}$$

Theorem 1 (sequential construction) *Let G be a graph and $G_1, G_2, \ldots G_k$ be a sequence of 0-1 subgraphs of G. Let $x_1, x_2, \ldots x_{k+1} \in V(G)$. If $G_i = CTS(x_i, x_{i+1})$ for each i in the range $1 \ldots k$, then $f_k(G_1, G_2, \ldots G_k, x_1)$ is a convergent transfer subgraph from x_1 to x_{k+1} in G.*

Proof by induction on k, using lemma 6 (compositional closure). □

The previous theorem shows that CTSs can be sequentially combined, to achieve transitivity. The following result indicates the way to combine CTSs to maintain parallel transfer to the same destination.

Theorem 2 (parallel construction) *Let G be a graph and $G_1, G_2, \ldots G_k$ be a sequence of 0-1 subgraphs in G. Let $x_1, x_2, \ldots x_k, y \in V(G)$. If $G_i = CTS(x_i, y)$ (for* *each $i \in \{1 \ldots k\}$), then $(G_1 \oplus G_2 \oplus \ldots \oplus G_k)[x_j]$ is a CTS from x_j to y in G, for all $j \in \{1 \ldots k\}$.*

Proof by induction on k. It holds trivially for $k = 1$; we assume it holds for k and demonstrate $k+1$. Given the left-associativity of $\oplus$, we write $\oplus_{i=1}^{k+1} G_i = (G_1 \oplus \ldots \oplus G_k) \oplus G_{k+1} = (\oplus_{i=1}^{k} G_i) \oplus G_{k+1}$. By lemma 3 (subgraph containment), definition 5 (projection), and the premise, we have that $((\oplus_{i=1}^{k} G_i) \oplus G_{k+1})[x_{k+1}] = G_{k+1}$ is a CTS from x_{k+1} to y.

Let $j \in \{1 \ldots k\}$. We must show that $((\oplus_{i=1}^{k} G_i) \oplus G_{k+1})[x_j]$ is a CTS from x_j to y. The proof is similar to that of lemma 6. • The result is immediate if $x_j = x_{k+1}$. • If $x_j = y$ then $(\oplus_{i=1}^{k+1} G_i)[x_j] = (\{y\}, \emptyset)$ which is a trivial CTS from $x_j (= y)$ to y.

• If $x_{k+1} = y$ then $G_{k+1} = (\{y\}, \emptyset)$ is the trivial CTS from $x_{k+1} (= y)$ to y. We can invoke the induction hypothesis by showing that $\oplus_{i=1}^{k+1} G_i = \oplus_{i=1}^{k} G_i$. To simplify notation, let $A = \oplus_{i=1}^{k} G_i$ and $B = \oplus_{i=1}^{k+1} G_i = A \oplus G_{k+1}$. It is sufficient to show that $\mu_A(v) = \mu_B(v)$ for all $v \in V(G)$. Using lemma 2, there are three subcases. • If $v \in V(G_{k+1})$ it means that $v = y$, and $\mu_B(v) = \mu_{G_{k+1}}(v) = 0$. But, by the induction hypothesis, A is a CTS with sink y, which means $\mu_A(v) = 0$ too. • If $v \in V(A) \setminus V(G_{k+1})$ then $\mu_B(v) = \mu_A(v)$ by lemma 2. • If $v \notin V(A) \cup V(G_{k+1})$ then $\mu_B(v) = \perp$. By definition 4, $\mu_A(v) = \perp$ too since $v \notin V(A)$.

Now we may focus on the case where x_j, x_{k+1}, and y are pairwise distinct. We continue to use the abbreviations A and B. Let $v \in V(B[x_j])$. • Case 1: if $v \in V(G_{k+1})$ then there exists a path $P_{G_{k+1}}(v,y)$ which is also contained in B (by lemma 3) and $B[x_j]$ (because v is reachable from x_j). Otherwise, $v \in V(A)$ and there exists a path $P = P_{A[x_j]}(v,y)$. • Case 2a: if none of the paths of G_{k+1} intersects P at a vertex other than y, then P is contained in B and $B[x_j]$. • Case 2b: otherwise, let w be the closest intersecting vertex to v. Use the reasoning of cases 2.1 and 1 to construct a path from v to w and a path from w to y, respectively. This shows that $B[x_j]$ satisfies property 2 of definition 1. As in the proof of lemma 6, $B[x_j]$ satisfies the other properties of transfer subgraphs, and is acyclic. Hence, it is a CTS from x_j to y. $\square$

4 Conclusion

The construction of transfer sequences from non-deterministic specifications is an important problem for protocol conformance testing. We have defined two operators (composition and projection) and proved formally that they may be employed in the incremental construction of convergent transfer subgraphs within general directed graphs.

We believe this is the first *fully general* result to address CTS construction. Previous results took advantage of certain properties of the graph – the absence of cycles [3, 6, 5], or sparse non-determinism [3] – and thus have limited applicability. In future work, we expect this framework to be instrumental in the design of new space- and time-efficient algorithms for CTS construction, applied to real specifications.

References

[1] M. Ghriga and P. G. Frankl. Adaptive testing of non-deterministic communication protocols. In *Protocol Test Systems VI*, pages 347–362. Elsevier/North-Holland, 1994.

[2] M. Ghriga and P. Kabore. Un algorithme nondéterministe pour la génération de sous-graphes de transfer convergents. In *Actes du Colloque Francophone sur l'Ingénierie des Protocoles*. Hermes, 1999.

[3] M. Ghriga and P. Kabore. On the existence of convergent transfer subgraphs in labeled directed acyclic graphs. *Congressus Numerantium*, 136:207–214, 1999.

[4] H. Kloosterman. Test derivation from non-deterministic finite state machines. In *Protocol Test Systems V*, pages 297–308. Elsevier/North-Holland, 1993.

[5] W.-N. Li. Convergent transfer subgraph characterization and computation. In *Proc. Int'l Symp. on Circuits and Systems*, volume 3, pages 248–251. IEEE, May 2003.

[6] W.-N. Li, M. Ghriga, and P. Kabore. A polynomial time algorithm for deciding convergent transfer subgraphs in labeled directed acyclic graphs. *Congressus Numerantium*, 144:97–111, 2000.

[7] A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das. Non-deterministic state machines in protocol conformance testing. In *Protocol Test Systems VI*, pages 363–378. Elsevier/North-Holland, 1994.

[8] K. Sabnani and A. Dahbura. A protocol test generation procedure. *Computer Networks and ISDN Systems*, 15(4):285–297, 1988.

[9] P. Tripathy and K. Naik. Generation of adaptive test cases from non-deterministic finite state models. In *Protocol Test Systems V*, pages 309–320. Elsevier/North-Holland, 1993.