Real-time verification of wireless home networks using bigraphs with sharing

Muffy Calder, Alexandros Koliousis, Michele Sevegnani, Joseph Sventek

PII: S0167-6423(13)00197-4
DOI: 10.1016/j.scico.2013.08.004
Reference: SCICO 1588

To appear in: *Science of Computer Programming*

Received date: 3 February 2012
Revised date: 5 August 2013
Accepted date: 5 August 2013

Real-time verification of wireless home networks using bigraphs with sharing

Muffy Calder [1a], Alexandros Koliousis[b], Michele Sevegnani[a], Joseph Sventek[a]

[a]*School of Computing Science, University of Glasgow, UK*
[b]*Dept of Computing, Imperial College London, UK*

Abstract

Home wireless networks are difficult to manage and comprehend because of evolving locality, co-locality, connectivity and interaction. We define formal models of home wireless network infrastructure and policies and investigate how they can be used in a network management system designed to provide user-oriented support. We model spatial and temporal behaviour of network interactions and user-initiated network policies and define an online framework for generation of models from network and user-initiated events. The models are expressed in an extension to Milner's bigraphical reactive systems. Analysis of the models is carried out in real-time by a bespoke bigraph reasoning system based on checking predicates, which is encoded as bigraph matching. Real-time model generation and analysis is implemented on the experimental Homework system router and trialled with synthetic and actual network data.

Keywords: network management, verification, bigraphical reaction systems, bigraphs with sharing, runtime model generation, bigraph matching

1. Introduction

Wireless home networking is notoriously difficult to install and manage, especially for non-expert users. The Homework network management system [1] is an experimental system designed to provide user-oriented support in home wireless local area network (WLAN) environments. The Homework system is much more than a user interface for existing network infrastructure. It provides new network architectures that take into account the sociotechnical nature of home networking. For example, devices are brought into the home by family and friends, and users define policies for explicit management and access. It also encompasses new approaches to infrastructure measurement and monitoring and user focussed computational models for modelling and analysis in support of both design and user experience. In particular, the Homework system is a platform from which we can investigate how formal models can be used

[1]*Corresponding author.* e-mail: Muffy.Calder@glasgow.ac.uk.

iteratively and interactively to contribute to the question "is the proposed network infrastructure fit for purpose", and more generally, if and how seamfully exposing models of infrastructure and user behaviour to those being modelled is useful and can be carried out in real-time, without interruption or delay to the network management system.

The aim of this paper is to define suitable formal models of the infrastructure and policies and to investigate how they can be used in an extension to the basic Homework network management system.

1.1. The standard Homework system

The Homework system architecture consists of three complementary planes: data, signalling, and information. We focus on the last, which is a monitoring application that makes available information about network set-up, management and measurement. It uses a stream database to record (raw and derived) events. Events include network behaviours such as detecting that a new machine has joined the network, resulting in new links and granting a DHCP lease, and user-initiated behaviours such a enforcing or dropping a policy. Policies are defined by users through a novel user interface that allows drag and drop, comic-strip style interaction (see [2]). Typically, policies forbid or allow access to network resources; for example, a policy might block UDP and TCP traffic from a given website, or restrict internet access for certain users during given time periods.

1.2. Modelling wireless network management

Locality, co-location, interaction, connectivity, and user-perceived events are key aspects of user-oriented home networking. We require models that expose these aspects, and their temporal evolution, to both end users and system developers, and permit computation and analysis of properties in real-time. While various formalisms might fit these criteria to a greater or lesser extent, we propose that bigraphs with sharing, an extension of Milner's universal process algebra that encapsulates both dynamic and spatial behaviour [3], fit all these criteria particularly well. Specifically, bigraphical reactive systems (BRS) are well suited to the problem because a) the (human-oriented) graphical form provides an intuitive representation of locality, co-locality, and connectivity, b) there is an explict representation of user-perceived events by rewrite rules and c) there is a (machine readable) algebraic form for computation and verification of properties.

In our models, each BRS consists of a set of bigraphs that describes spatial and communication relationships between machines and entities in the network, and a set of bigraphical reaction rules that define how the bigraphs can evolve over time. We have extended the basic formalism of BRS to bigraphical reactive systems *with sharing*, to permit effective and intuitive representation of spatial locations that can overlap[2]. This extension is particularly relevant to

[2]Henceforth we refer to this extension simply as BRS.

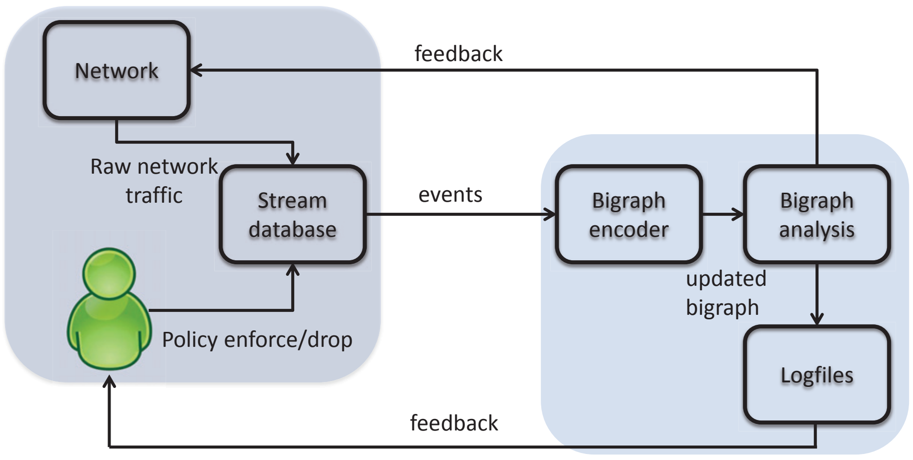


Figure 1: Real-time model generation, analysis and feedback in the Homework system.

our requirements, since multiple, overlapping signals are fundamental to wireless networks.

1.3. Real-time model generation, analysis and feedback in Homework

In our extension to the Homework system, models are generated from events recorded in the information plane and analysed without interruption or delay to the network management system.

The system is depicted in Figure 1. As we have indicated above, the *Stream database* is part of the standard Homework management system; all network and policy events are recorded as streams of tuples in the database. The *Bigraph encoder* component is new, and it encodes events as bigraphical reaction rules. The *Bigraph analysis* component is also new, and it has two roles. First, it generates the bigraphical representation of the current configuration of the WLAN, according to the sequences of reaction rules received from the *Bigraph encoder*. Second, it analyses the current configuration by checking properties, for example, whether or not a configuration violates a user-invoked access control policy. Properties are expressed as predicates that are encoded as instances of bigraph matching. The results are logged and can be fed back to the system, or to the user, using the graphical notation of bigraphs as explanation. This work flow is carried out in real-time, hence we refer to our approach as *real-time* verification.

While our long-term motivation is to aid users in their understanding of the state of their system (e.g. when and why it is “broken”), and to give feedback to developers about user experiences, in this paper we concentrate on the technical details of the representations of networks and policies and the analysis system itself.

1.4. Overview of paper

The main focus of this paper is to describe the bigraphical representations of networks topologies, the events that modify topologies and the access control policies, and how to represent and check predicates on bigraphs within the runtime system.

The contributions of the paper are the following:

- representations of network topologies as bigraphs and network events (such as a machine leaving and joining a network) as bigraphical reaction rules,
- representations of access control policies that forbid and allow behaviours as bigraphical reaction rules that constrain network evolutions,
- new reasoning techniques for predicates over bigraphs, encoded as instances of bigraph matching and implemented using a SAT solver,
- a solution for the problem of how to check for the non-existence of patterns in bigraphical reaction rules, and how to reason about topologies with arbitrary numbers of machines and communication channels, by tagging and untagging entities,
- on-line generation of bigraphical reactive system models from the current network topology and activated policies, as recorded in the Homework information plane, and
- empirical evidence demonstrating that generation and analysis of bigraph models can be carried out in real-time within the Homework system.

The paper is organised as follows. Section 2 contains an informal introduction to the bigraph notation, bigraphical reactive systems and bigraph matching. In Section 3 we describe how network topologies are represented as bigraphical systems and how network events, such as moving in and out of the router's range, and granting and revoking of leases, are encoded as reaction rules; in Section 4 we show how predicates are encoded as bigraphs, and thus can be checked by bigraph matching. In Section 5 the rules and predicates defined in Section 3 are used to generate sequences of models in real-time. Section 6 describes how policies that forbid and allow behaviour are represented as bigraphical reaction rules and how they constrain network evolutions. In Section 7 we describe how policy events such as *enforce a policy*, *drop a policy* or *check a policy*, are encoded as reaction rules, and we discuss the interplay between the (representations of) network and policy events. In Section 8 we show in detail how a bigraphical model of a WLAN is updated according to the stream of network and policy events generated in real-time. Section 9 discusses the role of state predicates in the analysis of network configurations and compliance with policies; in Section 10, we give an overview of the implementation. A discussion of the overall approach and the role of the bigraph abstraction is in Section 11 and related work is reviewed in Section 12; we conclude in Section 13.

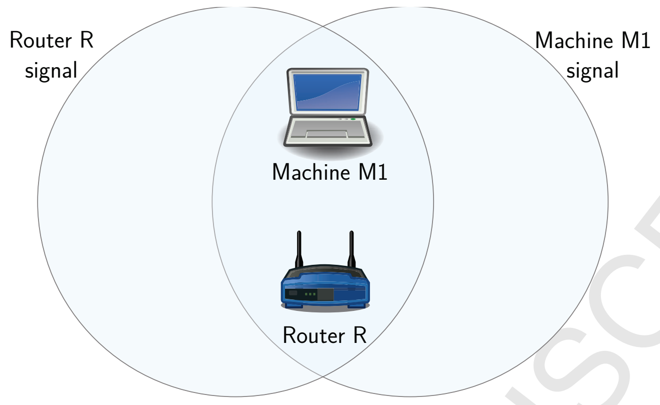


Figure 2: Simple WLAN with one machine and a router.

2. Bigraphs with sharing

In this section we give an informal overview of BRS, with some examples. The overview contains sufficient detail for this paper; a concise semantics of bigraphs with sharing is defined in [4]. Details of standard BRS (without sharing) are in [3].

A bigraph has a graphical and an algebraic form. In this paper, we use both forms, but primarily the graphical form. In the graphical form, an entity (real or virtual) is encoded by a *node* (oval or circle). Spatial placement of nodes is described by node nesting, which we have extended to directed acylic graphs. Thus, nodes can be placed in the intersection of other nodes. Each node is assigned a *control*. Interaction between nodes is represented by an edge called a *link* that connects *ports*. Each node can have zero, one or many *ports*, indicated by bullets. They can be thought of as sockets into which links can be plugged. A dashed rectangle denotes a *region* of adjacent parts of the system. A grey square indicates a *site*, which encodes part of the model that has been abstracted away. A link may be only partially specified, in which case it connects ports with a *name*. Name closure $/x\ A$ is used to disallow connections on name x in bigraph A.

As an example, consider the WLAN depicted in Figure 2: there is one machine and a router in the network, each associated with a signal.

This network is represented as a bigraph in Figure 3. There are three controls $\mathsf{M1}$, S, and R; the two signals (of the machine and the router) are represented by the nodes of control S, the router is indicated by the node of control R and the machine is represented by the node of control $\mathsf{M1}$. There are three links: a link between machine $\mathsf{M1}$ and its signal, a link between router R and its signal, and a link between machine $\mathsf{M1}$ and router R. There are no names in this bigraph.

The capabilities of a bigraph to interact with the external environment are given in its *interface*. For example, we write $A : 1 \to \langle 2, \{x, y\}\rangle$ to indicate

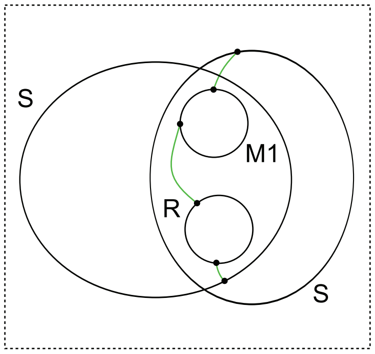


Figure 3: Bigraph representation of simple WLAN with one machine and a router.

that A has one site, two regions and the names x and y. Controls and links in a bigraph are classified by means of *sorts* (ranged over by a, b, ...) and a *formation rule* defines sorting properties a bigraph has to satisfy. For example, in a WLAN representation, a typical formation rule would be: an R node is always contained in an S node (i.e. a router has a signal). A sort may be a disjunction, which we denote as follows: $\widehat{\mathsf{ab}}$ means that a node can have sort a or b. The interface of a sorted bigraph is expressed as follows: $A : \mathsf{a} \to \langle \mathsf{bb}, \{x, y\}\rangle$. The notation indicates that the site has sort a and the two regions have sort b.

