On the construction of *convergent transfer subgraphs* in general labeled directed graphs

38th CGTC
7 March 2007

Christopher League*
Mohammed Ghriga

Long Island University

First, I'll explain what we mean by "Convergent Transfer Subgraphs." This is a technique from Protocol Conformance Testing…

Protocol conformance testing

implementation

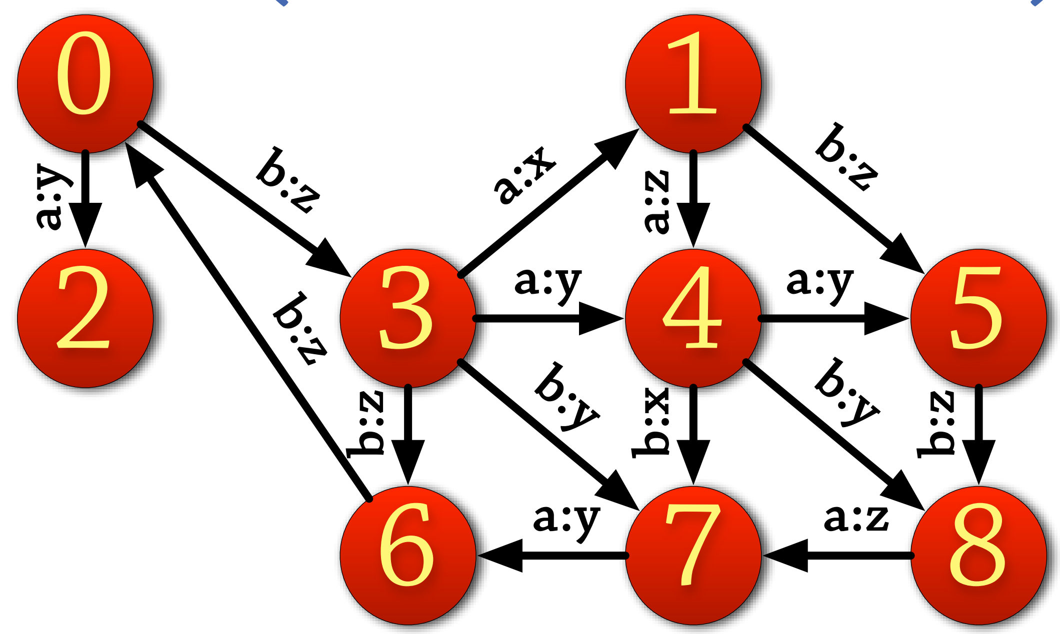
?

specification

…where we try to validate an implementation against a specification. We do **black box testing:** provide inputs to the implementation and observe its outputs, but don't look at the code. The specification…

Specification ≈ non-deterministic finite state transducer (automaton)

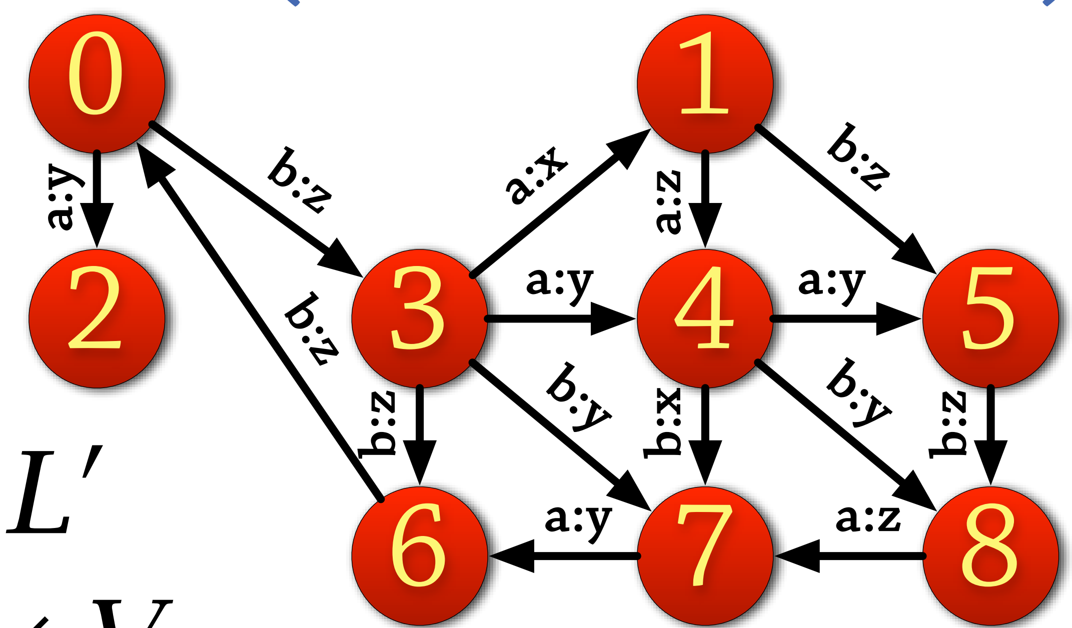
- Each transition labeled with accepted input & expected output



