Scaling Network Verification using Symmetry and Surgery

Gordon D. Plotkin* Nikolaj Bjørner‡ Nuno P. Lopes† Andrey Rybalchenko† George Varghese‡

LFCS, University of Edinburgh, UK* Microsoft Research, UK† Microsoft Research, USA‡

gordon.plotkin@gdp.inf.ed.ac.uk {nbjorner,nlopes,rybal,george}@microsoft.com

Abstract

On the surface, large data centers with about 100,000 stations and nearly a million routing rules are complex and hard to verify. However, these networks are highly regular by design; for example they employ fat tree topologies with backup routers interconnected by redundant patterns. To exploit these regularities, we introduce *network transformations*: given a reachability formula and a network, we transform the network into a simpler to verify network and a corresponding transformed formula, such that the original formula is valid in the network if and only if the transformed formula is valid in the transformed network.

Our network transformations exploit *network surgery* (in which irrelevant or redundant sets of nodes, headers, ports, or rules are "sliced" away) and *network symmetry* (say between backup routers). The validity of these transformations is established using a formal theory of networks. In particular, using Van Benthem-Hennessy-Milner style bisimulation, we show that one can generally associate bisimulations to transformations connecting networks and formulas with their transforms. Our work is a development in an area of current wide interest: applying programming language techniques (in our case bisimulation and modal logic) to problems in switching networks.

We provide experimental evidence that our network transformations can speed up by 65x the task of verifying the communication between all pairs of Virtual Machines in a large datacenter network with about 100,000 VMs. An all-pair reachability calculation, which formerly took 5.5 days, can be done in 2 hours, and can be easily parallelized to complete in minutes.

Categories and Subject Descriptors D.2.4 [*Software Engineering*]: Software/Program Verification

General Terms Verification, Theory

Keywords Network Verfication, Symmetries, Semantics

1. Introduction

Cloud services such as Dropbox, Google, iCloud, Amazon, and Azure contain up to a million inexpensive servers connected by a data center network. A single service request such as a Web search is split among hundreds of servers that communicate to produce the response. As our dependency on networks increases, the reliability of data center networks becomes increasingly critical. However, surveys show [19, 31] that network outages are quite common. Thus verification technologies that proactively prevent potential network disruptions are valuable.

A network, as shown in Figure 1, consists of boxes (routers, switches, firewalls, henceforth referred to as routers) that forward packets from input ports to output ports. We abstract the dataplane, or forwarding component of a router, as a set of rules that map predicates on packet headers (e.g., 32-bit IP addresses starting with 101) to output ports; some rules called Access Control Lists (ACLs) use complex predicates (e.g., sources from outside Microsoft that send SQL packets) on sets of fields to decide which packets must be dropped. The rules may also prescribe changes in packet headers. The output ports of each router are physically connected to the input ports of other routers as specified by a network topology. The network program, the composition of all the router programs following the topology, then maps packets from entry points in the network through the network to exit points.

This abstraction only enables qualitative reasoning, for example about packet reachability or about looping. It abstracts away quantitative issues, such as performance metrics (e.g., delay) and ignores the control plane that builds the forwarding rules. The correctness requirements [17] are also simple: they specify the headers that can communicate between hosts, together with predicates on the paths the headers traverse. So wherein, therefore, lies the complexity that justifies the emerging area of network verification [15–17, 22, 30]?

First, manual rules added by operators interact with the rules computed by automatic routing protocols. Second, the forwarding rules typically involve load balancing by which a packet can be sent by many possible routes to its destination. Third, routers sometimes rewrite packets. Fourth, while the program structure is simple, the state space is large. Forwarding rules operate on at least the IP and TCP headers and other fields such as MPLS [18]. Thus conservatively, we have a space of 2^{80} headers, millions of rules, and a large number N of servers (as many as 1 million) that can communicate. Checking reachability across N^2 pairs of stations for 2^{80} headers to find a network policy violation is a scaling challenge.

Early systems [22] found a single violation using SAT solvers. Later systems [15, 17] found all violations for medium sized networks using symbolic execution. NetPlumber [16] introduced efficient incremental analysis. Finally, Yang and Lam [30] used predicate abstraction to rewrite router rules in a form that is much more efficient to verify. However, these improvements do not target the scaling challenge. For example, it was reported in [10] that the cost of verifying reachability for all pairs of stations took around 1 day, even on a small university network. Verifying all pairs in a large data center in a few minutes (the time scale at which a network can be reconfigured) seemed out of reach for earlier techniques.

1.1 Scaling Verification Using Network Transformations

Fortunately, most large networks are designed using design patterns that enforce regularities that we can exploit. For example, data center networks are often arranged in fat-trees (Figure 1), which

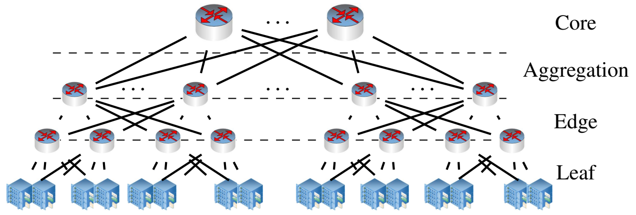

Figure 1. Data center network schematic

are leveled graphs where routers at a given level are symmetrically connected to multiple routers at adjacent levels. This is done for load balancing and resiliency reasons. For example, in Figure 2 R_3 and R_4 are symmetrically placed, for if they are interchanged and the ports on R_3, R_4 and the corresponding ports of their neighbors R_1, R_2 and R_5 are correspondingly interchanged, then the global topology stays the same.

Thus, by design, one may expect the rule sets of symmetrically placed routers at a given level (e.g., R_3 and R_4) to be symmetrical. This suggests that symmetrical routers such as R_3 and R_4 can be replaced by a single equivalent (with respect to packet forwarding) router as in Figure 2 without changing the qualitative properties of the network. Doing so reduces the rules in the new network, and in the limit can transform a "fat tree" into a "thin tree".

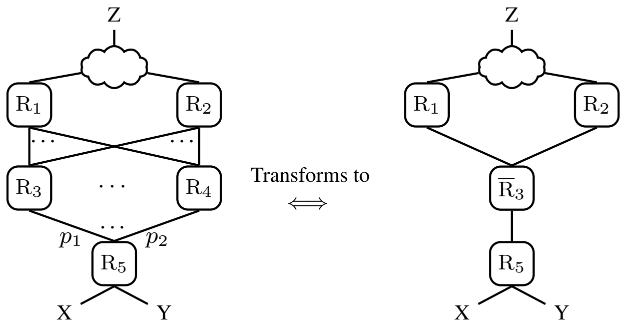

Figure 2. Suppose that interchanging R_3 and R_4 as well as corresponding marked ports on their neighbours, e.g., p_1 and p_2 on R_5, leaves reachability the same. Then R_3 and R_4 can be merged into a single router in the transformed network on the right.

Further, the hierarchical structure implies that communication within subtrees should stay local. Thus two stations X and Y attached to the same edge (or ToR, for Top of Rack switch) R_5 (see Figure 2) should typically communicate only via R_5. Thus, for verification of the communication between X and Y, the rest of the network is irrelevant as long as one can prove that traffic from X to Y does not "leak" out from the ToR R_5. This suggests a general notion of slicing away irrelevant rules and portions of the network. We refer to such slicing of ports and rules as network "surgery".

One can also slice networks in terms of headers. For example, if in reality the two backup routers R_3 and R_4 of Figure 2 are nearly identical except for two local addresses h_1 in R_3 and h_2 in R_4, we might hope to slice the network into two equivalent networks operating on disjoint sets of headers, one of which has R_3 and R_4 perfectly symmetric, and one of which has the standard topology but with a smaller set of rules (say dealing with only h_1 and h_2).

Both symmetry-induced and slicing-based transformations are *network transformations* that transform the original network into a sequence of simpler networks with equivalent forwarding, but with smaller size in terms of boxes, rules, and links. If the verification effort scales with size, and the transformations are efficient, the overall verification time will be faster.

To proceed, we need a model of networks and a way to write reachability specifications. To this end we use an operational semantics treating each router as a state machine. We then use a standard modal logic defined on the states to describe desired behaviors. The logic permits assertions that certain (anonymous) transitions occur, that a packet header has reached a certain port, or that a packet header has a certain property.

To connect network transformations with the logic we generally show that they yield *bisimulations* (in the sense of Van Benthem and Park and Milner [28]) between a network $\mathcal{N}$ and its transform $\bar{\mathcal{N}}$. We then employ the Van Benthem-Hennessy-Milner principle, as used in modal logic and process calculus [24, 28], to show the validity of a transformation using bisimulation. In process calculus this principle states that if there is a bisimulation between two processes P and $\overline{P}$, then P has a given property φ, if, and only if, $\overline{P}$ does. In our case, as, for example, $\bar{\mathcal{N}}$ may have different ports from $\mathcal{N}$, it may be necessary to modify the formula φ asserted of $\mathcal{N}$ to another, $\overline{\varphi}$, asserted of $\bar{\mathcal{N}}$ (the analogous situation in process calculus would be when comparing processes with different action alphabets, see, e.g., [4]).

We propose two different ways to employ the Van Benthem-Hennessy-Milner technique for network verification. One is to reduce the size of $\mathcal{N}$, for example, by slicing away parts of the network irrelevant to the proposition, or else by exploiting symmetry either to remove essentially duplicate rules, or to merge symmetrically placed routers with symmetrical semantics. The other is to reduce the number of properties to be verified, by finding symmetries between propositions about symmetrically placed parts of the network or by identifying equivalent packet headers. In doing so, it may be useful to pass through a sequence of such transformations, producing a sequence $\mathcal{N}_1 \sim_1 \ldots \sim_{n-1} \mathcal{N}_n$ of bisimilar networks.

Our contributions (with a paper outline) are:

1. A Theory of Network Dataplanes: To formalize network transformation, we introduce a network model (Sections 2.1, 2.2), a modal logic (Section 2.3), and a proof technique based on bisimulation to verify networks (Section 3). After describing surgeries, we describe a compositional way of generating bisimulations (Section 5). A side effect of all this machinery is the ability to formalize earlier concepts such as slicing [15], and (a generalized version of) Yang-Lam equivalence [30].

2. A Toolbox of Network Transformations: We provide a toolbox of network transformations (Examples 1 through 4) based on surgery (Section 4) and header merging and symmetry (Section 6) that can be applied iteratively to simplify the verification task. There are generally bisimulations corresponding to each. As with earlier work (e.g., [7]), we exploit symmetry over state spaces. However, to do this we exploit the structure of the network domain where the state consists of both headers and locations, and the program is distributed over boxes and rules, enabling symmetries to be constructed from *both* topological and semantic components.

2. Scaling Comprehensive Network Verification: We show (Section 7) how one might scale comprehensive network evaluation for large production data centers over all pairs of stations. In particular, for a Microsoft data center in Singapore with ∼820K rules, we reduce the time to verify reachability between all pairs of VMs from 5.5 days to 2 hours on a single core using only simple transformations, a 65× reduction. Earlier work [15–17, 30] did not report the running times for reachability between all N^2 pairs.

2. Networks and Their Logics

As a foundation for our theory, we provide a formal definition of switching networks as graphs of interconnected boxes of various kinds, such as routers, switches, bridges, or firewalls. This is network *syntax*. Having a syntax available, we then define network *semantics* in terms of suitable kinds of transition relations. The semantics, in its turn, supports a suitable *modal logic*, which can be used to specify network properties that we wish to verify.

2.1 Network Syntax

Networks are constructed using a *box signature* that consists of a set of *boxes* $b \in \mathrm{Box}$, with each box having a given finite collection $K \subseteq_{\mathrm{fin}} \mathbb{N}$ of ports, written $b : K$; we write Box for the signature. (We use a set, rather than a sequence, of ports to prepare the ground for network surgery in which we slice away ports to transform a network.) Given such a box signature, a network $\mathcal{N}$ consists of

- a finite set $\mathrm{Node}_{\mathcal{N}}$ of *nodes*,
- a *box assignment* function $\beta_{\mathcal{N}} : \mathrm{Node}_{\mathcal{N}} \to \mathrm{Box}_{\mathcal{N}}$,
- a symmetric, 1-1, irreflexive, *port connection* or *link* relation $\gamma_{\mathcal{N}}$ on the set $\mathrm{Port}_{\mathcal{N}}$ of *network ports* where:

$$\mathrm{Port}_{\mathcal{N}} =_{\mathrm{def}} \{(n, i) \mid n \in \mathrm{Node}_{\mathcal{N}},\ \beta_{\mathcal{N}}(n) : K, \text{ and } i \in K\}$$

The idea behind having both boxes and nodes is that the same box (e.g., backup router) can be replicated at different points of the network; one can think of the nodes as being box id's attached by $\beta_{\mathcal{N}}$ to their boxes. We use p to range over ports, and write the pairs (n, i) as $n.i$. The connection relation $\gamma_{\mathcal{N}}$ specifies how ports are connected to each other and hence encodes the network topology. It is symmetric as packets can flow in either direction; it is 1-1 to model physical connection; it is irreflexive to avoid tight loops. Not all ports need be connected to another port; such unconnected ports are called *external*; the others are called *internal*.

Definitions in this style of various forms of networks made up out of boxes with ports can be found in other contexts, for example sharing graphs [12], bigraphs [25], or cyclic networks [13]; other definitions can be found in the network literature, for example [32].

2.2 Network Semantics

For the network semantics, we first assume given a set $\mathrm{Pac}_{\mathcal{N}}$ of *packet headers* ranged over by h (below we may just say "packet"); we call $\mathrm{Pac}_{\mathcal{N}}$ the *header space (of $\mathcal{N}$)*. Packet headers are typically bit strings – hence the 2^{80} possibilities alluded to in the introduction — but one may use any convenient set. For each box $b : K$ we then have the set of *box-located packets*, where such a located packet is a pair, written $h \mathbin{@} i$, of a packet h and a box location, that is, a box port i (with $i \in K$). In other words, a box-located packet is a specific packet located at a specific port in the network.

Next, for each box $b \in \mathrm{Box}$, we assume given a transition relation

$$h \mathbin{@} i \longrightarrow_{\mathrm{Box},\, b} h' \mathbin{@} i'$$

between box-located packets. Note that packets may be changed (rewritten) in a transition as well as their location. When the signature can be understood from the context, we will generally just write $h \mathbin{@} i \longrightarrow_b h' \mathbin{@} i'$.

These transitions are typically given by the data plane of a router, via forwarding tables, ACLs (access control lists), and packet rewriting. We do not specify here what these may be, or how they induce transition relations (however, apart from Section 7, our work is independent of such considerations). The box transitions arise from packets moving from an input port of a router, through its internal ports, and, possibly changed, on to an output port, following its set of rules.

In many switching networks only packet locations change, as there are no packet rewriting rules. That is, packet headers are not changed by any of the boxes in the network. Formally, for any box $b : K$ in the network we have:

$$h \mathbin{@} i \longrightarrow_{\mathrm{Box},\, b} h' \mathbin{@} i' \implies h' = h$$

for all $i, i' \in K$ and h, h'. This holds, for example, for the Singapore network considered in Section 7.