The structure of a bigraph can also be specified in algebraic form by combining elementary bigraphs and bigraphical operations. A summary is given in Table 1. Except for sharing, the notation is fairly straightforward. An explanation of the notation for sharing is the following. Sharing is a specialised version of nesting: share F by ϕ in G denotes the bigraph in which the regions of bigraph F can be placed inside the sites of bigraph G. The association between F's regions and G's sites is specified by *placing* ϕ, which is a bigraph without nodes. This allows the expression of shared nodes, i.e. nodes situated in the intersection of other nodes. Numbering of regions and sites proceeds from left to right starting from zero. Therefore, placings can be expressed by a vector of sets indicating unambiguously how regions are shared by sites. For example,

$$\text{share } F \text{ by } \phi \text{ in } G$$

where $F \overset{\text{def}}{=} \mathsf{A} \parallel \mathsf{B}$, $G \overset{\text{def}}{=} \mathsf{C} \mid \mathsf{D}$ and placing $\phi \overset{\text{def}}{=} [\{0\}, \{0, 1\}]$, is depicted in Fig. 4. This Figure also indicates the difference between the more familiar Venn diagram graphical notation that we use, and the usual stratified notation. Here, ϕ has length 2 and indicates that the first F region (the region containing the A-node) is placed in the first G site (the site in the C-node) while the second F region (the region containing the B-node) is shared between the first and the second of G's site (the sites in the C and D-nodes, respectively). Regarding elementary bigraphs, 1 denotes an empty region and 0 expresses a site that is not within a region; the latter only exists because of sharing. Identities are

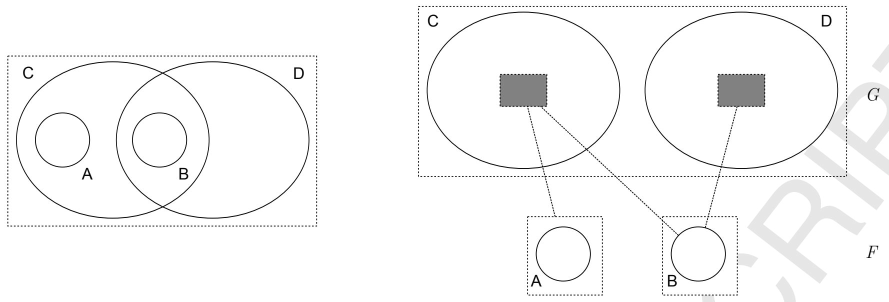


Figure 4: Two representations of bigraphical term share F by ϕ in G, where $F \overset{\text{def}}{=} \mathsf{A} \parallel \mathsf{B}$ and $G \overset{\text{def}}{=} \mathsf{C} \mid \mathsf{D}$. Graphical notation using Venn diagrams (left) and stratified notation highlighting placing $\phi \overset{\text{def}}{=} [\{0\}, \{0,1\}]$ (right).

indicated with $\mathsf{id}_{n,X}$ where $n \in \mathbb{N}$ and X is the elements of a set of names. We sometimes write id_n when $X = \emptyset$ and id for id_1.

We note that while it is possible to encode sharing in standard BRS, these encodings suffer several disadvanges (see [4] for details); moreover, an advantage of explicit sharing is that it overcomes the asymmetric treatment of roots and sites in standard bigraphs.

Evolution in a BRS is defined by rewrite rules, called *bigraph reaction rules*, which induce a transition relation on bigraphs. Reaction rules are written with an arrow thus: $\longrightarrow\!\!\blacktriangleright$, whereas transitions between bigraphs are written with an arrow thus: $\longrightarrow\!\!\triangleright$. We also use $\longrightarrow\!\!\triangleright^*$ to indicate zero or more transitions. As an example, consider the evolution of a WLAN consisting of two machines and a router, to one machine and a router, as depicted in Figure 5a. On the left-hand side, the two machines are part of the network. They can both sense the router, but not each other. On the right-hand side, one machine has left the network. This evolution can be represented formally with two bigraphs, W_0 and W_1, as shown in Figure 5b. Note that on the left-hand side, each signal is linked to its device and the three devices are linked together to indicate they all are part of the WLAN. On the right-hand side, M2 and its signal disappear. The link representing the WLAN now only connects M1 and R. Observe that both bigraphs W_0 and W_1 respect the formation rule described above (i.e. an R node is always contained in an S node).

Now consider how the transformation of W_0 into W_1 is specified by the *reaction rule* given in Figure 6. In general, the left-hand side of a reaction rule identifies the parts of a bigraph that are to be modified (this is also called *bigraph matching*), and the right-hand side describes how to modify them. In this example, bigraph R identifies M2 and its signal as the sub-parts of W_0 that are to be modified. The site indicates that other nodes can be present inside the S node. Similarly, name r represents the fact that M2 can be linked to other nodes. When the reaction rule is applied to W_0, the site is associated to R and r to M1. The two regions surrounding node M2, together with the site inside S, are necessary to express that the site and M2 are in different parts of the

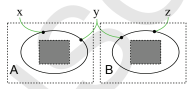
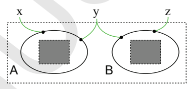
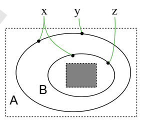
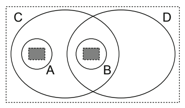
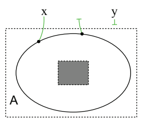
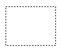
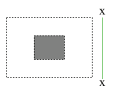
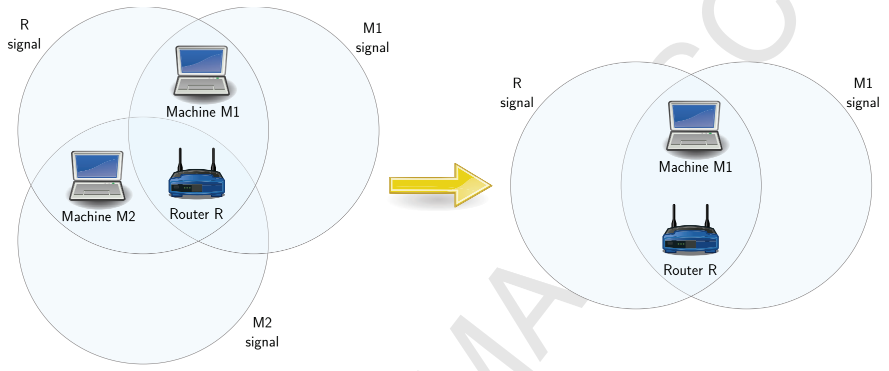
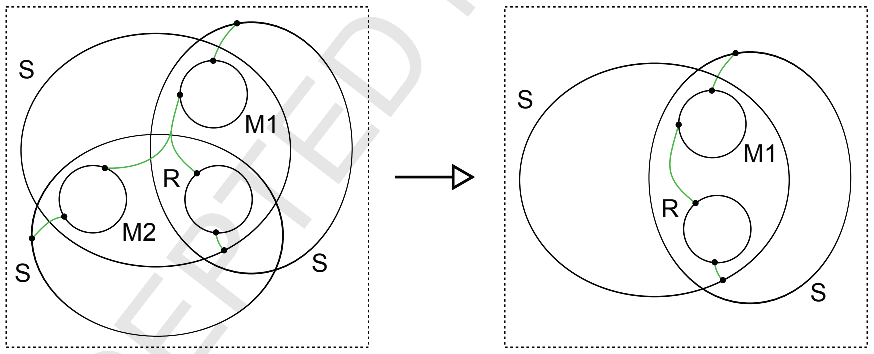
Description	Algebraic	Graphical
Parallel product	$\mathsf{A}(x, y) \parallel \mathsf{B}(y, z)$	
Merge product	$\mathsf{A}(x, y) \mid \mathsf{B}(y, z)$	
Nesting	$\mathsf{A}(x, y).\mathsf{B}(x, z)$	
Sharing	share $\mathsf{A} \parallel \mathsf{B}$ by ϕ in $\mathsf{C} \mid \mathsf{D}$ $\phi \overset{\text{def}}{=} [\{0\}, \{0, 1\}]$	
Closure and new	$/z\ \mathsf{A}(x, z) \parallel y$	
Empty region	1	
Site not within a region	0	
Identity	$\mathsf{id}_{1,x}$	

Table 1: Elementary bigraphs and operations on bigraphs.



(a) A machine leaves the WLAN



(b) A bigraphical representation: $W_0 \longrightarrow\!\triangleright W_1$

Figure 5: Evolution of a WLAN: network diagram (a) and bigraphical representation (b).

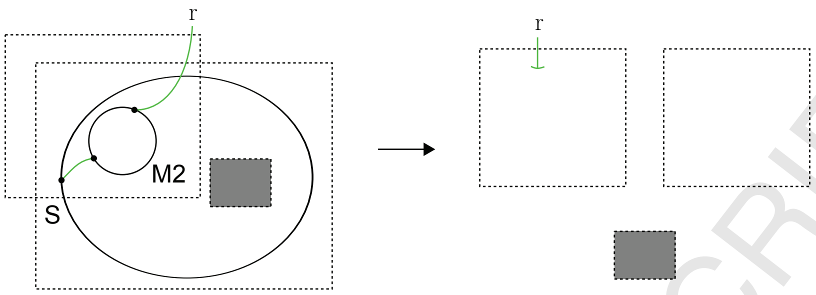


Figure 6: Reaction rule $R \longrightarrow R'$: machine M2 leaves the WLAN.

system. This is shown in W_0, where M2 and R are in different intersections of S nodes. Right-hand side R' specifies that the sub-parts of W_0 matched by R are substituted by two regions, a site and a closed link on name r, i.e. M2 and its signal are removed. When the occurrence of R in W_0 is replaced with R', we indeed obtain the updated WLAN encoded by bigraph W_1. Note that this rule highlights a difficulty of simple Venn diagrams for representing complex spatial relationships. For example, on the right hand side the grey site is not within the parent region of S (i.e. the upper right hand region) because this would impose a relationship between the grey site and this region. While there was a relationship between this site and signal S on the left hand side of the rule, when S is no longer present, there is no relationship.

Bigraph matching and rewriting

Like in any rule-based system, a given reaction rule is applicable to a given bigraph (the target) when the the left-hand side of the reaction rule (the pattern) *matches* the target. Thus bigraph matching is fundamental to the transition relation $\longrightarrow\!\triangleright$. Bigraph matching was first defined in [5] by a set of inference rules characterising the occurrence of an abstract pattern in an abstract target. However, these rules do not lead to an efficient implementation, nor can they be extended in an efficient way to bigraphs with sharing. In particular, there is only one way to extend the rules to deal with sharing and it increases significantly the amount of unnecessary blind search into the inference process. Whereas the matching problem without sharing is (in general) an instance of the subforest isomorphism problem, in most cases (for example, when a reaction rule is applied) it is an instance of the subtree isomorphism problem, which can be efficiently solved in polynomial time. However, the matching problem for bigraphs with sharing is a special case of the subgraph isomorphism problem, which is NP-complete. We have defined and implemented an efficient algorithm for matching bigraphs with sharing based on a SAT-encoding, which has proven effective for solving several other NP-complete problems (e.g. graph colouring problem, bounded model checking). Since (standard) bigraphs are a special case of bigraphs with sharing, our algorithm works for standard bigraphs as well. Full details of the algorithm are given in [4]. Details of experiments with synthetic and actual network data are given in Section 10, where we note the slowest update (including checking several predicates) is less than 0.1s.

A rewriting paradigm we encounter several times when modelling home wireless network management is the need to apply a rewrite rule(s) a fixed number of times to certain terms in the representation. For example, we may require to apply a reaction rule to (the represesentation of) every machine in the network. But, we are dealing with a dynamic network topology, and we do not have a fixed number of machines. Alternatively, we may need to distinguish between machines that are connected to the network and those that are not, so that we can apply a treatment to only one type. Our solution is to "tag" terms that have been treated (or conversely, have still to be treated). This means adding additional reaction rules to apply and remove the markings, a process we refer to as "tagging" and "untagging". The first occurrence of this paradigm is in Section 3.4 where we consider granting leases to machines in the network.

3. Bigraphical models of network topology and network events

In this section we outline how a given network topology is represented by a bigraph, and then how network events, such as moving in/out of the router's range and granting/revoking leases, are represented by reaction rules.

3.1. Network topology

We use a node to represent each entity present in the network, which can be physical e.g. router, wireless signal, machines, or virtual e.g. configuration properties, the Internet, communication channels. Links connect related entities. For instance, a machine is linked to its signal and to its properties. The sorting discipline ensures that only bigraphs with a meaningful structure are constructed. For example, it enforces that a node representing a machine lies within a node representing its signal.

The controls and sorts used to represent the network are listed in Table 2. An explanation is as follows. Sort p is assigned to controls indicating MAC addresses, such as control $\mathsf{01{:}23{:}45{:}67{:}89{:}ab}$. We use a special control MAC, to indicate a generic MAC address, controls $\mathsf{Hostname}$ and IP, to indicate a generic host-name and IP address, respectively. The formation rule is given in Table 3. Informally, it states that most of the entities are atomic (e.g. machines, input, output, etc.) and each machine is placed inside a signal and is connected to it. Analogously, the router lies within its signal and is linked to it. Machines are also connected to a property box that contains various configuration details. Note that whereas in the introductory material in Section 2 we had two controls for machines, i.e. $\mathsf{M1}$ and $\mathsf{M2}$ in Figure 4, here we have only one control for machines, M. Individual machines are distinguished by a link to their MAC address. Machines that are part of the WLAN share a link with the w-node inside the router. Finally, property boxes (and the Internet) are linked to each other via a pair of communication channels. These are represented by an i-node linked to an o-node.

