Equivalence and generalization in a layered network model: full paper

Steven Fortune[1]

August 25, 2014

Abstract

We investigate basic algorithmic and structural properties of a formal model of layered telecommunication networks. The network model includes ports, which are access points to data streams; links, which transmit data streams; and adapters, which convert data streams from one layer, i.e. encoding, to another. Two ports *communicate* if there is a path from one to the other in which every adaptation is balanced by a reverse adaptation. An instance of the model captures a snapshot of the configured state of a telecommunication network.

Two networks M and N with the same public ports are *(externally) equivalent* if for any other network Q, two ports in the composition $M \circ Q$ communicate exactly if they communicate in $N \circ Q$. M *generalizes* N if whenever two ports communicate in $N \circ Q$, the ports also communicate in $M \circ Q$.

We show the following. If adaptation is "simple", i.e. one or more data streams can be adapted into only a single data stream, then two networks are equivalent exactly if their normal forms, appropriately defined, are isomorphic. An immediate consequence is a linear time algorithm for equivalence. An extension yields a syntactic test and linear time algorithm for generalization as well. If adaptation models "reverse multiplexing", i.e. one or more data streams can be converted into more than one data stream, isomorphism of normal forms is sufficient but not necessary for equivalence; however, there is a quadratic-time algorithm for equivalence and generalization based on a notion of "black-box" testing. Finally, if adaptation models "protection switching," then testing equivalence is co-NP-complete.

1 Introduction

A telecommunication network is often modelled as a graph, with a vertex representing a network element or a physical location, and an edge representing a link over which data can be transmitted. This simple model does not capture the complex layered nature of many networks, where a logical link used by one technological level is provided as a service by a different level.

We study the formal properties of a network model that explicitly captures the layered nature of networks. The model is based on the concepts of layering and adaptation. A *layer* is a particular encoding of a stream of data. *Adaptation* is the conversion between layers; adaptation models both encapsulation and multiplexing. An instance of the model includes links, which transmit data and adapters, which perform adaptation.

This model is simple; however it is rich enough to apply uniformly and simultaneously to most connection-oriented network technologies (SONET, ATM, DWDM, Connection-oriented Ethernet, etc., as well as protocols that establish connections over connectionless networks, e.g. TCP over IP, RSVP over MPLS). An instance of the model represents the flow of data through a network, where communication at one layer may depend upon links at a different layer. We begin by describing the model informally (a more formal definition appears in Section 2.1) and then give examples of the utility of the model.

[1]Bell Laboratories, Alcatel-Lucent Technologies, 600 Mountain Ave, Murray Hill, NJ 07974. sjf@bell-labs.com.

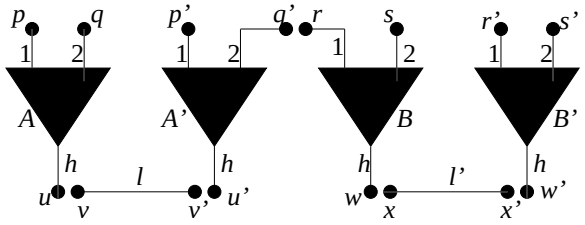

Figure 1: A, A', B, B' are adapters all of the same type; l, l are links; $1, 2$ are guest labels; h is a host label. Guest ports (p, q etc.) are on the top flat side; host ports (u, u', w, w', all with label h) are on the bottom pointed side. Ports v, v' are end ports of link l. Connections ($u \cdot v$, $q' \cdot r$, etc.) are drawn–in this figure only–by physical proximity of ports. Ports v and v' communicate, in one step, as do p and p', q and q'. Ports q and r' communicate in two steps. Ports p and q' do not communicate.

The model. A *link* represents the ability to transmit data from one place to another. A link has two endpoint *ports*, which are access points to the stream of data flowing on the link. Data injected into one port traverses the link and is ejected from the other port. A link is bidirectional, so data injected into the second port will be ejected from the first. A link is drawn as a line segment, with a port drawn as a dot; see Figure 1.

Links and ports can be interpreted concretely, as physical objects. For example, a link could be a copper or fiber cable and a port could be the physical connector at the end of the cable. More generally, a link could be a wireless connection and the port might be some connector attached to antenna circuitry. Links and ports can also be interpreted more abstractly. For example, a port might correspond to a subset of the data stream present at a physical port that can be independently switched by a network element. Similarly, a link might correspond to several physical links connected in sequence.

Ports and links are assigned a *layer*. A layer is essentially a type, and represents the relevant set of information required to describe the data stream, possibly including bandwidth, bit encoding scheme, error correcting codes, header information, etc. A link and its end ports always have the same layer.

Adaptation converts between layers. Adaptation represents both encapsulation, the encoding of one data stream within another, for example by adding extra header information, and multiplexing, where several data streams are combined into a single stream. An *adapter* performs adaptation. An adapter has a labelled set of *guest* ports and a labelled set of *host* ports, with layers and labels determined by the type of the adapter. An adapter is drawn as a triangle pointing down, with guest ports at the bottom and host ports at the top; see Figure 1.

Consider an adapter with a single guest port and a single host port. Any data stream of the appropriate layer can be injected into the guest port, where it is adapted to the layer of the host port and then ejected. If the resulting data stream is injected into the host port of another adapter of the same type, it is reconverted to the original layer and ejected from the guest port.

Like links, adapters are bidirectional. However, there is an asymmetry between guest and host ports: an arbitrary data stream of the appropriate layer can be injected into a guest port, and it will be adapted and ejected from the host port. However, the only data stream it is sensible to inject into a host port is one that results from a previous adaptation of the same type.

An adapter with multiple guest ports and a single host port models multiplexing. Several data streams can be injected into the guest ports of such an adapter; they are combined together and ejected from the host port. If the data stream is subsequently injected into the host port of an adapter of the same type, the individual streams are demultiplexed and ejected from the guest ports. The labels on guest ports serves to identify the individual data streams, so a data stream injected into the host port with label l of the first adapter will be ejected from the host port labelled l of the second adapter. The label can model time slot (in time-division multiplexing), wavelength (in wavelength-division multiplexing), channel identifier (ATM), etc.

An adapter with a single host port and several guest ports models reverse multiplexing: an arbitrary data stream can be split into several data streams. Unadaptation of reverse multiplexing recombines the data streams into a single stream. Reverse multiplexing is relatively rare, but occasionally a single high-bandwidth data stream is split into several low-bandwidth streams for transport. (A generalization models protection switching; see Section 5.)

A *network signature* describes the layers and adapter types allowed in the network; every network has an associated signature. In the model, a layer is abstract and is not decomposed in any way. All information about layers is defined by the set of adapter types, which specifies the set of possible adaptations between layers.

Networks are built by *connecting* two ports of the same layer together. Informally, data ejected from one port is injected to the other. A port can be connected to exactly one other port; a port that is not connected is a *free* port. A network is built from links and adapters by connecting ports together. In Figure 1, connections are depicted by proximity of ports. In all subsequent figures, two connected ports are shown drawn on top of each other.

A central notion for networks is communication. Two ports *communicate* if they behave like the end ports of a link, i.e. a data stream injected in one port is eventually ejected from the other port. Necessarily the ports must be of the same layer. Somewhat more formally, two ports communicate if there is a path through the network using links, adapters and connections for which every adaptation is balanced with a corresponding unadaptation of the same type. (See Section 2.1 for a more precise definition.) In Figure 1, ports v and v' communicate since they are the endpoints of a link; q and q' communicate since they are guest ports of adapters of the same type with the same label, and the host ports are connected to communicating ports v and v'. Similarly p and p' communicate, as do q and r'; however p and q' do not, since ports p and q' have different guest labels.

Representation of networks. An instance of the network model describes a snapshot of the configured state of a telecommunication network, in particular, the flow of data streams through the network. The main purpose of this paper is to explore the formal properties of the model, rather than the descriptive value of the model. However, for reference we briefly discuss how the model can describe network elements and networks, using legacy Sonet technology.

A typical Sonet network element, an add-drop multiplexer, has two "high-speed" OC48[1] ports, "east" and "west", and potentially a variety of "low-speed" ports, say at OC3 and STS1. The add-drop multiplexer can be configured to demultiplex the incoming OC48 stream from the east port into lower bandwidth streams, and either route the substreams to low-speed ports or to be remultiplexed and sent out on the west port. For this technology data flow is usually symmetric, so for any substream demultiplexed from the east port to a low-speed port there is a stream in the opposite direction multiplexed from the low-speed port to the east port; the west port is symmetric in the same way.

In this case the network signature would consist of three layers, STS1, OC3, and OC48, and two adaptation types, a three-one multiplexing of STS1 into OC3, and a eight-one multiplexing of OC3 into OC48. The first adaptation type would have labels 1,2,3 for the guest ports, corresponding to time slots within a SONET data stream. Similarly the second adaptation type would have 8 guest labels. With these layers and adaptation types, a configured add-drop multiplexor could be modeled using adapters as in Figure 2(a). (We emphasize that the dashed box in Figure 2(a) and the notion of 'network element' are not part of the formal model described in this paper; the dashed box is simply a pictorial convenience indicating the logical grouping of adapters that describes the network element). The add-drop multiplexor in Figure 2(a) is configured so that the data stream from the east port is demultiplexed into OC3 streams, one of which–time slot 3–is routed to a low-speed port. The stream in time slot 2 is further demultiplexed into STS1 streams, one of which is routed to a low-speed port, and the others are remultiplexed into the west port. In this representation some ports, e.g. the host ports of the OC3-OC48 adapters, represent physical objects, specifically a socket on the add-drop multiplexor into which a cable could be plugged. Other ports and the connections between them, for example the connections between guest ports at layer STS1, describe

[1] OC48 is 2.5 Gbps; OC3 is 155 Mbps; STS1 is 55 Mbps. STS1 is multiplexed three-one into OC3; OC3 is usually multiplexed eight-one into OC48, with the remaining capacity used for protection. Protection is ignored in this discussion.

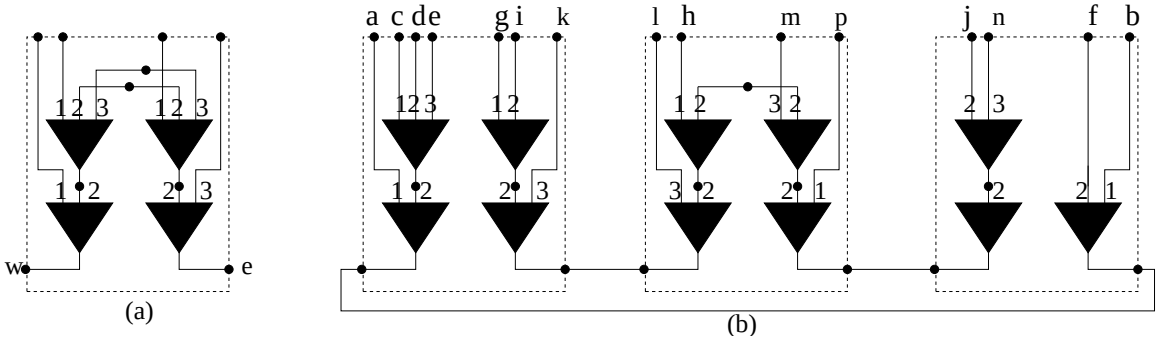

Figure 2: Dotted boxes outline aggregation of adapters into Sonet add-drop multiplexors (dotted lines are not part of the model). The host ports of the bottom row of adapters are OC48; the guest ports of the bottom row and the host ports of the top row are OC3; and the guest ports of the top row are STS1.

the configuration of the add-drop multiplexor, e.g. as set through a management interface. These ports and connections may not have any physical interpretation.

Data between Sonet network elements is carried on an optical-fiber cable. In the model, a cable is represented as a link. The fact that a cable is plugged into a network element is represented using a connection, as in Figure 2(b). In this case a 'connection' has a physical interpretation, specifically that a cable is plugged into a socket. Of course, links could be used to describe cables that carry lower-rate traffic as well, i.e. with connections to the low-speed ports of an adapter.

The description of a complete network can be synthesized out of the descriptions of individual network elements and links. Figure 2(b) depicts a simple SONET ring. Behavior emerges from the complete description; for example, ports a and b communicate.

By choosing layers and adaptation types appropriately, many different network technologies can be modeled. For example, optical wavelength-division multiplexing can be modeled with layers representing optical fiber, individual wavelengths, and client data streams (OC48, OC192, OC768, etc.) with adaptation types that represent the multiplexing of different client streams onto the fiber. In this case, the adapter guest port labels might represent the particular wavelength used by a client data stream. ATM and MPLS are similar; for example with ATM the guest port labelling can represent virtual channel index and virtual path index.

Results. The results in this paper concern the formal properties of the model, in particular, the external behavior of a network and the algorithmic question of deciding whether two networks are equivalent, or whether one is more general than the other. The external behavior of a network is determined by a set of distinguished "public" ports; a public port must be free. In Figure 2(b), we might decide that the low-speed ports—with labels a through p—define the external interface of the Sonet ring and hence define the public ports. For any given pair of public ports, there may be no relation between the ports or the two ports may communicate, in which case the behavior of the two ports is indistinguishable from a link. However, there may also be a potentially complex set of adaptations and unadaptations relating the two ports. For example, in Figure 2(b), ports c and f do not communicate but are related, in particular the STS1 data stream at port c is one of the three streams multiplexed together at OC3 port f.

We define equivalence and generalization in terms of composition and communication. The composition $M \circ_\sigma N$ of two networks is the network obtained by connecting some of the public ports of M and N, where σ specifies the connections. Suppose networks M and N have the same (or corresponding) public ports. M and N are *externally equivalent*, $M \equiv N$, if for every network Q, every σ, ports p and q communicate in $M \circ_\sigma Q$ if and only if they communicate in $N \circ_\sigma Q$. M *generalizes* N, $N \preceq M$, if whenever two ports communicate in $N \circ_\sigma Q$, they also communicate in $M \circ_\sigma Q$.

In Figure 3, all networks A through E have the same public ports p and q. However only D and E

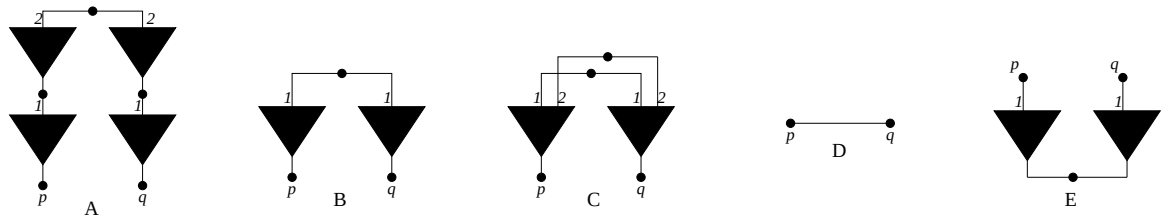

Figure 3: In this and subsequent figures, connected ports are superimposed. Host port labels are omitted. In each of A through E, ports p and q are public and are the only free ports. We have $A \preceq B \preceq C \preceq D \equiv E$.

