# Instituto Tecnológico y de Estudios Superiores de Monterrey
## Campus Guadalajara

### Escuela de Graduados en Ingeniería y Arquitectura (EGIA)

### Maestría en Ciencias Computacionales

# Applying Model Checking to Detect Conflicts in Rule-Based Internet Plans

**AUTOR:** Alvaro Maza Maza

**ASESOR:** Dr. Gerardo Padilla Zarate

Guadalajara (Jal), 01 de Octubre de 2015

# Abstract

TBD

# Acknowledgments

TBD

Dedicated to my parents

# Table of Contents

# List of Tables

# List of Figures

# Listings

# CHAPTER 1

# Introduction

## 1.1 Background

Internet is a collection of inter-connected private networks. These networks are operated by many Internet service providers (ISPs) or Network Operators in the world. How these networks interact with each other has strongly changed over the past years.

Just a decade ago, Internet users were just starting to use high-speed broadband access. By that time, there were only 569 million Internet users and we could download a song in around 10 minutes. Now, things changed and we are over 2.27 billion Internet users in the world and Real-time entertainment is, by far, the largest category of Internet traffic.

In order to sustain profitable business models, network operators are now looking at new ways of pricing models, beyond the standard expectation of volume-based plans [1].

Particularly, mobile operators have realized that they need to offer a number of flexible and smartly designed data packages, aligned to the budgets of different customer segments and target groups. As an example, several mobile operators are now offering free access to certain applications such as Facebook or WhatsApp.

## 1.2 Problem Statement and Motivation

During the development of Broadband Plans, Network Operators specify several rules to make each plan. A rule generally defines an action to either allow or block certain applications. More complex rules, add additional constraints, such as a predefined bandwidth or a limited amount of bytes to be consumed over a period of time.

While a Broadband Plan is composed of one or more of these rules, they are usually designed individually and developed independently to offer new plan options to customers. When rules are added to a particular plan, it may conflict with existing rules of the plan in some unexpected ways. For instance, when a new rule, such as *Unlimited Facebook*, which allows unlimited access to Facebook traffic, is added to a plan already including a rule to limit up to 1GB of monthly usage, a conflict happens in the case when Facebook traffic is generated and these two rules are configured. The reason is that these two rules can be applied to the Facebook traffic.

Such conflicts are best resolved as early as possible – ideally before such rules are implemented. This work is motivated by the desire to detect rule conflicts at the specification stage, using Model Checking. This may help prevent time-consuming and high-cost investment in implementations based on incorrect specifications. And this may also help in fixing the requirements of the plans or rules. For example, a solution to solve the conflict above is to assign a higher priority to the *Unlimited Facebook* rule than to the general rule of the plan.

In order for us to use Model Checking to detect conflicts between rules within a plan, we first need to design a generic model to specify the different plans. Afterwards, we can use a formal verification language to exhaustively and automatically check whether the model meets the plan specification and to detect conflicts between the rules in the plan.

## 1.3    Aim and Objectives

### 1.3.1     Aim

- Detect conflicts between rules, within a rule-based Internet Plan, using Model Checking.

### 1.3.2     Objectives

- Use a verification modeling language to specify the Rule-Based Internet plans with their rules.

- Exhaustively and automatically check whether the model meets the specification of the plan.

- Detect if there is any conflict between two or more rules within the rule-based Internet plan.

## 1.4    Scope, Limitations, and Delimitations

The model described in this Thesis was based on the fact that Internet plans, and Network Policies, have been commonly expressed in the form of rules and actions. Network specifications evolved from Network Policy languages, found in [2, 3, 4], to the well-known and currently widely-used 3GPP Architecture, found in [5, 6].

Network operators using 3GPP-based technologies, requires to specify their plans in the form of rules and actions. 3GPP defines a 5-tuple set of classifiers: source IP address, destination IP address, source port number, destination port number, and protocol ID of the protocol above IP to identify a flow. However, many Network Operators are also extending them, to other classifications such as protocol/application, location and device.

For the verification of the model, Linear temporal logic (LTL) was used to exhausti-vely and automatically check whether the model meets the plan specification and to detect conflicts between the rules within the plan.

## 1.5     Justification

World's population is expected to exceed 7.6 billion early in 2020, up from the current 7.2 billion; while the number of internet-connected devices is expected to double in 5 years, from 25 billion, now in 2015, to 50 billion by 2020 [7].

With this proliferation of Internet-connected devices, essentially the number of Internet-connected devices in the world has grown faster than the number of people in the World, network operators are trying to differentiate themselves by offering new ways of pricing models, beyond the usual vectors of volume and bandwidth.

In the world of fast-food restaurants, we have seen an evolution in the last 5 years. At the very beginning, the meals were fixed and simple – for example a meal could consist in one hamburger, one portion of fries, and one drink. Now the meal options got more varied and, in some fast-food chains, it is even possible to choose all the *features* in your meal: which kind of bread, which kind of ham, which dressing, and so on [8].