Control	Meaning	Sort	Graphical notation
R	Router	r	Circle
S	Wireless signal	s	Oval
M	Wi-Fi enabled machine	m	Circle
Internet	Outside world	j	Box
Properties, . . .	Configuration settings	b	Box
W	WLAN	w	Circle
I, . . .	Input	i	Small rectangle
O, . . .	Output	o	Small arrowhead
MAC, . . .	MAC address	p	Rounded box
Hostname, . . .	Hostname	p	Rounded box
IP, . . .	IP address	p	Rounded box

Table 2: Controls and sorts for WLAN.

all $\widehat{\mathsf{mwiop}}$-nodes are atomic
all children of an s-node have sort $\widehat{\mathsf{rm}}$
an r-node has a w-child
all p-nodes are children of a b-node
all $\widehat{\mathsf{io}}$-nodes are children of a $\widehat{\mathsf{bj}}$-node
all s-nodes are always linked to a $\widehat{\mathsf{rm}}$-child
a b-node is always linked to an m-node
a w-node may only be linked to m-nodes
an i-node may only be linked to an o-node
an o-node may only be linked to an i-node

Table 3: Formation rule for WLAN.

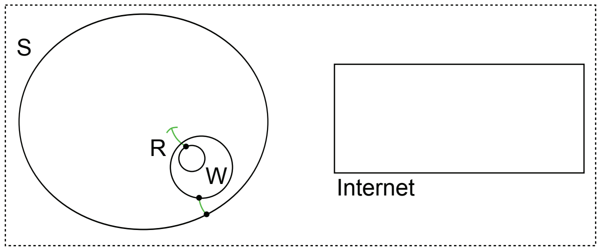


Figure 7: Initial configuration S_0.

The initial configuration of a WLAN is given by bigraph S_0 in Figure 7. It models the scenario in which only the router and the external world are present. The interface is $S_0 : \epsilon \rightarrow \langle \widehat{\mathsf{sj}}, \emptyset \rangle$, ϵ indicates no sorted site, i.e. the interface of S_0 is a constant. The algebraic form is $S_0 \overset{\text{def}}{=} /x\ /y\ (\mathsf{S}(x).\mathsf{R}(x).\mathsf{W}(y).1) \mid \mathsf{Internet}.1$

Now we turn our attention to the reaction rules that represent the network events, which include moving in and out of the router's range, and the granting and revoking of DHCP leases. We discuss each event in turn, using the graphical form. A summary of all the reaction rules is given in algebraic form in Table 4, and the interfaces are in Table 5, respectively.

3.2. Moving into the signal range of the router

The first reaction rule, given in Figure 8, models the appearance of a new machine in the signal range of the router. On the left-hand side, in the expression denoted by R_1, the router is in the range of its signal and possibly other signals. This is expressed by the region surrounding the r-node. On the right-hand side, in the expression denoted by R_1' (n.b. in general, the text accompanying a rule describes the right-hand side), a new machine is in the range of the router's signal. The router senses the new machine's signal and possibly other signals. This is expressed by nodes R and M being in the intersection of the two s-nodes and the region surrounding R. A property box (i.e. a b-node) is also linked to M. Note that the only configuration setting specified at this stage is the MAC address of the new machine M. This is witnessed by the p-node placed inside Properties.

Observe that this reaction rule forces all m-nodes to be shared by only two s-nodes. This means our model does not capture any interference between the signals of the machines in the system: our model is based solely on information provided by the router. In other words, we only model what the router senses.

3.3. Moving out of the signal range of the router

Another reaction rule, given in Figure 9, models the evolution of the system when a machine is no longer in the router's signal range. This happens because either a machine switches off its network interface or it moves into a location not reachable by the router's signal. On the left-hand side, in expression R_2, an m-node is linked to a b-node and placed within an s-node. These correspond

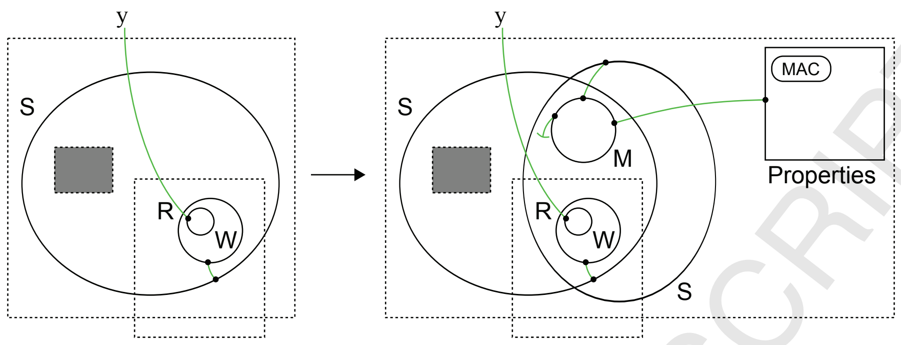


Figure 8: Reaction rule $R_1 \longrightarrow\!\!\!\blacktriangleright R'_1$: a new machine appears in the router's signal range.

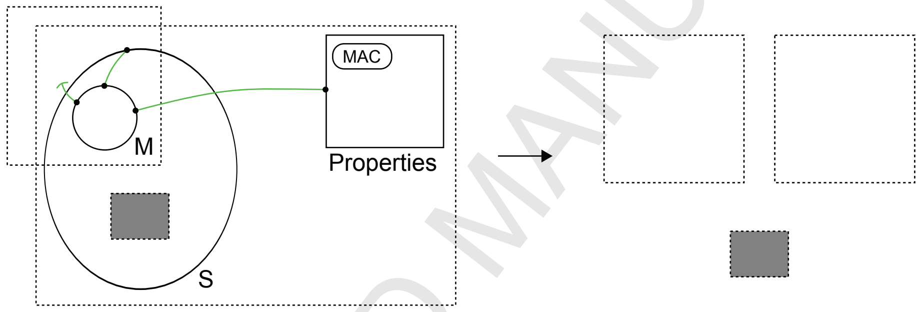


Figure 9: Reaction rule $R_2 \longrightarrow\!\!\!\blacktriangleright R'_2$: a machine is no longer in the router's signal range.

to a machine, its configuration properties and its signal range, respectively. The extra region enclosing M and the site are necessary to allow the machine modelled in R_2 to be in the range of the router and possibly other machines. On the right-hand side, in expression R'_2, all the nodes have disappeared and only the bigraphical interface is preserved (see Table 5). This models the absence of the machine from the system. Note that on the left-hand side, there could be another entity in the site (e.g. the router), which would persist even after we remove the signal S.

3.4. Granting leases

The next three reaction rules describe how the system changes when a machine joins the WLAN and a DHCP lease is granted. This requires distinguishing between the new machine and those already in the network. We do so by tagging the latter. The first rule, $R_{3a} \longrightarrow\!\!\!\blacktriangleright R'_{3a}$, implements the tagging, the second rule, $R_{3b} \longrightarrow\!\!\!\blacktriangleright R'_{3b}$, establishes the network aspects of the untagged machine (i.e. IP address etc.), and the third rule, $R_{3c} \longrightarrow\!\!\!\blacktriangleright R'_{3c}$, establishes the communication channels between the new machine and the tagged machines and then it revokes the tags.

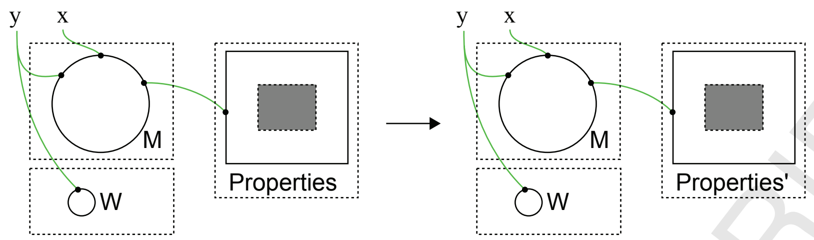


Figure 10: Reaction rule $R_{3a} \longrightarrow R'_{3a}$. A new machine joins the WLAN: all stations already in the WLAN are tagged.

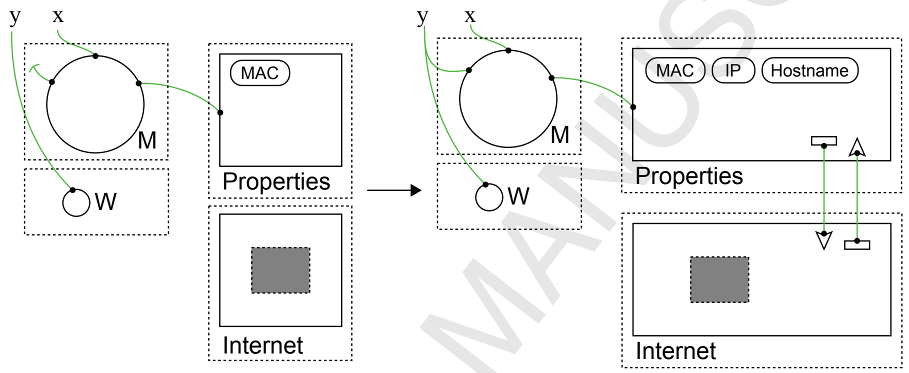


Figure 11: Reaction rule $R_{3b} \longrightarrow R'_{3b}$. A new machine joins the WLAN: Hostname and IP address are set and communication channels with the Internet are established.

Reaction $R_{3a} \longrightarrow R'_{3a}$, in Figure 10, is used to tag all the machines in the system that are already part of the WLAN. On the left-hand side we have an m-node linked to the w-node. The actual tagging is implemented in the right-hand side, where a node of control Properties′ takes the place of the corresponding node of control Properties in R_{3a}.

Reaction rule $R_{3b} \longrightarrow R'_{3b}$ models the DHCP server granting a lease to the machine, as depicted in Figure 11. On the left-hand side, a machine is not part of the network and the only configuration property already specified is the MAC address. This is shown by the absence of a link between the m-node and the w-node and the absence of a site inside the node of control Properties. On the right-hand side, R'_{3b}, the machine joins the WLAN, IP address and hostname are set, and two communication channels with the external world are established. Note that the channels are directional.

In reaction rule $R_{3c} \longrightarrow R'_{3c}$ a pair of communication channels is established between the new machine and the machines already part of the WLAN, see Figure 12. On the left-hand side, R_{3c}, a node of control Properties and a node of control Properties′ specify the configurations of the new machine and a machine already in the WLAN, respectively. On the right-hand side, R'_{3c}, a pair of

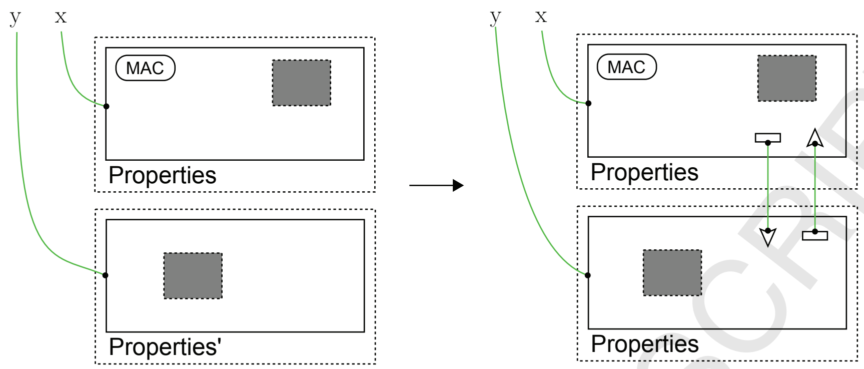


Figure 12: Reaction rule $R_{3c} \longrightarrow R'_{3c}$. A new machine joins the WLAN: Communication channels are created between the station and all the machines already present in the WLAN.

communication channels is established and a node of control Properties replaces the corresponding node of control Properties′ in R_{3c}.

We note that initially, all machines that have already joined the WLAN are tagged, using reaction $R_{3a} \longrightarrow R'_{3a}$. This means the reaction is applied n times, where n is the number of machines in the network. The resulting interleaving of applications is confluent, therefore, only one sequence need be considered. Reaction $R_{3b} \longrightarrow R'_{3b}$ is applied once. Finally, reaction $R_{3c} \longrightarrow R'_{3c}$ is applied n times. Again, due to confluence, only one sequence need be considered.

3.5. Revoking leases

Now consider a machine leaves the WLAN and the lease is revoked, which is represented by two rules. Note, this does not automatically imply that the machine is also leaving the router's signal range.

Reaction rule $R_{4a} \longrightarrow R'_{4a}$ is given in Figure 13. R_{4a} specifies a property box for the machine and a pair of channels. The site also allows the reaction to be applied when other nodes are inside the node of control Properties. On the right-hand side the interface is preserved and only the two communication channels are removed.

Reaction rule $R_{4b} \longrightarrow R'_{4b}$ revokes the machine's DHCP lease. This is encoded by the removal of nodes of control Hostname and IP and the breaking of the link between M and W, as depicted in Figure 14.