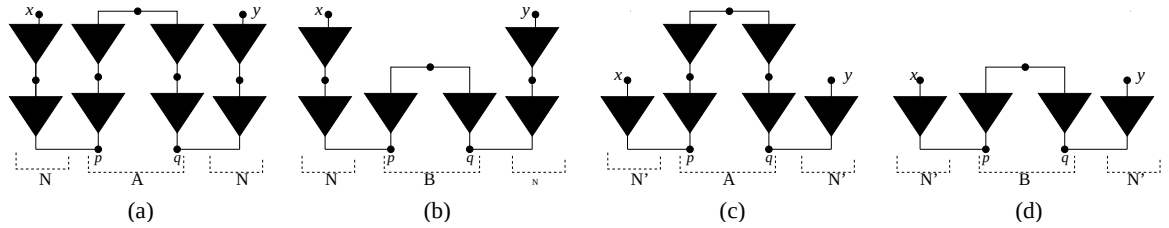

Figure 4: Networks A and B are redrawn from Figure 3 with guest labels omitted for clarity. Ports x and y communicate in networks $A \circ_\sigma N$ (a), $B \circ_\sigma N$ (b), and $B \circ_\sigma N'$ (d). In network $A \circ_\sigma N'$ (c), x and y do not communicate, so A and B are not equivalent.

are equivalent, informally because communication and links are indistinguishable. Networks A and B are redrawn in Figure 4 each composed with a network N with a pairing σ that connects p and q to ports of N, and similarly to a network N'. In Figure 4(a) ports x and y communicate in $A \circ_\sigma N$; similarly x and y communicate in $B \circ_\sigma N$. However, in Figure 4(d), x and y communicate in $B \circ_\sigma N'$, but they do not communicate in $A \circ_\sigma N'$ (Figure 4(c)). Hence A and B are not equivalent;, however from the results below it follows that B generalizes A, $A \preceq B$. In Figure 3 we have $A \preceq B \preceq C \preceq D \equiv E$.

An important component of our results is the notion of normal form. In effect, the normal form of a network is the simplest equivalent representation that preserves public ports. The normal form can be obtained by replacing communication with links, and then eliminating links whenever possible. Figure 2(a) is already in normal form if external ports are considered public. Figure 5 is the normal form of Figure 2(b),

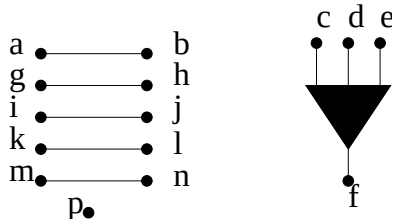

Figure 5: Normal form of network in Figure 2(b), assuming the public ports are the labelled ports a through p. Ports a and b communicate in Figure 2 so the normal form is a link, and similarly for the other links. Ports c, d and e become the guest ports of an adapter with host port f. Port p is public in Figure 2 but cannot communicate in any network composition, so it becomes isolated.

where the labelled ports are considered public.

We show the following. First consider networks with simple adaptation, that is, any adapter has only a single host port. Then two networks M and N with the same public ports are equivalent exactly if their normal forms are isomorphic. An immediate consequence is that there is a test for equivalence that runs in time linear in the sum of the sizes of M and N. A variant of this result gives a syntactic condition on normal forms that holds exactly if one network generalizes the other; again a linear time algorithm can test generalization.

A network instance can have adapters with several host ports; this models so-called reverse adaptation, where a single high-bandwidth data stream can be split into several lower-bandwidth data streams for transport. In this case isomorphism of normal forms certainly implies equivalence of the original networks, but the converse is not true. Nonetheless, there is a quadratic algorithm for testing whether one network generalizes another, and hence of testing whether two networks are equivalent.

A final result concerns a protection-switching variant, which can model the routing of communication over two disjoint paths, or more generally the routing of k communicating pairs over $l > k$ paths. In this case it turns out that equivalence is co-NP-complete, using a reduction from testing equivalence of monotone boolean circuits.

Applications. The network model in this paper was intended as the formal basis for NetML[3]. NetML is an XML-based language that describes a configured instance of a multilayer, multitechnology telecommunication network in a manner that is uniform across technologies and layers. NetML in fact extends to connectionless networks such as IP and captures the interconversion between connection-oriented and connectionless networks. The intent was to have a uniform description that could be the basis for many network analysis tasks. For example, given that two ports communicate, it may be of interest to estimate the reliability of communication. With a NetML description, it is possible to trace the communication path through its constituent network elements and links, and then to form an aggregate reliability estimate from individual reliability estimates. At least in principle, the same analysis code could be used for any network technology or combination of technologies. A description in NetML could also be used for a wide variety of other network analysis tasks, e.g. determining link utilization, summarizing traffic properties, planning circuit rearrangements, root-cause alarm analysis, reporting of recent changes, etc. NetML received some, though not widespread, use; for example NetML was used to describe IP networks as discovered by NetInventory [4], and was used to describe Sonet networks for the purpose of time-slot defragmentation [1].

A specific potential application of the equivalence and generalization algorithms is to assist with network design tools. The network design problem can be described generically as 'Given a set of client service demands, an available physical topology, and the cost and capacity of network elements, design the minimal-cost network infrastructure that provides the requested services'. A client demand is typically a specified amount of bandwith from one location to another; the physical topology specifies links (e.g. fiber) connecting one location to another that are available for use. Algorithms for network design are potentially quite complicated, including for example the routing of client demands over the underlying physical network, choosing disjoint protection routes, selecting wavelengths in optical networks, and in the case of multiple technologies, e.g. Ethernet over optical, simultaneously designing at multiple network layers. Such algorithms are often computationally challenging (NP-hard) and have intricate technological constraints, e.g. on capacity of network elements or allowable routing and cross-connect rules.

The equivalence and generalization algorithms could allow a verification that a network design provides the requested services, in a way that is independent of the tools that produced the network design. Suppose both the requested services S and the network design D are expressed as networks using the model in this paper. If $S \equiv D$, then the network design exactly captures the request; if $S \preceq D$, then the network design provides all the requested functionality plus possibly more. For example, Figure 5 could be the requested network S and Figure 2(b) the designed network D. A link from a to b in S is request for bandwidth, which is provided by the communicating pair of ports a and b in D. The specific novelty here is that a demand may be more general than just a 'link', but may also include a specification of adaptation, e.g. the multiplexing of c, d, e into f. While such adaptation as a design requirement is perhaps not common, it can occur if the overall network is composed of independently-designed autonomous networks, and data

streams multiplexed together in one network are demultiplexed in another. The clear attraction is a uniform language for expressing both network requirements and network design, and a way of relating the two.

The verification that requested services are actually provided could apply to operating networks as well as to designs of networks. In many cases the management-level interface to a network element can provide the configured state of the network element, as for example in figure 2(a). If this configured state is known for every network element, and the link connectivity is known or can be inferred, then a global description of the network can be constructed as in figure 2(b). From this the equivalence and generalization algorihtms can verify that requested services are provided. Alternately, the normal-form algorithm can give a minimal representation of the services that are actually configured in the operating network.

Other work. Optical networks are a prime examples of connection-oriented networks, with a wealth of literature. A general reference on optical network design is [17], covering optical add-drop multiplexors, regeneration, routing, wavelength assignment, grooming, etc. Some representative research includes investigation of optical-network reliability [5], multilayer network design [11, 16], and capacity dimensioning [14].

Multilayer network models appear in industrial standards documents [10, 18] including ITU G-805 [9] and related research [7, 8, 12]. ITU G-805 is an ambitious attempt to 'describe the functional architecture of transport networks in a technology independent way' [9]. ITU G-805 is complex, with simultaneous definition of more than 50 concepts; it distinguishes between 'adaptation', which defines how data in a client layer is embedded in the data of a service layer, and 'termination', which is the addition of monitoring information required for reliability; it has various concepts of 'link connection', 'trail', 'network connection,' 'tandem connection' that are all related to the notion of a link. ITU G-809 [10] extends the definitions to packet-switched networks. A helpful informal introduction to ITU G-805 is available [6]; see [2] for some concerns about the definitions in ITU G-805.

The model presented here is certainly motivated in part by ITU G-805, and is intended to capture the essence of ITU G-805 in a clear, minimal, and rigorous way (though certainly not all the details described by ITU G-805). ITU G-805 has a visual language for representing networks using the concepts that it defines, and has many pictures that are similar to the figures in this paper. In particular, the visualization of adaptation as a triangular shape is borrowed from ITU G-805. Unfortunately, it is hard to tell if ITU G-805 gives rigorous rules for what constitutes a valid visual representation of a network, and what properties can be inferred from the representation.

We do not know any prior work on a model of multilayer networks that allows a mathematically rigorous definition of the normal form of a multilayer network, or what it would mean for one network to be equivalent or generalize another, or to provide algorithms that decide these questions.

To avoid possible confusion, we remark that the network model has some misleading syntactic similarity to the Calculus of Communicating Systems (CCS) model of Milner [13]. In the network model here, port labels identify subchannels and the composition operator $\circ_\sigma$ explicitly indicates the pairing σ of ports; in CCS, port labels are instead used to identify port pairing under composition. CCS is vastly more ambitious than the network model here; indeed it would be easy to model the intended operational semantics of links and adapters in CCS. However, there would be no advantage to doing so, since such modeling would obscure the structural theorems relating equivalence and normal forms given below.

2 Basic Properties

2.1 Formal definition of networks

A *network signature* $(\mathcal{L}, \mathcal{T})$ consists of a set $\mathcal{L}$ of *layers* and a set $\mathcal{T}$ of adapter types. An *adapter type* $T = (G, H, l)$ consists of disjoint nonempty sets G and H of *guest labels* and *host labels*, respectively, and a *layer function* $l : G \cup H \to \mathcal{L}$. The sets $l(G)$ and $l(H)$ are the *guest layers* and *host layers*, respectively; guest and host layers are usually but not necessarily disjoint.

Links and adapters are defined using an abstract set of *ports*. A *link* is a set of two ports, its *end* ports. Ports and links have layers; the layer of a link and its ports must be the same.

An *adapter* $A = (P, \lambda)$ of type $T = (G, H, l)$ is a set P of ports and a labelling function $\lambda : P \to G \cup H$. Port p is a *guest* or *host* port as $\lambda(p) \in G$ or $\lambda(p) \in H$, respectively. Port p must have layer $l(\lambda(p))$. Adapter A can have at most one port with label k; if it exists, the port is denoted A_k.

A *pairing* of a set P of ports is a set of pairs $\{\{p_i, q_i\}\}$ so that for each i, $p_i, q_i \in P$, $p_i \neq q_i$, p_i and q_i have the same layer, and each port of P appears in at most one pair.

A *network* N with signature $(\mathcal{L}, \mathcal{T})$ is a tuple (P, L, A, C, B) where P and L are sets of ports and links, respectively, with layers in $\mathcal{L}$; A is a set of adapters with types in $\mathcal{T}$; C, the set of *connections*, is a pairing of P; and $B \subseteq P$ is the set of *public* ports. Ports p and q are *connected* in N if $\{p, q\} \in C$ (note that by the definition of pairing, a port can be connected to at most one other port). If a port appears in any adapter or link, it can appear at most one such adapter or link in the network; a port that does not so appear is *isolated*. A port is *free* if it is not connected to any other port. All public ports must be free. The *size* of N, $|N|$, is the number of its ports. Figure 1 depicts a small network.

A port is *guest-connected* if it is connected to a guest port, and similarly for *host-connected* and *link-connected*.

We write ports(N), pub(N) and conn(N) for the ports, public ports, and connections of N, respectively. We write '$a \cdot b$' to mean the pair $\{a, b\}$ and '$a \cdot b$ in N' to mean that $\{a, b\} \in \text{conn}(N)$.

Network N is a *subnetwork* of network M if every port, link, adapter, or connection of N is a port, link, adapter, or connection of M, respectively. (Two ports connected in M need not be connected in N even if present in N.) If N is a subnetwork of M, M is a *supernetwork* of N.

Suppose M and N have the same signature. A bijection τ from the ports, links, and adapters of M to the ports, links, and adapters of N, respectively, is a *local isomorphism* if ports p and $\tau(p)$ have the same layer; port p is public iff $\tau(p)$ is public; p is an end port of link l iff $\tau(p)$ is an end port of $\tau(l)$; adapters A and $\tau(A)$ have the same type; A has a port with label l iff $\tau(A)$ has a port with label l; and $\tau(A_l) = \tau(A)_l$. If in addition for all ports $p, p' \in M$, $p \cdot p'$ in M iff $\tau(p) \cdot \tau(p')$ in N, τ is an *isomorphism*. Two networks are *isomorphic* if there is an isomorphism from one to the other.

A network can be represented with a data structure in the obvious way, with a node in memory for each adapter, port and link. We assume constant time mappings from a link to its end ports, from a port to its containing adapter or link, from an adapter and a label to the corresponding port, and from a port to the port to which it is connected, if any.

2.2 Communication

Ports a and b *communicate* in a network N, $a \overset{*}{\longleftrightarrow} b$, if they can be shown to communicate by the following inductive definition:

1. a and b are the end ports of a link; or
2. there are adapters A and B and a guest label g so that $a = A_g$, $b = B_g$, and for each host label h, either A_h is connected to B_h, or A_h is connected to a port a', B_h is connected to b', and a', b' communicate; or
3. there are ports $a = a_1, b_1, a_2, b_2, \ldots, a_k, b_k = b$ so that b_i is connected to a_{i+1}, $i = 1, \ldots, k-1$ and a_i and b_i communicate by rules (1) and (2), $i = 1, \ldots, k$.

If either of the first two cases hold, a and b *communicate in one step*, $a \leftrightarrow b$; if the third case holds, a and b *communicate in k steps*. See Figure 1. The following propositions is useful, though its proof is immediate.

Proposition 2.1 *If M is a subnetwork of N and $p \overset{*}{\longleftrightarrow} p'$ in M, then $p \overset{*}{\longleftrightarrow} p'$ in N.*

The *level* of a communicating pair $\{a, b\}$ is defined by induction: in case (1), the level of a pair of link end ports is 1; in case (2), the level is 1 plus the maximum level of any pair $\{a', b'\}$, or just 1 if all corresponding host pairs are connected; in case (3), the level is 1 plus the maximum level of any pair $\{a_i, b_i\}$.

2.3 Composition, tests, equivalence, and generalization

Suppose N and M are networks with the same signature and disjoint sets of ports, and ϕ is a pairing of the public ports of N and M. The *composition* $N \circ_\phi M$ is the network obtained by taking the union of the ports, links, adapters, connections, and public ports of M and N, respectively, then adding ϕ to the connections and removing the ports connected in ϕ from the public ports. Composition is symmetric, i.e. $N \circ_\phi M = M \circ_\phi N$ and associative as long as it is defined (if σ pairs a port in N to a port in P, $(N \circ_\phi M) \circ_\sigma P$ may be defined but $N \circ_\phi (M \circ_\sigma P)$ is not).