Next, packet exit ports j may be independent of their entry ports i. Formally, we additionally have for any box $b : K$ in the network:

$$h \mathbin{@} i_1 \longrightarrow_{\mathrm{Box},\, b} h' \mathbin{@} i' \iff h \mathbin{@} i_2 \longrightarrow_{\mathrm{Box},\, b} h' \mathbin{@} i'$$

for all $i_1, i_2, i' \in K$ and h, h'. This happens when the forwarding tables are "centrally located" and are used to route all incoming messages to exit ports, independently of their entry port, and when ACLs are given per exit port; the Singapore network again provides an example. In this case the the semantics can be equivalently described in terms of "rules" of the form $h \mapsto I$ where $I \subseteq K$, with one such rule for each h, namely the one where

$$I =_{\mathrm{def}} \{j \in K \mid \exists i \in K.\ h \mathbin{@} i \rightarrow_{\mathrm{Box},\, b} h \mathbin{@} j\}$$

With the box transition semantics available, we define the network semantics in terms of two transition relations between states: an *internal* one, moving to internal network nodes, and an *external* one, moving to external network nodes.

The set, $\mathrm{States}_{\mathcal{N}}$, of states of the network $\mathcal{N}$ consists of the *network-located packets*; these are pairs, written $h \mathbin{@} p$, of a packet h and a network location, that is, a network port p.

The *internal network transition relation*:

$$h \mathbin{@} n.i \longrightarrow_{\mathcal{N}} h' \mathbin{@} n'.i'$$

holds if, and only if, for some j, $h \mathbin{@} i \longrightarrow_{\beta(n)} h' \mathbin{@} j$ and $n.j \, \gamma \, n'.i'$. In some sense we are doing two transitions in one step in that a packet first follows a box transition at input port i to output port i' in the same box, and then moves to the input port j of the router to which i is connected (as specified by the port connection relation γ). A similar "short-circuiting" is done in the Header Space model [15].

The *external network transition relation*:

$$h \mathbin{@} n.i \twoheadrightarrow_{\mathcal{N}} h' \mathbin{@} n'.i'$$

holds iff $h \mathbin{@} i \longrightarrow_{\beta(n)} h' \mathbin{@} i'$ where $n.i'$ is an external port. We write $\rightsquigarrow_{\mathcal{N}}$ for the *total* network transition relation $\rightarrow_{\mathcal{N}} \cup \twoheadrightarrow_{\mathcal{N}}$.

Below, when they can be understood from the context, we generally omit network suffixes from $\mathrm{Node}_{\mathcal{N}}$, $\beta_{\mathcal{N}}$ and so on. However, network suffixes are essential when we describe transformations between networks.

The double packet/location nature of states gives the subject a special interest. Note that our states only keep track of the location of one packet, not of several as would be natural if we were interested in concurrent aspects of network operation. Our purpose, as elsewhere in the network verification literature, see, e.g., [33], is rather to verify properties such as reachability, concerning only one packet at a time.

A notion of *network slice* will prove important. In its most general form, a slice is a subset S of States, the set of states. We say that a slice is a *network invariant* if, in addition, we have:

$$h \mathbin{@} p \in S \wedge (h \mathbin{@} p \rightsquigarrow_{\mathcal{N}} h' \mathbin{@} p) \implies h' \mathbin{@} p' \in S$$

A less general, but useful case, already considered in [15], is where the slice is a product $H \times P$ of sets H and P of packets and ports.

2.3 A Network Logic

We next spell out a small natural modal logic for such networks that supports the application of the van Benthem-Hennessy-Milner principle while being enough to cover typical network requirements such as reachability relations between hosts.

We first assume given a collection of *packet formulas* α to specify properties of packets, together with a satisfaction relation

$$h \models \alpha$$

for them, where $h \in \mathrm{Pac}_{\mathcal{N}}$. For example such packet formulas may consist of wildcard expressions such as 101xxxxx for packets with (let's say) eight bit destination IP addresses that start with 101 as in Header Space Analysis [15].

One can then define a largely standard modal logic by the following grammar of $\mathcal{N}$*-formulas*:

$$\varphi ::= \alpha \mid \, @p \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \mathrm{F}\varphi$$

The modal worlds are the network states. The formula α expresses that α holds of the state packet; $@p$ expresses that the state location is p; $\Diamond\varphi$ expresses that one can reach a state where φ holds in zero or more internal steps from the current state; and $\mathrm{F}\varphi$ expresses that one can reach a state where φ holds by an external step from the current state. Other connectives are definable: the other propositional ones; the $\Box$ modality, expressing necessarily reaching, having taken zero or more internal steps; $\Diamond_{\mathrm{T}}\varphi$ abbreviating $\Diamond(\varphi \vee \mathrm{F}\varphi)$ and the corresponding $\Box_{\mathrm{T}}\varphi$; and $@p_1, \ldots, p_n$ (for $n > 0$) abbreviating $@p_1 \vee \ldots \vee @p_n$. More expressive logics can be obtained, for example, by adding fixed-points $\mu X.\varphi[X]$, but the simpler logic seems sufficiently expressive.

Given a network $\mathcal{N}$, and a semantics for it, we obtain a semantics for the logic, following the above ideas. As usual, we define a satisfaction relation:

$$h\,@\,p \models_{\mathcal{N}} \varphi$$

– that in network $\mathcal{N}$ the packet h satisfies the formula φ when located at port p — by structural induction on formulas. The clauses for the boolean formulas are as usual; the others are:

$$\begin{array}{lll}
h\,@\,p \models_{\mathcal{N}} \alpha & \iff_{\mathrm{def}} & h \models \alpha \\
h\,@\,p \models_{\mathcal{N}} @p' & \iff_{\mathrm{def}} & p = p' \\
h\,@\,p \models_{\mathcal{N}} \Diamond\varphi & \iff_{\mathrm{def}} & \exists h', p'.\, h\,@\,p \rightarrow_{\mathcal{N}}^{*} h'\,@\,p' \\
 & & \quad\wedge\; h'\,@\,p' \models \varphi \\
h\,@\,p \models_{\mathcal{N}} \mathrm{F}\varphi & \iff_{\mathrm{def}} & \exists h', p'.\, h\,@\,p \twoheadrightarrow_{\mathcal{N}} h'\,@\,p' \\
 & & \quad\wedge\; h'\,@\,p' \models \varphi
\end{array}$$

There are then two natural validity notions for a given network $\mathcal{N}$ (reflecting the double nature of states): one where a header validates a formula if the header satisfies the formula when located at any port; and the other where all headers satisfy the formula when located at any port:

$$h \models_{\mathcal{N}} \varphi \iff \forall p.\, h\,@\,p \models_{\mathcal{N}} \varphi$$

and

$$\models_{\mathcal{N}} \varphi \iff \forall h.\, h \models_{\mathcal{N}} \varphi$$

(There is also a third validity $p \models_{\mathcal{N}} \varphi$, but we do not know of a use for it.)

Let us give a few examples to illustrate the expressiveness of the logic. First,

$$\models_{\mathcal{N}} \alpha \wedge @p_1 \Rightarrow \Diamond(\,@p_2, p_3 \wedge \Diamond\mathrm{F}(\alpha' \wedge @p_4))$$

holds if, whenever a network packet satisfying α is at port p_1 then it can reach an external port p_4 via p_2 or p_3, being meanwhile transformed into a packet satisfying α'.

In the same vein, suppose a network slice S is defined by a boolean combination of atomic formulas, φ. Then the slice is a network invariant if, and only if,

$$\models_{\mathcal{N}} \varphi \Rightarrow \Box_{\mathrm{T}}\varphi$$

holds. Suppose that S' is another slice, defined by a boolean combination of atomic formulas, φ'. Then

$$\models_{\mathcal{N}} \varphi \Rightarrow \neg\Diamond_{\mathrm{T}}\varphi'$$

holds if the second slice cannot be accessed from the first. This models the standard notion of slicing (a form of network virtualization that requires isolation between slices) considered in [15, 17], but our formalism can express alternative notions.

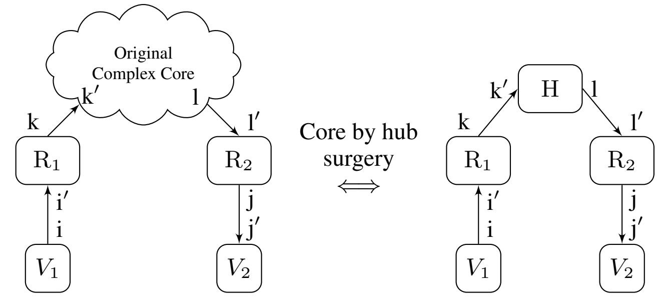

Figure 3. Replacing the core by a hub.

Continuing, suppose that p is an internal port. Then

$$\models_{\mathcal{N}} \alpha \wedge @p \;\Rightarrow\; \Box\,@p$$

holds if any packet with header satisfying α and reaching p is dropped (or loops through p forever).

Next,

$$h \models_{\mathcal{N}} @p \Rightarrow \Diamond(\neg\,@p \wedge \Diamond\,@p)$$

holds if any packet with header h, loops through p via some other port, returning to p with its header possibly being transformed. This is a generic loop, in the terminology of [15]. However, if one has an invariant α available, one can detect some infinite loops:

$$\models_{\mathcal{N}} \alpha \wedge @p \Rightarrow \Diamond(\neg\,@p \wedge \Diamond(\alpha \wedge @p))$$

holds if any packet with header satisfying α loops through p, via some other port, with its header, however transformed, still satisfying α.

2.4 A First Surgery

Our first example of a surgery shows how to replace a core by a hub, while preserving reachability formulas.

Example 1: Replacing a Core by a Hub

Suppose our network $\mathcal{N}$ consists of a "core" $\mathcal{N}_{\mathrm{core}}$ through which ToRs (or other boxes) communicate via a single link. Suppose too that packet headers are not changed by any of the boxes in the network. Then if the core is obstruction-free, in a suitable sense, reachability queries between network machines accessing the core via those boxes hold if, and only if, they hold for a greatly reduced network in which the core is replaced by a hub: see Figure 3.

As shown there, the core network $\mathcal{N}_{\mathrm{core}}$ has external ports k' and l (at nodes a and b, say). Fix a "traffic set" $\mathrm{T} \subseteq \mathrm{Pac}_{\mathcal{N}}$ of packet headers; typically this will be the packets addressed to the VMs handled by R_2. Then we assume the core is *obstruction-free for* T *between* $\mathrm{a.k}'$ *and* $\mathrm{b.l}$, i.e., that, for all $h \in \mathrm{T}$ we have:

$$h\,@\,\mathrm{a.k}' \longrightarrow^{*}_{\mathcal{N}_{\mathrm{core}}} h\,@\,\mathrm{b.l}$$

The network $\bar{\mathcal{N}}$ is the same as $\mathcal{N}$ except that the core has been replaced by a hub H with ports k' and l. Regarding its semantics we assume only that, for all h:

$$h\,@\,\mathrm{k}' \rightarrow_{\mathrm{H}} h\,@\,\mathrm{l}$$

In both cases we have boxes (ToR's, say) R_1 and R_2 (at nodes r_1 and r_2, say) to which (for example, virtual machines) V_1 and V_2 are connected, with ports as shown. Regarding logic, both networks have the same packet formulas, and the evident collections of port formulas.

Then, for any h and p, we have that the reachability formula φ_{reach}, namely:

$$\alpha_{\mathrm{T}} \wedge @\,\mathrm{r}_1.\mathrm{i}' \;\Rightarrow\; \Diamond\, @\,\mathrm{v}_2.\mathrm{j}'$$

is satisfied by $h\,@\,p$ in $\mathcal{N}$ if, and only if, it is satisfied by $h\,@\,p$ in $\bar{\mathcal{N}}$ (we are assuming that we have a packet formula α_{T} defining T, i.e., such that $\mathrm{T} = \{h \mid h \models \alpha_{\mathrm{T}}\}$). This is easily verified by direct consideration of the semantics of the formula in each of the two networks, coupled with the no-obstruction assumption. For either $\mathcal{N}$ or $\bar{\mathcal{N}}$, $h\,@\,p \models \varphi_{\mathrm{reach}}$ holds if, and only if, we have

$$h \in \mathrm{T} \,\wedge\, p = \mathrm{r}_1.\mathrm{i}' \wedge h\,@\,p \models \Diamond\, @\,\mathrm{v}_2.\mathrm{j}'$$

and we can then calculate, for $h \in \mathrm{T}$:

$$\begin{array}{ll}
h\,@\,\mathrm{r}_1.\mathrm{i}' \models_{\mathcal{N}} \Diamond\, @\,\mathrm{v}_2.\mathrm{j}' \iff & h\,@\,\mathrm{r}_1.\mathrm{i}' \rightarrow^*_{\mathcal{N}} h\,@\,\mathrm{v}_2.\mathrm{j}' \\
\iff & h\,@\,\mathrm{i}' \rightarrow_{\mathrm{R}_1} h\,@\,\mathrm{k} \,\wedge\, h\,@\,\mathrm{a.k}' \rightarrow^*_{\mathcal{N}_{\mathrm{core}}} h\,@\,\mathrm{b.l} \,\wedge \\
& \quad h\,@\,\mathrm{r}_2.\mathrm{l}' \rightarrow_{\mathrm{R}_2} h\,@\,\mathrm{v}_2.\mathrm{j}' \\
\iff & h\,@\,\mathrm{i}' \rightarrow_{\mathrm{R}_1} h\,@\,\mathrm{k} \,\wedge\, h\,@\,\mathrm{r}_2.\mathrm{l}' \rightarrow_{\mathrm{R}_2} h\,@\,\mathrm{v}_2.\mathrm{j} \\
\iff & h\,@\,\mathrm{i}' \rightarrow_{\mathrm{R}_1} h\,@\,\mathrm{k} \,\wedge\, h\,@\,\mathrm{k}' \rightarrow_{\mathrm{H}} h\,@\,\mathrm{l} \,\wedge \\
& \quad h\,@\,\mathrm{r}_2.\mathrm{l}' \rightarrow_{\mathrm{R}_2} h\,@\,\mathrm{v}_2.\mathrm{j}' \\
\iff & h\,@\,\mathrm{r}_1.\mathrm{i}' \rightarrow^*_{\bar{\mathcal{N}}} h\,@\,\mathrm{v}_2.\mathrm{j}' \\
\iff & h\,@\,\mathrm{r}_1.\mathrm{i}' \models_{\bar{\mathcal{N}}} \Diamond\, @\,\mathrm{v}_2.\mathrm{j}'
\end{array}$$

In the calculation we use the fact that the network does not alter packet headers and, in the third equivalence, the fact that $\mathcal{N}_{\mathrm{core}}$ is obstruction-free for T between $\mathrm{a.k}'$ and b.l.

3. Network Bisimulations

To show the validity of network transformations, our proof technique is to define bisimulations $\sim$ between two networks $\mathcal{N}$ and $\overline{\mathcal{N}}$ built using the same box signature, but possibly with different box semantics, for example between the networks on the left and right in Figure 2.