- States of implementation are not observable, just stimuli & responses

…is normally represented as a **Non-Deterministic Finite State Transducer,** or automaton. It can be generated from specifications written in some formal language, like LOTOS or Estelle. ¶ Each transition is labeled with an accepted input and expected output. The states of the implementation are not directly observable. Some notation…

Specification ≈ non-deterministic finite state transducer (automaton)



graph $G = (V, E)$

input, output alphabets L, L'

edge set $E \subseteq V \times L \times L' \times V$

set of input symbols on outgoing edges:

$$out_G(v) = \{a \mid (v, a, \_, \_) \in E\}$$

set of outgoing edges for a given input:

$$E_G(v, a) = \{e \in E \mid e = (v, a, \_, \_)\}$$

$$d_G^-(v, a) = |E_G(v, a)|$$

The graph consists of a set of vertices, and a set of edges. L and L' are sets of input and output symbols. So each edge is specified as a source vertex, input symbol, output symbol, and destination. ¶ Given a vertex v, the notation out(v) indicates the set of input symbols on outgoing edges. Example: out(3)={a,b}; out(5)={b}; out(7)={a}. ¶ For a vertex v and input symbol a, E(v,a) is the set of outgoing edges for a given input; and d− is the size of that set. ¶ So, non-determinism is present when d−(v,a) > 1 for some v,a.

Non-deterministic spec means testing must be adaptive

- With deterministic state machine, preset test cases can be derived in advance
- But many protocols are non-deterministic,
- So stimuli provided to machine may depend on its previous responses

Now, testing basically tries to cover the graph, visiting each transition and checking that the outputs of the system occur as specified. ¶ When the protocol is deterministic, we can create a bunch of static test cases in advance, and know exactly what to expect from the implementation. ¶ But many protocols are non-deterministic, so the stimuli we provide to the machine may depend on its previous responses. (More like a 2-way conversation between tester and system.) We call this **adaptive testing.**

To check transitions, find three kinds of traces in graph

- **Synchronizing sequence** takes machine from any state back to start state (reset)
- **Transfer sequence** moves from one given state to another (source of next transition to check)
- **Unique input output sequence** serves as signature to distinguish given set of states
- Goal: test plan that maximizes graph coverage

Adaptive test plans are not just sequences, but trees (or DAGs): if the system system responds this way, we'll pursue that plan; otherwise try this other plan. ¶ To construct adaptive tests for a given specification, it is very helpful to find 3 kinds of traces in the graph… [READ]

Convergent Transfer Subgraph

- A representation of an adaptive transfer sequence from one node to another

$G' \in CTS_G(v, v')$ if $G' \subseteq G$, G' is acyclic, and:

So, a **Convergent Transfer Subgraph** is a representation of an adaptive transfer sequence from one node to another. I'll go through the parts of the definition, then show an example. ¶ G' is a convergent transfer subgraph from v to v' in G if: **G' is an acyclic subgraph of G,** and…

Convergent Transfer Subgraph

- Source and sink are vertices of subgraph

$G' \in CTS_G(v, v')$ if $G' \subseteq G$, G' is acyclic, and:

1. $v, v' \in V(G')$

...v and v' are **in** the subgraph...

Convergent Transfer Subgraph

- All other vertices in subgraph are reachable from source, and sink is reachable from them

$$G' \in CTS_G(v, v') \text{ if } G' \subseteq G, G' \text{ is acyclic, and:}$$

1. $v, v' \in V(G')$
2. $\forall v_i \in V(G'), v_i \in R_{G'}(v) \wedge v' \in R_{G'}(v_i)$

…every node in the subgraph is reachable from the source v, and the sink v' is reachable from every node…

Convergent Transfer Subgraph

- Exactly one input symbol on all outgoing edges from each node (except sink)

$G' \in CTS_G(v, v')$ if $G' \subseteq G$, G' is acyclic, and:

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. $\forall v_i \in V(G'), |out_{G'}(v_i)| \leq 1$
   $\wedge (|out_{G'}(v_i)| = 0 \leftrightarrow v_i = v')$

And here come the tricky bits: for each node, all outgoing edges share the same input symbol… and the only node with no outgoing edges is the sink, v'. Finally…

Convergent Transfer Subgraph

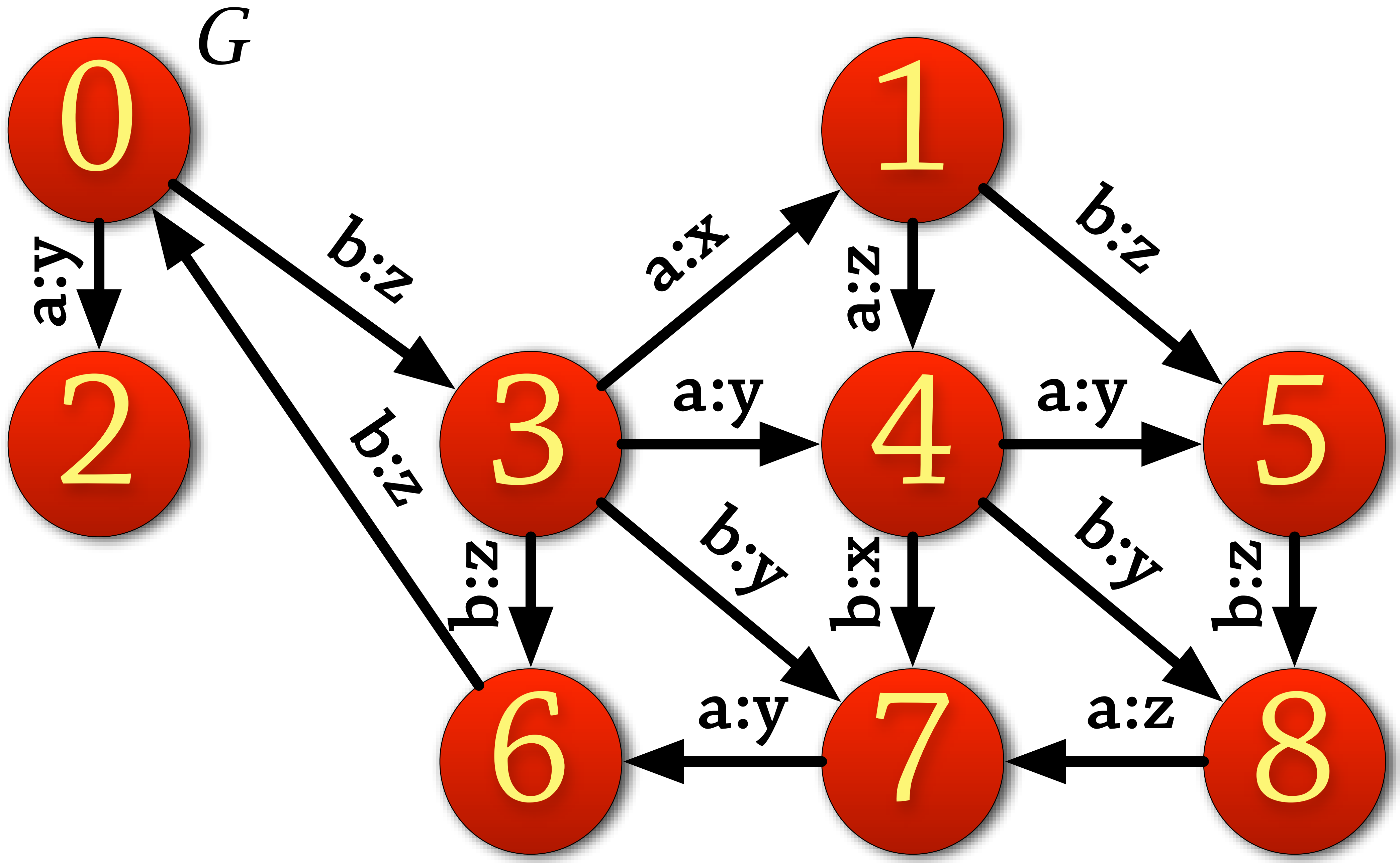
- All non-determinism on that unique input symbol must be preserved

$G' \in CTS_G(v, v')$ if $G' \subseteq G$, G' is acyclic, and:

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. $\forall v_i \in V(G'), \forall a \in L, a \in out_{G'}(v_i) \rightarrow$
   $d^-_{G'}(v_i, a) = d^-_G(v_i, a)$