A *test* for a set C of ports is a tuple (T, p, q, ϕ) where T is a network, $p, q \in C \cup \text{pub}(T)$, and ϕ is a pairing of $C \cup \text{pub}(T)$. T is the *test network* and p, q are the *test ports*. A network N with $C \subseteq \text{pub}(N)$ *satisfies* the test if p and q communicate in $T \circ_\phi N$.

Suppose M and N have the same signature and $\text{pub}(M) \subseteq \text{pub}(N)$. Network M *is generalized by* N (or N *generalizes* M), $M \preceq N$, if every test that satisfies M also satisfies N. Networks M and N are *(externally) equivalent*, $N \equiv M$, if $M \preceq N$ and $N \preceq M$ (equivalently if they have the same public ports and every test that satisfies one also satisfies the other). It is immediate that generalization is transitive and that external equivalence is an equivalence relation. In Figure 3, E acts like a link; using Theorems 3.3 and 3.12 below, it follows that $A \preceq B \preceq C \preceq D \equiv E$, while A, B, C, and D are not equivalent.

2.4 The communication graph

A set $S = \{p_1, q_1, \ldots, p_k, q_k\}$ is a *communicating chain (of length k)* if $k \geq 1$ and $p_1 \leftrightarrow q_1 \cdot p_2 \leftrightarrow q_2 \ \ldots \ p_k \leftrightarrow q_k$. S is *maximal* if it is not a subset of any other communicating chain. If S is maximal, it is a *cycle* if $q_k \cdot p_1$, otherwise p_1 and q_k are the *extremal* ports of the chain. Necessarily, for each i, p_i and q_i are end ports of a link, or p_i and q_i are both identically-labelled guest ports of adapters of the same type. S is *proper* if it is maximal and neither p_1 nor q_k is guest-connected (so either host-connected or free). An easy induction shows the following.

Proposition 2.2 *If $p \stackrel{*}{\longleftrightarrow} q$, then there is a unique maximal communicating chain containing p and q.*

The *communication graph* of network N is a directed graph. Its nodes are maximal communicating chains. There is an arc from node P to node Q if there are $A_g, A'_g \in P$ and $b, b' \in Q$ so that $A_g \leftrightarrow A'_g$, $A_h \cdot b$, and $A'_h \cdot b'$ for some adapters A and A', guest label g, and host label h. (Necessarily b and b' are the end ports of a maximal communicating chain.) The *level* of a node P, $l(P)$, is the level of the pair of extremal ports of P.

Proposition 2.3 *The communication graph is acyclic.*

Proof: If there is an arc from P to Q in the communication graph, then $l(P) > l(Q)$. □

Proposition 2.4 *The communication graph of network N can be constructed in time $O(|N|)$.*

Proof: (Sketch) Maintain a set L of communicating chains. L is initialized with all pairs $\{p_i, q_i\}$ so that $p_i \leftrightarrow q_i$ at level 1 (in one step); such pairs are either endpoints of links or corresponding guest ports of a pair of adapters with all corresponding pairs of host ports connected, and can be found by a linear-time scan of the network. There are two types of events that modify L: merging, i.e. replacing $\{p_1, \ldots, q_i\}$, $\{p'_1, \ldots, q'_j\}$ so that $q_i \cdot p'_1$ with $\{p_1, \ldots p_i, p'_1 \ldots q'_j\}$, and adapter traversal, i.e. if A, A' are adapters of the same type, and for each host label h the ports connected to A_h and A'_h form the extremal ports of some chain in L, then for each guest label g, the pairs $\{A_g, A'_g\}$ are added to L. An easy induction on level shows that, after all events are processed, L contains exactly the maximal chains of N. The events can be implemented to take constant time for each pair (p_i, q_i) so that $p_i \leftrightarrow q_i$ (in one step), and hence linear time overall. □

The proof of the following is easy.

Proposition 2.5 *A chain in the communication graph with positive indegree is proper.*

3 Simple multiplexing

An adapter type is *simple* if it has one host label. A network is *simple* if it only contains adapters with simple adapter types. Throughout this section, Section 3, all networks are simple.

A network signature may have an adapter-type cycle, that is, a set of adapter types $\mathcal{T}_1, \ldots, \mathcal{T}_k, \mathcal{T}_{k+1} = \mathcal{T}_1$ so that a host layer of $\mathcal{T}_i$ is a guest layer of $\mathcal{T}_{i+1}$. We assume up through section 3.3 that there are no such cycles; this assumption somewhat simplifies the presentation. An immediate consequence of this assumption is that the network does not have an adapter cycle, that is, a set of adapters $A_1, \ldots, A_k, A_{k+1} = A_1$ so that a host port of A_i is connected to a guest port of A_{i+1}. In Section 3.4 we show that we can remove this assumption, essentially by showing that adaptation cycles cannot contribute to communication.

3.1 Normal forms

Network M *reduces in one step* to N if N is obtained from M in one of the following ways:

> *Link collapse*: M contains a link l with ports p and q, not both of which are public. Link l is deleted. If $p \cdot p'$ and $q \cdot q'$, then the connections to p and q are deleted and the connection $p' \cdot q'$ is added. If one port, say q, is public, and p is connected to a port p', then in addition p' is deleted and replaced by q.
>
> *Adapter elimination*: M contains a pair of adapters A and B with connected host ports. Adapters A and B are deleted. If A and B had the same type, then for each guest label g so that A and B both have a port with label g, A_g and B_g become the end ports of a new link.
>
> *Free port elimination*: A port p that is free and not public is deleted. If p is the end port of a link, then the link is deleted. If p is a host port or the last guest port of an adapter, the adapter is deleted.
>
> *Isolated port elimination*: If port p is isolated and not public, it is deleted.

For example, in Figure 3, E reduces in one step to D. After a link or adapter is deleted, various ports can become isolated or free but not public. Such ports are deleted by the last two rules. For example, consider a link with its two end ports connected to each other. The link is deleted by the 'link collapse' rule, and then the two ports are deleted by the 'isolated port elimination rule'.

A network is *reducible* if some reduction applies, otherwise it is *irreducible*.

Lemma 3.1 *Let p and g be the number of ports and guest ports of M, respectively. M reduces in at most $p + g$ steps to a unique irreducible network, denoted* $\mathrm{nf}(M)$*, satisfying* $M \equiv \mathrm{nf}(M)$.

Proof: It is easy to check that one step of reduction reduces the sum $p + g$. A case analysis shows that if M reduces in one step to both N and P, then either $N = P$, N reduces in one step to P (or P to N), or there is a network Q so that both P and N reduce in one step to Q; uniqueness of irreducible network follows.

Suppose M reduces in one step to N. It is easy to check that M and N have the same public ports; for public ports p and q, $p \overset{*}{\longleftrightarrow} q$ in M iff $p \overset{*}{\longleftrightarrow} q$ in N; and for any T, σ, $M \circ_\sigma T$ reduces in one step to $N \circ_\sigma T$. $M \equiv \mathrm{nf}(M)$ follows. □

Lemma 3.2 *Let N be a simple network.*

1. *If $p \cdot q$ in* $\mathrm{nf}(N)$*, then p and q are both guest ports, or one is a host and the other a guest port.*
2. *All free ports of* $\mathrm{nf}(N)$ *are public.*
3. $p \overset{*}{\longleftrightarrow} q$ *in* $\mathrm{nf}(N)$ *iff p and q are public end ports of a link.*
4. *The normal form of N can be constructed in time $O(|N|)$.*

Proof: Parts 1 through 3 are immediate. For part 4, each possible reduction is determined by a constant-sized portion of the network. It suffices to maintain a queue of all possible reductions in the current network. The queue can be initialized by a linear scan of the network. Applying a reduction and checking locally for any newly possible reduction each take constant time, hence linear overall by Lemma 3.1. □

Clearly any guest port in a normal-form network is either public, guest-connected, or host-connected. In Figure 3 A, B, C, and D are in normal form; E reduces in one step to D.

An *adaptation tree* is network consisting only of adapters and with only guest-host connections (there are no adaptation cycles by the assumption that there are no adapter-type cycles). It follows immediately from Lemma 3.2 that if all guest-guest connections are deleted, a normal form network decomposes into a set of links and a set of adaptation trees. An adaptation tree has a unique free host port, the *root* port. If host port h is not the root port in an adaptation tree, then its *parent* is the host port of the adapter with a guest port connected to h. If h is the host port of an adapter A in an adaptation tree, the *subtree rooted at* h is the adapter A with its ports, any adapters whose host port connects to a guest port of A, any adapters whose host ports connects to those adapters, and so forth, together with any internal guest-host connections.

3.2 External equivalence

Theorem 3.3 *Suppose M and N are simple networks with the same public ports. Then $M \equiv N$ if and only if there is an isomorphism mapping $\mathrm{nf}(M)$ to $\mathrm{nf}(N)$ that is the identity on public ports.*

By Lemma 3.2, we can assume that M and N are in normal form. For simplicity we assume that neither has any isolated ports–such ports cannot participate in any communication. One direction of the theorem is obvious: if the isomorphism exists, it is immediate that $M \equiv N$.

Define the 'copy' network $C = C(M)$ to be an isomorphic copy of the adaptation trees comprising M after guest-guest connections are deleted. More precisely, for each adapter A of M, C contains an adapter $c(A)$ of the same type, with a port $c(A)_l$ for each port A_l of M; $c(A_l)$ is defined to be $c(A)_l$. The connection $c(g) \cdot c(h)$ exists in C iff $g \cdot h$ in M and one of g, h is a guest port and the other is a host port. All ports free in C are public. Note that if g and g' are connected guest ports, then $c(g)$ and $c(g')$ are free and public in C. See Figure 6.

Define σ to be the connections $\{h \cdot c(h)\}$, where h varies over free host ports of C. Explicitly, in the composition $M \circ_\sigma C$, the only connections between M and C are connections between host ports.

Proposition 3.4 *Let g be a guest port of M and consider communication in $M \circ_\sigma C$. Then*

1. *$c(g) \leftrightarrow g$ (in one step), and the communication chain is maximal if g is free.*
2. *Let g, g' be guest ports of M. Then the following are equivalent.*
   - *(a) $c(g) \stackrel{*}{\longleftrightarrow} c(g')$*
   - *(b) $g \cdot g'$*
   - *(c) $c(g) \leftrightarrow g \cdot g' \leftrightarrow c(g')$ and the communication chain is maximal.*

Proof: For part 1, an easy induction using the definition of C shows $c(g) \leftrightarrow g$; $c(g)$ is host-connected or free, so if g is free, the communication is maximal. For part 2, we show (a) implies (c); (c) implies (b) and (b) implies (a) are immediate. Suppose $c(g) \stackrel{*}{\longleftrightarrow} c(g')$. We have $g \neq c(g')$ since the former is in M and the latter in C. By part 1; $c(g) \leftrightarrow g$, since the communication continues to $c(g')$, we must have g connected to some guest port g'', necessarily in M since there are no guest-guest connections in σ or C. We also have $g'' \leftrightarrow c(g'')$ by part 1. Any communication chain must stop at a guest port of C, since such a port is either public or host-connected. This implies $c(g') = c(g'')$ and $g' = g''$. The communication chain is maximal since $c(g)$ and $c(g')$ are each either free or host-connected. □

Proposition 3.5 *There are no guest ports g, g' of N that communicate in $N \circ_\sigma C$.*

Proof: Suppose to the contrary; choose guest ports g, g' in N of minimal level so that $g \stackrel{*}{\longleftrightarrow} g'$ in $N \circ_\sigma C$. The corresponding host ports h, h' in N are not connected to each other since N is in normal form. Let $r \cdot h$ and $r' \cdot h'$. We must have $r \stackrel{*}{\longleftrightarrow} r'$ since $g \stackrel{*}{\longleftrightarrow} g'$; r and r' must be guest ports since no communication can start with a host port; and r, r' are in N since there are only host-host connections in σ. This contradicts the minimality of g, g'. □

For the following we develop structural information about the compositions $M \circ_\sigma C$ and $N \circ_\sigma C$. This information will be used both in this section (the case $M \equiv N$) and in the next section (the case $M \preceq N$). Hence in what follows we are explicit about which hypothesis $M \equiv N$ or $M \preceq N$ is used.

If $M \preceq N$ and M contains a link $\{p, q\}$, then easily $\{p, q\}$ must be a link of N: since p communicates with q in M, p and q must communicate in N, which is only possible if $\{p, q\}$ is a link in N as N is in normal form. For simplicity we now assume that M contains no links (and with the stronger assumption $M \equiv N$ we could also assume that N contains no links).

Proposition 3.6 *Suppose $M \preceq N$. Every guest port $c(g) \in C$ is the endpoint of a proper communicating chain.*

Proof: First suppose $c(g)$ is a free guest port of C. Then g is either public or guest-connected in M. By Proposition 3.4, in the former case, $c(g) \stackrel{*}{\longleftrightarrow} g$ in $M \circ_\sigma C$; in the latter case $c(g) \stackrel{*}{\longleftrightarrow} c(g')$, where $g \cdot g'$ in M and $c(g')$ is public in C. Since $M \preceq N$ and the communications are between public ports, the same communications hold in $N \circ_\sigma C$, and easily both are proper. Second, suppose $c(g)$ is connected, necessarily to some host port $c(h)$. By the above, every free guest port in the adapter subtree rooted at $c(h)$ communicates with some port. For this to be possible $c(g)$ must communicate with some port, and by Proposition 2.5, the communicating chain is proper. □

Suppose $M \preceq N$. We define a mapping e from M to $N \circ_\sigma C$ as follows. For g a guest port of M, we let $e(g)$ be the port of $N \circ_\sigma C$ with which $c(g)$ communicates in one step, $c(g) \leftrightarrow e(g)$. Such a port exists by Proposition 3.6. We extend e to adapters: if g is a guest port of adapter A, then $e(A)$ is the adapter containing $e(g)$ as a guest port. This is well-defined, since if g, g' are distinct guest ports of the same adapter A and $c(g) \leftrightarrow e(g)$ and $c(g') \leftrightarrow e(g')$, then it is easy to verify that $e(g)$ and $e(g')$ must be guest ports of the same adapter, since the communication is in one step. We extend e to host ports in the obvious way. Mapping e is one-one since c and one-step communication are one-one. It is easy to check that e is a local isomorphism of M into the range of e, e.g. adapters A and $e(A)$ have the same type, etc. If $M \equiv N$, we will eventually show that e is the desired isomorphism of M to N.