We take these to be relations between network-located packets which are both bisimulations between ${\longrightarrow_{\mathcal{N}}}^*$ and ${\longrightarrow_{\overline{\mathcal{N}}}}^*$ and bisimulations between $\twoheadrightarrow_{\mathcal{N}}$ and $\twoheadrightarrow_{\overline{\mathcal{N}}}$, each in the usual sense (note they are then also bisimulations between $\rightsquigarrow^*_{\mathcal{N}}$ and $\rightsquigarrow^*_{\overline{\mathcal{N}}}$). It is natural to consider, in particular, *one-step* bisimulations between $\mathcal{N}$ and $\overline{\mathcal{N}}$; these are bisimulations between both $\longrightarrow_{\mathcal{N}}$ and $\longrightarrow_{\overline{\mathcal{N}}}$ and $\twoheadrightarrow_{\mathcal{N}}$ and $\twoheadrightarrow_{\overline{\mathcal{N}}}$ (and so also between $\rightsquigarrow_{\mathcal{N}}$ and $\rightsquigarrow_{\overline{\mathcal{N}}}$).

One can form relations between states from relations $\sim_{\mathrm{Pa}}$ and $\sim_{\mathrm{Po}}$ between the respective sets of packets and ports by:

$$h\,@\,p \sim \overline{h}\,@\,\overline{p} \iff_{\mathrm{def}} h \sim_{\mathrm{Pa}} \overline{h} \,\wedge\, p \sim_{\mathrm{Po}} \overline{p}$$

Then, given a relation $\sim_{\mathrm{Po}}$ between ports, there is a largest relation $\sim_{\mathrm{Pa}}$ between packets which, so combined with $\sim_{\mathrm{Po}}$, forms a (one-step) network bisimulation between $\mathcal{N}$ and $\overline{\mathcal{N}}$. We remark that, while there certainly are largest such (one-step) bisimulations, they do not seem very useful. The bisimulations relative to a fixed port relation are sensitive to ports visited and so do prove useful; they are analogous to those considered in [4].

Given a bisimulation $\sim$ between $\mathcal{N}$ and $\overline{\mathcal{N}}$ we wish to transfer properties between the first network and the second, as is usual with bisimulations, so that proving a formula in the first network is equivalent to proving a transformed formula in the second network. As we have two logics here, speaking of different sets of packets and ports, we need to correlate formulas. To that end we define a correlation relation $\sim_{\mathrm{For}}$ between $\mathcal{N}$-formulas and $\overline{\mathcal{N}}$-formulas by taking $\varphi \sim_{\mathrm{For}} \overline{\varphi}$ to hold if, and only if, for all $h, p, \overline{h}, \overline{p}$ we have:

$$h\,@\,p \sim \overline{h}\,@\,\overline{p} \implies (h\,@\,p \models \varphi \iff \overline{h}\,@\,\overline{p} \models \overline{\varphi})$$

We then have the expected proposition expressing the Van Benthem-Hennessy-Milner principle for networks:

PROPOSITION 1. *The formula correlation relation $\sim_{\mathrm{For}}$ is closed under the logical connectives, that is the following implications hold:*

$$\bot \sim_{\mathrm{For}} \bot \qquad \frac{\varphi \sim_{\mathrm{For}} \overline{\varphi}}{\neg\varphi \sim_{\mathrm{For}} \neg\overline{\varphi}} \qquad \frac{\varphi \sim_{\mathrm{For}} \overline{\varphi} \quad \psi \sim_{\mathrm{For}} \overline{\psi}}{\varphi \wedge \psi \sim_{\mathrm{For}} \overline{\varphi} \wedge \overline{\psi}}$$

$$\frac{\varphi \sim_{\mathrm{For}} \overline{\varphi}}{\Diamond\varphi \sim_{\mathrm{For}} \Diamond\overline{\varphi}} \qquad \frac{\varphi \sim_{\mathrm{For}} \overline{\varphi}}{\mathrm{F}\varphi \sim_{\mathrm{For}} \mathrm{F}\overline{\varphi}}$$

PROOF. This is a standard proof. There are five cases:

1. As $\bot$ never holds, we have $\bot \sim_{\mathrm{For}} \bot$.
2. Suppose that $\varphi \sim_{\mathrm{For}} \overline{\varphi}$ and that $h\,@\,p \sim \overline{h}\,@\,\overline{p}$. Then $h\,@\,p \models_{\mathcal{N}} \neg\varphi$ holds iff $h\,@\,p \models_{\mathcal{N}} \varphi$ does not hold iff $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \overline{\varphi}$ does not hold (as we have $\varphi \sim_{\mathrm{For}} \overline{\varphi}$) iff $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \neg\overline{\varphi}$ holds.
3. Suppose that $\varphi \sim_{\mathrm{For}} \overline{\varphi}$ and that $\psi \sim_{\mathrm{For}} \overline{\psi}$ and $h\,@\,p \sim \overline{h}\,@\,\overline{p}$. Then $h\,@\,p \models_{\mathcal{N}} \varphi \wedge \psi$ holds iff $h\,@\,p \models_{\mathcal{N}} \varphi$ and $h\,@\,p \models_{\mathcal{N}} \psi$ both hold iff $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \overline{\varphi}$ and $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \overline{\psi}$ both hold (as we have $\varphi \sim_{\mathrm{For}} \overline{\varphi}$ and $\psi \sim_{\mathrm{For}} \overline{\psi}$) iff $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \overline{\varphi} \wedge \overline{\psi}$ holds.
4. Suppose that $\varphi \sim_{\mathrm{For}} \overline{\varphi}$ and that $h\,@\,p \sim \overline{h}\,@\,\overline{p}$. Assume that $h\,@\,p \models_{\mathcal{N}} \Diamond\varphi$ holds, in order to show that $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \Diamond\overline{\varphi}$ does. Note that then, for all h', p' such that $h\,@\,p \rightarrow^*_{\mathcal{N}} h'\,@\,p'$, $h'\,@\,p' \models_{\mathcal{N}} \varphi$ holds.

   Assuming given $\overline{h}', \overline{p}'$ such that $\overline{h}\,@\,\overline{p} \rightarrow^*_{\mathcal{N}} \overline{h}'\,@\,\overline{p}'$, we have to show $\overline{h}'\,@\,\overline{p}' \models_{\overline{\mathcal{N}}} \overline{\varphi}$ holds. As $\sim$ is a bisimulation between $\mathcal{N}$ and $\bar{\mathcal{N}}$ and $h\,@\,p \sim \overline{h}\,@\,\overline{p}$, there are h', p' such that $h\,@\,p \rightarrow^*_{\mathcal{N}} h'\,@\,p'$ and $h'\,@\,p' \sim \overline{h}\,@\,\overline{p}'$. By the assumption we then have $h'\,@\,p' \models_{\mathcal{N}} \varphi$, and so, as $\varphi \sim_{\mathrm{For}} \overline{\varphi}$, $\overline{h}'\,@\,\overline{p}' \models_{\overline{\mathcal{N}}} \overline{\varphi}$, holds as required.

   The proof that $h\,@\,p \models_{\mathcal{N}} \Diamond\varphi$ holds if $\overline{h}\,@\,\overline{p} \models_{\overline{\mathcal{N}}} \Diamond\overline{\varphi}$ does is similar.
5. This is similar to the previous case.

□

In case $\sim$ is built from relations $\sim_{\mathrm{Pa}}$ (on packets) and $\sim_{\mathrm{Po}}$ (on ports), as above, there are natural sufficient conditions for the correlations to hold between atomic formulas. Recall that atomic formulas are either of the form α (sets of packets) or $@\,p_1, \ldots, p_n$ (sets of ports). So, $\alpha \sim_{\mathrm{For}} \overline{\alpha}$ holds if

$$\forall h, \overline{h}.\, h \sim_{\mathrm{Pa}} \overline{h} \implies (h \models \alpha \iff \overline{h} \models \overline{\alpha})$$

and $@\,p_1, \ldots, p_m \sim_{\mathrm{For}} @\,\overline{p}_1, \ldots, \overline{p}_n$ holds if

$$\forall p, \overline{p}.\, p \sim_{\mathrm{Po}} \overline{p} \implies (p \in \{p_1, \ldots, p_m\} \iff \overline{p} \in \{\overline{p}_1, \ldots, \overline{p}_n\})$$

These conditions are also necessary, except in evident trivial cases.

We remark that there is no commitment to the above logic for purposes of network verification. Proposition 1 demonstrates a wide range of properties that can be transferred between bisimilar networks. However one is free to express such properties using whatever logical or automata-theoretic means one desires. For example one can translate our logic into Datalog, so one can use [20].

4. Network Surgery

We next give three examples of network transformations by various forms of surgery, whether at the levels of graphs, headers, or rules. The networks are related to their transforms by bisimulations; these, in their turn, allow application of appropriate versions of the Van Benthem-Hennesy-Milner principle, justified by Proposition 1. We also revisit Example 1, and show why the obstruction-freeness assumption made there does not justify a helpful bisimulation.

Our first example of a network transformation may appear too trivial to be useful (removing disconnected sets of nodes). How-

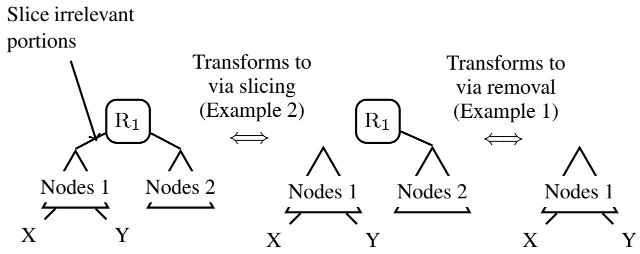
Slice irrelevant portions
Transforms to via slicing (Example 2)
$\Longleftrightarrow$
Transforms to via removal (Example 1)
$\Longleftrightarrow$
R$_1$
Nodes 1
Nodes 2
X
Y
R$_1$
Nodes 1
Nodes 2
X
Y
Nodes 1
X
Y

Figure 4. When considering the reachability between stations in the same subtree, the remainder of the network can be sliced away (Example 3), leaving a disconnected set of Nodes that can be removed by Example 2.

ever, as Figure 4 shows, it becomes effective after the following example, a more complex surgery called slicing.

Example 2: Removing Disconnected Components

When a network $\mathcal{N}$ consists of two disconnected subnetworks we can verify their properties separately. Suppose its nodes split into two disjoint subsets Node_1 and Node_2 which are disconnected in that for all ports $n_1.i_1$ and $n_2.i_2$:

$$n_1 \in \mathrm{Node}_1 \;\wedge\; n_1 \in \mathrm{Node}_2 \implies \neg\, n_1.i_1 \,\gamma\, n_2.i_2$$

Then $\mathcal{N}$ naturally splits into two subnetworks $\mathcal{N}_1$ and $\mathcal{N}_2$ with respective node sets Node_1 and Node_2 and with $\beta_{\mathcal{N}_i j}$ and $\gamma_{\mathcal{N}_j}$ being the evident restrictions of $\beta_\mathcal{N}$ and $\gamma_\mathcal{N}$, for $j = 1, 2$. Note that a port is internal (external) in an $\mathcal{N}_j$ iff it is in $\mathcal{N}$.

We then have that $\mathcal{N}$ is bisimilar to each of the $\mathcal{N}_j$. For $j = 1, 2$, one can take as (one-step) bisimulation relation

$$h\,@\,n.i \sim_j \overline{h}\,@\,\overline{n}.\overline{i} \iff_{\mathrm{def}} h\,@\,n.i = \overline{h}\,@\,\overline{n}.\overline{i} \wedge n \in \mathrm{Node}_j$$

The logic for $\mathcal{N}_j$ has as formulas those of $\mathcal{N}$ that only mention ports with nodes in Node_j. For such formulas, Proposition 1 tells us that that for all packets h and $\mathcal{N}_j$-ports $n.i$ we have:

$$h\,@\,n.i \models_\mathcal{N} \varphi \iff h\,@\,n.i \models_{\mathcal{N}_j} \varphi$$

We now introduce the more powerful slicing transformation illustrated in Figure 4. When verifying the reachability between certain pairs of stations such as X and Y, under certain conditions it is possible to slice away irrelevant portions of the network which we formalize as slicing.

Example 3: Slicing

Suppose we have a slice of the form $S = H \times P$ of a network $\mathcal{N}$, where H is a subset of the set of headers and P is a subset of the set of ports. We wish to define a corresponding slice network $\mathcal{N}|S$ obtained by cutting off all the ports not in P and restricting the header space to H. We restrict ourselves to the case where P is *connection invariant*, meaning that for all $p, p' \in \mathrm{Port}_\mathcal{N}$, we have:

$$p\gamma_\mathcal{N}p' \implies (p \in P \iff p' \in P)$$

Thus we are going to cut off the port at one end of a link if, and only if, we cut off the port at its other end; in other words, we are cutting off either external ports or whole links.

For slicing to work, we will need the slice to be a network invariant, as defined above. In the case of slices of the form $H \times P$ network invariance can be rephrased in terms of natural no-leakage and header invariance conditions on network boxes. For every box $b\!:\!K$ occurring in the network (i.e., in the range of β) say that a box port pair $(i, j) \in K^2$ is *P-leaky* if, and only if, for some $p \in P$ and $q \notin P$ we have:

$$\exists n.\, \beta(n) = b \wedge p = n.i \wedge n.j\,\gamma\, q$$

and say that a box port pair $(i, j) \in K^2$ is *P-connected* if, and only if, for some $p, q \in P$ we have:

$$\exists n.\, \beta(n) = b \wedge p = n.i \wedge (n.j\,\gamma\, q \vee (q = n.j \wedge q \text{ is external}))$$

Then the slice is a network invariant if, and only if, the following two conditions hold:

- *No leakage* For all $b\!:\!K$, $i \in K$, and $h \in H$:
$$h\,@\,i \longrightarrow_b h'\,@\,j \implies (i, j) \text{ is not } P\text{-leaky}$$
- *Header invariance* For all $b\!:\!K$, $h \in H$, and P-connected pairs (i, j):
$$h\,@\,i \longrightarrow_b h'\,@\,j \implies h' \in H$$

In the case where the slice is only on headers it evidently suffices to check invariance of H for all pairs of ports of all boxes occurring in the network.

To define the slice network $\mathcal{N}|S$, we begin by defining the slice signature $\mathrm{Box}|S$. For every box $b : K$ and every $I \subseteq K$, we assume available a $\mathrm{Box}|S$ box $b|I : I$; intuitively, $b|I$ is obtained from b by slicing off all of b's ports that are in $K\backslash I$. We take the header space to be H and its semantics is the restriction of that of b to H, that is that, for all $h, h' \in H$ and $i, i' \in I$ we have:

$$h\,@\,i \rightarrow_{\mathrm{Box}|S, b|I} h'\,@\,i' \iff h\,@\,i \rightarrow_{\mathrm{Box}, b} h'\,@\,i'$$

The slice network $\mathcal{N}|S$ is then obtained as follows: its node set is that of $\mathcal{N}$; its box assignment function is given by:

$$\beta_{\mathcal{N}|S}(n) = \beta_\mathcal{N}(n)|\{i \in K \mid n.i \in P\}$$

where $\beta(n) : K$; and its connection relation $\gamma_{\mathcal{N}|I}$ is the restriction of $\gamma_\mathcal{N}$ to $\mathcal{N}|I$'s ports, i.e., to P. We note that the set of ports of $\mathcal{N}|S$ is P, as anticipated. (For $n.i$ is such a port iff $i \in K$ and $n.i \in P$, where $\beta(n) : K$. But if $n.i \in P$ then $n.i \in \mathrm{Port}_\mathcal{N}$ and so $i \in K$; so we see that $n.i$ is indeed such a port iff it is in $\mathrm{Port}_{\mathcal{N}|S}$.) Note too that, as P is connection invariant, a port $p \in P$ is an internal (external) port of $\mathcal{N}|S$ if, and only if, it is of $\mathcal{N}$.