Note that reaction $R_{4a} \longrightarrow R'_{4a}$ is applied first, until no other channels can be removed. Again, the order in which the channels are removed is not important and only one sequence of reactions need be considered. Second, reaction rule $R_{4b} \longrightarrow R'_{4b}$ is applied once.

Reaction rule	Algebraic form
$R_1 \longrightarrow R'_1$	$R_1 \overset{\text{def}}{=} /x \Big(\text{share } \mathsf{R}(x).\mathsf{W}(y).1 \parallel \mathsf{id}$
	$\text{by } \phi$
	$\text{in } \mathsf{S}(x).(\mathsf{id} \mid \mathsf{id}) \parallel \mathsf{id}_{1,y,x}\Big)$
	$R'_1 \overset{\text{def}}{=} /x\, /z\, /p \Big(\text{share } \mathsf{R}(x).\mathsf{W}(y).1 \parallel /y\, \mathsf{M}(y,z,p).1 \parallel \mathsf{id}$
	$\text{by } \phi'$
	$\text{in } (\mathsf{S}(x).(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{S}(z) \mid \mathsf{Properties}(p).\mathsf{MAC}.1)$
	$\parallel \mathsf{id}_{1,y,x,z,p}\Big)$
	$\phi \overset{\text{def}}{=} [\{1,2\},\{0\}] \qquad \phi' \overset{\text{def}}{=} [\{1,2,3\},\{1,2\},\{0\}]$
$R_2 \longrightarrow R'_2$	$R_2 \overset{\text{def}}{=} /x\, /p \Big(\text{share } /y\, \mathsf{M}(y,x,p).1 \parallel \mathsf{id}$
	$\text{by } \phi$
	$\text{in } (\mathsf{S}(x).(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{Properties}(p).\mathsf{MAC}.1) \parallel \mathsf{id}_{1,x,p}\Big)$
	$R'_2 \overset{\text{def}}{=} 1 \parallel 1 \parallel 0$
$R_{3a} \longrightarrow R'_{3a}$	$R_{3a} \overset{\text{def}}{=} /p\, (\mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel \mathsf{Properties}(p))$
	$R'_{3a} \overset{\text{def}}{=} /p\, (\mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel \mathsf{Properties}'(p))$
$R_{3b} \longrightarrow R'_{3b}$	$R_{3b} \overset{\text{def}}{=} /p\, (/y\, \mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel P \parallel \mathsf{Internet})$
	$R'_{3b} \overset{\text{def}}{=} /p\, /h\, /l\, (\mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel P' \parallel I)$
	$P \overset{\text{def}}{=} \mathsf{Properties}(p).\mathsf{MAC}.1$
	$P' \overset{\text{def}}{=} \mathsf{Properties}(p).$
	$(\mathsf{MAC}.1 \mid \mathsf{Hostname}.1 \mid \mathsf{IP}.1 \mid \mathsf{I}(h).1 \mid \mathsf{O}(l).1)$
	$I \overset{\text{def}}{=} \mathsf{Internet}.(\mathsf{id} \mid \mathsf{I}(l).1 \mid \mathsf{O}(h).1)$
$R_{3c} \longrightarrow R'_{3c}$	$R_{3c} \overset{\text{def}}{=} \mathsf{Properties}(x).(\mathsf{id} \mid \mathsf{MAC}.1) \parallel \mathsf{Properties}'(y)$
	$R'_{3c} \overset{\text{def}}{=} /l\, /h\, (\mathsf{Properties}(x).C \parallel \mathsf{Properties}(y).C')$
	$C \overset{\text{def}}{=} \mathsf{id} \mid \mathsf{MAC}.1 \mid \mathsf{I}(l).1 \mid \mathsf{O}(h).1$
	$C' \overset{\text{def}}{=} \mathsf{id} \mid \mathsf{I}(h).1 \mid \mathsf{O}(l).1$
$R_{4a} \longrightarrow R'_{4a}$	$R_{4a} \overset{\text{def}}{=} /l\, /h\, (\mathsf{Properties}(y).C \parallel (\mathsf{I}(h).1 \mid \mathsf{O}(l).1))$
	$R'_{4a} \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid \mathsf{MAC}.1) \parallel 1$
$R_{4b} \longrightarrow R'_{4b}$	$R_{4b} \overset{\text{def}}{=} /p\, (\mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel P)$
	$R'_{4b} \overset{\text{def}}{=} /p\, (/y\, \mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel \mathsf{Properties}(p).\mathsf{MAC}.1)$
	$P \overset{\text{def}}{=} \mathsf{Properties}(p).(\mathsf{MAC}.1 \mid \mathsf{Hostname}.1 \mid \mathsf{IP}.1)$

Table 4: Reaction rules for network events.

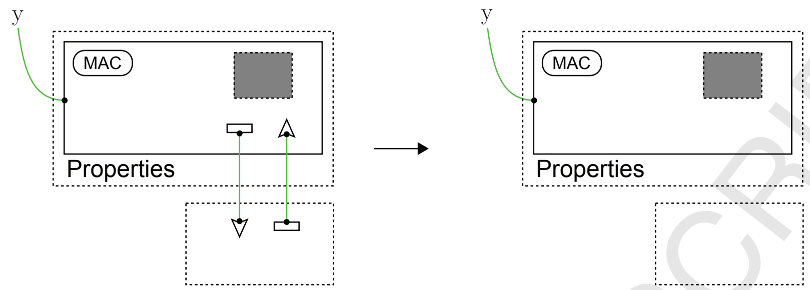


Figure 13: Reaction rule $R_{4a} \longrightarrow R'_{4a}$. A machine leaves the WLAN: Pairs of communication channels are removed.

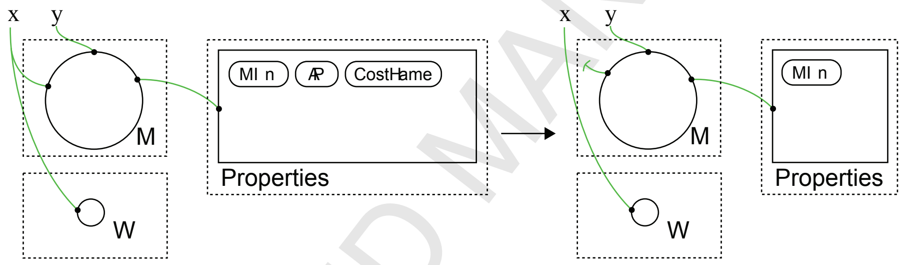


Figure 14: Reaction rule $R_{4b} \longrightarrow R'_{4b}$. A machine leaves the WLAN: DHCP leases are revoked.

Interfaces
$R_1 : \widehat{\mathsf{mr}} \to \langle \widehat{\mathsf{sbr}}, \{y\} \rangle$
$R'_1 : \widehat{\mathsf{mr}} \to \langle \widehat{\mathsf{sbr}}, \{y\} \rangle$
$R_2 : \widehat{\mathsf{mr}} \to \langle \widehat{\mathsf{sbm}}, \emptyset \rangle$
$R'_2 : \widehat{\mathsf{mr}} \to \langle \widehat{\mathsf{sbm}}, \emptyset \rangle$
$R_{3a} : \widehat{\mathsf{pio}} \to \langle \mathsf{mwb}, \{x, y\} \rangle$
$R_{3b} : \widehat{\mathsf{io}} \to \langle \mathsf{mwbj}, \{x, y\} \rangle$
$R_{3c} : \widehat{\mathsf{pio}}\widehat{\mathsf{pio}} \to \langle \mathsf{bb}, \{x, y\} \rangle$
$R_{4a} : \widehat{\mathsf{pio}} \to \langle \mathsf{b}\widehat{\mathsf{io}}, \{x, y\} \rangle$
$R_{4b}, R'_{4b} : \epsilon \to \langle \mathsf{mwb}, \{x, y\} \rangle$

Table 5: Rule interfaces for network events.

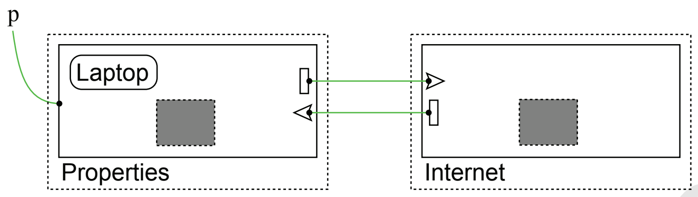


Figure 15: Bigraph encoding predicate “Laptop has Internet connection”.

4. Predicates

Predicates for bigraphs can be expressed in the logic BiLog [6]. However, if we restrict to an intensional fragment of BiLog, which omits the Boolean operators and product and composition adjuncts, then any predicate can be encoded (syntactically) as a bigraph (see [4] for formal details), which can be then be checked by reduction to bigraph matching. We have found this restriction to be suitable for the predicates we require in this application. For example predicates typically express spatial, static properties of the systems such as “TCP traffic is blocked for machine with IP address 192.168.0.3”,“Machine 01:23:45:67:89:ab is in the range of the router’s signal”, and “Laptop has Internet connection”. The latter property is represented by the bigraph in Figure 15.

Definition: Let α be a predicate and B_α its bigraph encoding. Let S be a bigraph. We define $S \models \alpha$ iff B_α is a match in S. $S \not\models \alpha$ denotes B_α is not a match in S.

We often require to reason about whether or not a machine is in the system or part of the WLAN, especially in the context of enforcing or revoking a policy. We therefore define the following two predicates (parameterised by a machine address):

- φ_{MAC} is true iff the machine MAC is present in the system,
- ψ_{MAC} is true iff the machine MAC is part of the WLAN.

The corresponding algebraic forms are:

$$B_{\varphi_{\mathsf{MAC}}} \stackrel{\text{def}}{=} \mathsf{Properties}(p).(\mathsf{id} \mid \mathsf{MAC}.1)$$
$$B_{\psi_{\mathsf{MAC}}} \stackrel{\text{def}}{=} /p\,(\mathsf{M}(y,x,p).1 \parallel \mathsf{W}(y).1 \parallel \mathsf{Properties}(p).(\mathsf{id} \mid \mathsf{MAC}.1))$$

These predicates are encoded by bigraphs $B_{\varphi_{\mathsf{MAC}}}$ and $B_{\psi_{\mathsf{MAC}}}$, depicted in Figure 16.

5. Generating models of network events in real-time

The model of the current configuration is generated and stored in the *Bigraph analysis* component. Note, we generate and store the algebraic form, whereas we use the graphical form for feedback.

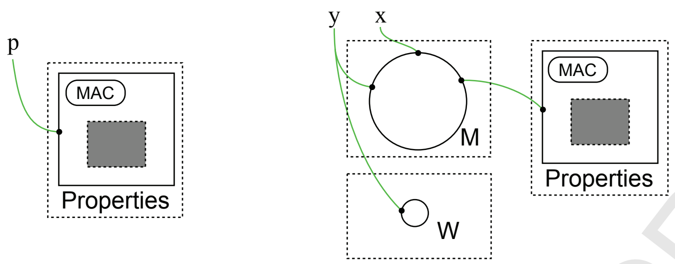


Figure 16: Bigraphs $B_{\varphi_{\mathsf{MAC}}}$ (left) and $B_{\psi_{\mathsf{MAC}}}$ (right).

The reaction rules and predicates defined in the previous section are used to generate sequences of models, e.g. S_0, S_1, ..., from network events. For a given model S_n in a sequence, we generate a successor model S_{n+1} when $S_n \rightarrowtriangle^* S_{n+1}$. Strictly, any model S such that $S_n \rightarrowtriangle^* S$ is a successor model, however, often we store only the model obtained after several rewriting steps, for example when tagging and untagging is required. This means that the sequence of stored models corresponds exactly to the sequence of events. Generation is carried out in real-time, without interruption or delay to the network management system.

An example illustrates the generation process.

Assume the *Stream database* generates a (derived) network event specifying that machine A is present in the system and a DHCP lease has been granted. Let the current model be denoted by S_n, and assume the generated event has been sent to the *Bigraph encoder* component. The sequence of reaction rules to be applied to S_n is determined by whether or not machine A is already present in the system and if it has joined the WLAN. Therefore, the *Bigraph analysis* component is queried to check if $S_n \models \varphi_{\mathsf{A}}$ and $S_n \models \psi_{\mathsf{A}}$. The results are sent back to the *Bigraph encoder* component. We then have three cases of model generation, summarised as follows:

- If $S_n \models \varphi_{\mathsf{A}}$ and $S_n \models \psi_{\mathsf{A}}$, then the system remains unchanged and no reaction rule is applied.
- If $S_n \models \varphi_{\mathsf{A}}$ but $S_n \not\models \psi_{\mathsf{A}}$, then machine A has to join the WLAN. The generated sequence of reactions is: $R_{3a} \longrightarrow R'_{3a}$, $R_{3b} \longrightarrow R'_{3b}$, $R_{3c} \longrightarrow R'_{3c}$, which is sent to the *Bigraph analysis* component to update the model: $S_n \rightarrowtriangle^*_{3a} \rightarrowtriangle_{3b} \rightarrowtriangle^*_{3c} S_{n+1}$. For brevity, we denote this sequence of applications as $S_n \rightarrowtriangle^*_3 S_{n+1}$.
- If $S_n \not\models \varphi_{\mathsf{A}}$, then machine A has to appear in the range of the router and then to join the WLAN. The generated sequence is: $R_1 \longrightarrow R'_1$, $R_{3a} \longrightarrow R'_{3a}$, $R_{3b} \longrightarrow R'_{3b}$, $R_{3c} \longrightarrow R'_{3c}$, which is sent to the *Bigraph analysis* component to update the model: $S_n \rightarrowtriangle_1 \rightarrowtriangle^*_3 S_{n+1}$.