Proposition 3.7 *Suppose $M \preceq N$ and g is a guest port of M. Then the maximal communication chain starting at $c(g)$ in $N \circ_\sigma C$ is one of the following:*

1. *$c(g) \leftrightarrow c(g')$, if $e(g)$ is in C and hence $e(g) = c(g')$, for some guest port g' of M, or*
2. *$c(g) \leftrightarrow e(g) \cdot e(g') \leftrightarrow c(g')$, if $e(g)$ is a guest port of N and is guest-connected, where also $e(g')$ is a guest port of N, or*
3. *$c(g) \leftrightarrow e(g)$, if $e(g)$ is a guest port of N and is public or host-connected.*

These three cases are mutually exclusive.

In Figure 7, case (2) holds for ports g and g', that is $c(g) \leftrightarrow e(g) \cdot e(g') \leftrightarrow c(g')$ in $N \circ_\phi C$. Case (1) holds for port g'', that is $c(g'') \leftrightarrow c(q)$, where $c(q) = e(g'')$.

Proof: By definition $c(g) \leftrightarrow e(g)$. Clearly $e(g)$ is a guest port. One of the cases $e(g)$ is in C, $e(g)$ is in N and is guest-connected, or $e(g)$ in N and is public or host-connected must hold, and they are mutually exclusive. If the first case holds, $e(g)$ in C, then the communication chain must stop at $e(g)$ since any guest port in C

is either public or host-connected, and the communication chain cannot extend. Thus $e(g) = c(g')$ for some guest port g' of M. The argument for the third case is similar.

Suppose the second case holds, $e(g)$ is in N and is guest-connected, $e(g) \cdot r$ for guest port r. Since neither C nor σ has guest-guest connections, r is in N. The maximal communicating chain from $c(g)$ in $N \circ_\sigma C$ is proper by Proposition 3.6, hence must extend beyond $e(g)$, so $c(g) \leftrightarrow e(g) \cdot r \stackrel{*}{\longleftrightarrow} t$ in $N \circ_\sigma C$, for some port t. By Proposition 3.5, t cannot be in N, so t is in C, and $t = c(g')$, some guest port g' in C. It must be that $r = e(g')$, for if $r \neq e(g')$ then $c(g) \leftrightarrow e(g) \cdot r \stackrel{*}{\longleftrightarrow} r' \cdot e(g') \leftrightarrow c(g')$, for some r' necessarily in N because guest-guest connections are in N, again contradicting Proposition 3.5. Hence $c(g) \leftrightarrow e(g) \cdot e(g') \leftrightarrow c(g')$. □

Proposition 3.8 *Suppose $M \equiv N$ and g is a guest port of M. Then $e(g)$ is in N (so case 1 of Proposition 3.7 does not hold).*

Proof: We show $M \preceq N$ and $e(g)$ is not in N implies $N \not\preceq M$. Suppose $M \preceq n$ and $e(g)$ is not in N. Then $e(g)$ is in C, so $e(g) = c(g')$ for some guest port g' in M. We have $c(g) \leftrightarrow c(g')$ in one step in $N \circ_\sigma C$. Let $c(h)$ and $c(h')$ be the host ports of the adapters A and A', containing $c(g)$ and $c(g')$, respectively. It cannot be that $c(h) \cdot c(h')$, since such a connection is neither in σ nor C. It cannot be that $c(h)$ is connected to the host port of an adapter distinct from A', since $c(g) \leftrightarrow c(g')$ in one step. Hence $c(h) \cdot c(r)$ and $c(h') \cdot c(r')$, some guest ports r, r' in M, and $c(r) \stackrel{*}{\longleftrightarrow} c(r')$ in $N \circ_\sigma C$. Let H be the union of the subnetwork of C rooted at $c(h)$ with the subnetwork rooted at $c(h)'$. Let $C \setminus H$ be the subnetwork of C obtained by deleting the adapters in H and connections to them, with $c(r)$ and $c(r')$ as public ports. The adapters in H cannot contribute to the communication between $c(r)$ and $c(r')$ in $N \circ_\sigma C$, hence $c(r) \stackrel{*}{\longleftrightarrow} c(r')$ in $N \circ_\sigma (C \setminus H)$. Now consider communication in $M \circ_\sigma C$. Since r is not guest-connected, by Proposition 3.4, $c(r)$ does not communicate with $c(r')$ in $M \circ_\sigma C$, hence not in $M \circ_\sigma (C \setminus H)$. Thus $N \not\preceq M$. □

Proposition 3.9 *Suppose $M \preceq N$ and for guest port g of M, $e(g)$ is in N (cases 2 and 3 of Proposition 3.7). Let h be the host port of the adapter containing g in M. Then in $N \circ_\sigma C$ either*

1. *$c(h) \cdot e(h)$, or*
2. *$c(h) \cdot c(g')$, for some guest port g' of M, $e(g')$ is a guest port of N, and $e(h) \cdot e(g')$ in N.*

Proof: Suppose $c(h)$ and $e(h)$ are not connected to each other. Then $c(h) \cdot r$ and $e(h) \cdot s$ for some ports r, s with $r \stackrel{*}{\longleftrightarrow} s$. Ports r and s cannot be host ports, since $c(g) \leftrightarrow e(g)$ in $N \circ_\sigma C$ in one step. Neither r nor s can be a link port of C since C has no links. Port s cannot be a link port of N since $s \cdot e(h)$, $e(h)$ in N, and N in normal form. Hence port r cannot be a link port of N, since $r \stackrel{*}{\longleftrightarrow} s$ and N in normal form implies $\{r, s\}$ a link of N and s a link port. Hence r and s must be guest ports of adapters. Port s must be in N since there are no connections in σ from a port of N to a guest port of C. Port r must be in C, since r in N would contradict Proposition 3.5. Hence $r = c(g')$ for some guest port g' of M. Since the maximal chain from $c(g')$ ends at s, a nonpublic port of N, case 3 of Proposition 3.7 must hold, so $s = e(g')$ and $e(h) \cdot e(g')$. □

A public host port p of M is *link-replaced* if p is an end port of a link of N.

Proposition 3.10 *Suppose $M \preceq N$. If p is a public port of M, then $e(p) = p$ or p is link replaced. If p is link replaced, then for some host port p' of M, $c(p) \cdot p \leftrightarrow p' \cdot c(p')$ in $N \circ_\sigma C$ and $\{p, p'\}$ is a link of N. If $M \equiv N$, then always $e(p) = p$.*

Proof: By assumption M has no links, so p is not a link port. So suppose p is a public guest port of M, then $c(p)$ is public in N. We have $c(p) \leftrightarrow p$ in $M \circ_\sigma C$ by Proposition 3.4; since $M \preceq N$, $c(p) \stackrel{*}{\longleftrightarrow} p$ in $N \circ_\sigma C$. By Proposition 3.7(1), the communication must be in one step as p is in N and not C. Hence $c(p) \leftrightarrow p$ and $p = e(p)$.

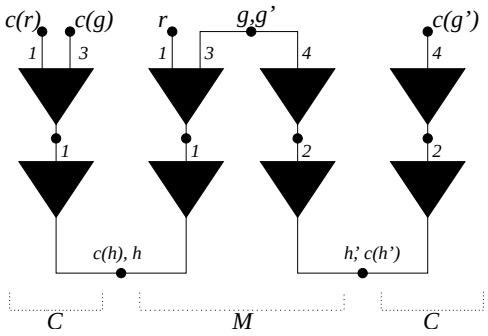

Figure 6: An example of $M \circ_\sigma C$. $1, 2, 3, 4$ are guest labels; no host labels shown. Ports g and g' are connected together in M. M has two public host ports h, h' with $h \cdot c(h)$, $h' \cdot c(h') \in \sigma$; r is a public guest port of M.

Now suppose p is a public host port of M, then $c(p) \cdot p$ in both $M \circ_\sigma C$ and $N \circ_\sigma C$. For some guest port g of the adapter containing p, we have $c(g) \leftrightarrow e(g)$ in $N \circ_\sigma C$. Then $e(p)$ is the host port of the adapter containing $e(g)$, and in $N \circ_\sigma C$ we either have $c(p) \cdot e(p)$ or $c(p) \cdot p \stackrel{*}{\longleftrightarrow} q \cdot e(p)$ for some port q. In the former case, $e(p) = p$, since $c(p) \cdot e(p)$ in $N \circ_\sigma C$. In the latter case, p and q must be in N and $e(p)$ in C. Ports p and q cannot be guest ports by Proposition 3.5 nor host ports. So p and q must be end ports of a link, and $\{p, q\}$ is a link in N since N is in normal form. We have $e(p) = c(p')$ for some host port p' of C and $c(p') \cdot p'$ in both $M \circ_\sigma C$ and $N \circ_\sigma C$, so $q = p'$, and $c(p) \cdot p \leftrightarrow p' \cdot c(p')$. Note that in this last case $M \not\equiv N$, since $p \stackrel{*}{\longleftrightarrow} q$ in $N \circ_\sigma C$ but not in $M \circ_\sigma C$. □

Proof: (of Theorem 3.3) Suppose $M \equiv N$. As mentioned, we can assume that M and N have no links, since easily a link of one is a link of the other.

By Proposition 3.8, e maps every guest port of M to a guest port of N; since e is one-one, M has at least as many guest ports as N. However, the whole construction is symmetric in M and N, so M and N have the same number of guest ports. The same argument holds for adapters and host ports, so e is a local isomorphism from M onto N.

To finish the proof, we need to show that e is an isomorphism, i.e. e preserves connections. Suppose guest port g is connected to host port h in M. Then some guest port $c(g')$ of the adapter containing $c(h)$ communicates with $e(g')$, and by Proposition 3.8, $e(g') \in N$. Hence by Proposition 3.9(2), applying case 2 and interchanging the roles of g and g', $e(g) \cdot e(h)$. Now suppose guest ports g, g' are connected in M. By Proposition 3.4, $c(g) \stackrel{*}{\longleftrightarrow} c(g')$ in $M \circ_\sigma C$, so by equivalence $c(g) \stackrel{*}{\longleftrightarrow} c(g')$ in $N \circ_\sigma C$. By Proposition 3.7(2), $e(g) \cdot e(g')$.

For the converse, suppose $e(g) \cdot e(h)$ in N, then neither $e(g)$ nor $e(h)$ are public in N. M and N have the same public ports, and all other ports are connected. In particular $g \cdot h'$ some h', so $e(g) \cdot e(h')$ as just argued, but $e(g)$ can be connected to at most one port, so $e(h') = e(h)$ and $e(g) \cdot e(h)$. □

3.3 Generalization

A *doubled* network is a network of the form $T \circ_\rho T'$, where T, T' are isomorphic adaptation trees, $T' = \phi(T)$ for isomorphism ϕ, and ρ is the pairing $\{g, \phi(g)\}$ for all guest ports g of T not connected to a host port. More informally, a doubled network consists of two isomorphic adaptation trees so that a guest port in one tree is either connected to a host port in the same tree, or connected to the corresponding guest port in the other tree. The only free ports in a doubled network are the two root ports. In Figure 3, A, B, and C are doubled networks. Notice that if an adapter A in one tree has a guest label l, the guest port A_l is not required to be in the tree, but if A_l is in the tree, then because the trees are isomorphic, the other tree has a corresponding guest port, and the two guest ports must be connected together.

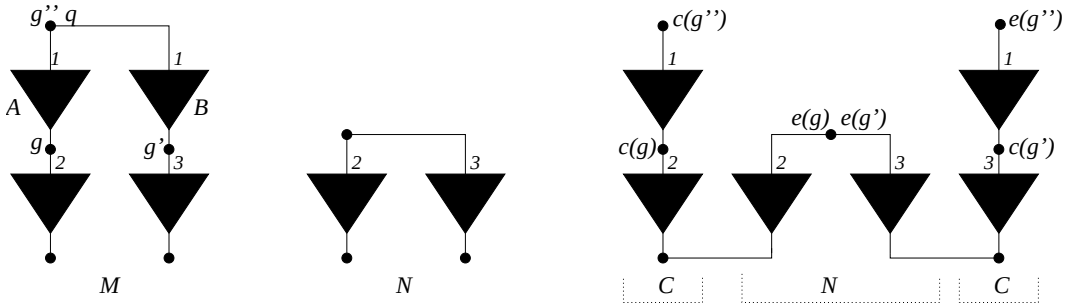

Figure 7: $M \prec N$. Ports g'' and q are connected and are guest ports of adapters A and B, respectively, with the same label 1. In $N \circ_\sigma C$, guest port g of M is mapped, i.e. $e(g) \in N$, while g'' is doubled, i.e. $e(g'') \in C$. Note $e(g'') = c(q)$. The host ports h and h' connected to g and g' in M (not labelled) form a critical pair.

Let M be a simple network. Suppose host ports $\{p, q\}$ form the root pair of a doubled subnetwork of M. Network N is obtained by *strengthening* the pair $\{p, q\}$ if N results by replacing the doubled subnetwork with a link l with end ports p and q, preserving any connections to p and q, and then applying the link elimination rule of Section 3.1 to l, if possible. For example, if $p \cdot p'$ and $q \cdot q'$ in M, then $p' \cdot q'$ results in N. A network is a *strengthening* of M if it is obtained by strengthening zero or more root pairs of doubled networks. Easily the strengthening is in normal form if M is in normal form. In Figure 3, B is a strengthening of A, and D is a strengthening of A, B, and C.

This proposition is easy.

Proposition 3.11 *Suppose $p \overset{*}{\longleftrightarrow} q$ in M and N is a strengthening of M. Then $p \overset{*}{\longleftrightarrow} q$ in N.*

Most of the rest of this section is devoted to the proof of the following theorem.

Theorem 3.12 *Suppose M and N are simple networks with the same public ports. Then $M \preceq N$ if and only if a strengthening of $\mathrm{nf}(M)$ is isomorphic to a subnetwork of $\mathrm{nf}(N)$.*

One direction of the theorem is obvious: if a strengthening of $\mathrm{nf}(M)$ is isomorphic to a subnetwork of $\mathrm{nf}(N)$, then $M \preceq N$ follows from Propositions 2.1 and 3.11. For the other direction, we let C, σ and e be as in the previous section. Also as in the previous section, we can assume M has no links.

A port or adapter of M is *doubled* if its image under e is in C. Clearly an adapter is doubled exactly if its guest and host ports are doubled. Also clearly, if adapter A is doubled, $e(A) = c(A')$, some adapter A', then A' is doubled and $e(A') = c(A)$. (Looking ahead, the terminology comes from considering the composition $\phi = c^{-1} \circ e$ that maps ports of M to ports of M, which by Proposition 3.14 yields a doubled subnetwork of M.) A port or adapter of M is *mapped* (to N) if its image under e is in N. In Figure 7, port g'' is doubled with $e(g'') = c(q)$, while $c(g)$ is mapped, $e(g) \in N$. In the last section under the assumption $M \equiv N$, all ports and adapters were mapped (i.e. case (1) of Proposition 3.7 did not hold, by Proposition 3.8). With the weaker assumption $M \preceq N$, some ports and adapters can be doubled.