In the particular case when we are only slicing on headers, that is when P is the set of all ports, $\mathcal{N}|S$ can be taken to be $\mathcal{N}$, although one evidently still needs the restricted signature.

We now show that, restricted to S, the transition relations of $\mathcal{N}$ and $\mathcal{N}|S$ are the same.

LEMMA 1. *1. The internal (external) transition relation of $\mathcal{N}|S$ is contained in that of $\mathcal{N}$.*

2. The restriction of the internal (external) transition relation of $\mathcal{N}$ to the slice S is contained in that of $\mathcal{N}|S$.

PROOF.

1. Suppose, first, that $h\,@\,n.i \longrightarrow_{\mathcal{N}|S} h'\,@\,n'.i'$ in order to show that $h\,@\,n.i \longrightarrow_\mathcal{N} h'\,@\,n'.i'$. Then for some j we have $h\,@\,i \longrightarrow_{\mathrm{Box}|S,\, \beta(n)} h'\,@\,j$ and also $n.j\,\gamma_{\mathcal{N}|S}\, n'.i'$. We then have $h\,@\,i \longrightarrow_{\mathrm{Box},\, \beta(n)} h'\,@\,j$ and $n.j\,\gamma_\mathcal{N}\, n'.i'$ and so $h\,@\,n.i \longrightarrow_\mathcal{N} h'\,@\,n'.i'$.

   Next, suppose instead that $h\,@\,n.i \twoheadrightarrow_{\mathcal{N}|S} h'\,@\,n'.i'$. Then we have $h\,@\,i \longrightarrow_{\mathrm{Box}|S,\, \beta(n)} h'\,@\,i'$ and $n.i'$ is an external port of $\mathcal{N}|S$. We then have that $h\,@\,i \longrightarrow_{\mathrm{Box},\, \beta(n)} h'\,@\,i'$ and, by the above remark on external ports of $\mathcal{N}|S$, that $n.i'$ is an external port of $\mathcal{N}$, and so $h\,@\,n.i \twoheadrightarrow_\mathcal{N} h'\,@\,n'.i'$.

2. For the second part consider two states $h\,@\,n.i$ and $h'\,@\,n'.i'$ in S. Then suppose, first, that $h\,@\,n.i \longrightarrow_\mathcal{N} h'\,@\,n'.i'$ to show that $h\,@\,n.i \longrightarrow_{\mathcal{N}|S} h'\,@\,n'.i'$. Then $h\,@\,i \longrightarrow_{\mathrm{Box},\, \beta_\mathcal{N}(n)} h'\,@\,j$ and $n.j\,\gamma_\mathcal{N}\, n'.i'$, for some j.

   So, first, as $n.i \in P$ (since $h\,@\,n.i$ is in S), i is in the set $I =_{\mathrm{def}} \{i \in K \mid n.i \in P\}$ where $\beta_\mathcal{N}(n) : K$. Then, as $n'.i' \in P$ (since $h'\,@\,n'.i' \in S$) and as P is connection invariant and

$n.j\,\gamma_{\mathcal{N}}\,n'.i'$, we see that $n.j \in P$ and so that $j \in I$ also holds. It follows that $h\,@\,i \longrightarrow_{\mathrm{Box},\, \beta_{\mathcal{N}}(n)|I} h'\,@\,j$. Then, as $h, h' \in H$, we finally have $h\,@\,i \longrightarrow_{\mathrm{Box}|S,\, \beta_{\mathcal{N}|S}(n)} h'\,@\,j$.

Next as $n.j\,\gamma_{\mathcal{N}}\,n'.i'$ and $n.j, n'.i \in P$ we further have that $n.j\,\gamma_{\mathcal{N}|S}\,n'.i'$. Combining the last two facts, we have $h\,@\,n.i \longrightarrow_{\mathcal{N}|S} h'\,@\,n'.i'$, as required.

Next, suppose instead, that $h\,@\,i \longrightarrow_{\mathrm{Box},\, \beta_{\mathcal{N}}(n)} h'\,@\,i'$ and $n.i'$ is an external port of $\mathcal{N}$. Here, as $h\,@\,n.i$ and $h'\,@\,n'.i'$ are in S, $n.i$ and $n'.i'$ are in P; therefore, arguing as above, we see that $h\,@\,i \longrightarrow_{\mathrm{Box}|S,\, \beta_{\mathcal{N}|S}(n)} h'\,@\,i'$ holds. But $n.i'$ is an external port of $\mathcal{N}|S$ as it is an external port of $\mathcal{N}$. So we find that $h\,@\,n.i \twoheadrightarrow_{\mathcal{N}|S} h'\,@\,n'.i'$, concluding the proof.

□

Turning to logic, for the logic of $\mathcal{N}|S$, we take the $\mathcal{N}$-formulas that mention only ports of $\mathcal{N}|S$. There is a natural potential bisimulation $\sim_S$ between $\mathcal{N}$ and $\mathcal{N}|S$, defined by:

$$h\,@\,p \sim_S \overline{h}\,@\,\overline{p} \iff_{\mathrm{def}} h\,@\,p = \overline{h}\,@\,\overline{p} \in S$$

THEOREM 1. *The relation $\sim_S$ is a one-step bisimulation between $\rightarrow_{\mathcal{N}}$ and $\rightarrow_{\mathcal{N}|S}$ if, and only if, S is a network invariant of $\mathcal{N}$. In that case, for any $\mathcal{N}|S$ formula and any state $h\,@\,p \in S$ we have:*

$$h\,@\,p \models_{\mathcal{N}} \varphi \iff h\,@\,p \models_{\mathcal{N}|S} \varphi$$

PROOF. Suppose that S is an invariant. To show that $\sim_S$ is a bisimulation between the $\rightarrow_{\mathcal{N}}$ and $\rightarrow_{\mathcal{N}|S}$, we suppose that $h\,@\,p \sim_S \overline{h}\,@\,\overline{p}$ (i.e., that $h\,@\,p = \overline{h}\,@\,\overline{p} \in S$) and verify the usual two conditions.

First, if $h\,@\,p \rightarrow_{\mathcal{N}} h'\,@\,p'$ then, as S is an invariant, we have $h'\,@\,p' \in S$. So, applying Lemma 1, part 2, we have that $h\,@\,p \rightarrow_{\mathcal{N}|S} h'\,@\,p'$. As $h'\,@\,p' \sim_S h'\,@\,p'$ (as $h'\,@\,p' \in S$) we have verified the first condition. Second, if $h\,@\,p \rightarrow_{\mathcal{N}|S} h'\,@\,p'$ then $h'\,@\,p' \in S$ and so $h\,@\,p \rightarrow_{\mathcal{N}} h'\,@\,p'$, by Lemma 1, part 1. The second condition follows. One shows $\sim_S$ an external transition bisimulation similarly.

Conversely, suppose that $\sim_S$ is a one-step bisimulation between $\mathcal{N}$ and $\mathcal{N}|S$. To show S invariant, suppose that $h\,@\,p \in S$ and $h\,@\,p \rightsquigarrow_{\mathcal{N}} h'\,@\,p'$. Then $h\,@\,p \sim_S h\,@\,p$ and so, as $\sim_S$ is a $\rightsquigarrow$ bisimulation, we have that $h\,@\,p \rightsquigarrow_{\mathcal{N}|S} \overline{h}'\,@\,\overline{p}'$ for some $\overline{h}'\,@\,\overline{p}'$ with $h'\,@\,p' \sim_S \overline{h}'\,@\,\overline{p}'$, that is, with $h'\,@\,p' = \overline{h}'\,@\,\overline{p}' \in S$. Thus S is indeed an invariant.

The final part follows from the assumption that $\sim_S$ is a bisimulation by applying Proposition 1 and the remarks made immediately thereafter. □

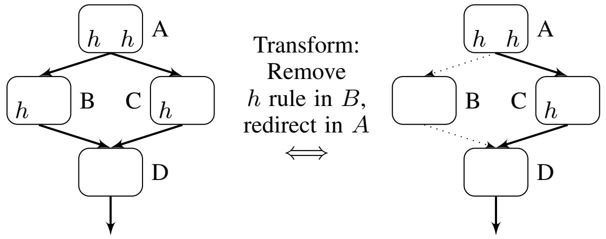

Figure 5. When considering forwarding header h, redirecting traffic from A to only use C and dropping the relevant rule in B does not impact the reachability of h

Our next surgery (redirection), formalized in Example 4, keeps the topology unchanged but drops rules. This, in turn, involves redirecting the traffic covered by the dropped rules. First, notice in Figure 5 that A, in Figure 5, forwards packets to header h to both B and C. This is done for good reasons in practice such as load balancing.

However, given that certain conditions hold, to compute the reachable set of packets addressed to h it suffices to redirect traffic at A to only go to C, and drop the rules dealing with this traffic from B. The advantage of this transformation is that it can be applied repeatedly to gradually drain the rules out of backup routers, to make a much simpler reduced network.

Example 4: Redirecting Traffic

Suppose we have a network $\mathcal{N}$ in which traffic $\mathrm{T} \subseteq \mathrm{Pac}_{\mathcal{N}}$ flows from a node a to a node b and then, without change, to a node d, but there is an alternative flow, in which the same traffic moves from a to yet another node c from which it also flows without change to d.

Then, as long as the box at d treats the traffic in the same way, by whichever of the two routes it arrives, we can transform the network by rerouting the traffic T through b through c instead, and removing relevant rules from the box at b. This results in another network $\overline{\mathcal{N}}$, with less rules.

While this transformation is clearly far from general, it seems widely applicable to data center switching networks because of the symmetry present there. In Section 6 below we point out how one could pick out candidate nodes a, b, c, etc., using symmetry considerations.

We next present the transformation and its background assumptions precisely. The nodes a, b, c, and d are supposed all distinct.

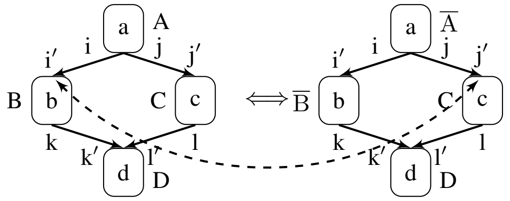

Figure 6. The formal setup for Example 4

Let the boxes at these nodes be $\mathrm{A} : \mathrm{K_A}$, $\mathrm{B} : \mathrm{K_B}$, $\mathrm{C} : \mathrm{K_C}$, and $\mathrm{D} : \mathrm{K_D}$, respectively. Suppose that a and b are connected by ports a.i and b.i′; that b and d are connected by ports b.k and d.k′; that a and c are connected by ports a.j and c.j′; and that c and d are connected by ports c.l and d.l′. Note that we then also have that i and j are distinct, as are i′ and k; j′ and l; and k′ and l′. This scheme is depicted in Figure 6.

The background assumptions are that for any $h \in \mathrm{T}$ the following hold:

$$h\,@\,\mathrm{i}' \longrightarrow_{\mathrm{B}} h'\,@\,q \iff h' = h \wedge q = \mathrm{k}$$

which says that T traffic entering at i′ is forwarded by B unchanged at k,

$$h\,@\,\mathrm{j}' \longrightarrow_{\mathrm{C}} h'\,@\,q \iff h' = h \wedge \ q = \mathrm{l}$$

which says that T traffic entering at j′ is forwarded by C unchanged at l, and

$$h\,@\,\mathrm{k}' \longrightarrow_{\mathrm{D}} h'\,@\,q \iff h\,@\,\mathrm{l}' \longrightarrow_{\mathrm{D}} h'\,@\,q$$

which says that D treats T traffic at k′ and l′ identically.

The network $\overline{\mathcal{N}}$ is obtained from $\mathcal{N}$ by replacing the boxes $\mathrm{A} : \mathrm{K_A}$ and $\mathrm{B} : \mathrm{K_B}$ by boxes $\overline{\mathrm{A}} : \mathrm{K_A}$ and $\overline{\mathrm{B}} : \mathrm{K_B}$. The transition rules of $\overline{\mathrm{A}}$ are the same as those of A except for transitions to the ports i or j, where we put instead:

$$h\,@\,q \rightarrow_{\overline{\mathrm{A}}} h'\,@\,\mathrm{i} \iff_{\mathrm{def}} h\,@\,q \rightarrow_{\mathrm{A}} h'\,@\,\mathrm{i} \wedge h' \notin \mathrm{T}$$

$$h\,@\,q \rightarrow_{\overline{\mathrm{A}}} h'\,@\,\mathrm{j} \iff_{\mathrm{def}} h\,@\,q \rightarrow_{\mathrm{A}} h'\,@\,\mathrm{j} \vee$$

$$(h\,@\,q \rightarrow_{\mathrm{A}} h'\,@\,\mathrm{i} \wedge h' \in \mathrm{T})$$

The transition rules of $\overline{\mathrm{B}}$ are the same as those for B except for transitions between ports i′ and k where we have:

$$h\,@\,\mathrm{i}' \rightarrow_{\overline{\mathrm{B}}} h'\,@\,\mathrm{k} \iff_{\mathrm{def}} h\,@\,\mathrm{i}' \rightarrow_{\mathrm{B}} h'\,@\,\mathrm{k} \wedge h \notin \mathrm{T}$$

Turning to the bisimulation relation $\sim$, we take $h\,@\,p \sim \overline{h}\,@\,\overline{p}$ to hold if, and only if, $h = \overline{h}$ and one of the following two conditions also hold:

- $\overline{p} = p$ and, if $h \in \mathrm{T}$, then $p \neq \mathrm{b.i}'$
- $h \in \mathrm{T}$ and one of the following two conditions hold:

$$\begin{array}{lcl} p = \mathrm{b.i}' & \wedge & \overline{p} = \mathrm{c.j}' \\ p = \mathrm{d.k}' & \wedge & \overline{p} = \mathrm{d.l}' \end{array}$$

Note that this bisimulation relation is *not* the composition of separate relations between packets and ports.

THEOREM 2. *The relation $\sim$ is a one-step bisimulation between $\longrightarrow_{\mathcal{N}}$ and $\longrightarrow_{\bar{\mathcal{N}}}$.*

PROOF. We first show $\sim$ a bisimulation of the $\longrightarrow$ relation. We consider all the cases when a relation $h\,@\,p \sim \overline{h}\,@\,\overline{p}$ holds, and verify the bisimulation condition in each. We always have $\overline{h} = h$.