Encodings for the four network events, e.g. move in and out of range, grant and revoke leases, are summarised in Table 6.

Event	Encoding	Notation
Move in range	$\rightarrowtriangle_1$	—
Grant lease	$\rightarrowtriangle^*_{3a}$ $\rightarrowtriangle_{3b}$ $\rightarrowtriangle^*_{3c}$	$\rightarrowtriangle^*_3$
Revoke lease	$\rightarrowtriangle^*_{4a}$ $\rightarrowtriangle_{4b}$	$\rightarrowtriangle^*_4$
Move out range	$\rightarrowtriangle_2$	—

Table 6: Encodings for network events.

Control	Meaning	Sort	Graphical notation
Port, ...	Port number	p	Bold rounded box
WWW, ...	External host	p	Bold rounded box
P, ...	Protocol	p	Bold rounded box
BLOCKED	All traffic forbidden	p	Bold rounded box

Table 7: New controls and sorts for modelling policies.

6. Bigraphical models of policies

Now we turn our attention to the representation of access control policies by reaction rules. Access control policies constrain behaviours, for example they can constrain traffic between machines, or types of traffic. New entities are therefore required. For example, new controls are needed to express the ban of a given port or protocol. The additional controls are listed in Table 7, which we call *constraints*. The formation rule given in Table 3 is also modified by allowing $\widehat{\mathsf{io}}$-nodes to be linked to p-nodes.

Policies are categorised as *forbid* policies or *allow* policies. The latter are relatively simple to represent because matching can detect the existence of a constraint that is required to be removed. However, the representation of *forbid* policies is a little more complex.

The key idea of representing a *forbid* policy is to link chains of p-nodes to communication channels. A chain of constraints represents a conjunction of constraints, and several chains linked to a channel represent a disjunction of constraints. Some policies can be represented by a single reaction rule, whereas others require several when a form of tagging is needed in the representation (because we consider arbitrary network topologies). We illustrate the possible forms of representation with three example *forbid* policies. A summary of the reaction rules (algebraic form) for these policies is given in Table 8.

Policy 1: Consider a policy that forbids the machine named Laptop from receiving incoming traffic from remote host WWW, defined by the reaction rule in Figure 17. This can match only Laptop's properties box, its out-going channel to the external world and Internet box. In the right-hand side, constraint WWW is attached to the channel's link. Note that constraints like WWW are always placed within the sender's $\widehat{\mathsf{bj}}$-box. The inverse reaction $P_1' \longrightarrow\!\!\!\blacktriangleright P_1$ models the policy being dropped.

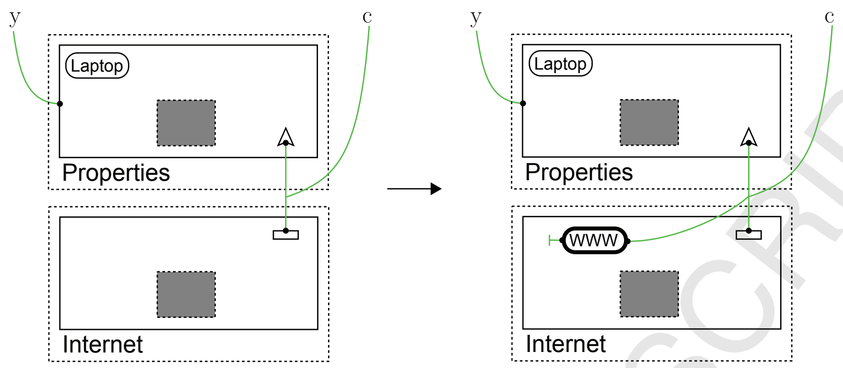


Figure 17: Reaction rule $P_1 \longrightarrow P_1'$. All incoming traffic from WWW to Laptop is blocked.

While this policy (P_1) is represented by a single reaction rule, we note that it must be applied carefully, to avoid multiple or inconsistent applications. The following example illustrates the problem. Consider a bigraph S in which machine Laptop is already forbidden from receiving traffic from WWW, i.e. a WWW-node is already linked to the channel from Laptop to Internet (this is indicated by the open link on name c in P_1). The reaction rule $P_1 \longrightarrow P_1'$ could be applied to this bigraph, and as a result of the rule application, we would obtain a bigraph in which *two* copies of the same constraint are linked to the channel. To avoid this, we must check, before any rule applications for the policy, whether traffic from WWW to Laptop is forbidden. Specifically, the *Bigraph analysis* component is queried to check whether $S \models \varphi_{P_1'}$, where predicate $\varphi_{P_1'}$ corresponds to the bigraph P_1'. The reaction rule for the policy is applied only if the predicate is false (i.e. P_1' is not a match in S). Since the predicate holds for S, reaction rule $P_1 \longrightarrow P_1'$ would not be applied in this case.

Policy 2: A more complex model arises when TCP connections with any host using destination ports 8080 or 6881 and source port 6882 are forbidden. First, rule $P_{2a} \longrightarrow P_{2a}'$ is applied once to all the channels in the system. This results in a bigraph in which all $\widehat{\mathsf{io}}$-nodes are tagged, which is necessary in order to ensure that rule $P_{2b} \longrightarrow P_{2b}'$ is applied only once. Second, rule $P_{2b} \longrightarrow P_{2b}'$ is applied to all the tagged channels; this is depicted in Figure 18. The left-hand side P_{2b} matches any tagged channel. On the right-hand side P_{2b}', the constraints are placed by linking them to the channels and $\widehat{\mathsf{io}}$-nodes are untagged. Constraints on source ports are placed inside the box containing node O (i.e. sender's Properties box), while constraints on destination ports are inside the box containing node I (i.e. receiver's Properties box). The order in which channels are tagged and untagged is irrelevant. Thus, only one interleaving need be considered.

As in the previous example, the *Bigraph analysis* component is queried prior to the application of the reaction rules modelling this policy in order to avoid double entries and inconsistent constraints.

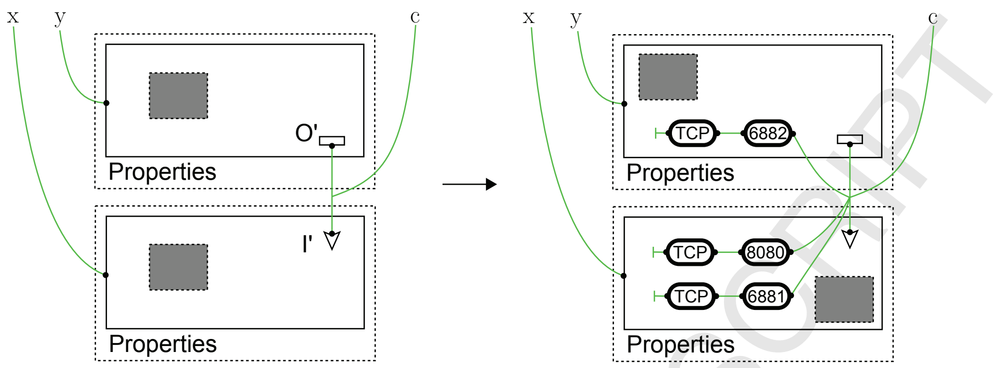


Figure 18: Policy reaction rule $P_{2b} \longrightarrow P'_{2b}$. TCP connections with any host using destination ports 8080 or 6881 and source port 6882 are blocked.

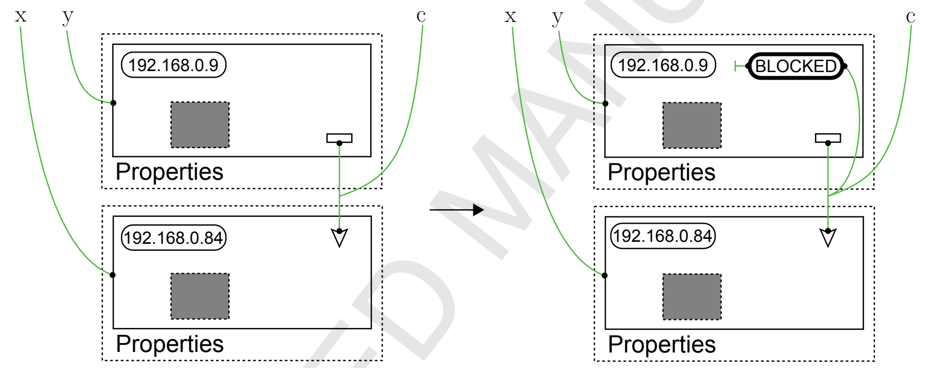


Figure 19: Policy reaction rule $P_3 \longrightarrow P'_3$. Traffic from 192.168.0.9 to 192.168.0.84 is forbidden.

Policy 3: Finally, consider a policy that forbids traffic from host 192.168.0.9 to host 192.168.0.84, defined as a reaction rule in Figure 19. The left-hand side matches the channel blocked by the policy. On the right-hand side, special constraint BLOCKED is linked to the channel.

7. Generating models of policy events in real-time

Reaction rules describing policies are used by the *Bigraph analysis* component to generate sequences of models encoding the policy events generated by the *Stream database* at runtime. The possible policy events are *enforce*, *drop* or *check* policy compliance. *Forbid* policy events are more difficult to encode than *allow* policy events, and so we consider these first.

7.1. Encoding forbid *policy events*

A *forbid* policy is represented by linking constraints (p-nodes) to channels. Again, we employ tagging to indicate when rules may or may not be applicable.

Reaction rule	Algebraic form
$P_1 \longrightarrow P_1'$	$P_1 \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid \mathsf{Laptop}.1 \mid \mathsf{I}(c).1) \parallel \mathsf{Internet}.(\mathsf{id} \mid \mathsf{O}(c).1)$ $P_1' \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid \mathsf{Laptop}.1 \mid \mathsf{I}(c).1) \parallel \mathsf{Internet}.(\mathsf{id} \mid /h\ \mathsf{WWW}(c,h).1 \mid \mathsf{O}(c).1)$
$P_{2a} \longrightarrow P_{2a}'$	$P_{2a} \overset{\text{def}}{=} \mathsf{O}(c).1 \parallel \mathsf{I}(c).1$ $P_{2a}' \overset{\text{def}}{=} \mathsf{O}'(c).1 \parallel \mathsf{I}'(c).1$
$P_{2b} \longrightarrow P_{2b}'$	$P_{2b} \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid \mathsf{O}'(c).1) \parallel \mathsf{Properties}(x).(\mathsf{id} \mid \mathsf{I}'(c).1)$ $P_{2b}' \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid C_s \mid \mathsf{O}'(c).1) \parallel \mathsf{Properties}(x).(\mathsf{id} \mid C_{d1} \mid C_{d2} \mid \mathsf{I}'(c).1)$ $C_s \overset{\text{def}}{=} /q\ (6882(c,q).1 \mid /r\ \mathsf{TCP}(q,r).1)$ $C_{d1} \overset{\text{def}}{=} /q_1\ (8080(c,q_1).1 \mid /r\ \mathsf{TCP}(q_1,r).1)$ $C_{d2} \overset{\text{def}}{=} /q_2\ (6881(c,q_2).1 \mid /r\ \mathsf{TCP}(q_2,r).1)$
$P_3 \longrightarrow P_3'$	$P_3 \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid 192.168.0.9().1 \mid \mathsf{O}(c).1) \parallel \mathsf{Properties}(x).(\mathsf{id} \mid 192.168.0.84().1 \mid \mathsf{I}(c).1)$ $P_3' \overset{\text{def}}{=} \mathsf{Properties}(y).(\mathsf{id} \mid 192.168.0.9().1 \mid /e\ \mathsf{BLOCKED}(c,e).1 \mid \mathsf{O}(c).1) \parallel \mathsf{Properties}(x).(\mathsf{id} \mid 192.168.0.84().1 \mid \mathsf{I}(c).1)$

Table 8: Reaction rules for example policies

In the case of enforce, we employ tagging to ensure that constraints are only added once. In the case of checking policy compliance, the use of tagging is more subtle. The problem we need to overcome is how to check for the *non-existence* of a pattern in a bigraph, namely, we require to check that we *cannot* (bigraph) match the left hand-side of a policy enforcement rule. So, we tag channels that comply with the policy. If all the channels are tagged, then a (bigraph) match is *not* possible, denoted by $\cancel{match}$, and we can conclude the entire model complies with the policy. Thus, for a policy P, we denote by φ_{P} the predicate for compliance with policy P and $B_{\varphi_{\mathsf{P}}}$ the corresponding bigraph.