The following proposition is similar to Proposition 3.9, but handles the case of doubled adapters and ports. The proof, similar to those of Proposition 3.8 and 3.9, is omitted.

Proposition 3.13 *Suppose $M \preceq N$ and adapter A of M with host port h is doubled, i.e. $e(A)$ is in C and $e(h) = c(h')$ for some host port h' in M. Then either*

1. *$c(h) \cdot c(g)$ for some guest port g of M, $e(g)$ is in C hence g is doubled, and $e(h) \cdot e(g)$, or*
2. *$c(h) \cdot c(g)$ for some guest port g of M, $e(g)$ is a guest port of N hence g is mapped, $e(h) \cdot e(g)$, and $e(g)$, h', and $c(h') = e(h)$ are public ports of M, N, and C, respectively, or the symmetric case where a guest port of N is connected to $c(h)$ and $c(h')$ is connected to a guest port in C, or*

3. *$c(h)\cdot c(g)$ and $c(h')\cdot c(g')$ for some guest ports g, g' of M, and $c(g) \leftrightarrow e(g)\cdot e(g') \leftrightarrow c(g')$ where $e(g), e(g')$ are guest ports of N, hence both g and g' are mapped.*

4. *h, h' are public in M, $\{h, h'\}$ is a link of N, and $c(h)\cdot h$ and $c(h')\cdot h'$ in $M \circ_\sigma C$.*

Suppose the host port of adapter A is connected to a guest port of adapter B. Propositions 3.9 and 3.13 imply that if A is doubled, B can be mapped or doubled, but if A is mapped, B must be mapped. Hence in a sequence of adapters $A_1, A_2, \ldots A_n$, where A_{i+1} is the parent of A_i, some possibly empty initial sequence of adapters is doubled, and the remaining sequence, possibly empty, is mapped. A doubled host port pair $\{h, h'\}$ of M is *critical* if either h or h' is a root port, hence public, or at least one of the corresponding parent host ports $\{p, p'\}$ is mapped. In Figure 7, the host ports of adapters A and B form a critical pair.

Proposition 3.14 *If $\{p, p'\}$ is a doubled host port pair, then $\{p, p'\}$ is the root pair of a doubled subnetwork of M.*

Proof: Let M_g be the network M with links and guest-guest connections removed, then M_g is a collection of adaptation trees. Port p appears in one such adaptation tree $\hat{T}$; let T be the subtree of $\hat{T}$ rooted at p; clearly T is a subnetwork of M. Clearly any port or adapter in T is doubled. Similarly let T' be the subtree rooted at p'.

The map $\phi = c^{-1} \circ e$ is easily seen to be a local isomorphism of T into its range $\phi(T)$. We wish to show ϕ is the desired isomorphism of T into T'. Suppose host port h of T is not the root port p, then $h \cdot g$ for some guest port g in T. Thus $c(h) \cdot c(g)$ and $e(g)$ is in C, so only case 1 of Proposition 3.13 applies, and $e(h) \cdot e(g)$. Hence any guest-host connection in T is preserved in $e(T)$, and the range $e(T)$ is an adaptation tree isomorphic to T. This range contains $e(h')$, so the range $e(T)$ is a subnetwork of $c(T')$, and the range of $\phi(T)$ is a subnetwork of T'. Hence T' has at least as many adapters and ports as T. However, the situation is symmetric, so that T and T' have the same number of ports and adapters, and ϕ is an isomorphism from T to T'. (Easily ϕ maps T' to T as well.)

Suppose g is a guest port of one tree, say T, and g is not connected to a host port. To show that T, T' form a doubled network, we must show $g \cdot \phi(g)$ in M. Port g cannot be free, for then it would be public and also in N; furthermore we would have $c(g) \leftrightarrow e(g) = g$ in $M \circ_\sigma C$ by Proposition 3.4, so also $c(g) \stackrel{*}{\longleftrightarrow} e(g)$ in $N \circ_\sigma C$, with $e(g)$ in N, contradicting g doubled. So g must be connected to a guest port g' in M. By Proposition 3.4, $c(g) \leftrightarrow g \cdot g' \leftrightarrow c(g')$ in $M \circ_\sigma C$, so also $c(g) \stackrel{*}{\longleftrightarrow} c(g')$ in $N \circ_\sigma C$. Since g is doubled, the communication is in one step, so $c(g') = e(g)$ or $g' = c^{-1}(e(g)) = \phi(g)$. □

Let M^* be the network obtained from M by strengthening every critical host port pair; clearly M^* includes all mapped ports and adapters. Let e^* be the restriction of e on the adapters of M^* and their ports, and the identity on links of M^*. Let N^* be the subnetwork of N that is in the range of e^*. The following completes the proof of Theorem 3.12.

Proposition 3.15 *Mapping e^* is an isomorphism of M^* to N^*.*

Proof: By arguments similar to the proof of Theorem 3.3, e^* maps an adapter M^* and its ports to an adapter of N^* and its ports; any guest-guest or guest-host connection in M^* that is also in M is preserved in N^*. It remains to consider connections and links that result from strengthening M to M^*. So suppose $\{h, h'\}$ are a critical root pair; clearly Proposition 3.13 applies. If both h, h' are connected to mapped guest ports, $h \cdot p, h' \cdot p'$, then the connection $p \cdot p'$ results by strengthening $\{h, h'\}$. Only case (3) of Proposition 3.13 holds, so $e(p) \cdot e(p')$. Similarly if both h, h' are public, then strengthening $\{h, h'\}$ results in a link in M^* with end ports h, h' (this link was not present in M). By case (4) of Proposition 3.13, $\{h, h'\}$ is a link of N. The case that, say, h' is public and h is connected to a guest port p of M (so $c(h) \cdot c(p)$), is similar, using case (2) of Proposition 3.13. □

Theorem 3.16 *Let M and N be simple networks with the same public ports. There are algorithms to decide if $M \equiv N$ or $M \preceq N$ that run in time $O(|M| + |N|)$.*

Proof: To decide $M \equiv N$, compute $\mathrm{nf}(M)$ and $\mathrm{nf}(N)$ and then decide if there is an isomorphism from $\mathrm{nf}(N)$ to $\mathrm{nf}(M)$. Testing isomorphism in linear time is easy: a partial isomorphism ϕ (the identity) is given for public ports; ϕ can be extended through links, connections, and from one port of an adapter to all ports, since in each case the extension is uniquely determined. Either ϕ extends to all of $\mathrm{nf}(N)$, or a proof of nonisomorphism results. The procedure to decide $M \preceq N$ is similar. □

3.4 Adaptation cycles

In this section we remove the assumption that the network signature had no adapter-type cycles. This requires a slight extension of the notion of normal form.

The *adaptation graph* of a network N is a directed graph whose nodes are guest ports and with an arc from h_1 to h_2 if h_1 is the guest port of an adapter with a host port connected to h_2. The *acyclic core* $\mathrm{acyclic}(N)$ is the network obtained from N by deleting, for each guest port g that appears in a cycle in the adaptation graph, the adapter containing g, all of its nonpublic ports, and any connection to a deleted port. Any public port of a deleted adapter becomes isolated.

Proposition 3.17 *Suppose there is an arc from p to q in the adaptation graph, and a node P containing p in the communication graph. Then the communication graph contains a node Q containing q and an arc from P to Q.*

Proof: Since p is part of a communicating chain, $p \leftrightarrow p'$ for some port p'. Since there is an arc from p to q in the adaptation graph, $p = A_g$ and $A_h \cdot q$ for some adapter A. Since $p \leftrightarrow p'$ and $p = A_g$, $p' = A'_g$ for some adapter A', $A'_h \cdot q'$ for some port q', and $q \stackrel{*}{\longleftrightarrow} q'$. Hence the required node and arc exist. □

Proposition 3.18 *If $p \stackrel{*}{\longleftrightarrow} q$ in N, then $p \stackrel{*}{\longleftrightarrow} q$ in* $\mathrm{acyclic}(N)$.

Proof: Let P be all ports contained in nodes of the communication graph, and let N^* be the subnetwork of N induced by P, i.e., the network containing all ports of P, all connections between ports of P, any links containing ports in P, and any adapters containing ports in P together with all their host ports. Clearly $p \stackrel{*}{\longleftrightarrow} q$ in N^*. Any port on a cycle in the adaptation graph cannot be a port of N^*, else N^* would contain a cycle by Proposition 3.17. Hence N^* is a subnetwork of $\mathrm{acyclic}(N)$, so $p \stackrel{*}{\longleftrightarrow} q$ in $\mathrm{acyclic}(N)$ by Proposition 2.1. □

Corollary 3.19 $N \equiv \mathrm{acyclic}(N)$.

Proof: Suppose (T, p, q, ϕ) is a test for $\mathrm{pub}(N)$. By Proposition 2.1, $p \stackrel{*}{\longleftrightarrow} q$ in $\mathrm{acyclic}(N) \circ_\phi T$ implies $p \stackrel{*}{\longleftrightarrow} q$ in $N \circ_\phi T$. Conversely, $p \stackrel{*}{\longleftrightarrow} q$ in $N \circ_\phi T$ implies $p \stackrel{*}{\longleftrightarrow} q$ in $\mathrm{acyclic}(N \circ_\phi T)$ by Proposition 3.18. We have that $\mathrm{acyclic}(N \circ_\phi T)$ is a subnetwork of $\mathrm{acyclic}(N) \circ_\phi T$ as any cycle in the adaptation graph of N is still a cycle in the adaptation graph of $N \circ_\phi T$. Hence $p \stackrel{*}{\longleftrightarrow} q$ in $\mathrm{acyclic}(N) \circ_\phi T$. □

The normal form network $\mathrm{nf}(M)$ may have adaptation cycles, while $\mathrm{acyclic}(\mathrm{nf}(\mathrm{N}))$ may have isolated or free nonpublic ports. The *acyclic normal form* of N, $\mathrm{anf}(N)$, is $\mathrm{nf}(\mathrm{acyclic}(\mathrm{nf}(\mathrm{N})))$. Clearly $\mathrm{anf}(N) \equiv N$.

Proposition 3.20 *The adaptation graph of* $\mathrm{anf}(N)$ *is acyclic.*

Proof: There are certainly no instances of link collapse or adapter elimination rules in $\mathrm{nf}(N)$, hence none in $N^* = \mathrm{acyclic}(\mathrm{nf}(\mathrm{N}))$. The free port and isolated port rules may apply to N^*, but these cannot introduce cycles. □

In the case of adapter-type cycles, it suffices to use $\mathrm{anf}(N)$ in place of $\mathrm{nf}(N)$ for Theorems 3.3, 3.12, and 3.16.

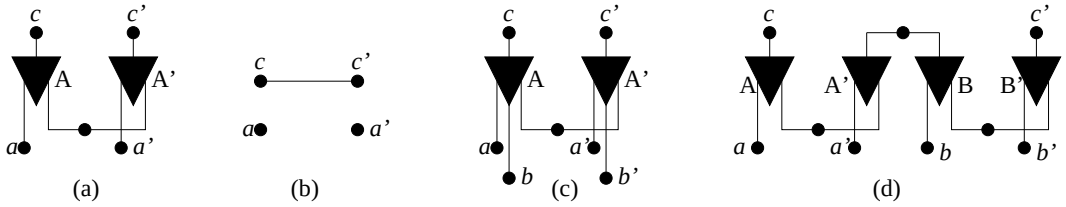

Figure 8: Reverse multiplexing examples: (b) generalizes (a); (c) and (d) are equivalent.

4 Reverse multiplexing

We now consider the questions of equivalence and generalization for networks with 'reverse multiplexing', i.e. containing adapters with more than one host port. The external behavior of such networks is more complicated than with simple multiplexing, hence the techniques required are more complex. For example, isomorphism of normal forms implies equivalence, but the converse is not true. The two networks of 8(c) and 8(d) are equivalent and in normal form, but are not isomorphic.

Since what follows has many technicalities, we first give an informal overview of the next few sections.

Consider the network N in Figure 8(a), which has two adapters each with two host ports, with one pair connected together. As depicted, no pair of ports communicate. However, ports c and c' *tentatively communicate*: they do not communicate, but in any supernetwork where c communicates with some port, for example in a supernetwork with $a \cdot a'$, c can only communicate with c'. The two adapters are *tentatively paired*, and ports a and a' are *guards*, in the sense that they 'guard' the communication of c and c'. Tentative communication and tentative pairing propagate "upwards," for example, if two identically-typed simple adapters had host ports connected to c and c', respectively, than the two adapters would be tentatively paired and every pair of identically-labelled guest ports would tentatively communicate.

Tentative pairing makes normal forms more complex; for example all the networks in Figure 8 are in normal form, even though there are host-host connections. Section 4.5 gives a structure theorem for normal forms, which can be interpreted as follows. A normal-form network consists of a set of *backbone* adapters and links and a set of tentatively-paired adapters. The tentatively-paired adapters form a family of pairs of subnetworks. Each pair consists of two isomorphic, acyclic subnetworks, where the isomorphism maps an adapter to its tentatively paired adapter. The backbone has the structure of a normal form as in Section 3, i.e. a family of adaptation trees with guest-guest connections, except that some number of connections are replaced with tentative connections, i.e. a connection $a \cdot b$ could be replaced with $a \cdot a'$, $b' \cdot b$, where b tentatively communicates with b'. In Figure 8(d), there are two pairs of subnetwork, one pair consisting of A and A', the other B and B'. In Figure 10, with subnetwork C removed, A is tentatively paired with A' and p is tentatively connected to q.

The next few sections develop the notion of normal forms. Section 4.1 gives some easy technical propositions about pasting paths. Section 4.2 gives the formal definition of tentative communication and pairing, while Section 4.3 extends the notion of reduction. The notion of adaptation cycles is more complex with reverse multiplexing. Section 4.4 shows that, as in Section 3.4, cycles can be removed while preserving equivalence; an important consequence is an induction order on normal forms. Finally, Section 4.5 gives the normal form characterization.

4.1 Pasting paths

For $\mathcal{S}$ a pairing of $\mathrm{pub}(N)$, we write $N[\mathcal{S}]$ for the network obtained by adding $\mathcal{S}$ to the connections of N and remove ports paired in $\mathcal{S}$ from $\mathrm{pub}(N)$. Equivalently $N[\mathcal{S}]$ is $N \circ_{\mathcal{S}} \epsilon$, where ϵ is the empty network. For simplicity we write e.g. $N[p \cdot q]$ for $N[\{p \cdot q\}]$.

The following two lemmas are needed in Section 4.7. The proof of Lemma 4.1 is easy (though the lemma is false if the hypothesis '$t \xleftrightarrow{*} t'$ in N' is replaced with '$t \xleftrightarrow{*} t'$ in $N[s \cdot s']$').