At each node, all outgoing edges from the original graph **sharing the chosen input symbol** must be present in the subgraph. Conceptually, that source of non-determinism must be preserved. ¶ Let's see an example…

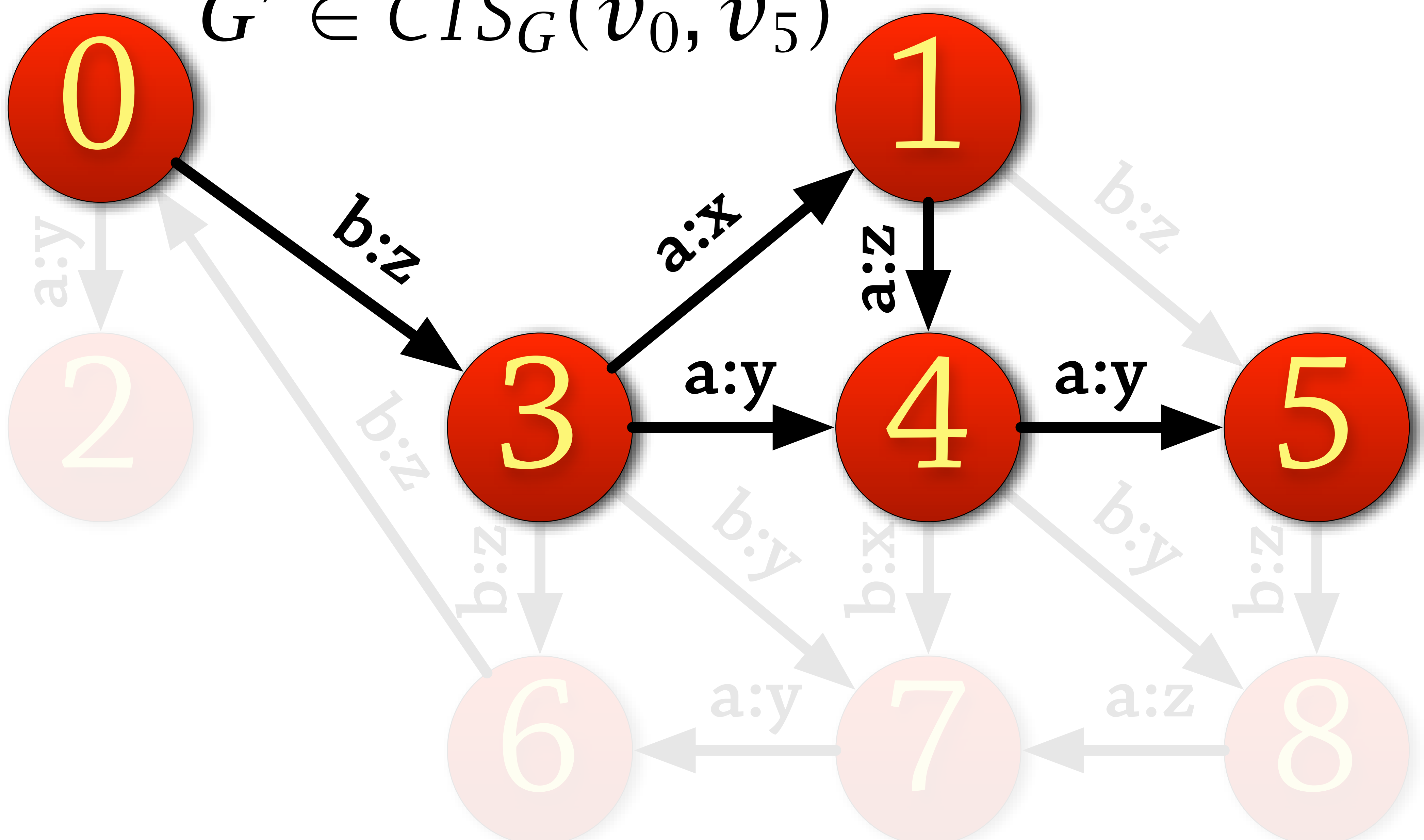
Example of Convergent Transfer Subgraph



This is **not** a CTS yet… this is the original specification graph, G. Suppose we want to effect a transfer from state 0 to state 5. ¶ We would input **b** and take the transition to state 3. ¶ From there, we'd input **a,** but we don't know in advance which edge the system will follow. According to the definition, we must include **both edges** in the CTS, and in either case, we need a plan for getting to state 5.