Suppose first that $\overline{p} = p$. Then either $h \notin \mathrm{T}$ or else $p \neq \mathrm{b.i}'$. There are two cases:

- In the first case, we assume first that p is not a port of the node a. In this case, a $\rightarrow$-successor state of $h\,@\,p$ of either network cannot be of the form $h'\,@\,\mathrm{b.i}'$ and so is $\sim$-related to itself.

  Further, both networks have the same $\rightarrow$-successor states, as the relevant transitions are the same in both networks. This is evident in the case of the c and the d nodes where the corresponding boxes are the same in both networks. In the case of the node b, the boxes B and $\overline{\mathrm{B}}$ have the same transitions, except for those from i′; however in this case we then have $h \notin T$ and so B and $\overline{\mathrm{B}}$ have the same transitions from $h\,@\,\mathrm{i}'$.

  As the $\rightarrow$-successor states are the same in both networks and are $\sim$-related to themselves, the bisimilarity condition holds in this case.

- In the second case we assume that p has the form $a.q$, and consider the possible forms $h'\,@\,q'$ of the A and $\overline{\mathrm{A}}$ successors of $h\,@\,q$.

  These are the same if $h' \notin \mathrm{T}$ or else $q' \neq i$ and $q' \neq j$. The corresponding $\rightarrow$-successor states in the two networks are then equal and in the $\sim$ relation to themselves.

  Otherwise, $h' \in \mathrm{T}$ and $q' = i$ or $q' = j$, and we note:

  – Every A successor $h'\,@\,\mathrm{i}$ can be paired off with the $\overline{\mathrm{A}}$ successor $h'\,@\,\mathrm{j}$, and $h'\,@\,\mathrm{b.i}' \sim h'\,@\,\mathrm{c.j}'$.

  – Every A successor $h'\,@\,\mathrm{j}$ can be paired off with the $\overline{\mathrm{A}}$ successor $h'\,@\,\mathrm{j}$, and $h'\,@\,\mathrm{c.j}' \sim h'\,@\,\mathrm{c.j}'$; and this accounts for all the remaining $\overline{\mathrm{A}}$ successors of the form $h''\,@\,\mathrm{j}$ with $h'' \in \mathrm{T}$.

  – There are no $\overline{\mathrm{A}}$ successors of $h\,@\,q$ of the form $h'\,@\,i$.

  This verifies the bisimilarity condition in this case.

Suppose next that $\overline{p} \neq p$. Then $h \in \mathrm{T}$, and there are two cases:

- In the first case $p = \mathrm{b.i}'$ and $\overline{p} = \mathrm{c.j}'$. So, employing the first two background assumptions, the only possible $\rightarrow$-successor state of $\mathcal{N}$ is $h\,@\,\mathrm{d.k}'$, and only possible next such state of $\bar{\mathcal{N}}$ is $h\,@\,\mathrm{d.l}'$. As $h\,@\,\mathrm{d.k}' \sim h\,@\,\mathrm{d.l}'$, the bisimulation condition holds in this case.

- In the second case, $p = \mathrm{d.k}'$ and $\overline{p} = \mathrm{d.l}'$. By the third background assumption the $h'\,@\,q$ reachable from $h\,@\,\mathrm{k}'$ and those reachable from $h'\,@\,\mathrm{l}'$ by a D transition are the same, and so too, therefore are the states of the two networks reachable from $h\,@\,p$ and $h'\,@\,\overline{p}$ by a $\rightarrow$-transition.

  Further, for such a q, d.q will either be an external port or else be connected to a port other than b.i′ (as that port is connected to another port). So if $h'\,@\,p'$ is reachable by a $\rightarrow$ transition from $h\,@\,p$ (equivalently from $h\,@\,\overline{p}$) then we have $h'\,@\,p' \sim h'\,@\,p'$. The bisimulation condition therefore holds in this case too.

We next show that $\sim$ is a bisimulation of the $\twoheadrightarrow$ relation. The two networks have the same $\twoheadrightarrow$ relation. Also, $h\,@\,p \sim h\,@\,p$ whenever there is a $\twoheadrightarrow$ transition from $h\,@\,p$. The only case where $\sim$ relates two different states, one of which may make a $\twoheadrightarrow$ transition, is $h\,@\,\mathrm{d.k}' \sim h\,@\,\mathrm{d.l}'$, with $h \in \mathrm{T}$, and in that case one can make a $\twoheadrightarrow$ transition to a state $h'\,@\,p'$ iff the other can, as D treats T traffic at k′ and l′ identically. Putting these together, we see that $\sim$ is a bisimulation of $\twoheadrightarrow$. □

Turning to logic we take the logic of $\bar{\mathcal{N}}$ to be that of $\mathcal{N}$, and consider formula correlations. For packet header formulas we evidently always have

$$\alpha \sim_{\mathrm{For}} \alpha$$

For port formulas we assume available a boolean combination φ_T of packet header formulas defining T. We then have the following correlations:

$$@\,p \ \sim_{\mathrm{For}}\ @\,p$$

if p is not b.i′, c.j′, d.k′, or d.l′,

$$\neg\varphi_{\mathrm{T}} \wedge @\,p \ \sim_{\mathrm{For}}\ \neg\varphi_{\mathrm{T}} \wedge @\,p$$

if p is c.j′, d.k′, or d.l′,

$$\varphi_{\mathrm{T}} \wedge (\,@\,\mathrm{b.i}' \vee @\,\mathrm{c.j}') \sim_{\mathrm{For}}\ \varphi_{\mathrm{T}} \wedge @\,\mathrm{c.j}'$$

and

$$\varphi_{\mathrm{T}} \wedge (\,@\,\mathrm{d.k}' \vee @\,\mathrm{d.l}') \sim_{\mathrm{For}}\ \varphi_{\mathrm{T}} \wedge (\,@\,\mathrm{d.k}' \vee @\,\mathrm{d.l}')$$

So, in particular, if we are "away from the surgery" the same assertions hold. More precisely, if p is not b.i′, c.j′, d.k′, or d.l′, and φ does not mention any of these ports, then, for any h, we have:

$$h\,@\,p \models_{\mathcal{N}} \varphi \iff h\,@\,p \models_{\bar{\mathcal{N}}} \varphi$$

There are variations on this transformation. Regarding the definition of $\overline{\mathrm{A}}$, one can instead leave its transitions to i unchanged, when the proof that $\sim$ is a one-step bisimulation relation goes through unchanged. It may also happen that the transitions to j are unchanged, as the ones being added are already there. This is common in the core where ports such as i and j are treated symmetrically for the usual redundancy and performance reasons.

Next, let us assume, as is common in switching networks, that packets are forwarded without being transformed. We can then weaken the condition that D treats T traffic at k and l identically. Given a subset G of ports (to be "guarded"), we define a *forwarding relation* $\equiv_{\mathrm{TG}}$ on ports. It is the least relation such that $p \equiv_{\mathrm{TG}} q$ if, and only if, one of the following hold:

- $p = q$, or
- $p, q \notin \mathrm{G}$ and both of the following hold
  - $\forall h \in \mathrm{T}, p'.\ h\,@\,p \rightarrow_{\mathcal{N}} h\,@\,p' \Rightarrow \exists q'.\, h\,@\,q \rightarrow_{\mathcal{N}} h\,@\,q' \wedge p' \equiv_{\mathrm{TG}} q'$
  - $\forall h \in \mathrm{T}, q'.\, h\,@\,q \rightarrow_{\mathcal{N}} h\,@\,q' \Rightarrow \exists p'.\, h\,@\,p \rightarrow_{\mathcal{N}} h\,@\,p' \wedge p' \equiv_{\mathrm{TG}} q'$

Then $\equiv_{\mathrm{TG}}$ is an equivalence relation and if $p \in \mathrm{G}$ and $p \equiv_{\mathrm{TG}} q$ then $p = q$. The weaker condition on D is then that $\mathrm{d.k}' \equiv_{\mathrm{TG}} \mathrm{d.l}'$.

One can define a bisimulation exactly as before, except that one replaces the second clause in the case where $h \in \mathrm{T}$ (i.e., the one that $p = \mathrm{d.k}'$ and $\overline{p} = \mathrm{d.l}'$) by $p \equiv_{\mathrm{TG}} \overline{p}$. As before, the same assertions hold if we are away from the surgery, i.e., the ports $\mathrm{b.i}'$, $\mathrm{c.j}'$, or any ports in the $\equiv_{\mathrm{TG}}$-relation to a different port, and so, in particular, any port in $\overline{\mathrm{G}}$. As before, one may also leave the transitions to i of $\overline{\mathrm{A}}$ unchanged.

Returning to Example 1, let us see why there need not be a relevant bisimulation between the two networks, by which we mean a bisimulation allowing the two atomic formulas $@\,\mathrm{r}_1.\mathrm{i}'$ and $@\,\mathrm{v}_2.\mathrm{j}'$ to be transferred. The problem is that although a packet $h \in \mathrm{T}$ can go from $\mathrm{r}_1.\mathrm{i}'$ to $\mathrm{v}_2.\mathrm{j}'$ via the core, it might also enter the core and then go somewhere else and then be dropped. Now consider the following reachability formula (generally stronger than φ_{reach})

$$\alpha_{\mathrm{T}} \wedge @\,\mathrm{r}_1.\mathrm{i}' \;\Rightarrow\; \Box\Diamond @\,\mathrm{v}_2.\mathrm{j}'$$

This holds in $\bar{\mathcal{N}}$, but may fail to hold in $\mathcal{N}$. However, if that is the case, then, by Proposition 1, there can be no bisimulation between $\mathcal{N}$ and $\bar{\mathcal{N}}$ that allows the two atomic formulas to be transferred.

5. Composite Bisimulations

We now see how composite bisimulations can be obtained. These use "topological" relations between networks and their signatures to compose together bisimulation relations between the transition relations of boxes to form bisimulations between the networks. This contrasts with the bisimulations in Section 4 which are rather constructed in an "ad hoc" way, according to the need at hand.

Suppose we have two signatures Box and $\overline{\mathrm{Box}}$. Then a *signature relation* between the two signatures consists of:

- a *box relation* $\sim_{\mathrm{B}}$ between Box and $\overline{\mathrm{Box}}$, and
- for each so-related pair of boxes $b\!:\!K$, $\overline{b}\!:\!\overline{K}$, a *box port relation* $\sim_{\mathrm{Po}}^{b,\overline{b}} \subseteq K \times \overline{K}$ between their ports.

Suppose next that we have two networks $\mathcal{N}$ and $\overline{\mathcal{N}}$ built over the two signatures. Then a *topological relation* between $\mathcal{N}$ and $\overline{\mathcal{N}}$ consists of a signature relation between the two signatures, as above, together with a *node relation* $\sim_{\mathrm{N}}$ between their sets of nodes such that:

- if $n \sim_{\mathrm{N}} \overline{n}$ then $\beta_{\mathcal{N}}(n) \sim_{\mathrm{B}} \beta_{\overline{\mathcal{N}}}(\overline{n})$, and
- the *network port relation* $\sim_{\mathrm{Po}}$ between $\mathrm{Port}_{\mathcal{N}}$ and $\mathrm{Port}_{\overline{\mathcal{N}}}$ defined by:

  $$n.i \sim_{\mathrm{Po}} \overline{n}.\overline{i} \quad \Longleftrightarrow_{\mathrm{def}} \quad n \sim_{\mathrm{N}} \overline{n} \,\wedge\, i \sim_{\mathrm{Po}}^{\beta_{\mathcal{N}}(n),\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{i}$$

  is a bisimulation of γ by $\overline{\gamma}$. (Note that then if $n.i \sim_{\mathrm{Po}} \overline{n}.\overline{i}$ then $n.i$ is external iff $\overline{n}.\overline{i}$ is.)

Turning to the box transition relations , assume first that we have two header spaces $\mathrm{Pac}_{\mathcal{N}}$ and $\mathrm{Pac}_{\overline{\mathcal{N}}}$. Then a *packet header relation* is a relation $\sim_{\mathrm{Pa}}$ between the two header spaces, $\mathrm{Pac}_{\mathcal{N}}$ and $\mathrm{Pac}_{\overline{\mathcal{N}}}$. Assume next that we have a semantics for each of the signatures, that is, we have families of transition relations $\longrightarrow_{\mathcal{N},b}$ ($b \in \mathrm{Box}$) and $\longrightarrow_{\overline{\mathcal{N}},\overline{b}}$ ($\overline{b} \in \overline{\mathrm{Box}}$) for the two signatures, as considered above. Then, given a signature relation between the two signatures as above, a packet header relation $\sim_{\mathrm{Pa}}$ is a *signature bisimulation* between the two signatures if, and only if the *box-located packet relation* $\sim_{\mathrm{Blp}}^{b,\overline{b}}$, defined by:

$$h\,@\,i \;\sim_{\mathrm{Blp}}\; \overline{h}\,@\,\overline{i} \quad \Longleftrightarrow_{\mathrm{def}} \quad h \sim_{\mathrm{Pa}} \overline{h} \,\wedge\, i \sim_{\mathrm{Po}}^{b,\overline{b}} \overline{i}$$

is a bisimulation between $\longrightarrow_b$ and $\longrightarrow_{\overline{b}}$, whenever $b \sim_{\mathrm{B}} \overline{b}$.

Given a topological relation (comprising a signature relation and a node relation) and a packet relation as above, we have a relation $\sim_{\mathrm{Pa}}$ between the two sets of packets and a relation $\sim_{\mathrm{Po}}$ between the two sets of ports. So we can define a relation $\sim_{\mathrm{Net}}$ between the corresponding located packet sets as remarked in Section 3, i.e., by putting:

$$h\,@\,p \;\sim_{\mathrm{Net}}\; \overline{h}\,@\,\overline{p} \quad \Longleftrightarrow_{\mathrm{def}} \quad h \sim_{\mathrm{Pa}} \overline{h} \,\wedge\, p \sim_{\mathrm{Po}} \overline{p}$$

We then have:

PROPOSITION 2. *If $\sim_{\mathrm{Pa}}$ is a signature bisimulation, then $\sim_{\mathrm{Net}}$ is a one-step bisimulation relation between $\mathcal{N}$ and $\overline{\mathcal{N}}$.*

PROOF. Assume $\sim_{\mathrm{Pa}}$ is a signature bisimulation. We give the first halves of the proofs that $\sim_{\mathrm{Net}}$ is an internal transition relation bisimulation and that it is an external one; the second halves are similar.

Suppose $h\,@\,n.i \sim_{\mathrm{Net}} \overline{h}\,@\,\overline{n}.\overline{i}$. Then $h \sim_{\mathrm{Pa}} \overline{h}$, $n \sim_{\mathrm{N}} \overline{n}$, and $i \sim_{\mathrm{Po}}^{\beta_{\mathcal{N}}(n),\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{i}$, and so $h\,@\,i \sim_{\mathrm{Blp}}^{\beta_{\mathcal{N}}(n),\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}\,@\,\overline{i}$.