Lemma 4.1 *Suppose s, s', t, t' are all distinct and have the same layer, $p \xleftrightarrow{*} p'$ in $N[s \cdot s']$ and $t \xleftrightarrow{*} t'$ in N. Then $p \xleftrightarrow{*} p'$ in $N[s \cdot t, s' \cdot t']$.*

Lemma 4.2 *Suppose s_i, s'_i, t_i, t'_i, $i = 1, \ldots, m$, are all distinct, and for each i, s_i, s'_i, t_i, t'_i have the same layer. Let $\mathcal{S} = \{s_i \cdot s'_i : i = 1, \ldots, m\}$ and $\mathcal{U}_j = \{s_i \cdot t_i,\ s'_i \cdot t'_i : i \leq j\}$, $j = 0, \ldots, m$. If $p \xleftrightarrow{*} p'$ in $N[\mathcal{S}]$ and $t_j \xleftrightarrow{*} t'_j$ in $N[\mathcal{U}_{j-1}]$, $j = 1, \ldots, m$, then $p \xleftrightarrow{*} p'$ in $N[\mathcal{U}_m]$.*

Proof: Let $\mathcal{T}_j = \{s_i \cdot s'_i : i > j\} \cup \mathcal{U}_j$, $j = 1, \ldots, m$. An inductive proof for $j = 1, \ldots, m$ using Lemma 4.1 shows $p \xleftrightarrow{*} p'$ in $N[\mathcal{T}_j]$. The lemma follows since $\mathcal{T}_m = \mathcal{U}_m$. □

4.2 Tentative pairing and communication

We define "tentative communication" of ports and "tentative pairing" of adapters simultaneously by the following recursive definition. Two guest ports A_g and B_g with the same label *tentatively communicate*, written $A_g \leftrightsquigarrow B_g$, if adapters A and B are tentatively paired. Two adapters A and B of the same type are *tentatively paired* if for some host label h, either

1. $A_h \cdot B_h$; or
2. For some sequence of guest ports $c_1, d_1, \ldots, c_n, d_n$, $A_h \cdot c_1 \leftrightsquigarrow d_1 \cdot c_2 \leftrightsquigarrow d_2 \ \ldots \ c_{n-1} \leftrightsquigarrow d_n \cdot B_h$.

In Figure 8(a), the two adapters are tentatively paired and $c \leftrightsquigarrow c'$, and similarly in Figure 8(c).

Ports a and b are *tentatively connected* if there is a sequence $c_1, d_1, \ldots, c_n, d_n$, $n \geq 1$, so that $a \cdot c_1 \leftrightsquigarrow d_1 \cdot c_2 \leftrightsquigarrow d_2 \ \ldots \ c_n \leftrightsquigarrow d_n \cdot b$.

Suppose adapters A and A' are tentatively paired. A host port A_h is a *guard* if it is not connected or tentatively connected to A'_h; the pair $\{A_h, A'_h\}$ is a *tentative guard pair*. In Figure 8(a), $\{a, a'\}$ are a tentative guard pair, and in Figure 8(c), $\{a, a'\}$ and $\{b, b'\}$ are tentative guard pairs.

Proposition 4.3 *Suppose $p \leftrightsquigarrow q$ in network N, and for some composition $N \circ_\sigma T$, $p \leftrightarrow r$ in $N \circ_\sigma T$. Then $q = r$.*

4.3 Reduction

Network M *reduces in one step* to N if N is obtained from M by the link collapse, free port elimination, or isolated port elimination rules as in Section 3.1, or by one of the following rules (the first of which replaces the previous adapter elimination rule):

> *Adapter elimination:* M contains a pair of adapters A and B of the same type, and for each host label h, $A_h \cdot B_h$. N is obtained by deleting A and B. In addition, for each guest label g so that both A and B have ports with label g, A_g and B_g become the end ports of a new link.
>
> *Adapter mismatch:* M contains a pair of adapters A and B with connected or tentatively connected host ports, and either the adapter types do not match or the host labels do not match. N is obtained by deleting A and B.
>
> *Multiple tentative pairing:* M contains distinct adapters A, B, and C so that A is tentatively paired with both B and C. N is obtained by deleting A, B, and C.

A network is *irreducible* if no reduction applies. By the second rule, in an irreducible network an adapter is tentatively paired with at most one other adapter. As in Section 3, a network N reduces in a finite number of steps to a unique irreducible network $\mathrm{red}(N)$ while preserving public ports and equivalence.

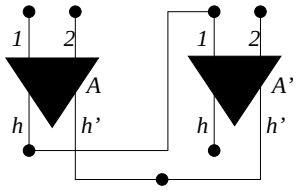

Figure 9: Adapters A and A' are tentatively paired, so e.g. $A_2 \leftrightsquigarrow A'_2$; however there is no supergraph in which $A_2 \leftrightarrow A'_2$. The adaptation graph is acyclic. The tc-graph has nodes $\{A_1, A'_1\}$ and $\{A_2, A'_2\}$, each of which has an arc to itself, because $A_h \cdot A'_1$.

4.4 The tc graph

In Section 3.4, we simplified networks by removing subnetworks that caused cycles in the adaptation graph. This removal preserved equivalence, since the cycle-inducing subnetworks could never be used to form communicating pairs. With reverse multiplexing a similar simplification is necessary; however, the definition of adaptation graph must be generalized. Consider the network N in Figure 9. Adapters A and A' are tentatively paired, so e.g. $A_2 \leftrightsquigarrow A'_2$. The adaptation graph of N is acyclic. However, it is easy to check that there is no supernetwork of N in which $A_2 \leftrightarrow A'_2$; hence if N were to appear as a subnetwork of another network, it could be removed. The tc-graph of N, to be defined, will consist of a cycle.

Let N be an irreducible network. A *tc-chain* is a set of guest ports $\{p_1, p_2, \ldots p_n\}$, $n \geq 1$, such that for $i = 1, \ldots, n-1$, either $p_i \cdot p_{i+1}$ or $p_i \leftrightsquigarrow p_{i+1}$; it is *maximal* if it is not a subset of any other tc-chain. It is easy to check that any guest port is in a unique maximal tc-chain.

The *tc-graph* of N is a directed graph. Its nodes are maximal tc-chains. There is an arc from tc-chain P to tc-chain Q if there is $p \in P$ and $q \in Q$ so that p is the guest port of an adapter with host port connected to q. See Figure 9. The *tentatively communicating acyclic core*, $\mathrm{tca}(N)$, is the network obtained from N by deleting, for each guest port $p \in P$ where P appears in a cycle in $\mathrm{tca}(N)$, the adapter containing P and all its nonpublic ports. For the network N in Figure 9, $\mathrm{tca}(N)$ consists only of isolated ports, the ports public in N.

The proofs of the following are similar to the proofs of Proposition 3.17 and corollary 3.19.

Proposition 4.4 *Suppose there is a directed arc from P to Q in the tc-graph, and a node P' in the communication graph with $P \subseteq P'$. Then the communication graph contains a node Q' with $Q \subseteq Q'$ and an arc from P' to Q'.*

Lemma 4.5 $N \equiv \mathrm{tca}(N)$.

4.5 Normal forms

A *backbone* adapter is an adapter that is not tentatively paired; its ports are *backbone* ports. The *normal form* of network N, $\mathrm{nf}(N)$, is red(tca(red(N))).

Lemma 4.6 *Let N be a network.*

1. $N \equiv \mathrm{nf}(N)$.
2. *The tc-graph of* $\mathrm{nf}(N)$ *is acyclic*
3. *If backbone ports p and q are tentatively connected in* $\mathrm{nf}(N)$*, then at least one of p and q is a guest port.*
4. *All free ports of* $\mathrm{nf}(N)$ *are public.*

5. *If A, A' are tentatively paired adapters of* $\mathrm{nf}(N)$*, and h is a host label, then either A_h is a guard, $A_h \cdot A'_h$, or A_h is tentatively connected to A'_h.*

Proof: (1), (2), (4), and (5) are straightforward. For (3) suppose p and q are both tentatively connected host ports. By the adapter mismatch rule, they must be identically-labelled host ports of adapters of the same type, but then the adapters are tentatively paired, contradicting the assumption that p and q are backbone ports. □

4.6 Basic tests

The results on equivalence and generalization in Section 4.7 depend upon the notion of basic tests. Each basic test of M is determined by a peak of M, where a peak generalizes a guest-guest connection or a free guest port in a simple-network normal form. A basic test is constructed using a 'copy' network, as in Section 3.2, in a way that guarantees that requisite tentative communication becomes actual communication. By corollary 4.8 of this section, M satisfies its own basic tests; by Theorem 4.9 of the next section, if some other network N also satisfies all of the basic tests of M, then $M \preceq N$. This theorem immediately yields algorithms to test if $M \preceq N$ or $M \equiv N$.

Suppose k is a node of the tc-graph of N. Network N_k is the subnetwork of N obtained as follows: for every tc-graph node r reachable from k in the tc-graph, and for every guest port $g \in r$, N_k contains g, its adapter, and every host port of the adapter. Two ports of N_k are connected if they are connected in N.

A *peak* is a node of the tc-graph with indegree 0. Peaks can be classified as follows.

- A *full peak* is of the form $p_1 \cdot p_2 \leftrightsquigarrow p_3 \ \ldots \ p_{n-2} \leftrightsquigarrow p_{n-1} \cdot p_n$, i.e. it starts and ends with a connection; necessarily $n \geq 2$ is even. Both p_1 and p_n must be guest ports of backbone adapters, for if either adapter were tentatively paired, then one of p_1 and p_n would tentatively communicate with another port, contradicting the maximality of the tc-chain.

- A *half peak* is of the form $p_1 \cdot p_2 \leftrightsquigarrow p_3 \ \ldots \ p_{n-1} \leftrightsquigarrow p_n$, i.e. starting with a connection and ending with tentative communication, or the reverse. Port p_1 must be a guest port of a backbone adapter, and p_n must be free, hence public. Possibly $n = 1$, in which case p_1 is a public guest port of a backbone adapter.

- A *link peak* is either a link or of the form $p_1 \leftrightsquigarrow p_2 \ \ldots \ p_{n-1} \leftrightsquigarrow p_n$, i.e. starting and ending with tentative communication; necessarily $n \geq 2$ and p_1 and p_n are both public.

In Figure 10, $\{p, e, f, q\}$ is a full peak; in Figure 6, $\{r\}$ is a half peak.

Let N be a network in normal form. The 'copy' network $C = C(N)$ is a copy of the backbone of N, with tentative connections collapsed to connections. More precisely, for each backbone adapter A of N, C has an adapter $c(A)$ of the same type, and for each port A_l of A, $c(A)$ has a port $c(A)_l$ with the same layer and label. We extend c to ports by defining $c(A_l) = c(A)_l$. Two ports $c(g)$ and $c(h)$ are connected in C if g and h are connected or tentatively connected backbone ports in N, and one of g, h is a guest port and the other a host port. All free ports of C are public. Notice that C has no links. See Figure 10.

A port p is *tentatively free* if it is free or $p \cdot e_1 \leftrightsquigarrow f_1 \cdot e_2 \leftrightsquigarrow f_2 \ \ldots \ e_n \leftrightsquigarrow f_n$ and f_n is free; the *representative* of p, $r(p)$, is p itself in the former case and f_n in the latter case.

Let $\phi = \{r(h) \cdot c(h) : h \text{ a tentatively free nonguard host port of } N\}$.

For k a peak, let C_k be the subnetwork of C obtained from the backbone adapters of N_k. Let $\phi_k = \{r(h) \cdot c(h) \in \phi : \ h \text{ in } N_k\}$. Let G be the set of tentative guard pairs of N and $G_k = \{\{g, g'\} \in G : g, g' \text{ in } N_k\}$. (We use '$g$' for guard, even though a guard is actually a host port.)

The *surrogate* $s(g)$ of guard g is $r(g)$ if g is tentatively free in N; otherwise $s(g)$ is $c(m)$, where m is the backbone guest port tentatively connected to g. See Figure 10. For $G^* \subseteq G$ define $\mathcal{S}(G^*) = \{s(g) \cdot s(g') : \{g, g'\} \in G^*\}$.

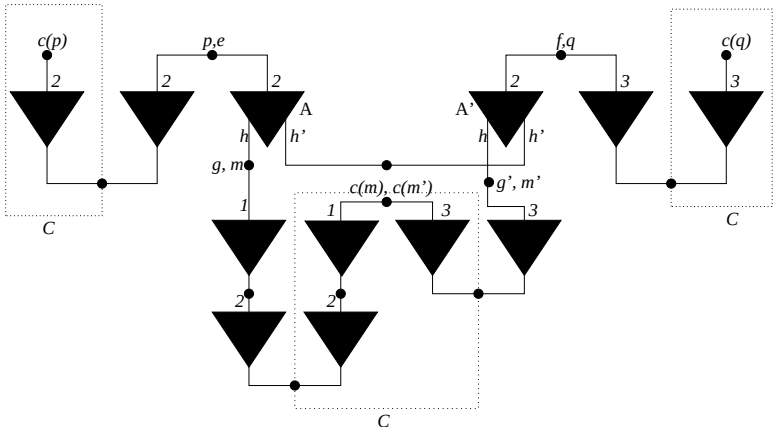

Figure 10: Example of copy network construction. Subnetwork inside dotted boxes is C; the remainder is N. $1, 2, 3$ are guest labels; h, h' are host labels. Ports g and g' are guards; m, m' guest ports; $g \cdot m$, $g' \cdot m'$. The surrogate $s(g) = c(m)$, similarly $s(g') = c(m')$; $k = \{p, e, f, q\}$ is a full peak, where $p \cdot e \leftrightsquigarrow f \cdot q$. The test ports are $c(p)$ and $c(q)$, which communicate in $C \circ_\phi N$.

The *basic test for peak* k of network N is the tuple (C_k, a, b, ω), where $\omega = \phi_k \cup \mathcal{S}(G_k)$ and the test ports a, b are defined as follows. Suppose $k = \{p_1, \ldots, p_n\}$. If k is a link peak, $\{a, b\} = \{p_1, p_n\}$. If k is a full peak, $\{a, b\} = \{c(p_1), c(p_n)\}$. If k is a half peak, with p_1 the backbone guest port, then $p_n = r(p_1)$ and we set $\{a, b\} = \{c(p_1), p_n\}$.

Lemma 4.7 *Let (C_k, a, b, ω) be the basic test for peak k of network N, and let $T = C_k \circ_\omega N$.*

1. *If p is a backbone guest port of N_k, p and $c(p)$ communicate in T.*
2. *If $p \leftrightsquigarrow q$ in N_k, then $p \leftrightarrow q$ in T.*

Proof: Both statements are shown by simultaneous induction on the ordering implied by the tc-graph of N.