Example of Convergent Transfer Subgraph

$G' \in CTS_G(v_0, v_5)$



So here's the Convergent Transfer Subgraph from state 0 to state 5. (We could have chosen either edge from state 1.) **[Check time...]**

Algorithms to construct convergent transfer subgraphs

Ghriga & Kabore '99	polynomial, if sparsely
acyclic	non-deterministic
Li, Ghriga, & Kabore '00	square
acyclic	$O(n(n+e))$
Li '03	linear
acyclic	$O(n+e)$

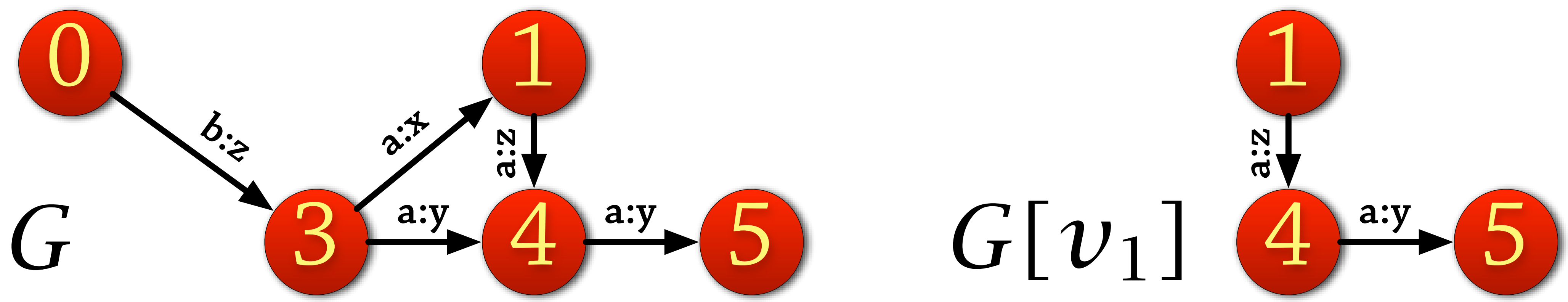
So, to construct (or find) these subgraphs, a number of algorithms have been proposed. ¶ Ghriga & Kabore offered an algorithm that was polynomial but only if "sparsely deterministic" … logarithmic amount of non-determinism… would become exponential otherwise. ¶ They collaborated later with Wing-Ning Li, and developed an n^2 algorithm, which Li later improved to linear. ¶ It's truly hard to imagine beating that, but **[CLICK]** in all these cases the input graphs are acyclic. Iteration can be very important in protocols, so we'd like to investigate lifting this restriction.

This contribution:

- Algebraic framework for incremental construction and manipulation of convergent transfer subgraphs
  - within general labeled directed graphs (cycles okay)

Our current work is an algebraic framework supporting the incremental construction and manipulation of convergent transfer subgraphs, within general labeled directed graphs. ¶ It is not itself an algorithm, but we expect it will lead to a variety of algorithms useful for conformance testing. ¶ So, here's what it looks like…

Projection operator *G[x]* induces subgraph reachable from *x*



Theorem. If $G' \in CTS_G(v, w)$ and $x \in V(G')$ then $G'[x] \in CTS_G(x, w)$

There are two operators. ¶ The first is a projection operator, G[x], which induces the subgraph of nodes and edges **reachable** from x. That's pretty simple, but the important point is that **[CLICK]** if you start with a CTS and apply the projection operator, **you still have a CTS.** ¶ The other operator needs a longer explanation…

Associate subgraphs with characteristic functions

For "0–1 subgraphs" $G' \subseteq G$ where $\forall v \in V(G'), |out_{G'}(v)| \leq 1$:

$$\mu_{G'} : V(G') \rightarrow L \cup \{0, \perp\}$$

$$\mu_{G'}(v) = \begin{cases} 0 & \text{if } v \in V(G') \wedge out_{G'}(v) = \emptyset \\ a & \text{if } v \in V(G') \wedge out_{G'}(v) = \{a\} \\ \perp & \text{if } v \notin V(G') \end{cases}$$

Can easily convert from G' to $\mu_{G'}$ and back.

We start by associating certain subgraphs with characteristic functions. We call them **0–1 subgraphs**, because each node has at most 1 input symbol on all its outgoing edges. (All CTSs are 0–1 subgraphs, but not all 0–1 subgraphs are CTSs) ¶ For these subgraphs, a characteristic function maps each vertex to its outgoing input symbol, to zero if that vertex is a sink, or to 'undefined' (bottom) if the vertex is not present in the subgraph.