An explanation of the sequence of reaction rules that encode a *forbid* policy is given below, the rules are summarised in Table 9, assuming a current model S. The rules are grouped according to three functions: **tag**, **enforce/remove/check**, and **untag**. We note that tagging and untagging is required when enforcing or checking a (forbid) policy because we are dealing with an arbitrary topology (with an unknown number of communication channels). Without tagging we would be unable to determine how many channels to check. Moreover, we can't just match patterns of the form $/c(\mathsf{O}(c).1 \parallel \mathsf{I}(c).1)$ to search for channels without constraints because if such match does not exist, it does not assure us that the policy holds. It may not hold because the channel is linked to other constraints.

When a *forbid* policy is to be *enforced*, generate and apply the following

Event	Encoding	Notation
Enforce policy P	$\overbrace{\mathrel{-\!\!\triangleright}^*}^{\text{tag}} \overbrace{\mathrel{-\!\!\triangleright}^*}^{\text{enforce}} \overbrace{\mathrel{-\!\!\triangleright}^*}^{\text{untag}}$	$\mathrel{-\!\!\triangleright}^*_{\mathsf{P}}$
Drop policy P	$\overbrace{\mathrel{-\!\!\triangleright}^*}^{\text{remove}}$	$\mathrel{-\!\!\triangleright}^*_{\not{\mathsf{P}}}$
Check policy P	$S \overbrace{\mathrel{-\!\!\triangleright}^*}^{\text{tag}} T$ B_{φ_P} match $T \Longrightarrow S \not\models \varphi_\mathsf{P}$ B_{φ_P} $\cancel{\text{match}}$ $T \Longrightarrow S \models \varphi_\mathsf{P}$ $T \overbrace{\mathrel{-\!\!\triangleright}^*}^{\text{untag}} S$	—

Table 9: Encodings for *forbid* policy events.

sequences of rules, in order, to the current model:

1. **(tag)** a sequence of rules that tag channels in the model that comply with the policy (i.e. tag the channels that are linked to the appropriate constraint),
2. **(enforce)** a sequence of rules that link the constraint specified by the policy to the un-tagged channels, and then tag these channels so they are not considered again,
3. **(untag)** a sequence of rules that removes the tags applied in steps 1 and 2.

When a forbid policy is *dropped*, generate and apply one sequence of rules:

1. **(remove)** a sequence of rules that removes the policy constraints from channels.

When a *forbid* policy is *checked*, generate and apply the following sequences of rules, in order, to the current model:

1. **(tag)** a sequence of rules that tag channels in the model that comply with the policy (i.e. tag the channels that are linked to the appropriate constraint),
2. **(check)** whether the predicate φ_P holds for the tagged model (from step 1), by attempting to match B_{φ_P}. If a match is possible, i.e. there is an un-tagged channel, then conclude $S \not\models \varphi_\mathsf{P}$, otherwise conclude $S \models \varphi_\mathsf{P}$,
3. **(untag)** a sequence of rules that removes the tags applied in step 1.

7.2. Encoding allow *policy events*

Allow policies are much easier to encode because constraints are removed, instead of being added to the model. Thus, we can take advantage of the fact

Event	Encoding	Notation
Enforce policy P	$\overbrace{\rightarrowtriangle}^{\text{enforce}}{}^{*}$	$\rightarrowtriangle^{*}_{\mathsf{P}}$
Check policy P	$B_{\varphi_{\mathsf{P}}}$ match $S \Longrightarrow S \not\models \varphi_{\mathsf{P}}$ $B_{\varphi_{\mathsf{P}}}$ ~~match~~ $S \Longrightarrow S \models \varphi_{\mathsf{P}}$	—

Table 10: Encodings for *allow* policy events.

that bigraph matching tests for the *existence* of a pattern. An overview of *allow* policy enforce/check is the following, which is also summarised in Table 10. Again, assume current model S.

When an *allow* policy is to be enforced, simply generate and apply a sequence of rules that enforce the policy by removing the relevant constraints. There is no need for tagging.

When an *allow* policy is checked, simply attempt to match $B_{\varphi_{\mathsf{P}}}$. Again, if a match is possible, then conclude $S \not\models \varphi_{\mathsf{P}}$, otherwise conclude $S \models \varphi_{\mathsf{P}}$.

We note that is not possible to drop an *allow* policy. If the user wishes to block some behaviour, it has to be specified explicitly as a *forbid* policy.

7.3. Interplay between network and policy events

When a network event occurs, the *Bigraph analysis* component applies a sequence of reaction rules as described in Section 5. However, this may lead to a system in which some policies are not enforced. For example, assume a current model, S_n, of a WLAN where every machine is forbidden to receive data from remote host WWW. Further, assume a new machine joins the WLAN. As a result, the *Bigraph analysis* component updates S_n to S_{n+1} thus: $S_n \rightarrowtriangle_1 \rightarrowtriangle_3^* S_{n+1}$. But in model S_{n+1}, the new machine is not forbidden from receiving data from WWW, thus the policy has to be re-enforced.

Let us call a policy that has been enforced, but not dropped, an *active* policy. In general, in the bigraph model, active policies need to be checked/enforced/dropped (by the *Bigraph analysis* component) *before* and/or *after* network and policy events. The exact sequence depends on the event. Informally, consider each possible event and the requirements to check/enforce/drop active policies:

1. **grant** a lease - enforce active policies *after* granting a lease, then check all active polices
2. **revoke** a lease - drop active policies *before* revoking a lease, enforce all active polices *afterwards*, then check all active policies
3. move **in**to signal range - check active policies *after* moving into signal range
4. move **out** of signal range - check active policies *after* moving out of signal range
5. **enforce** a policy - enforce new policy, and add new policy to active set, then check all active policies

6. **drop** a policy - drop the policy and remove from active set, then check all active policies.

We can make this more precise as follows, referring to network and policy events by the abbreviations above. For policy ϕ, set of active policies Φ, and sequence of network and policy events S, we define the expansion of a sequence of events according to the function [| _ |] as follows:

1. [| grant; S |] Φ = grant; enforce Φ; check Φ; ([| S |] Φ)
2. [| revoke; S |] Φ = drop Φ; revoke; enforce Φ; check Φ; ([| S |] Φ)
3. [| in; S |] Φ = in; check Φ; ([| S |] Φ)
4. [| out; S |] Φ = out; check Φ; ([| S |] Φ)
5. [| enforce ϕ; S |] Φ = enforce ϕ; check $\{\phi\} \cup \Phi$; ([| S |] ($\{\phi\} \cup \Phi$))
6. [| drop ϕ; S |] Φ = drop ϕ; check $\Phi \setminus \{\phi\}$; ([| S |] ($\Phi \setminus \{\phi\}$))

We illustrate the process, in detail, in the next section.

8. Example of interplay between network events and policy events in real-time

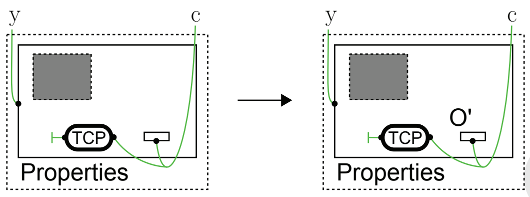
We show step-by-step how updates are made to the current bigraphical model of the WLAN, according to the events from the *Stream database*. We indicate sequences of WLAN models by S_0, S_1, ... Due to confluence properties, we consider only one possible sequence of updates.

Initially, no stations are present, as given by bigraph S_0 in Figure 7. Now consider the following scenario, a summary of which is given in Table 11.

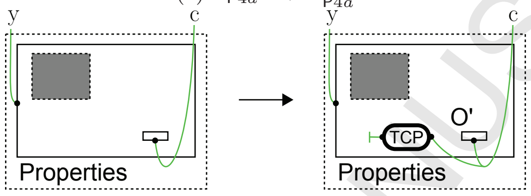
1. The user specifies and enforces a new policy that all out-going TCP traffic for any machine is forbidden. This user-action generates a policy event, which triggers the generation of the reaction rules for a *forbid* policy. We denote the policy by $\mathsf{P4}$ and give the reaction rules in Figure 20. A brief explanation of the rules is the following. Reaction rule $P_{\mathsf{P4}a} \longrightarrow P'_{\mathsf{P4}a}$ tags any out-going channel of any machine that is part of the WLAN[3] and complies with $\mathsf{P4}$. Reaction rule $P_{\mathsf{P4}b} \longrightarrow P'_{\mathsf{P4}b}$ matches any untagged channel and thus it enforces the policy. On the right hand-side, $P'_{\mathsf{P4}b}$, a TCP-node is linked to the matched channel and the channel is tagged (to avoid further treatment). Untagging reaction rule $P_{\mathsf{P4}c} \longrightarrow P'_{\mathsf{P4}c}$ removes the tags. Bigraph $B_{\varphi_{\mathsf{P4}}}$ matches any untagged out-going channel and as described in Section 7, $\mathsf{P4}$ is violated when $B_{\varphi_{\mathsf{P4}}}$ is a match in the temporary state in which all blocked out-going channels are tagged. At this point, the *Bigraph analysis* component enforces $\mathsf{P4}$ on S_0 and we have $S_0 \rhd^{*}_{\mathsf{P4}} S_0$, i.e. no reaction rule is applicable because no machines are present in S_0. Policy $\mathsf{P4}$ is also checked. Since $B_{\varphi_{\mathsf{P4}}}$ is not a match, $\mathsf{P4}$ holds.

2. Machine $\mathsf{MAC1}$ *enters a location covered by the router's signal.* Since $S_0 \not\models \varphi_{\mathsf{MAC1}}$, the *Bigraph encoder* component instantiates[4] reaction rule $R_1 \longrightarrow R'_1$

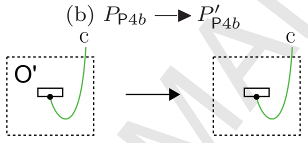
[3]A channel is present thus a DHCP lease has already been granted.
[4]Special control MAC is replaced by actual address $\mathsf{MAC1}$.



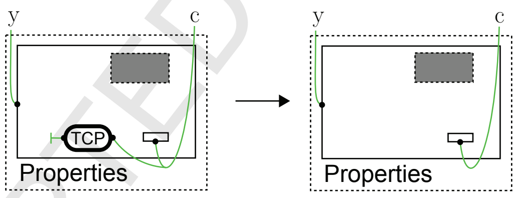
(a) $P_{\mathsf{P4}a} \longrightarrow P'_{\mathsf{P4}a}$



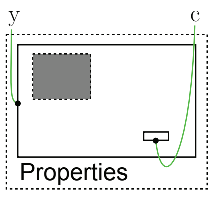
(b) $P_{\mathsf{P4}b} \longrightarrow P'_{\mathsf{P4}b}$



(c) $P_{\mathsf{P4}c} \longrightarrow P'_{\mathsf{P4}c}$



(d) $P_{\mathsf{P4}d} \longrightarrow P'_{\mathsf{P4}d}$



(e) $B_{\varphi_{\mathsf{P4}}}$

Figure 20: Policy $\mathsf{P4}$: forbid all out-going TCP traffic for any machine. Tag sequence is $\longrightarrow\!\!\triangleright^*_{\mathsf{P4}a}$, enforce sequence is $\longrightarrow\!\!\triangleright^*_{\mathsf{P4}b}$, untag sequence is $\longrightarrow\!\!\triangleright^*_{\mathsf{P4}c}$, and drop sequence is $\longrightarrow\!\!\triangleright^*_{\mathsf{P4}d}$. $B_{\varphi_{\mathsf{P4}}}$ is the bigraph for predicate φ_{P4}.

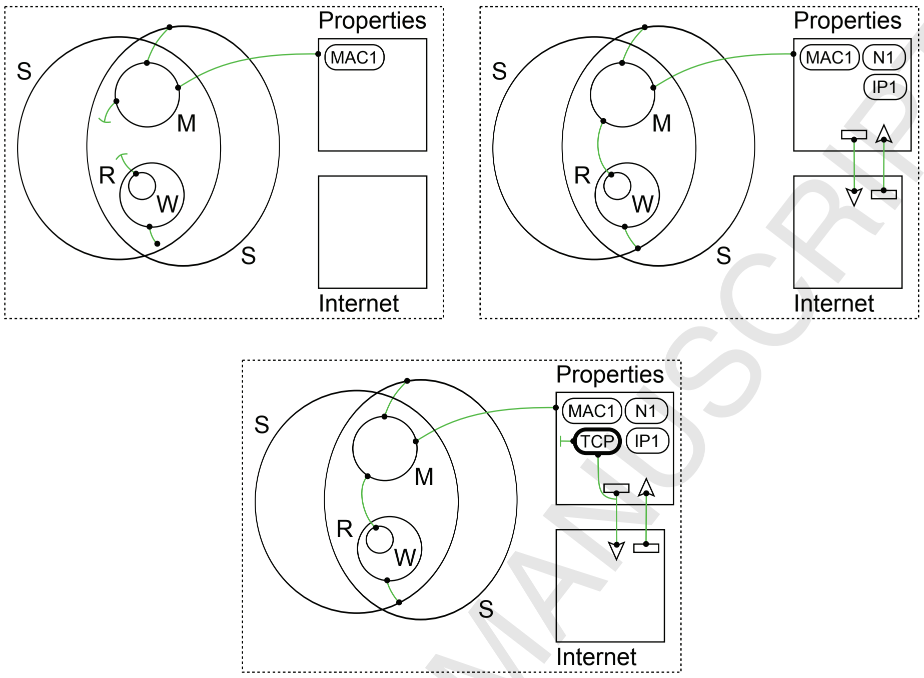


Figure 21: States S_1 (upper left), S_2 (upper right), S_3 (lower). In State S_3 MAC1 is part of the network and P4 is enforced.

and the *Bigraph analysis* component updates the system: $S_0 \rightarrowtriangle_1 S_1$. After this step, the *Bigraph encoder* component checks whether P4 is violated. In this case, reaction rule $P_{\mathsf{P4}a} \longrightarrow P'_{\mathsf{P4}a}$ is not applicable and $B_{\varphi_{\mathsf{P4}}}$ is not a match in S_1. Therefore, the policy is not violated.