Suppose p is a backbone guest port, $p = A_g$ for some adapter A and guest label g. We show that for each host label h, A_h is connected or indirectly connected to $c(A)_h$ in T (i.e. $A_h \cdot q \stackrel{*}{\longleftrightarrow} q' \cdot A'_h$, some q, q'); this implies $p \leftrightarrow c(p)$ in T. If A_h is free in N, then by definition $A_h \cdot c(A)_h$ in T. Suppose A_h is not free but is tentatively free, i.e. $A_h \cdot c_1 \leftrightsquigarrow d_1 \ \ldots \ c_n \leftrightsquigarrow d_n$ in N with d_n free. Then $d_n = r(A_h)$ and $d_n \cdot c(A)_h$ in T; by induction hypothesis, $c_i \leftrightarrow d_i$ in T. Hence A_h is indirectly connected to $c(A)_h$. Finally suppose A_h is tentatively connected to some guest port A'_g of adapter A' in N. By using the induction hypothesis, A_h is indirectly connected to A'_g in T. Again by the induction hypothesis A'_g communicates with $c(A')_g$ in T. By definition $c(A')_g \cdot c(A)_h$ in C_k. Hence again A_h is indirectly connected to $c(A)_h$ in T.

Suppose $p \leftrightsquigarrow q$, where $p = A_g$ and $q = A'_g$ for some tentatively paired adapters A and A' and some guest label g. We show that for each host label h, A_h is connected or indirectly connected to A'_g. By Lemma 4.6, either $A_h \cdot A'_h$, or A_h tentatively connected to A'_h, or A_h, A'_h both guards. There is nothing to show in the first case; the second case follows using the induction hypothesis. So suppose A_h, A'_h are both guards. By definition, ϕ connects the surrogate $s(A_h)$ with the surrogate $s(A'_h)$. If the surrogate $s(A_h)$ is not A_h, then $A_h \cdot r$ and $s(A_h) = c(r)$ for some guest port r; by induction hypothesis r communicates with $c(r)$. Similarly, A'_h is either its own surrogate, or is connected to a port that communicates with its surrogate. Hence A_h is connected or indirectly connected to $c(A)_h$. □

Corollary 4.8 *Network N satisfies its own basic tests.*

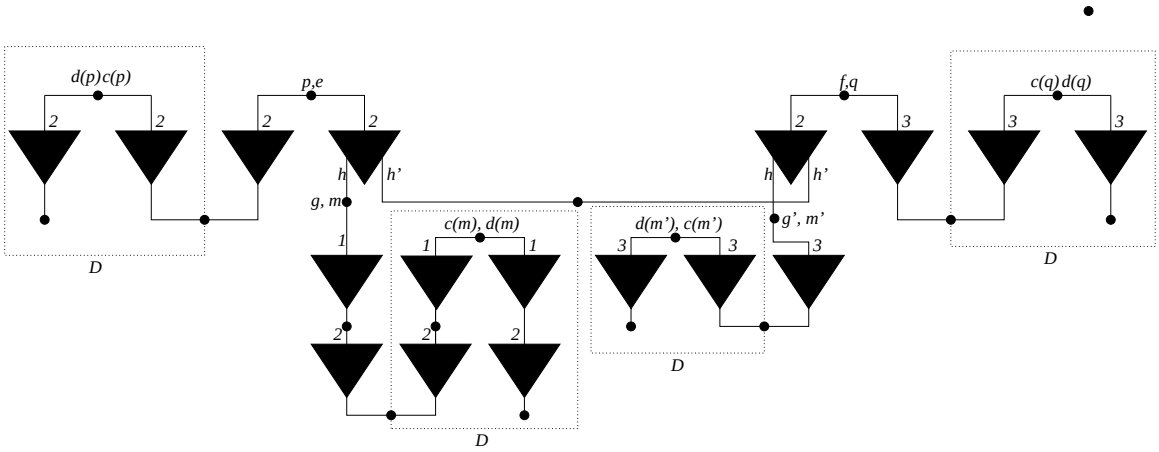

Figure 11: The network $M \circ_\phi D[\mathcal{T}(\{g, g'\})]$. We have $s(g) = c(m)$, $t(g) = d(m)$, $s(g') = c(m')$, and $t(g') = c(m')$. The pairing $S(\{g, g'\})$ would connect $c(m) \cdot c(m')$.

Proof: Suppose $k = \{p_1, \ldots, p_n\}$ is a peak and (C_k, ϕ, a, b) the corresponding basic test. If k is a link peak, then $a = p_1 \leftrightsquigarrow p_2 \cdot p_3 \leftrightsquigarrow p_4 \ \ldots \ p_{n-1} \leftrightsquigarrow p_n = b$. Using Lemma 4.7, a and b communicate. If k is a full peak, then $a = c(p_1)$, $p_1 \cdot p_2 \leftrightsquigarrow p_3 \ \ldots \ p_{n-2} \leftrightsquigarrow p_{n-1} \cdot p_n$, and $b = c(p_n)$. Again by Lemma 4.7, a and b communicate. The case that k is a half peak is similar. □

4.7 Generalization

Theorem 4.9 *Suppose* $\mathrm{pub}(M) \subseteq \mathrm{pub}(N)$. *Then* $M \preceq N$ *iff* N *satisfies every basic test of* M.

Trivially $M \preceq N$ implies that N satisfies every basic test of M. The rest of this section is devoted to the proof of the converse. Throughout the section, M and N are networks that satisfy the hypothesis of the theorem and (Q, y, y', σ) is a test that satisfies M, i.e. $y \stackrel{*}{\longleftrightarrow} y'$ in $M \circ_\sigma Q$. The theorem is proved if we show $y \stackrel{*}{\longleftrightarrow} y'$ in $N \circ_\sigma Q$.

By Lemma 4.6 we can assume that M is in normal form. Let n be the maximal communication chain containing y, y'; by pruning M and Q as necessary, we can also assume that $M \circ_\sigma Q = (M \circ_\sigma Q)_n$. Thus every port of $M \circ_\sigma Q$ appears in some communication chain. It simplifies the argument a bit to assume that there is no connection $m \cdot m' \in \sigma$ with m, m' in M; any such connection could be replaced with connections in σ from m, m' to the end ports of a new link in Q. Similarly, we can assume $y, y' \in Q$.

> *Remark:* The proof that follows has details that obscure its structure, so we give a brief preview. The communication $y \stackrel{*}{\longleftrightarrow} y'$ is established inductively in $M \circ_\sigma Q$, and we wish to establish the same communication in $N \circ_\sigma Q$. However, we know nothing about the ports in N, so it is not obvious how to copy the induction to $M \circ_\sigma Q$. Instead, we construct a network D that sits between N and Q, in particular forming the composition
>
> $$N \circ_\phi D \circ_{\sigma'} Q$$
>
> where ϕ is as in Section 4.6 and and σ' is derived from σ. An 'observable' port is a backbone port of M or a port of Q; for each observable port p there is a corresponding port $d(p)$, where $d(p) \in D$ if $p \in M$, and $d(p) = p$ if $p \in Q$. For any observable ports p, p' so that $p \stackrel{*}{\longleftrightarrow} p'$ in $M \circ_\sigma Q$, we will show that $d(p) \stackrel{*}{\longleftrightarrow} d(p')$ in $N \circ_\phi D \circ_{\sigma'} Q$. In particular, $y = d(y) \stackrel{*}{\longleftrightarrow} d(y') = y'$

in $N \circ_\phi D \circ_{\sigma'} Q$. By construction, the network D is a doubled network, so Proposition 3.11 immediately implies that $y \overset{*}{\longleftrightarrow} y'$ in $N \circ_\sigma Q$.

The doubled network $D = D(M)$ consists of $C = C(M)$ and another network isomorphic to C, with corresponding guest ports connected. More precisely, for each backbone adapter A of N, D has an adapter $d(A)$ of the same type, and for each port A_l of A, $d(A)$ has a port $d(A)_l$ with the same layer and label. We extend d to ports by defining $d(A_l) = d(A)_l$. If $d(p)$ is a free guest port in D, then $c(p)$ is as well; $c(p)$ and $d(p)$ are connected in D. See Figure 11.

An *observable* port is a port of Q or a backbone port of M. We extend d to be the identity on Q, i.e. $d(p) = p$ if p is a port of Q; then d is defined for all observable ports.

Recall ϕ from Section 4.6 connects $r(h)$ to $c(h)$, for each tentatively free nonguard host port h of M. The pairing σ' is obtained from σ by replacing any connection to $r(h)$ with a connection to $d(h)$ and by deleting any connections to the representatives $r(g)$, g a tentatively free guard.

Remark. For each peak k, we know N satisfies the basic test for k, (C_k, a, a', ω). We would like to argue that the test subnetwork C_k is a subnetwork of D, and hence use Proposition 2.1 to argue that $a \overset{*}{\longleftrightarrow} a'$ in $N \circ_\phi D \circ_{\sigma'} Q$. However, C_k is not actually a subnetwork of D: in C_k, for each tentative guard pair $\{g, g'\}$, the surrogates $s(g)$ and $s(g')$ are connected (Figure 10), whereas in D, the surrogate $s(g)$ is connected to a different port $t(g)$, as described below (Figure 11). We need to use the mechanism of Lemma 4.2 to argue that $a \overset{*}{\longleftrightarrow} a'$ in $N \circ_\phi D \circ_{\sigma'} Q$ (Lemma 4.10). This requires that D be parameterized by the connections to surrogates; we now describe the notation.

Recall G is the set of tentative guard pairs of M. Let $\{g, g'\}$ in G. By assumption, corresponding guest ports of the adapters containing g, g' communicate in $M \circ_\sigma Q$, so g, g' are indirectly connected, though also by assumption $g \cdot g' \notin \sigma$.

We define $t(g)$ as follows. If g is tentatively free in M, and $r(g)$ is the representative of g, then the surrogate $s(g) = r(g)$ is a public port of M, $s(g) \cdot q \in \sigma$ for some port $q \in Q$; we define $t(g) = q$. Otherwise g is tentatively connected to backbone port p in M, then $s(g) = c(p)$; we define $t(g) = d(p)$. See Figure 11. The following restatement is useful below: if p is the observable port indirectly connected to g, then $t(g) = d(p)$.

For $G^* \subseteq G$, let

$$\mathcal{T}(G^*) = \{s(g) \cdot t(g), s(g') \cdot t(g') : \{g, g'\} \in G^*\}$$

and recall that

$$\mathcal{S}(G^*) = \{s(g) \cdot s(g')) : \{g, g'\} \in G^*\}.$$

For p an observable port, we let $l(p)$ denote the level of the maximal communication chain containing p in $M \circ_\sigma Q$; for g a guard, we let $l(g) = l(p)$, where $g \cdot p$. Easily $l(y) = l(y')$ and if $\{g, g'\} \in G$, $l(g) = l(g') < l(y)$. For $m = 0, 1, \ldots, l(y)$, let $H_m = \{\{g, g'\} \in G : l(y) < m\}$, so $H_{l(y)} = G$, and define

$$N_m = N \circ_\phi D[\mathcal{T}(H_m)] \circ_{\sigma'} Q.$$

The *communication hypothesis at level m* is the following statement:

Suppose p, p are observable ports, $p \overset{*}{\longleftrightarrow} p'$ in $M \circ_\sigma Q$, and $l(p) = m$. Then $d(p) \overset{*}{\longleftrightarrow} d(p')$ in N_m.

Lemma 4.10 *Suppose k is a peak of M, a, a' are the test ports of the basic test for k, and the communication hypothesis holds at levels less than $m = l(k)$. Then $a \overset{*}{\longleftrightarrow} a'$ in N_m.*

Proof: For the rest of this proof, index the elements of $G_k = \{\{g_i, g_i'\} : i = 1, \ldots, n\}$ so that $i < j$ implies $l(g_i) \leq l(g_j)$. Clearly $l(g_i) < m$ for any $\{g_i, g_i'\} \in G_k$. For $j = 0, \ldots, n$, let $G_{kj} = \{\{g_i, g_i'\} : i \leq j\}$, so G_{k0} is empty and $G_{kn} = G_k$.

Let (C_k, a, a', ω) be the basic test for k in network M, where $\omega = \phi_k \cup \mathcal{S}(G_k)$. Since N satisfies all the basic tests of M, $a \stackrel{*}{\longleftrightarrow} a'$ in $N \circ_\omega C_k$. We have C_k a subnetwork of D and $\phi_k \subseteq \phi$, so $N \circ_\omega C_k$ is a subnetwork of $N \circ_\phi D[\mathcal{S}(G_k)]$, and hence of $N \circ_\phi D[\mathcal{T}(H_l \setminus G_k) \cup \mathcal{S}(G_k)]$. By Lemma 2.1,

$$a \stackrel{*}{\longleftrightarrow} a' \text{ in } N \circ_\phi D[\mathcal{T}(H_l \setminus G_k) \cup \mathcal{S}(G_k)] \circ_{\sigma'} Q. \tag{1}$$

Choose $\{g_i, g_i'\} \in G_k$. Then g_i, g_i' are indirectly connected to observable ports p, p', respectively, with $p \stackrel{*}{\longleftrightarrow} p'$ in $M \circ_\sigma Q$. By the induction hypothesis, $t(g_i) = d(p) \stackrel{*}{\longleftrightarrow} d(p') = t(g_i')$ in $N \circ_\phi D[\mathcal{T}(H_{l(g_i)})] \circ_{\sigma'} Q$. We also have $H_{l(g_i)} \subseteq G_{ki-1} \cup H_m \setminus G_k$. Hence

$$t(g_i) \stackrel{*}{\longleftrightarrow} t(g_i') \text{ in } N \circ_\phi D[\mathcal{T}(G_{ki-1} \cup H_m \setminus G_k)] \circ_{\sigma'} Q. \tag{2}$$

Using equations (1) and (2) and Lemma 4.2

$$a \stackrel{*}{\longleftrightarrow} a' \text{ in } N \circ_\phi D[\mathcal{T}(G_{kn} \cup H_m \setminus G_k)] \circ_{\sigma'} Q,$$

Since $G_k = G_{kn}$, the lemma is established. □

Lemma 4.11 *The communication hypothesis holds for all levels.*

Proof: The proof is by induction on level. Suppose $p \stackrel{*}{\longleftrightarrow} p'$ in $M \circ_\sigma Q$. We show that $d(p) \stackrel{*}{\longleftrightarrow} d(p')$ in N_m in two steps: first if $p \leftrightarrow p'$ in one step, and next if $p \stackrel{*}{\longleftrightarrow} p'$ in more than one step.

Suppose $p \leftrightarrow p'$ in $M \circ_\phi Q$ in one step and p, p' are end ports of a link. If the link is in M, p, p' must be public, and easily p, p' are end ports of a link in N. Otherwise the link is in Q. In either case $p \stackrel{*}{\longleftrightarrow} p'$ in N_l.