Composition operator over range of μ

Remember, $\mu_{G'} : V(G') \rightarrow L \cup \{0, \perp\}$

For $x, y \in L \cup \{0, \perp\}$:

$$x \oplus y = y \quad \text{if } y \in L$$

$$x \oplus 0 = 0$$

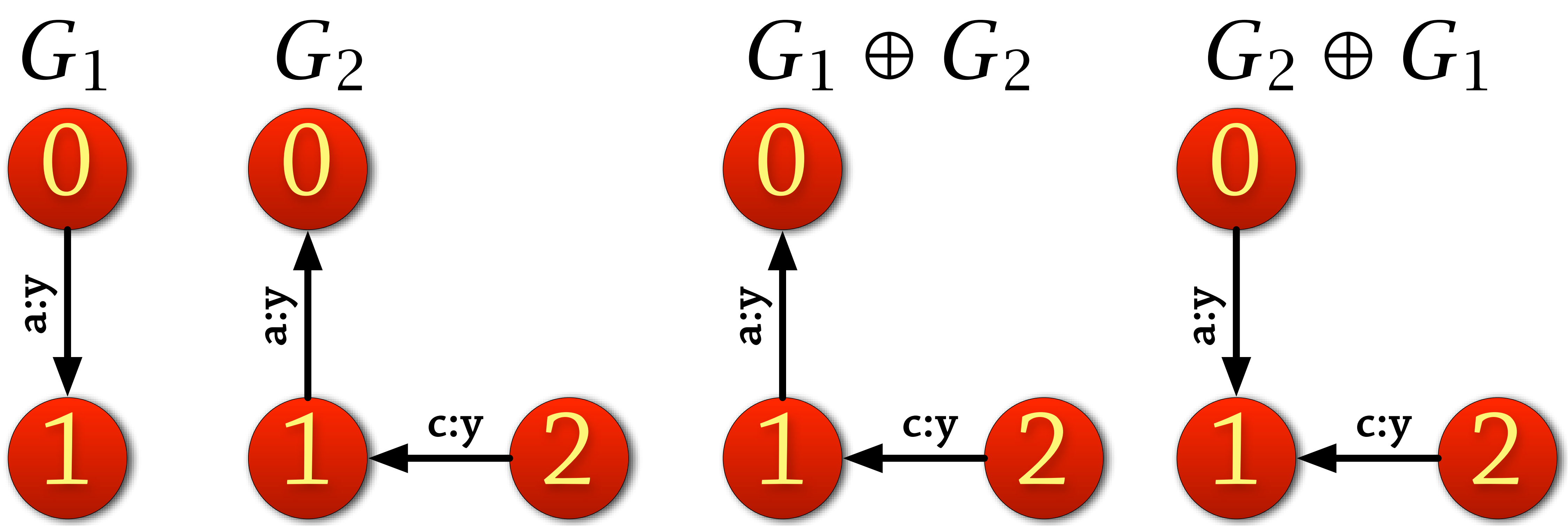
$$x \oplus \perp = x$$

Now, over that extended input set, we define a composition operator "o-plus". It always returns it right parameter, **unless that parameter is undefined.**

Composition operator over 0–1 subgraphs

$G_1 \oplus G_2$ defined as graph characterized by

$$\mu_{G_1 \oplus G_2}(v) = \mu_{G_1}(v) \oplus \mu_{G_2}(v) \quad \forall v \in V(G)$$



Using the characteristic function, we can apply that composition operator to 0–1 subgraphs as follows. G1 o-plus G2 (where G1,G2 are both subgraphs of the same graph G) is the graph characterized by the composition of the characteristic functions of G1 and G2, applied to all vertices in the underlying graph G. **[CLICK… examples]**

Composition operator over 0–1 subgraphs

Theorem. If $G_1 \in CTS_G(x, y)$ and $G_2 \in CTS_G(y, z)$ then $(G_1 \oplus G_2)[x] \in CTS_G(x, z)$

It's somewhat trickier to prove, but like the projection operator, the **set of convergent transfer subgraphs is closed over composition.** Specifically, [READ]

These operators permit incremental construction of CTS

- With them, we expect to build new efficient algorithms useful for conformance testing against non-deterministic finite state transducers

So, because of these properties, we expect this framework to be a useful tool for building traces used in adaptive conformance testing against non-deterministic specifications.

Thanks!

christopher.league@liu.edu