3. A DHCP lease is granted to machine MAC1. In the current state we have $S_1 \models \varphi_{\mathsf{MAC1}}$ and $S_1 \not\models \psi_{\mathsf{MAC1}}$. Therefore, the *Bigraph analysis* component updates the system by applying instantiated rules $R_{3a} \longrightarrow R'_{3a}$, $R_{3b} \longrightarrow R'_{3b}$, and $R_{3c} \longrightarrow R'_{3c}$: $S_1 \rightarrowtriangle^*_{3a} \rightarrowtriangle_{3b} \rightarrowtriangle^*_{3c} S_2$. After the topology update, the *Bigraph analysis* component enforces P4 in S_2. The following updates are performed: $S_2 \rightarrowtriangle^*_{\mathsf{P4}a} \rightarrowtriangle^*_{\mathsf{P4}b} \rightarrowtriangle^*_{\mathsf{P4}c} S_3$[5]. We indicate this sequence of rules by $\rightarrowtriangle^*_{\mathsf{P4}}$. States S_1, S_2 and S_3 are shown in Figure 21.

4. MAC2 *enters a location covered by the router's signal.* Since $S_3 \not\models \varphi_{\mathsf{MAC2}}$, the *Bigraph analysis* component performs the same sequence described in step *2* above when machine MAC1 entered the signal range. Specifically, first, the topology is updated with $S_3 \rightarrowtriangle_1 S_4$. Second, P4 is enforced by the usual tagging, matching, untagging sequence: $S_4 \rightarrowtriangle^*_{\mathsf{P4}} S_4$. Observe that no update is performed because reaction rule $P_{\mathsf{P4}b} \longrightarrow P'_{\mathsf{P4}b}$ is not applicable (there are no

[5]In this case $\rightarrowtriangle^*$ is $\rightarrowtriangle$ because only one machine is part of the network in S_2.

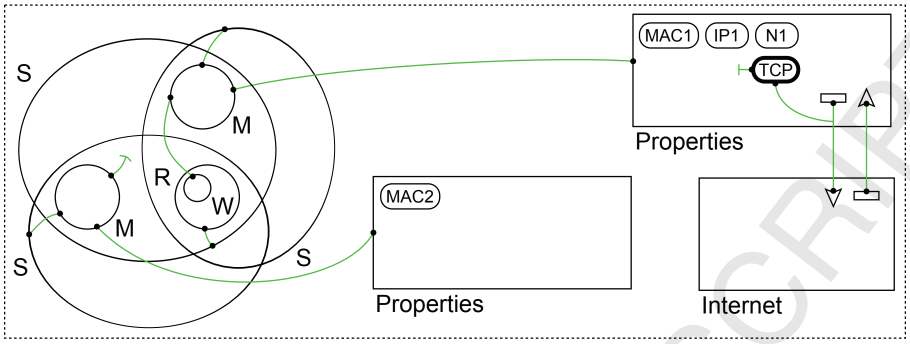


Figure 22: State S_4: MAC2 enters the router's signal range, MAC1 is part of the WLAN and P4 is enforced.

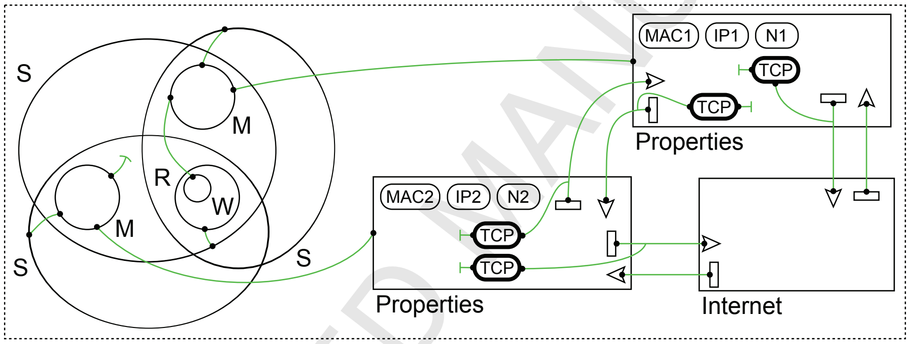


Figure 23: State S_5: MAC1 and MAC2 joined the WLAN and P4 is enforced.

out-going channels requiring to be blocked in machine MAC2). Finally φ_{P4} is checked. Since $B_{\varphi_{\mathsf{P4}}}$ is not a match in T_4, P4 is not violated. The bigraph for updated model S_4 is given in Figure 22.

5. A DHCP lease is granted to machine MAC2. The status of the configuration is $S_4 \models \varphi_{\mathsf{MAC2}}$ and $S_4 \not\models \psi_{\mathsf{MAC2}}$. Hence, the *Bigraph analysis* component updates the model by applying the sequence of reaction rules encoding a join event: $S_4 \longrightarrow\!\!\triangleright_3^* S_4'$. Then, the policy is enforced with $S_4' \longrightarrow\!\!\triangleright_{\mathsf{P4}}^* S_5$. Finally, the *Bigraph analysis* component checks whether φ_{P4} holds. Since $B_{\varphi_{\mathsf{P4}}}$ does not occur in T_5, we have $S_5 \models \varphi_{\mathsf{P4}}$. The bigraph for S_5 is given in Figure 23.

The sequence of events and the corresponding model updates described are summarised in Table 11.

9. Bigraph model analysis

At any point in the model generation process we can check whether the bigraphical representation of the current system satisfies compliance with a policy, or an invariant, or indeed any property that can be defined in the fragment

Event	Updates	WLAN model	Policy
1. $\mathsf{P4}$ enforced	$S_0 \rightarrowtriangle^*_{\mathsf{P4}} S_0$ $S_0 \rightarrowtriangle^*_{\mathsf{P4}a} T_0$ $B_{\varphi_{\mathsf{P4}}}$ ~~match~~ T_0 $T_0 \rightarrowtriangle^*_{\mathsf{P4}c} S_0$ check $\mathsf{P4}$	S_0	$S_0 \models \varphi_{\mathsf{P4}}$
2. $\mathsf{MAC1}$ in signal range	$S_0 \not\models \varphi_{\mathsf{MAC1}}$ $S_0 \rightarrowtriangle_1 S_1 \rightarrowtriangle^*_{\mathsf{P4}} S_1$ $S_1 \rightarrowtriangle^*_{\mathsf{P4}a} T_1$ $B_{\varphi_{\mathsf{P4}}}$ ~~match~~ T_1 $T_1 \rightarrowtriangle^*_{\mathsf{P4}c} S_1$ check $\mathsf{P4}$	S_1	$S_1 \models \varphi_{\mathsf{P4}}$
3. $\mathsf{MAC1}$ lease granted	$S_1 \models \varphi_{\mathsf{MAC1}}$ and $S_1 \not\models \psi_{\mathsf{MAC1}}$ $S_1 \rightarrowtriangle^*_3 S_2 \rightarrowtriangle^*_{\mathsf{P4}} S_3$ $S_3 \rightarrowtriangle^*_{\mathsf{P4}a} T_3$ $B_{\varphi_{\mathsf{P4}}}$ ~~match~~ T_3 $T_3 \rightarrowtriangle^*_{\mathsf{P4}c} S_3$ check $\mathsf{P4}$	S_3	$S_3 \models \varphi_{\mathsf{P4}}$
4. $\mathsf{MAC2}$ in signal range	$S_3 \not\models \varphi_{\mathsf{MAC2}}$ $S_3 \rightarrowtriangle_1 \rightarrowtriangle^*_{\mathsf{P4}} S_4$ $S_4 \rightarrowtriangle^*_{\mathsf{P4}a} T_4$ $B_{\varphi_{\mathsf{P4}}}$ ~~match~~ T_4 $T_4 \rightarrowtriangle^*_{\mathsf{P4}c} S_4$ check $\mathsf{P4}$	S_4	$S_4 \models \varphi_{\mathsf{P4}}$
5. $\mathsf{MAC2}$ lease granted	$S_4 \models \varphi_{\mathsf{MAC2}}$ and $S_4 \not\models \psi_{\mathsf{MAC2}}$ $S_4 \rightarrowtriangle^*_3 \rightarrowtriangle^*_{\mathsf{P4}} S_5$ $S_5 \rightarrowtriangle^*_{\mathsf{P4}a} T_5$ $B_{\varphi_{\mathsf{P4}}}$ ~~match~~ T_5 $T_5 \rightarrowtriangle^*_{\mathsf{P4}c} S_5$ check $\mathsf{P4}$	S_5	$S_5 \models \varphi_{\mathsf{P4}}$

Table 11: Generation of models $S_0 \rightarrowtriangle \cdots \rightarrowtriangle S_5$.

of BiLog. For example, we check invariants after every update of the system, logging any violations and reporting them, as required, to the system and/or user. We can also detect conflicting policies, as follows. We assume the right-hand side of reaction rules for policies as invariants. A new policy conflicts with an existing one whenever its application invalidates an invariant. As a simple example, consider reaction rule $P_1 \rightarrowtriangle P_1'$ and its inverse $P_1' \rightarrowtriangle P_1$ introduced in Section 6. Assume that Laptop is the only machine in the system and no constraints are in place. Call the bigraph representing this state S_n. When the system is updated by $P_1 \rightarrowtriangle P_1'$, right-hand side P_1' is adopted as an invariant. The evolution of the system is given by $S_n \rightarrowtriangle_1 S_{n+1}$. Now consider an application of the inverse rule, so the system evolves: $S_{n+1} \rightarrowtriangle_{1'} S_{n+2} = S_n$. At this point the invariant is checked. Since P_1' is not a match in S_{n+2} (node of control WWW cannot be matched), then $S_{n+2} \not\models P_1'$, thus indicating a conflict between the two policies. This is a simple example: a policy and its inverse are trivially in conflict and in this case the policies are implemented by single rules.

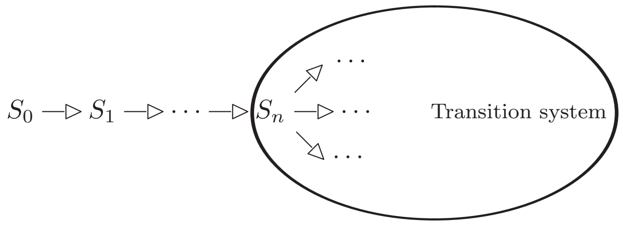


Figure 24: Generating all possible evolutions from current state S_n.

In general, checking for conflicts will be more complicated because a policy is implemented by a sequence of reactions (e.g. because of tagging/untagging). In any event, the run-time system can either indicate this to the user, deny the enforcement of the second policy, or just keep track of conflicts in a logfile.

It is possible to reason about the evolution of the system with *temporal* properties such as "*Eventually* machine 01:23:45:67:89:ab will be connected to Laptop", "TCP traffic is *always* blocked for machine with IP address 192.168.0.3", "A lease is granted to machine 01:23:45:67:89:ab *until* it is not in the range of the router's signal". These properties can be expressed in an appropriate (e.g. linear or branching time) temporal logic and then checked in a transition system of all possible evolutions, generated from the current state. See Figure 24 for an illustration. In order to generate a finite structure, a fixed set of machines and policies would have to be specified. Further, to reflect likely user behaviours, allowable events have to be specified (otherwise, from any state we could return to S_0). For example, we might wish to reason about future behaviour, based on the assumption that *no* machines leave the network, or *no* new policies are enforced.

Given a finite transition system, checking a temporal property involves bigraph matching for state formulae and standard model checking techniques for the temporal operators. The latter is computationally expensive and may not be tractable in real-time, depending on the number of machines and policies and on the temporal formula. So far, we have not found a need for temporal properties: state formulae are currently sufficient for all verification needs expressed by the Homework system users.

10. Implementation

A prototype system is fully implemented on the Homework router, which is hosted on a variety of small form-factor PCs. The bigraph generation is implemented in OCaml; the matching engine is based on the MiniSat solver [7] and is written in C++. The *Bigraph encoder* and *Bigraph analysis* components are part of the more extensive BigraphER (Bigraph Evaluator and Rewriting) System [8]. The software runs on a standard Linux Ubuntu distribution. Access control is enforced via NOX (which implements the custom DHCP server) and Open vSwitch, as dictated by the Ponder2 policy engine [9], based on events recorded in the Homework database.