Now suppose $p \leftrightarrow p'$ in $M \circ_\phi Q$ in one step and $p = A_g$, $q = A_g'$ for some adapters A and A' and guest label g. Consider a host label h of A. In $M \circ_\phi Q$, either $A_h \cdot A_h'$ or for some ports s and s', $A_h \cdot s$, $s' \cdot A_h'$, and s communicates with s'. We assume the latter case; the former is similar and easier. We show $d(A_h) \cdot d(s)$ or $d(A_h) \cdot s \stackrel{*}{\longleftrightarrow} c(s) \cdot d(s)$. By construction $d(A_h) \cdot d(s)$ unless A is an adapter of Q and s is a guest port of M. In this case $d(A_h) = A_h$ and $A_h \cdot s$ but $s \neq d(s)$. By Lemma 4.7, s communicates with $c(s)$. By construction $c(s) \cdot d(s)$. Similarly either $d(A_h') \cdot d(s')$ or $d(A_h') \cdot s' \stackrel{*}{\longleftrightarrow} c(s') \cdot d(s')$. By induction hypothesis, $d(s)$ communicates with $d(s')$, so $d(A_h)$ is indirectly connected to $d(A_h')$. Since this argument holds for all host labels h, $d(A_g)$ communicates with $d(A_g')$ in N_l.

Now suppose $p \stackrel{*}{\longleftrightarrow} p'$ in $M \circ_\phi Q$ in more than one step, $p = p_0 \leftrightarrow p_1 \ \ldots \ p_{n-1} \leftrightarrow p_n = p'$. Consider the sequence obtained by replacing each peak $p_i \ldots p_j$ of M as follows:

full peak:	$\ldots \leftrightarrow\ p_i \cdot p_{i+1}\ \ldots\ p_{j-1} \cdot p_j\ \leftrightarrow \ldots$	with	$\ldots \leftrightarrow\ d(p_i) \cdot c(p_i) \stackrel{*}{\longleftrightarrow} c(p_j) \cdot d(p_j)\ \leftrightarrow \ldots$
half peak:	$\ldots \leftrightarrow\ p_i \cdot p_{i+1}\ \ldots\ p_{j-1} \cdot p_j \cdot \ldots$	with	$\ldots \leftrightarrow\ d(p_i) \cdot c(p_i) \stackrel{*}{\longleftrightarrow} p_j \cdot \ldots$
link peak:	$\ldots \cdot\ p_i \leftrightarrow p_{i+1}\ \ldots\ p_{j-1} \leftrightarrow p_j \cdot \ldots$	with	$\ldots \cdot\ p_i \stackrel{*}{\longleftrightarrow} p_j\ \cdot\ \ldots$

The initial and final '$\leftrightarrow$' or '$\cdot$' indicates the relationship of the peak sequence $p_i \ldots p_j$ with the rest of the communication sequence; in each case the substitution preserves this relationship. If p_i or p_j is the initial or final port in the sequence, then the corresponding '$\cdot$' does not appear. There is a symmetric case for half peaks with '$\leftrightarrow$' and '$\cdot$' reversed; it is also possible for a half peak that $i = j$.

The result is a sequence of ports of N_l that are asserted to be related by communication or connection. Within the subsequence of ports from $d(p)$ to $d(p')$, we claim that all the relations hold in N_l. This proves the lemma.

Suppose two adjacent ports are related by connection: $a \cdot b$. The substitution introduces new connections only between a port $c(p)$ and $d(p)$, some backbone port p, and this connection exists in N_l by construction. Any other connection is either a connection between two private ports of Q or a connection in σ between two public nonhost ports of Q or N; in each case the connection must exist in N_l.

Suppose two adjacent ports are related by communication: $a \stackrel{*}{\longleftrightarrow} b$. If the communication was introduced by some substitution, then a and b are the test ports of some peak, so by Lemma 4.10, $a \stackrel{*}{\longleftrightarrow} b$. Otherwise

the communication must be $a = d(p_l) \stackrel{*}{\longleftrightarrow} b = d(p_{l+1})$ where $p_l \leftrightarrow p_{l+1}$ in $M \circ_\sigma Q$. This follows from the first part of the argument. □

Proof: (of Theorem 4.9) Recall that (Q, y, y', σ) is the assumed test of M. We have $d(y) = y$ and $d(y') = y'$ since $y, y' \in Q$. By Lemma 4.11, $y \stackrel{*}{\longleftrightarrow} y$ in $N_{l(y)} = N \circ_\phi D[\mathcal{T}(H_{l(y)})] \circ_{\sigma'} Q$ where $l = l(y)$. By Proposition 3.11 $y \stackrel{*}{\longleftrightarrow} y$ in $N \circ_\sigma Q$. □

Theorem 4.12 *Let M and N be networks. There is an algorithm that decides if $M \preceq N$ in time $O(|M|(|M|+|N|))$.*

Proof: Compute $\mathrm{nf}(M)$. For each basic test (C_k, a, b, ω) of M, decide if $a \leftrightarrow b$ in $C_k \circ_\omega N$. By Theorem 4.9, $M \preceq N$ exactly if N satisfies all such basic tests. There are $O(|M|)$ such tests, and it takes time $O(|N|+|M|)$ to test if a pair of ports communicate. □

5 Protection

We now consider *networks with protection*, which are defined by extending the previous definition of networks.

An adapter type now has another attribute, its *path count*, which is a pair (l, m) so that $1 \leq l \leq m$ and m is the number of host labels of the adapter type. The specification of a network now requires an additional set D of pairs of adapters. Each pair consists of two adapters of the same type; an adapter can appear in at most one pair. If $\{A, B\} \in D$ then A and B are *paired*.

Ports a and b *communicate* as before, except case 2 is modified to be

> there are paired adapters A and B of path count (l, m) and a guest label g so that $a = A_g$, $b = B_g$, and for at least l of the m host labels h, either $A_h \cdot B_h$, or $A_h \cdot a'$, $B_h \cdot b'$, and a' communicates with b'.

Adapter pairing is necessary (at least if $l \leq m/2$) to prevent a guest port A_g communicating with two different guest ports B_g, C_g. If every adapter has path count (m, m), some m, the definition is identical to the original definition except for the pairing requirement.

An adapter with a single guest port, two host ports, and path count (1,2) can be used to model 1+1 protection (where a data stream can be carried on either of two paths). More generally, an adapter with l guest ports, m host ports, and path count (l, m) adapter can be used to model $l : m$ protection (With $l : m$ protection, a protection-switching element uses m resource paths to route l data streams; as long as at least l of the resource paths have not failed, all the data streams will be routed).

The following is the main result of this section.

Theorem 5.1 *The problem of deciding equivalence of networks with protection is co-NP-complete.*

The following is quite basic, but apparently has not previously appeared.

Lemma 5.2 *The problem of deciding equivalence of monotone boolean formulae is co-NP-complete.*

Proof: [15] Inequivalence is clearly in NP. To show hardness, let C be a formula in 3-DNF on variables $x_1, \ldots, x_n$. It is NP-hard to falsify C, i.e. find an assignment to the x_i that makes C have value 0. Let D be the formula obtained from C by replacing, for each variable x_i, each complemented instance $\bar{x}_i$ by an instance of a new variable y_i. So D is a monotone formula on $x_1, \ldots, x_n, y_1, \ldots, y_n$. Let $F = \bigwedge_i (x_i \vee y_i)$ and $G = \bigvee_i (x_i \wedge y_i)$. Then $F \wedge \bar{G} \Rightarrow D$ is falsifiable exactly for an assignment for which $x_i = \bar{y}_i$ and D is 0, i.e. an assignment to the x_i for which C is 0. However, $F \wedge \bar{G} \Rightarrow D$ can be written as $F \vee (D \wedge G) \equiv F \wedge G$. Hence D is falsifiable exactly if $F \vee (D \wedge G)$ is not equivalent to $F \wedge G$. □

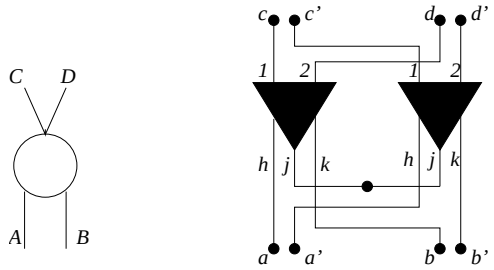

Figure 12: The network simulates an 'or' gate if the adapters are type (2,3), and an 'and' gate if type (3,3).

Proposition 5.3 *If $M \not\equiv N$, then there is a test (T, p, q, ω) with $|T| \leq \max(|M|, |N|)$ that distinguishes M and N, i.e. $p \stackrel{*}{\longleftrightarrow} q$ in one of $T \circ_\omega M$ and $T \circ_\omega N$ but not the other.*

Proof: Choose (T, p, q, ω) that distinguishes M and N with $|T|$ as small as possible. Assume $p \stackrel{*}{\longleftrightarrow} q$ in $T \circ_\omega M$ but not $T \circ_\omega N$. We can assume $\{p, q\}$ has the smallest level over all pairs $\{p', q'\}$ with $p', q' \in \text{ports}(T) \cup \text{pub}(M)$ and $p' \stackrel{*}{\longleftrightarrow} q'$ in $T \circ_\omega M$ but not $T \circ_\omega N$. Clearly T has no links nor isolated ports. Furthermore, for any guest port $t \in T$, $t \leftrightarrow u$ for some port u, otherwise we could delete t. We claim $u \in N$, which implies $|T| \leq |M|$. If instead $u \in T$, then the level of $\{t, u\}$ is less than the level of $\{p, q\}$, else we could delete t and u, hence $t \stackrel{*}{\longleftrightarrow} u$ in $T \circ_\omega N$, but then we could eliminate t, u by modifying T and ω to connect together the ports to which they are connected. This contradicts the minimality of T. □

Proof: (of Theorem 5.1) Network inequivalence is in NP by Proposition 5.3. To show hardness, we reduce monotone boolean formulae equivalence to network equivalence. Let F, G be two monotone boolean formulae on the same set of variables $X_1, \ldots, X_n$. We can convert them easily to two monotone circuits, where every node is either an input labelled with a variable, the unique output node, or a gate node, either 'and' or 'or' with at most two inputs and fanout at most 2.

Consider an 'or' gate with two inputs A and B and two outputs C and D and the subnetwork of two adapters depicted in Figure 12, where each adapter is of class $(2, 3)$. Each input and output of the gate is represented by a pair of ports, e.g. C is represented by the pair c, c'. The boolean output value 1 is encoded if the port pair communicates, and 0 is encoded if they do not. The connection between the output of one gate and the input of another is modeled by connecting the two pairs of ports. The subnetwork in Figure 12 is an 'or' gate: $c \leftrightarrow c'$ iff $d \leftrightarrow d'$ iff either a, a' are connected to a communicating pair of ports, or b, b' are connected to a communicating pair. An 'and' gate can be obtained simply by changing the class of the adapters to $(3, 3)$.

Given circuit F, we can construct a network F^* by replacing each gate with a subnetwork as described. Each input X_i to F is represented by a pair $\{x_i, x_i'\}$ of public ports. The output is represented by a pair of public ports q, q'. Similarly we can construct G^*, with the same set of public ports for inputs and outputs.

We claim that F is equivalent to G iff $F^* \equiv G^*$. If F is not equivalent to G, then there is some assignment to the $\{x_i\}$ for which they differ. We can construct a test (T, q, q', ω) so that ω connects a link in T to the pair x_i, x_i' iff x_i has value 1. Clearly, this test will distinguish F^* and G^*.

Conversely, suppose (U, p, p', ϕ) is a test that distinguishes F^* and G^*. An easy argument shows that q, q' are the only ports public in F^* that can communicate in any supernetwork of F^*, and similarly for G^*. Since the test distinguishes F^* and G^*, q, q must communicate in exactly one of $F^* \circ_\phi U$ and $G^* \circ_\phi U$. We can obtain an assignment to the X_i from the test using the rule $X_i = 1$ iff x_i, x_i' directly or indirectly connected in $F^* \circ_\phi U$. An easy induction shows that this assignment distinguishes F and G. □

References

[1] S. Acharya, B. Gupta, P. Risbood, A. Srivastava, Mobipack: optical hitless SONET defragmentation in near-optical cost, *INFOCOM 2004*, Twenty-third Annual Joint Confernece of the IEEE Computer and Communications Societies, March 2004.

[2] P. Barcelos, G. Guizzzardi, A. Garcia, Ontological Evaluation of the ITU-T Recommendation G. 805, *2011 18th International Conference on Telecommunications*.

[3] P. Bohannon, S. Fortune, C. Martin, The NetML network model. Manuscript, 2004. Available at `http://cm.bell-labs.com/who/sjf/pubs.html`.

[4] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, A. Silberschatz, Topology discovery in Heterogeneous IP Networks: the *NetInventory* System, *IEEE/ACM Transactions on Networking*, **12**:3, June 2004.

[5] J. Chen, L. Wosinksa, C. Mas Machuca, M. Jaeger, Cost vs Reliabiity Performance of Fiber Access Network Architectures, *IEEE Communications Magazine*, February 2010, 56–65.

[6] F. Dijkstra, B. Andree, K. Koymans, J. van der Ham, Introduction to ITU-T Recommendation G.805. SNE technical report SNE-UVA-2007-01, available from `http://www/science.uval.nl/research/sne/reports/`.

[7] F. Dijkstra, B. Andree, K. Koymans, J. van der Ham, P. Grosso, C. de Laat, A multi-layer network model based on ITU-T G.805. *Computer Networks* 52 (2008), pp. 1927–1937.

[8] H. van Helvoort, *SDH/Sonet explained in Functional Models: Modeling the Optical Transport Network*, John Wiley and Sons Limited, 2005.

[9] *ITU-T G.805: Generic functional architecture of transport networks*, International Telecommunication Union, Series G. Available from `www.itu.int`.

[10] *ITU-T G.809: Functional architecture of connectionless layer networks.*, International Telecommunication Union, Series G. Available from `www.itu.int`.

[11] M. Köhn, Comparison of SDH/SONET-WDM Multi Layer Networks with static and dynamic optical plane, *Conference on Optical Network Deisgn and Modeling*, pp. 403–412, February, 2005.

[12] A. McGuire, A.J. Nunn, A. Jackson, T.S. Brown, A.B.D. Reid, G.R. Hill, Functional architecture for optical networks, *BT Technical Journal*, **13:2, April 1995, pp. 145–155.**

[13] R. Milner, *Communication and Concurrency*, Prentice Hall, NJ, 1989.

[14] C. Pavan, R.M. Morais, A.N. Pinto, Estimating CapEx in Optical Multilayer Networks, *Proceedings of the 7th Conference on Telecommunications*, 2009.

[15] J. Radhakrishnan, L. Fortnow, private communication, 2004.

[16] M. Ruiz, O. Pedrola, L. Velasco, D. Careglio, J. Fernández-Palacios, G. Junyent, Survivable IP/MPLS-over-WSON Multilayer Network Optimization, *J. of Optical Communication and Networking*, 3:8, August 2011.

[17] J. Simmons, *Optical Network Design and Planning*, 2008, Springer.

[18] *TMF 513: Multitechnology network management business agreement, NML-EML interface*. TeleManagement Forum, May 2001. Available from `www.tmforum.org`.