If, first, $h\,@\,n.i \rightarrow_{\mathcal{N}} h'\,@\,n'.i'$, then $h\,@\,i \rightarrow_{\beta_{\mathcal{N}}(n)} h'\,@\,j$ and $n.j\,\gamma\,n'.i'$, for some j. As $\sim_{\mathrm{Blp}}$ is a bisimulation of $\rightarrow_{\beta_{\mathcal{N}}(n)}$ by $\rightarrow_{\beta_{\overline{\mathcal{N}}}(\overline{n})}$, there are $\overline{h}', \overline{j}$ such that $\overline{h}\,@\,\overline{i} \rightarrow_{\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}'\,@\,\overline{j}$ and $h'\,@\,j \sim_{\mathrm{Blp}}^{\beta_{\mathcal{N}}(n),\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}'\,@\,\overline{j}$ (and so $h' \sim_{\mathrm{Pa}} \overline{h}'$ and $j \sim_{\mathrm{Po}}^{\beta_{\mathcal{N}}(n),\beta_{\mathcal{N}}(\overline{n})} \overline{j}$). Next, as $n \sim_{\mathrm{N}} \overline{n}$ and $j \sim_{\mathrm{Po}}^{\beta_{\mathcal{N}}(n),\beta_{\mathcal{N}}(\overline{n})} \overline{j}$, $n.j \sim_{\mathrm{Po}} \overline{n}.\overline{j}$. So as $n.j\,\gamma\,n'.i'$ and $\sim_{\mathrm{Po}}$ is a bisimulation of γ by $\overline{\gamma}$, there are $\overline{n}'$ and $\overline{j}'$ such that $\overline{n}.\overline{j}\;\overline{\gamma}\;\overline{n}'.\overline{j}'$ and $n'.i' \sim_{\mathrm{Po}} \overline{n}'.\overline{j}'$.

Then, as $\overline{h}\,@\,\overline{i} \rightarrow_{\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}'\,@\,\overline{j}$ and $\overline{n}.\overline{j}\;\overline{\gamma}\;\overline{n}'.\overline{j}'$, we have that $\overline{h}\,@\,\overline{n}.\overline{i} \rightarrow_{\mathcal{N}} \overline{h}'\,@\,\overline{n}'.\overline{i}'$. Finally, as $h' \sim_{\mathrm{Pa}} \overline{h}'$ and $n'.i' \sim_{\mathrm{Po}} \overline{n}'.\overline{i}'$ we also have $h'\,@\,n'.i' \sim_{\mathrm{Net}} \overline{h}'\,@\,\overline{n}'.\overline{i}'$, as required.

If, instead, $h\,@\,n.i \twoheadrightarrow_{\mathcal{N}} h'\,@\,n'.i'$, then $h\,@\,i \rightarrow_{\beta_{\mathcal{N}}(n)} h'\,@\,i'$ and $n.i'$ is external. So, as $\sim_{\mathrm{Blp}}$ is a bisimulation of $\rightarrow_{\beta_{\mathcal{N}}(n)}$ by $\rightarrow_{\beta_{\overline{\mathcal{N}}}(\overline{n})}$, there are $\overline{h}', \overline{i}'$ such that $\overline{h}\,@\,\overline{i} \rightarrow_{\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}'\,@\,\overline{i}'$ and $h'\,@\,i' \sim_{\mathrm{Blp}}^{\beta_{\mathcal{N}}(n),\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}'\,@\,\overline{i}'$ (and so $h' \sim_{\mathrm{Pa}} \overline{h}'$ and $i' \sim_{\mathrm{Po}}^{\beta_{\mathcal{N}}(n),\beta_{\mathcal{N}}(\overline{n})} \overline{i}'$). Next, as $n \sim_{\mathrm{N}} \overline{n}$ and $i' \sim_{\mathrm{Po}}^{\beta_{\mathcal{N}}(n),\beta_{\mathcal{N}}(\overline{n})} \overline{i}'$, $n.i' \sim_{\mathrm{Po}} \overline{n}.\overline{i}'$. But then as $n.i'$ is external, we see that $\overline{n}.\overline{i}'$ is external too.

So, as $\overline{h}\,@\,\overline{i} \rightarrow_{\beta_{\overline{\mathcal{N}}}(\overline{n})} \overline{h}'\,@\,\overline{i}'$, we have $\overline{h}\,@\,\overline{n}.\overline{i} \twoheadrightarrow_{\overline{\mathcal{N}}} \overline{h}'\,@\,\overline{n}.\overline{i}'$ Finally, as $h' \sim_{\mathrm{Pa}} \overline{h}'$ and $n'.i' \sim_{\mathrm{Po}} \overline{n}'.\overline{i}'$ we also have that $h'\,@\,n'.i' \sim_{\mathrm{Net}} \overline{h}'\,@\,\overline{n}'.\overline{i}'$, as required. □

6. Network Symmetries and Merging Headers

We use composite symmetries in two ways. The first is to define various notions of symmetries, particularly the topological ones that typically hold in data centers, and then show how local symmetries can be found and also how to divide out networks by symmetry groups. The second is to construct equivalencies between packet headers that allow us to replace assertions about one header by an assertion about an equivalent one, or to go further and replace headers by their equivalence classes.

Example 5: Network Symmetry

At the most general level, a *symmetry* of a network $\mathcal{N}$ is a permutation $\pi_{\mathcal{N}}$ of network states which preserves and reflects both $\rightarrow_{\mathcal{N}}^*$ and $\twoheadrightarrow_{\mathcal{N}}^*$. That is, for all network states $h\,@\,p$ and $h'\,@\,p'$, we have:

$$h\,@\,p \longrightarrow_{\mathcal{N}}^* h'\,@\,p' \iff \pi_{\mathcal{N}}(h\,@\,p) \longrightarrow_{\mathcal{N}}^* \pi_{\mathcal{N}}(h'\,@\,p')$$

and similarly for $\twoheadrightarrow_{\mathcal{N}}^*$. More strictly, a *one step symmetry* of $\mathcal{N}$ is required to preserve and reflect $\rightarrow_{\mathcal{N}}$ and and $\twoheadrightarrow_{\mathcal{N}}^*$. One could also require that the permutation is composed from separate permutations of the headers and the network ports (note that if the header space is finite, then the reflection condition is redundant).

The network structure provides a good source of exploitable symmetries, and so we focus on *composite symmetries* which are simply those composite bisimulations between a network and itself where all the relations are bijections. (We note in passing the evident fact that all these various classes of symmetries form groups.)

Let us spell this out in functional terms. Assume given a signature Box. Then a *signature symmetry* over Box consists of:

- a *box permutation* $\pi_{\mathrm{Bo}} : \mathrm{Box} \cong \mathrm{Box}$, and
- for all boxes $b : K$, a *box port bijection* $\pi^b_{\mathrm{Po}} : K \cong \overline{K}$, where $\pi_{\mathrm{Bo}}(b) : \overline{K}$,

Next, given a network $\mathcal{N}$ over Box, a *topological symmetry* of $\mathcal{N}$ is a signature symmetry, together with a *node permutation* $\pi_{No} : \mathrm{Node} \cong \mathrm{Node}$ such that:

- For all n, we have $\pi_{\mathrm{Bo}}(\beta(n)) = \beta(\pi_{No}(n))$, and
- For all ports $n.i$ and $n^+.i^+$, we have:

$$n.i\,\gamma\,n^+.i^+ \iff \pi_{\mathrm{Po}}(n.i)\;\gamma\;\pi_{\mathrm{Po}}(n^+.i^+)$$

where the *port bijection* $\pi_{\mathrm{Po}} : \mathrm{Port} \cong \mathrm{Port}$ is defined by:

$$\pi_{\mathrm{Po}}(n.i) = \pi_{No}(n).\pi^{\beta(n)}_{\mathrm{Po}}(i) \quad (n.i \in \mathrm{Port})$$

We can associate a graph $\Gamma_{\mathcal{N}}$ to every network $\mathcal{N}$. It has nodes those of the network and has relation $R_{\mathcal{N}}$ where:

$$R_{\mathcal{N}}(n, n') \iff_{\mathrm{def}} \exists n.i, n'.i' \in \mathrm{Port}.\, n.i\,\gamma\,n'.i'$$

Then every topological symmetry π_{No} is a symmetry of $\Gamma_{\mathcal{N}}$.

Turning to the semantics, suppose we have a header space $\mathrm{Pac}_{\mathcal{N}}$, and families of transition relations $\longrightarrow_{\mathcal{N},b}$ ($b \in \mathrm{Box}$), as usual. Then, given a signature symmetry as above, a *packet bijection*

$$\pi_{\mathrm{Pa}} : \mathrm{Pac}_{\mathcal{N}} \cong \mathrm{Pac}_{\mathcal{N}}$$

is a *symmetry of the signature semantics*, if, for all $h, h', b : K$, and all $i, i' \in K$, we have:

$$h\,@\,i \to_b h'\,@\,i' \iff \pi_{\mathrm{Pa}}(h)\,@\,\pi^b_{\mathrm{Po}}(i) \to_{\pi_{\mathrm{Bo}}(b)} \pi_{\mathrm{Pa}}(h')\,@\,\pi^b_{\mathrm{Po}}(i')$$

Figure 2 provides an example of a topological symmetry where the boxes R_3 and R_4, and their nodes would be permuted, but all others would be fixed, and where the ports are permuted as shown. If, further, the two boxes contained the same rules (modulo box port bijections) then the identity packet bijection would provide a signature semantics symmetry.

Given a topological symmetry π_{No} (comprising a signature symmetry and a node permutation) and a packet bijection π_{Pa} as above, we can define a bijection of the located packet set by:

$$\pi_{Net}(h\,@\,n.i) =_{\mathrm{def}} \pi_{\mathrm{Pa}}(h)\,@\,\pi_{\mathrm{Po}}(n.i)$$

and we have the following corollary of Proposition 2:

COROLLARY 1. *If π_{Pa} is a symmetry of the signature semantics, then π_{Net} is a one-step symmetry of $\mathcal{N}$.*

Turning to the logic, assume that for each packet formula α there is a formula α_π such that:

$$h \models \alpha_\pi \iff \pi^{-1}_{\mathrm{Pa}}(h) \models \alpha$$

We then obtain an evident homomorphic definition of φ_π for $\mathcal{N}$-formulas φ, where:

$$(@p)_\pi = @\,\pi_{\mathrm{Po}}(p) \quad (\neg\varphi)_\pi = \neg\varphi_\pi \quad (\varphi \wedge \psi)_\pi = \varphi_\pi \wedge \psi_\pi$$

$$(\Diamond\varphi)_\pi = \Diamond\varphi_\pi \quad (\mathrm{F}\varphi)_\pi = \mathrm{F}\varphi_\pi$$

Proposition 1 then tells us, that, for any $\mathcal{N}$-formula φ, we have:

$$h\,@\,p \models \varphi \iff \pi_{\mathrm{Pa}}(h)\,@\,\pi_{\mathrm{Po}}(p) \models \varphi_\pi$$

We can use symmetry considerations to pick out candidates for traffic-redirection surgery. We would like to find two symmetrically placed nodes b and c which are candidates for redirecting traffic (from b to c), so wish to find a box symmetry π_{Bo} and an associated topological symmetry π_{No} which switches a and b. Let us say that the symmetry is *local* if π_{No} leaves all the other nodes invariant. For such a local symmetry a and b will have the same $\Gamma_{\mathcal{N}}$ neighbors. If there is only one link between any two nodes, as is common, the signature symmetry is also then determined.

So the suggestion is to look for two nodes with the same neighbors, and use the node connections to determine the port correlations between them (two ports are correlated if they link to the same neighbor). One can then choose pairs of such ports (other than any ports connecting a and b) to search for candidates for i′ and j′, or for k and l, for traffic redirection.

Figure 2 again provides an example. The two required nodes are those corresponding to the two boxes R_3 and R_4.

We next sketch how to quotient a network $\mathcal{N}$ by a group of topological symmetries. In this way we may be able to "slim" fat trees, as discussed above. First, let us briefly recall the relevant background material (and see, e.g. [3, §17]). An *action* of a group G on a set X is a map $\cdot : G \times X \to X$, such that the following two equations hold:

$$(g' \cdot g) \cdot x = g' \cdot (g \cdot x) \qquad e \cdot x = x$$

The *orbit* equivalence relation is then defined on X by:

$$x \sim_G y \iff_{\mathrm{def}} \exists g \in G.\, g \cdot x = y$$

and we write X/G for the set of equivalence classes $[x]$ of elements x of X under this equivalence relation.

Under componentwise composition, the topological symmetries $(\pi_{\mathrm{Bo}}, \pi_{\mathrm{Po}}, \pi_{No})$ form a group $\mathrm{Top}_{\mathcal{N}}$, with three actions: on Box; on $\mathrm{BoxPort} =_{\mathrm{def}} \{b.i \mid b : K, i \in K\}$; and on Node, where:

$$\begin{array}{lll} (\pi_{\mathrm{Bo}}, \pi_{\mathrm{Po}}, \pi_{No}) \cdot b & =_{\mathrm{def}} & \pi_{\mathrm{Bo}}(b) \\ (\pi_{\mathrm{Bo}}, \pi_{\mathrm{Po}}, \pi_{No}) \cdot (b.i) & =_{\mathrm{def}} & \pi_{\mathrm{Bo}}(b).\pi^b_{\mathrm{Po}}(i) \\ (\pi_{\mathrm{Bo}}, \pi_{\mathrm{Po}}\pi_{No}) \cdot n & =_{\mathrm{def}} & \pi_{No}(n) \end{array}$$

Let G be a subgroup of $\mathrm{Top}_{\mathcal{N}}$. Let Box/G, $\mathrm{BoxPort}/G$, and Node/G, be the collections of orbit equivalence classes for each of its three actions, inherited from $\mathrm{Top}_{\mathcal{N}}$. We define a signature on Box/G by setting $[b] : \{[b.i] \mid i \in K\}$, for each $b : K$ (ignoring that the $[b.i]$ are not natural numbers). Then we can define a quotient network $\mathcal{N}/G$ over the signature. It has node set Node/G, box assignment function $\beta_{\mathcal{N}/G}$, where

$$\beta_{\mathcal{N}/G}([n]) =_{\mathrm{def}} [\beta_{\mathcal{N}}(n)]$$

and connection relation $\gamma_{\mathcal{N}/G}$ where

$$\begin{array}{lll} m.q\,\gamma_{\mathcal{N}/G}\,m'.q' & \iff_{\mathrm{def}} & \exists\, n, n', b, b', i, i'.\, n.i\,\gamma_{\mathcal{N}}\,n'.i' \wedge \\ & & \beta_{\mathcal{N}}(n) = b \,\wedge\, \beta_{\mathcal{N}}(n') = b' \wedge \\ & & m = [n] \wedge m' = [n'] \wedge \\ & & q = [b.i] \,\wedge\, q' = [b.i'] \end{array}$$

Assume next that the identity packet permutation is a symmetry of the signature semantics, given any signature component of any element of G. Then we can define a transition relation for Box/G, keeping the packet set the same as that of $\mathcal{N}$, by:

$$\begin{array}{lll} h\,@\,q \to_c h'\,@\,q' & \iff_{\mathrm{def}} & \exists b, i, i'.\, h\,@\,i \to_b h'\,@\,i' \wedge \\ & & c = [b] \,\wedge\, q = [b.i] \,\wedge\, q' = [b.i'] \end{array}$$

So we have defined the syntax and semantics of the quotient network $\mathcal{N}/G$, as desired.