On the other hand, in the world of Retail, we see examples of a new micro payment economy. Rather than long term commitments, people are making compulsives small pur-chases. As an example we can see people renting a house for weekly basis, or hourly renting a car, or even purchasing certain songs and not entire albums.

In the world of broadband data, network operators are also rethinking the notion of the long term contracts and fixed tiered plans. Network Operators are basically moving from unlimited or volume-based models to value-added data offerings to basically offer

price transparency and access to certain applications for specified amounts of time, at affordable rates [9]. For example, Vox Telecom, a DSL provider in South Africa, partnered with a local retailer around the launch of the game "Call of Duty: Black Ops II" to provide 40GB of "gamer optimized" ADSL Bandwidth to customers who redeemed a voucher included with the purchase of the game [10]. Similarly, China Mobile currently offers "Lite Data Service Plans" at reduced prices, where heavy-data-usage applications, such as Peer-To-Peer applications, are blocked, in order to rationalize the use of the data plans [11].

With all this spectrum of possible plans based not only on the volume consumed but also based on the applications, device or location used, there is a potential need to identify conflicts amongst the rules that are defined within the plans. As time goes on, more smart plans will begin to emerge and Network Operators will definitely need tools to optimally find the rules that are compatible to make their plans – or to even allow their subscribers to make their own plans.

## 1.6   Hypothesis

This thesis will intend to prove the hypotheses listed below:

- **H1:** Rule-based Internet plans can be abstracted and specified in a verification modeling language.

- **H2:** Model Checking can be used to verify whether the model meets the specification of the plan and to detect conflicts between the rules within the plans.

## 1.7   Methodology

TBD.

## 1.8    Contribution of the Thesis

The main contributions of this Thesis are:

- Provide a framework to specify rule-based Internet plans via a verification modeling language.

- Provide a number of LTLs to exhaustively and automatically check whether the model meets the plan specification and to detect conflicts between the rules within the plans.

### 1.8.1    Target Audience

The following groups encompass the target audience for this Thesis:

- Network Operators.

- Internet Service Providers (ISPs).

- Future graduate students.

## 1.9    Organization of the Thesis

The Thesis is organized as follows:

1. In Chapter 1, the introduction to the thesis is provided which also includes our motivation and defines its scope.

2. In Chapter 2, we review the State of the Art.

3. In Chapter 3, we introduce our model to specify the Rule-based Internet Plans, and the LTL to be used to verify them.

4. In Chapter 4, we introduce MexCom, a not real Mexican Mobile Operator company offering use-cases we found in the Public Domain.
   .

5. In Chapter 5, we validate the use-cases described in the previous chapter with the model studied in chapter 3.

6. In Chapter 6, we analyse and discuss the findings obtained.

7. We conclude in Chapter 7 by discussing further improvements and future work.

# CHAPTER 2

# Literature Review

## 2.1  Mobile Broadband services

### 2.1.1  Evolution

There seems to be a consensus on how mobile broadband services have evolved and, more importantly, in which direction they go. In 2012, Kimbler and Taylor characterized the evolution of strategies for mobile broadband data services in three phases as follows [9]:

- Phase One: Unlimited data plans.

- Phase Two: Volume-based charging.

- Phase Three: Value-added mobile broadband services.

Ezziane was not wrong when in 2005 predicted that the traditional flat rate will no longer be valid. Instead, accounting and billing for emerging 3G services will be content-based and usage-based [12].

In 2013, Sen et al, introduced the term Smart Data Pricing (SPD), described as a broad set of ideas and principles that go beyond the traditional at-rate or byte-counting models. Such SDP models can include any of the following mechanisms: Time-based, location-based, application-based, quota-aware content distribution.

| Unlimited Data Plans | Volume-based Charging | Value-added Services |
|---|---|---|
| • Flat-Rate.<br>• No Control on Consumption. | • User pays a flat price up to a predetermined volume of traffic.<br>• Beyond their quota, the user may be charged in proportion to the usage. | • Application-specific services.<br>• Internet bundle.<br>• Partner content/application subscriber portals.<br>• Device-based plans.<br>• Location-based plans.<br>• Parental Control<br>• Family plans.<br>• Personalization. |

*Figure 2.1:* Evolution of the Mobile Broadband services

Later in 2013, Hart and Brown defined similar services to be enabled on an LTE network: Fair usage, Time of Day, Parental Control, Shared Data Plans, Turbo Boost, Bundling of Popular Services (e.g. Twitter, Facebook, etc.) and HD Voice and Video [13].

Similarly to the authors mentioned above, telecommunication companies have also published their research regarding this topic. For example in 2012, Elitecore Technologies listed a number of emerging use cases, such as: turbo boost, family plans, application-based & device-based plans, service personalization, etc [14]. While Sandvine, which offers solutions to communication service providers (CSPs), talks about: Application-based service tiers and bolt-ons, roaming packages, shared data plans, sponsored connectivity and third-party bundles [15]. Nowadays, Alcatel Lucent and Allot Communications, other telecommunication companies, are also offering solutions about application-based charging, turbo boost and Parental Control [16, 17].