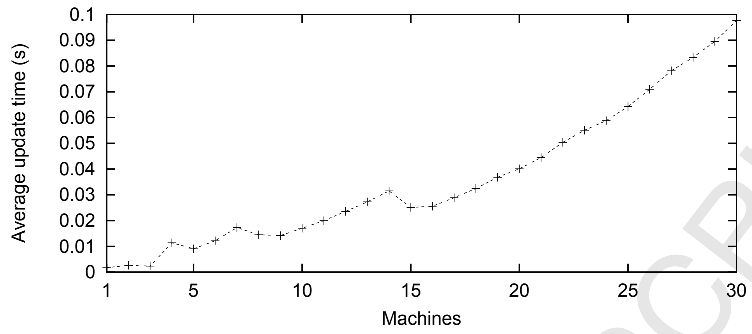


Figure 25: Average time to perform an update as a function of the number of machines in the network (x axis). Each update was performed 100 times and the average time is reported on the y axis.

We trialled the system with both synthetic and experimental data using a router hosted on an Asus Eee PC laptop with the following specification: 1.2GHz Intel Atom CPU, 2GB RAM, 200GB SATA HDD, 802.11b/g, 1Gbit ethernet, and a USB-to-ethernet adapter.

For the synthetic data, we added 30 stations to the initial configuration, firing reaction rule $R_1 \longrightarrow R_1'$ 30 times starting from bigraph S_0. The final state, a network with 30 stations, is a bigraph with 123 nodes. The time to update the (network) bigraphs increases with the number of nodes, as indicated in the graph of update times averaged over 100 runs, as shown in Figure 25. Note that the slowest update requires just under 0.10s.

Experimental data was taken from actual network trials. For example, the router sensed the signals of 6 stations, then 4 new devices joined the WLAN and were connected to the Internet. The final state was a bigraph with 71 nodes. The update times were similar to those shown above. Evidence from network trials suggests there are rarely more than 20 signals present in a home network and the rate of topology change is much slower than the times used in our (synthetic) experiment. Moreover, our times include a system overhead to generate and store on disk a graphical representation of each bigraph (involving an external invocation of the graph layout generator `dot`). While we expect that considerable speed-ups and optimisations are possible to the verification system, we conclude that the prototypical system can update and analyse the bigraphical representations of actual home networks in real-time. We note that the current implementation contains some optimisations. For example, when a DHCP lease is revoked we drop active policies *only* for the machine concerned (cf. Section 7.3). This means that we can skip the enforce step and so the sequence of events, for machine M, is: drop Φ on M; revoke; check Φ.

11. Discussion

In this section we give an overview of modelling and design decisions we have made and the implications for our overall approach.

11.1. Why model with BRS?

Our models are one of the first applications of BRS to a real world problem and modelling the management of wireless networks with BRS has some advantages over other process calculi. Arguably the most important advantage is the ability to express spatial aspects of computation in a natural, hierarchical way, a feature lacking in formalisms such as the π-calculus and CCS where the underlying spatial structure is assumed to be flat. Cardelli and Gordon's calculus of mobile ambients [10] is closer to BRS and allows the location hierarchy to be organised into a tree structure. Additionally, this structure can be represented graphically by using boxes to encode locations and the nesting of boxes to encode their topology. However, computation is encoded solely by changes in the spatial structure. In BRS, computation can also be encoded by changes in the link graph structure. While these process calculi are fundamentally equivalent, bigraphs allow for an easier representation of complex systems by keeping the concepts of space and computation separated. In contrast, as a consequence of the structural operational semantics of process calculi, non-trivial protocols are required when encoding complex state modifications.

Nevertheless we did encounter one drawback with BRS modelling, which arises from the declarative nature of reaction rules (and suffered by all process calculi). Recall, we employed tagging in our reaction rules to overcome problems in three scenarios:

1. the requirement to apply a rule n times, because we have *arbitrary* topologies,
2. the requirement to ensure that we do not duplicate the application of a rule,
3. the requirement to encode universal quantification (e.g. there does not exist a non-match for a given predicate).

Essentially, these problems are a consequence of the declarative nature of rewrite rules and rewriting: in these scenarios we require a notion of control. We could, for example, define that control explicitly, with a reaction rule for each value of n. But this is rather clumsy and would obscure the clarity of the model; it also assumes we can define a static upper bound. A better solution would be to introduce parameterised reaction rules into BRS, as syntactic sugar, thus avoiding the need for tagging. This is future work.

11.2. Do we have the right abstraction?

We currently model exactly what the router senses and the subsequent events stored in the *Stream database*. But we could add additional features such as the physical location of devices, or ownership of machines, if they can constrain behaviour. Moreover we could also add aspects of real-time, such as the rate of events or timed policies (e.g. a policy applies between 18:00 and 24:00 every day, or on a given day), if these become relevant. More generally, we aim to model the events recognised and stored by the Homework network management system; if the system monitors more detailed behaviour, then the abstraction and the subsequent formal models must reflect that.

11.3. Do we have the abstraction right?

In other words, are our models faithful to the actual system, do they abstract from the same semantics? This is a problem for any modelling approach. In the case of network events, it is fairly straightforward to see our abstraction is faithful. In the case of the policies, there are often subtleties about how exactly to interpret a policy, which may depend on user intentions. We have presented policy enforcement and dropping in detail deliberately, so as to expose these subtleties and why and how user comprehension and implementation can be difficult. Throughout the project we have discussed policy design and implementation with the developers of the policy engine, using the bigraph graphical notation. This has provided informal validation that bigraph models can help policy language designers agree on interpretations and resolve ambiguity.

11.4. Can we use the sequence of models in other ways?

The approach we have described here is a form of runtime verification, in that we construct a directed simulation and test for behaviors satisfying or violating certain properties. Our approach is not based on data traces or deadlock detection, rather it is a form of runtime formal modelling. While this has significant overheads, compared to more conventional runtime verification, we propose that the additional requirements of behaviour that is both spatial *as well as* temporal, can justify this. The result is much richer feedback that informs the user's cognitive model. For example, traditional verification might report that a packet was observed on a channel, or a variable has a particular value, whereas we can report a much more detailed model (of the state of the system) that explains current spatial relationships and communication links. We could make further use of the sequence of generated models, for example for debugging and for supporting understanding the causes of failure (e.g. we could rollback and replay), or for generating tests from the results of the failed verifications.

11.5. Can we improve feedback?

As we have stated earlier, our long-term motivation is to aid users in their understanding of the state of their system, and to give feedback to developers about user experiences. We have not yet carried out a formal evaluation of this, but informal evaluations from the (Homework) system developers is that the bigraphical feedback from the runtime verification is helpful to them. However, while bigraphs permit a straightforward and intuitive graphical representation, we conjecture that our system representations, which are based on Venn diagrams, may be too unfamiliar and/or detailed for non-expert users. We plan to experiment with other graphical representations that can be generated automatically from our bigraphs. For example, can we generate the cartoon in Figure 2 from the bigraph in Figure 3; can we generate the cartoons used in the Homework system drag and drop interface for policies? Are there constraints on the style of cartoons that can be generated from our bigraphs? Furthermore, we may not wish to reveal all the detail initially, but, for example, only present properties when the user rolls a mouse over the (representation of) a machine. This will be future work.

12. Related work

There is a significant body of work on analysis of policies and conflicts in the context of network management, for example [11], but we are unaware of any approach involving real-time analysis of process calculi models.

Runtime models for managing the complexity of evolving software behaviour while it is executing is a recent area of interest, particularly in the domain of self-adaptation (e.g. recent Dagstuhl seminar [12]). While our domain is different, our work has a similar goal in that it is reasoning about context-dependencies at runtime. We note a related online, event-driven formal modelling approach taken in [13] to checking configurations of a home-care application. In this case, the configuration data was streamed from log files of user activity; empirical use of the reasoning system also revealed state-based reasoning was sufficient and there was no need for temporal operators.

13. Conclusions

We have extended the Homework network management system with runtime verification comprising real-time generation and analysis of bigraphical models of network topology, network events and access policies. This work represents one of the first applications of bigraphical modelling to a real world problem, and a novel use of process algebraic modelling in a runtime verification context.

Both standard network events, such as a machine entering a signal range or having a DHCP lease revoked, and forbid and allow access policy enforcement or dropping, are represented by bigraph reaction rules. We have presented the full details of the representation so as to expose the precise details of how policies work and their interplay with network events.

Many rules involve the concept of "tagging" entities, to limit the scope of applicability of the rules. While this adds some complexity to the representation, it is an inevitable consequence of reasoning about complex computation with declarative rules.

The real-time generation of bigraph models is *event-driven*: we apply the reaction rules of a bigraphical reactive system, according to events captured in the Homework stream database. In essence, we generate a real-time simulation trace, where states are bigraph representations of the live system: at each step, the bigraph state is checked for invariants and violations are reported to the user.

Verification is done via a bespoke software component, based on reasoning about predicates by bigraph matching, encoded in a SAT solver. The verification system is fully implemented on the router and our experiments indicate that model generation and analysis can be carried out in real-time. We have outlined how to model-check temporal properties, but so far there is little evidence from user trials that temporal operators are required.

Future work will be in three areas: feedback, efficiency, and quantitive behaviour. The first includes developing and evaluating different forms and type of feedback about the outcomes of the verification process to both the user and the system. For example, we will extend the system so that the *Bigraph analysis* component communicates directly with the *Stream database* component. Whenever an invariant is not satisfied, the violation is recorded in a table in the database; users and diagnostic applications can subscribe to the table and obtain real-time updates of the system status. We will also explore (graphical) abstractions of bigraphical representations that may be more accessible and/or meaningful to users, and conduct user trials with them. We will consider extending the matching engine in the *Bigraph encoder* component to support regular expressions on controls. This can greatly reduce the number of matching instances, especially when reaction rules involve ranges of addresses. Finally, we will extend the modelling to stochastic bigraphs [14], to represent the rate of traffic and bandwidth capabilities, thus extending the range of policies we can consider.

Acknowledgments

This work is supported by the Homework Research Project, funded by the Engineering and Physical Sciences Research Council, under grant EP/F064225/1.

[1] J. Sventek, A. Koliousis, O. Sharma, N. Dulay, D. Pediaditakis, M. Sloman, T. Rodden, T. Lodge, B. Bedwell, K. Glover, An Information Plane Architecture Supporting Home Network Management, Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (2011) 1–8.

[2] R. Mortier, B. Bedwell, K. Glover, T. Lodge, T. Rodden, C. Rotsos, A. W. Moore, A. Koliousis, J. Sventek, Supporting novel home network management interfaces with openflow and nox, SIGCOMM Comput. Commun. Rev. 41 (4) (2011) 464–465. doi:10.1145/2043164.2018523.

[3] R. Milner, The space and motion of communicating agents, Cambridge University Press, 2009.

[4] M. Sevegnani, Bigraphs with sharing and applications in wireless networks, PhD thesis, University of Glasgow (2012).

[5] L. Birkedal, T. C. Damgaard, A. J. Glenstrup, R. Milner, Matching of bigraphs, Electronic Notes in Theoretical Computer Science 175 (4) (2007) 3 – 19, Proceedings of the Workshop on Graph Transformation for Concurrency and Verification (GT-VC 2006). doi:10.1016/j.entcs.2007.04.013.

[6] G. Conforti, D. Macedonio, V. Sassone, Spatial logics for bigraphs, in: L. Caires, G. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), Automata, Languages and Programming, Vol. 3580 of LNCS, Springer, 2005, pp. 766–778.

[7] N. Eén, N. Sörensson, An extensible SAT-solver, in: SAT 2003, LNCS vol. 2919, 2003, pp. 502–518.

[8] M. Sevegnani, BigraphER.
URL `http://www.dcs.gla.ac.uk/~michele/bigrapher.html`

[9] K. Twidle, N. Dulay, E. Lupu, M. Sloman, Ponder2: A policy system for autonomous pervasive environments, The Fifth International Conference on Autonomic and Autonomous Systemsdoi:10.1109/ICAS.2009.42.

[10] L. Cardelli, A. D. Gordon, Mobile ambients, Electr. Notes Theor. Comput. Sci. 10 (1997) 198–201.

[11] A. Bandara, J. Lobo, Calo, E. S. Lupu, A. Russo, M. Sloman, Toward a Formal Characterization of Policy Specification Analysis, in: Annual Conference of ITA (ACITA), University of Maryland, USA, 2007.

[12] U. Aßmann, N. Bencomo, B. H. C. Cheng, R. B. France, Models@run.time (Dagstuhl Seminar 11481), Dagstuhl Reports 1 (11) (2012) 91–123.
URL `http://drops.dagstuhl.de/opus/volltexte/2012/3379`

[13] M. Calder, P. Gray, C. Unsworth, Is my configuration any good: checking usability in a sensor-based activity monitor, Innovations in Systems and Software Engineering (2013) in press,doi:10.1007/s11334-013-0203-1.

[14] J. Krivine, R. Milner, A. Troina, Stochastic bigraphs, Electr. Notes Theor. Comput. Sci. 218 (2008) 73–96.