There is a one-step bisimulation between $\mathcal{N}$ and $\mathcal{N}/G$, formed by combining the identity relation on packets and the relation $\sim_{\mathrm{Po}}$ on ports, where:

$$n.i \sim_{\mathrm{Po}} [\overline{n}].[\overline{n}.\overline{i}] \iff_{\mathrm{def}} n \in [\overline{n}] \wedge n.i \in [\overline{n}.\overline{i}]$$

Finally, assuming the evident logic for $\mathcal{N}/G$, we have:

$$@\,p_1, \ldots, p_m \sim_{\mathrm{For}} [p]$$

if $[p] = \{p_1, \ldots, p_m\}$.

Figure 2 again provides an example, assuming it depicts a composite symmetry as discussed above. The permutation group G is that generated by the symmetry, when, for example, the box orbit equivalence classes would be singletons except for $\{\mathrm{R3}, \mathrm{R4}\}$. The right-hand-side of the figure depicts the net after division by the symmetry group, with, for example, $\overline{\mathrm{R3}}$ depicting $\{\mathrm{R3}, \mathrm{R4}\}$. In general, one might have a number of such pairs of symmetrically placed boxes, when G would be the group generated by the various pair symmetries.

Example 6: Merging Headers

Suppose we have a signature Box with semantics given as above. Then a *binary (signature) invariant* is a bisimulation $\sim_{\mathrm{Pa}}$ between the signature and itself. Spelling this out, what is required is that $\sim_{\mathrm{Pa}}$ is a relation on the set of packet headers such that, for any h, $\overline{h}$, if $h \sim_{\mathrm{Pa}} \overline{h}$ then, for all boxes $b : K$ and $i, i' \in K$:

$$(i)\ h\,@\,i \longrightarrow_b h'\,@\,i' \implies \exists \overline{h}'.\, h' \sim_{\mathrm{Pa}} \overline{h}' \wedge \overline{h}\,@\,i \longrightarrow_b \overline{h}'\,@\,i'$$

holds for all h', as does

$$(ii)\ \overline{h}\,@\,i \longrightarrow_b \overline{h}'\,@\,i' \implies \exists h'.\, h' \sim_{\mathrm{Pa}} \overline{h}' \wedge h\,@\,i \longrightarrow_b h'\,@\,i'$$

for all $\overline{h}'$. Taking the other relations to be the relevant identity relations, Proposition 2 applies, and one obtains a one-step bisimulation of $\mathcal{N}$ by itself.

Proposition 1 then tells us that for any formula φ and headers $h \sim_{\mathrm{Pac}_{\mathcal{N}}} \overline{h}$ we have:

$$h \models_{\mathcal{N}} \varphi \iff \overline{h} \models_{\mathcal{N}} \varphi$$

provided φ is built, using the connectives, from formulas of the form $@\,p$ or formulas that are $\sim_{\mathrm{Pa}}$*-invariant*, by which is meant that

$$\forall h, \overline{h}.\, h \sim_{\mathrm{Pa}} \overline{h} \implies (h \models_{\mathcal{N}} \psi \iff \overline{h} \models_{\mathcal{N}} \psi)$$

holds, and which are boolean combinations ψ of basic formulas.

We can go further, and identify equivalent packet headers. First, we can assume that $\sim_{\mathrm{Pa}}$ is a partial equivalence relation (i.e., that it is transitive and symmetric, but not necessarily reflexive); for if it is not, one need only consider its transitive symmetric closure. The cases where reflexivity holds are given by the domain of $\sim_{\mathrm{Pa}}$, $H =_{\mathrm{def}} \{h \mid h \sim_{\mathrm{Pa}} h\}$ (and $H \times \mathrm{Port}$ is easily seen to be a network invariant). Restricted to its domain, $\sim_{\mathrm{Pa}}$ is an equivalence relation and we can change the set of packet headers to its set of equivalence classes. That is, we change the header space to:

$$\mathrm{Pac}_{\mathcal{N}}/{\sim_{\mathrm{Pa}}} =_{\mathrm{def}} \{[h] \mid h \in H\}$$

Having done, so we can define a new semantics for boxes, where:

$$[h]\,@\,i \longrightarrow_b [h']\,@\,i' \iff_{\mathrm{def}} h\,@\,i \longrightarrow_b h'\,@\,i'$$

Then we can define the network $\mathcal{N}/\sim_{\mathrm{Pa}}$; this is the same as $\mathcal{N}$, except for the above changes in packet headers and box semantics. As may be expected, $\mathcal{N}$ and $\mathcal{N}/{\sim_{\mathrm{Pa}}}$ are one-step bisimilar, taking the packet header bisimulation $\in_{\mathrm{Pac}_{\mathcal{N}}}$ to be membership:

$$h \in_{\mathrm{Pac}_{\mathcal{N}}} [\overline{h}] \iff_{\mathrm{def}} h \in [\overline{h}] \quad (\iff h \sim_{\mathrm{Pa}} \overline{h})$$

and the other relations to be the relevant diagonals.

Turning to logic, we take the basic formulas of $\mathcal{N}/\sim_{\mathrm{Pa}}$ to be the $\sim_{\mathrm{Pa}}$-invariant boolean combinations ψ of basic formulas and set:

$$[h] \models \psi \iff h \models \psi$$

Then, applying Proposition 1, we find that, for any $\mathcal{N}/\sim_{\mathrm{Pa}}$ formula φ, and any $h \in H$:

$$h \models_{\mathcal{N}} \varphi \iff [h] \models_{\mathcal{N}/\sim_{\mathrm{Pa}}} \varphi$$

The maximal signature bisimulation provides a natural choice of binary invariant; it is automatically an equivalence relation. In the case where the signature semantics does not change packet headers, that is where packet headers are forwarded unchanged, one can give a more explicit form of the maximal signature bisimulation, first (implicitly) considered by Yang and Lam [30] in the particular case where the output ports of the signature transition relations do not depend on the input ports.

To see this, first note that in the case where packet headers are forwarded unchanged, $\sim_{\mathrm{Pa}}$ is a signature invariant if, and only if, whenever $h \sim_{\mathrm{Pa}} \overline{h}$ then, for all boxes $b : K$ and $i, i' \in K$, we have:

$$h\,@\,i \to_b h\,@\,i' \iff \overline{h}\,@\,i \to_b \overline{h}\,@\,i'$$

Next, identifying predicates on headers with sets of headers, given a set $\mathcal{H}$ of predicates, one can define an equivalence relation on headers by taking $h \sim_{\mathcal{H}} \overline{h}$ to hold if, and only if, for all $H \in \mathcal{H}$ we have:

$$h \in H \iff \overline{h} \in H$$

Then Yang and Lam's *atomic predicates* are the equivalence classes of this relation. Taking $\mathcal{H}$ to be the sets $\{h \mid h\,@\,i \to_b h\,@\,i'\}$, where $b : K$ and $i, i' \in K$, one obtains an equivalence relation $\equiv_{\mathrm{YL}}$ slightly generalising that of Yang and Lam, which we therefore call *Yang-Lam equivalence*.

Yang and Lam demonstrated that impressive efficiencies in verification could be achieved by replacing packet headers by (representations of) their equivalence classes (as discussed above) since there can be many fewer equivalence classes than headers. in the networks they considered. As we shall see this also holds for the Singapore network.

Comparing the above reformulation of signature invariants with Yang-Lam equivalence we see that $\sim_{\mathrm{Pa}}$ is a signature invariant if, and only if, $\sim_{\mathrm{Pa}} \subseteq \equiv_{\mathrm{YL}}$. This proves the first part of the following theorem:

THEOREM 3. *Under the assumption that the signature semantics does not change packet headers, the following are the same:*

- *the maximal signature bisimulation*
- *Yang-Lam equivalence*

Further, in case all boxes occur in $\mathcal{N}$ (i.e., are in the range of $\beta_{\mathcal{N}}$), then they are also the same as:

- *the maximal relation on headers which, when combined with the identity relation on ports (as in Section 3), forms a bisimulation.*

PROOF.

For the proof of the second part of the theorem, the condition that a packet header relation $\sim_{\mathrm{Pa}}$ forms a bisimulation when combined with the identity relation on ports is clearly equivalent to asking that, for all nodes n, $i \in K$ (where $\beta(n) : K$), and ports q we have:

$$h\,@\,n.i \to_{\mathcal{N}} h\,@\,q \iff \overline{h}\,@\,n.i \to_{\mathcal{N}} \overline{h}\,@\,q \quad (*)$$

Fix a node n, and an $i \in K$ (where $b : K$, setting $b =_{\mathrm{def}} \beta(n)$), and consider the possible q for which (*) holds. There are three cases. In the first $q = n.j$ for some j and q is external. Here, inspection of the definition of $\to_{\mathcal{N}}$ tells us that (*) is equivalent to

$$h\,@\,i \to_b h\,@\,j \iff \overline{h}\,@\,i \to_b \overline{h}\,@\,j$$

In the second $q = n'.j'$ and $n.j \gamma n'.j'$ for some port $n'.j'$ and $j \in K$. Inspection of the definition of $\to_{\mathcal{N}}$ again tells us that (*) is equivalent to

$$h\,@\,i \to_b h\,@\,j \iff \overline{h}\,@\,i \to_b \overline{h}\,@\,j$$

The third case is when q has neither of these forms, in which case (*) is trivially true (more precisely, both sides of implication are false).

As every port at any node is either internal or external, and as every box is in the range of β we see from the characterisation of signature bisimulations in the case that headers are left unchanged in signature transitions that $\sim_{\mathrm{P_a}}$ forms a bisimulation when combined with the identity relation on ports iff it is a signature bisimulation. □

7. Experiments

We describe our benchmark network and then show the use of variants of the surgeries of Examples 1 and 4, together with a header equivalence relation, to obtain large speedups compared to experiments described in earlier work [20]. The first is systematically applicable to all data center networks with a distinguished core; the others are applied automatically, in seconds, on large benchmarks; we also briefly describe how they are implemented.

7.1 Setup

As an initial experimental test of some of the ideas considered in this paper, we worked on a Microsoft production data center located in Singapore. This is a fairly large switching network, with 52 core routers, each with about 800 forwarding rules (but no ACLs), and with 90 ToRs with about 800 rules and 100 ACLs each. In total, this network has about 820K forwarding and ACL rules and is a reasonable example of a complex data center.

We used parsers to automatically extract routing tables from the Arista and Cisco devices in this network using a "show ip route" command. This produced a set of routing tables including the ECMP routing options. Each router is also annotated with a name from which it is easy to syntactically determine its level in the topology (e.g., router names starting with HL denote Host Leaf or Edge routers). This is standard practice; we did not have to add extra annotations manually. The rules in routers map headers to a set of next hop IP addresses; using a simple call to the Domain Name Service we map these next hops to router names to automatically extract the topology of the network.

We use the NoD tools [20] to encode the networks. All results were obtained by running NoD [20] queries on both the original network and various transformations of it.

7.2 Speeding Up All-Pairs Reachability

Our first experiment computed all-pairs reachability between client VMs on the Singapore data center; that is, for each pair of VMs we compute (a representation of) the set of headers of those packets that can reach one VM to the other. This is ideally what needs to be done periodically in operating data centers to help catch configuration errors in routers.

Naively done, this requires a quadratic number of queries over the number of client VMs (Virtual Machines), and these are usually of the order of 100,000s. For example, without exploiting the surgeries in this paper, NoD takes 131 hours ($\sim$5.5 days) to prove that client VMs can reach each other using a single query encoding all-pairs reachability at once.

We did the following experiment based on our knowledge of how this network operates. We followed the idea of the simplest surgery, described in Example 1, automatically rewriting the core by a hub, much as shown in Figure 3. The hub was, however, more complicated: it connected all pairs of ToRs, not just one, and it took account of the fact that ToRs could be connected to the core by more than one port. We then applied NoD to the transformed network, running the same all-pairs reachability query. This now took only 2 hours.

Of course, to be fair one has to count the time to *prove* that the core network does indeed behave like the hub i.e., that it does not filter out packets. The simplest approach is to use NoD itself to prove (see Figure 3) that for all pairs of edge routers R_1 and R_2, that all headers h (with source address corresponding to the prefix corresponding to R_1 and with destination address corresponding to the prefix corresponding to R_2) when sent from R_1 reach R_2. This "proof" does not even require rewriting the network; it only requires computing reachability between every pair of edge routers in the original network.

This "brute-force" verification that the core can be replaced by a hub still took only 1 hour and 40 minutes to complete. This is faster than the original 131 hours because all virtual machines such as V_1 connected to R_1 in Figure 3 are aggregated by a single prefix. Thus the original N^2 scaling is reduced to $(N/64)^2$ as edge routers typically have 64 ports. Below we will give a more sophisticated verification, which is much faster, taking only seconds, that exploits local rule surgery, much like Example 4.

However, even using brute-force verification that the core can be placed by a hub, the overall speedup is from 131 hours to 3.66 hours, i.e., 36×.

7.3 Accelerating Other Queries

We next apply the local surgery ideas introduced in Example 4 to speed up four sets of earlier experiments described in [20]. These checked common "beliefs" in the case of the Singapore data center. The first two beliefs checked were that neither Internet addresses nor customer VMs can access protected fabric controllers for security reasons. The third was that all "utility boxes" can reach all "fabric controllers", and the fourth was that all "service boxes" can reach all "fabric controllers", The definitions of fabric controllers, utility boxes, and service boxes are unimportant; the reader should think of these as classes of boxes with different reachability privileges.

The four queries each took several minutes to complete in the original network without the use of surgeries. Instead, we implemented a version of the Example 4 rule surgery, that used the fact that the network forwards packets unchanged and without regard to entry ports, when we can discuss the network transition relation using rules of the form $h \mapsto I$, as described in Section 2.2. We then obtained the results given in Table 1. Note that, in contrast to the times in minutes on the original network, the same queries on the transformed network often ran in a couple of seconds after surgery, owing to the reduction in the number of rules.

The essence of Example 4 is to remove rules with respect to some set of headers T that do not impact reachability of headers h in T with respect to some set of ports G. We take T to be a singleton consisting of a single header h, and G to be the set of ToR ports connecting to VMs, and write $\equiv_h$ for $\equiv_{\mathrm{TG}}$, the forwarding equivalence relation defined in Example 4.

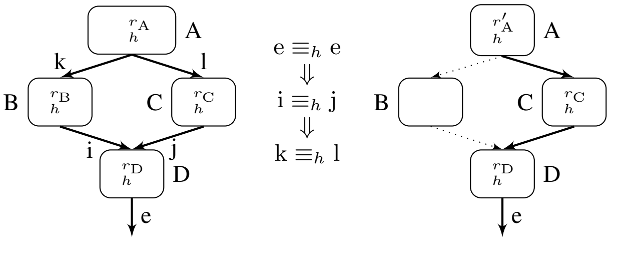

Figure 7. We fix a header h and then inductively establish the port forwarding equivalence relation $\equiv_h$. Some rules can then be dropped after redirecting traffic, in accordance with the equivalence relation.

For example, consider Figure 7, where, for illustration, we suppose that k, l, i, and j are not in G; the rule r_{A} sends h out only to k and l; the rule r_{B} sends it out only to i; the rule r_{C} sends it out only to j; and the rule r_{D} sends it out only to e. Then ports i and j at D are forwarding equivalent with respect to header h because they forward to the same external port e. This inductively implies that ports k at B and l at C are equivalent because they respectively forward to equivalent ports i and j.

We can then redirect the h-traffic through A to go only to C via l, changing r_{A} to the rule r'_{A}. Note that the new network has the same forwarding relation $\equiv_h$ as before, so we can do similar redirections for other rules. Now suppose that in Figure 7 the only way that h-traffic could reach B is via k. Then, after the redirections, the rule r_{B} will be inaccessible and so can be pruned, as can any other inaccessible rules.

We implemented this idea with two data structures. First, we have to deal with the large number of potential headers used in the forwarding plane. For IPv4, this is up to 2^{32} headers even if considering only the destination IP for building equivalence classes (as we do). We mapped the original set of potential headers into a smaller set of equivalence classes where headers are in the same equivalence class if there are no two rules that can distinguish them. For this purpose we used a data-structure called disjoint decomposed normal form (ddNF) data-structure, described in [5]. The asscoiated equivalence relation is theoretically coarser than Yang-Lam equivalence, but we observe in [5] that the number of extra partitions is insignificant.

Despite the fact that the number of rules in our data set was close to a million, the number of header equivalence classes was only around 4000, consistent with the results of [30]. It takes under four seconds to compute these equivalence classes for our network.

Next, we split rules so that each rule operates on a single header equivalence class. Then, for each such class h we compute the forwarding equivalence relation $\equiv_h$ on ports, illustrated above. The algorithm refines port equivalence relations $p \sim_h q$, represented as maps from header equivalence classes h to partitions on ports.

Initially, the map maps each such h to the discrete partition. Then, in the style of congruence closure algorithms, we use union-find structures to maintain partitions [29]: Until reaching the fixed-point $\equiv_h$, for each class of headers h, we merge partitions containing p and q not in G if $\{p' \mid h \,@\, p \longrightarrow_{\mathcal{N}} h \,@\, p'\}$ and $\{q' \mid h \,@\, q \longrightarrow_{\mathcal{N}} h \,@\, q'\}$ are element-wise $\sim_h$-equivalent, as are $\{p' \mid h \,@\, p \twoheadrightarrow_{\mathcal{N}} h \,@\, p'\}$ and $\{q' \mid h \,@\, q \twoheadrightarrow_{\mathcal{N}} h \,@\, q'\}$.

The element-equivalence check is fast because we can use the union-find root to find canonical equivalence classs representatives and we can maintain the sets as sorted lists. Finally, we rewrite all rules in all routers using the header and port equivalence relations: Each rule in every router is rewritten to redirect traffic to the canonical representative of the equivalence relation between ports. Then rules that are no longer reachable, because all the rules that previously directed headers to it had their ports renamed, are garbage collected.

After this transformation (which can be thought of as a *set* of transformations, one for each header equivalence class), we reran the original four queries described in the NoD paper [20], obtaining the results given in Table 1.

In this experiment, we transformed a network with nearly a million rules to a new network with just over 10K rules. Not shown is the time to parse text files containing the data center network (i.e., translate from CISCO format to Datalog) which is about 6 seconds and the time to perform the surgeries which is 4 seconds. The overall speedups obtained ranged from 15× to 360×.

We reiterate that both the identification and the rewriting involved are completely automatic and very fast. A major obstacle of the classical symmetry reduction program in model checking [6, 8, 14] is that is often computationally hard to even identify the symmetries. We are much faster because we do not aim to find all symmetries, and we work at the fine structure of rules. Even naively, finding which pairs of rules are equivalent is only quadratic in complexity (per header class). The congruence closure algorithm described above is even more efficient. It is noteworthy that this particular surgery does not even require identifying which routers are backups for each other because we focus on rule and not box equivalence.

Experiment	pre-op	post surgery
Internet Reaches Protected Fabric	12 min	2s
VM Reaches Protected Fabric	12 min	48s
Utility boxes can reach all fabric controllers	4 min	1.7s
Service boxes can reach all fabric controllers	6 min	1.6s

Table 1. Speedups for belief checking experiments in [20]

We complete the story by connecting the two experiments. Recall that Experiment 1 used a brute-force verification that the core can be replaced by a hub that took 1 hour and 40 minutes. We can use the rule surgeries just described to reduce to reduce even this time to under a second.

Suppose the Edge Routers of a data-center are $\mathrm{tor}_1, \ldots, \mathrm{tor}_n$ and we wish to check that, for example, tor_i is reachable on the address range that it owns from all the other ToRs on any of the ports by which they connect to the core (we use the terms ToR and Edge Router synonymously). Then the typical data-center configuration ensures that all the ToRs, other than tor_i, are forwarding equivalent for any packet h in the address range (by which we mean that the core ports they connect to are all $\equiv_h$-equivalent). We checked that this was the case for the Singapore data-center for all its ToRs, using the computed forwarding equivalence relations.

We observe that this cuts down a quadratic number of routes to a linear number of representatives for the pairwise routes, as, for example, to check reachability of tor_i from the others, one need only check reachability from any one of the core ports one of the others connects to. We checked these reachability queries on the Singapore data-center using a simple depth-first search algorithm on the transition graph of the heavily reduced network.

The bottom line is that with more efficient verification, the overall speedup for all-pairs verification dropped from 131 hours to 2 hours, a 65× speedup.

8. Related Work

Symmetry Reduction: Symmetry reduction has a long history in the model checking literature where it was used for verifying concurrent systems [6, 8, 14]. Ip and Dill [14] trace the use of symmetry for automatic verification to the 80's [1, 21]. Symmetry is formally defined in terms of a certain permutation of participating processes specified by a group G. After symmetries are identified, model checking is done on a simpler structure quotiented by G.

The permutations are usually discovered on the state components, i.e., they deal with the data aspects of the program. By contrast, we target particular permutations that arise from the replication of routers for load balancing and redundancy reasons. Perhaps it is fair to say that our symmetries are driven by the "control-flow graph" of the network. Further, exploiting symmetries does not require modifying the network verification engine since it is done by modifying the network itself (and adjusting the property).

Two practical difficulties with the classical symmetry reduction program are firstly finding and verifying the group G, and secondly dealing with the fact that real structures do not have perfect symmetries. We proceed differently. Rather than calculating the whole group, our aim is to find *particular* symmetries and divide the network by them, rather than its transition relation, intending to verify the simpler network in place of the original one. It is not hard to find such symmetries, at least as regards the topology of the network, especially for fat trees where we need only look at routers on the same level. However it may be that the topological symmetry does not preserve the transition relation, as there are a few differences in the rules of the two routers. So, rather than actually construct the quotiented network (which would anyway be expensive, even with perfect symmetry) we use the topological symmetry (possibly implicitly) to remove redundant rules, as illustrated in Section 4.

Software verification methods for multi-threaded programs usually explore symmetry in the local data maintained by (almost) identical processes by applying the so-called thread modular proof rule, where the induction principle is adjusted to reflect the symmetry [9]. Our approach performs a surgery before the verification engine is run, as opposed to *within* the verification engine.

Bisimulation: Bisimulation is at the core of NetKAT's [2] decision procedure for its equational theory [11]. However, its current implementation does not exploit symmetries. Figure 3 in [11] suggests that on fat tree topologies the running time appears to grow rapidly with the number of hosts.

Our approach, by contrast, relies on bisimulation in the preprocessing phase, but not the actual verification step. Our notions of symmetry and surgery might be beneficial for (and easily integrated into) the NetKAT decision procedure, perhaps as a preprocessing step. A similar approach could be adopted for other network verification tools that check forwarding properties [17, 30].

Flowlog [27] employs partial evaluation and weakening to simplify the network verification problem. Partial evaluation is not currently considered in our system but could be a valuable addition. Weakening is more challenging as our current setup is intimately connected with the notion of bisimulation, but not simulation.

Kuai [23] relies on partial order reduction when checking properties of SDNs. Our traffic redirection can be seen as an instance of partial order reduction that is applied statically.

Semantics: While NetKAT [2] and related work [26] provide a semantically-oriented theory of networks, there is currently no provision for exploiting symmetry and network transformation. Note also that the research program in NetKAT is primarily topdown: defining a policy language and synthesizing networks that meet these policies. By contrast, our agenda is bottom-up: we start with existing networks and analyze their reachability policies.

9. Conclusion

If network verification is to become an integral part of operational procedure in large networks, then the time for comprehensive verification (all pairs of stations, all properties) should be less than the average time of a network reconfiguration. Reconfiguration typically takes hours, but is set to decrease to seconds in the presence of virtual machine migration [16] to optimize resource usage.

Surveying the initial work on network verification, Zhang, Malik, and McGeer [33] say

> ... initial results are based on modest sized systems. However, overall, both FSM- and SAT-based approaches will need to be tested for larger scale systems, e.g. entire data centers or large scale enterprise networks. This will likely need development of new ideas in their solutions, or at the least adaptation of scaling techniques used in other domains. For example, large data centers are likely to have symmetry in their structure. This may enable the use of parametric model checking techniques [39], or symmetry reduction in model-checking and SAT-based techniques. Their application will open up new challenges.

Our paper addresses the challenge of verifying large scale networks by developing a theory of network transformation. Our experimental results show that the apparent complexity of a well-structured data center network (ostensibly 2^{32} headers, $\sim$820K rules) can reduce to $\sim$4,000 header equivalence classes and $\sim$10,000 rules after suitable transformations. In some sense, we are extending the research program of [30] which reduces the number of header equivalence classes but not the number of rules (or ports or routers). The final simpler underlying structure may not be as surprising as it seems at first: if it were more complex, it would be beyond the understanding of the humans who design and operate the network.

The initial experimental evidence in this paper shows that easy-to-code versions of simple transformations (Examples 1 and 4) can provide large speedups, reducing the time for comprehensive verification from days to 2 hours. Since the all-pairs verification task is easily parallelized, this suggests that comprehensive evaluation can be done using a 32-core machine in under 4 minutes which is practical for immediate deployment. Other transformations such as slicing (see Examples 2 and 3) may well bring this time down further. The aim would be to achieve verification in the order of seconds, possibly also using incremental verification techniques as in [16]. Note that earlier results in [16] and [30] were not for all pairs, but only for single queries and for much smaller networks.

The specific transformations we found useful for data center networks in our experiments may carry over to two other important classes of networks: enterprise networks and Internet Service Provider Networks. While neither uses fat-tree topologies, there are regularities in these designs that perhaps could be exploited using the methods of this paper. For example, most enterprise networks use a core network that interconnects a number of leaf networks, and Points of Presence (POPs) in ISP networks often use complete mesh topologies. Even if new transformations are needed, the bisimulation proof techniques may well still be applicable.

While we have only implicitly touched on modularity (when we replaced the core by a wire in Section 7), we plan to extend the theory in this paper to allow modular verification; this would correspond to the compositionality properties of bisimulation in the process calculus, though, as there, composing properties of subsystems will no doubt present challenges. Other avenues for research include scaling quantitative verification (e.g., bandwidth and delay and not just reachability) and control plane verification [10] using network transformations.

Finally, note that our semantics is relational, modeling nondeterminism. In particular partiality is used to model both dropped packets and infinite loops. One could instead adopt other semantic frameworks to model other aspects of networks, for example to distinguish packet dropping from infinite loops, or to model multicasting and/or probabilistic choice. We anticipate that one could then still follow the program of this paper and connect network verification with the then relevant notions of bisimulation.

References

[1] S. Aggarwal, R. Kurshan, and K. Sabnani. A calculus for protocol specification and validation. *Protocol Specification, Testing, and Verification*, 3(1), 1983.

[2] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker. NetKAT: semantic foundations for networks. In *POPL*, 2014.

[3] M. A. Armstrong. *Groups and Symmetry*. Springer, 1988.

[4] S. Arun-Kumar. On bisimilarities induced by relations on actions. In *SEFM*, 2006.

[5] N. Bjørner, G. Juniwal, R. Mahajan, S. A. Seshia, and G. Varghese. ddnf: An efficient data structure for header spaces. Technical report, Microsoft Research, November 2015. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=258188`.

[6] E. M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In *CAV*, 1993.

[7] E. Emerson and A. Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1-2):105–131, 1996.

[8] E. A. Emerson and A. P. Sistla. Symmetry and model checking. In *CAV*, 1993.

[9] C. Flanagan and S. Qadeer. Thread-modular model checking. In *SPIN*, 2003.

[10] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein. A general approach to network configuration analysis. In *NSDI*, 2015.

[11] N. Foster, D. Kozen, M. Milano, A. Silva, and L. Thompson. A coalgebraic decision procedure for NetKAT. In *POPL*, 2015.

[12] M. Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. PhD thesis, University of Edinburgh, 1997.

[13] M. Hasegawa, M. Hofmann, and G. Plotkin. Finite dimensional vector spaces are complete for traced symmetric monoidal categories. In *Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, pages 367–385. Springer Berlin Heidelberg, 2008.

[14] N. Ip and D. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1), 1996.

[15] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: static checking for networks. In *NSDI*, 2012.

[16] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real time network policy checking using header space analysis. In *NSDI*, 2013.

[17] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. VeriFlow: verifying network-wide invariants in real time. In *NSDI*, 2013.

[18] J. F. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002. ISBN 0201976994.

[19] Z. Li, M. Liang, L. O'Brien, and H. Zhang. The cloud's cloudy moment: A systematic survey of public cloud service outage. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, 2(5):321–331, 2013.

[20] N. P. Lopes, N. Bjørner, P. Godefroid, K. Jayaraman, and G. Varghese. Checking beliefs in dynamic networks. In *NSDI*, 2015.

[21] B. Lubachevsky. An approach to automating the veri
cation of compact parallel coordination programs. *Acta Informatica*, 21(2), 1984.

[22] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King. Debugging the data plane with Anteater. In *SIGCOMM*, 2011.

[23] R. Majumdar, S. D. Tetali, and Z. Wang. Kuai: A model checker for software-defined networks. In *FMCAD*, 2014.

[24] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[25] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[26] C. Monsanto, N. Foster, R. Harrison, and D. Walker. A compiler and run-time system for network programming languages. In *POPL*, 2012.

[27] T. Nelson, A. D. Ferguson, M. J. G. Scheer, and S. Krishnamurthi. Tierless programming and reasoning for software-defined networks. In *NSDI*, 2014.

[28] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4):15:1–15:41, May 2009.

[29] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[30] H. Yang and S. Lam. Real-time verification of network properties using atomic predicates. In *ICNP*, 2013.

[31] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown. Automatic test packet generation. In *CoNEXT*, 2012.

[32] S. Zhang and S. Malik. SAT based verification of network data planes. In *ATVA*, 2013.

[33] S. Zhang, S. Malik, and R. McGeer. Verification of computer switching networks: An overview. In *ATVA*, 2012.