It is clear that network operators have already started to create value added data services and specialized packages that can be sold on top of or instead of the traditional tiered volume-based data packages. This approach is allowing operators to create incremental data revenues and effectively monetize their data traffic in an smarter way.

## 2.1.2 Examples of value-added services

The table 2.1 below, provides an overview of some of the most common volume and value added data services available today. The real advantage of value-based data plans is that they are designed to meet the wants and needs of the end-users and customers [9].

| Pricing Policy | Description | Example of Global Telco's |
|---|---|---|
| *Access Network* | Policy based on the access network. e.g. 2G, 3G, 4G, WiFi, Wi-Max. | AT&T, Verizon, T-Mobile (USA), Movistar (Colombia), MTS(Russia), Aircel, Celcom, Singtel. |
| *Tiered Plans* | Different Quota-based plans with FUP and overage. | AT&T, Verizon, T-Mobile (USA), Celcom, Movistar (Colombia), Singtel, Aircel, Vodafone (India). |
| *Device-based plans* | Plans based on the accessed device. e.g. iPad, iPhone, Smart TV, Laptop | AT&T, Verizon, T-Mobile (USA),Celcom, Movistar (Colombia), MTS(Russia), Aircel, Singtel. |
| *Parner-based plans* | Partner-based plans enable operators offer content at subsidized rate. | AT&T, Verizon, T-Mobile (USA), Maxis, Celcom, SingTel, Aircel, Vodafone (India). |
| *Parner-based plans* | Partner-based plans enable operators offer content at subsidized rate. | AT&T, Verizon, T-Mobile (USA), Maxis, Celcom, SingTel, Aircel, Vodafone (India). |
| *Time-based plans* | Time of day, hourly plans, Seasonal plans. | AT&T, Movistar (Colombia), MTS (Russia), Maxis, Celcom. |
| *Service Personalization* | Subscribers can self select from webselfcare. | AT&T, Verizon, T-Mobile (USA), Movistar (Colombia), MTS (Russia), Maxis, SingTel, Aircel. |
| *Parental Control* | Quota management or Application control for child account. | AT&T, Verizon, Verizon, Maxis. |
| *Application/Service* | Application specific charges, e.g. Social networking bundles, emails, etc. | Movistar (Colombia), Maxis, Celcom, SingTel. |
| *Family-based Plans* | Data plans with common data pool and sharing | T-Mobile (USA). |

**Table 2.1:** Key use-cases listed by *Elitecore Technologies*.

## 2.2 Broadband communications technologies

Conflicts among policy rules within a particular plan are the core focus of this thesis. However, the interaction between technologies and the offered services should be well understood. Essentially, all plans are built on top of technologies.

In order to offer value-added services, network operators need comprehensive and powerful tools that enable them to analyze, manage and charge for mobile data traffic in more sophisticated ways that they have done in the past.

Several leading vendors including Comverse, Sandvine, Openet, Allot, Huawei and Amdocs have seized this market opportunity and offer comprehensive data traffic management solutions that combine:

- Policy Control (PCRF/PCEF).

- Deep Packet Inspection (DPI).

- On-line Data Rating and Charging.

- Bandwidth Control and QoS Management.

- Content Filtering, Caching and Compression.

- Business Intelligence and Analytics.

In the past, many operators used technologies like policy management or DPI to solve specific problems with network congestion, assure fair usage and satisfy regulatory requirements, but they rarely used them to generate new revenue streams.

According to Sandvine, operators are showing early interest in using a combination of DPI and policy management to associate specific applications or content types with a guaranteed quality of service that they can then charge a premium for. DPI can enable value-added data services that operators can bundle on top of their existing basic packages.

## 2.3 Model Checking

A model can be seen as a simplification of reality. We build models to better understand things, and most importantly, to describe part of a system from a particular perspective. Additionally, a model let us describe unambiguously the system itself or its properties. In this section, we focus on modeling languages and how can we prove or disprove the correctness of it with regards to a specification.

A modeling language is any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. One of the key benefits of using a modeling language, is that it will let us verify it formally against a given specification.

Formal verification is the process of applying a manual or automatic formal technique for establishing whether a given system satisfies a given property or behaves in accordance to some abstract description (formal specification) of the system. This verification is done by software tools as model checkers. A model checker thoroughly explores the state space to decide whether the system satisfies the set of properties.

Figure 2.2 below, illustrates the overall process of Model Checking [18]. In a first step, which is called modeling, the system description is converted into the system model. A system model is, for example, a Kripke structure, a labeled transition system, or a finite automaton. The requirements have to be manually formalized because they are mostly given in natural language. The result of this formalization is the formal specification given as formulas in a temporal logic such as LTL (Linear Temporal Logic).

As shown below, the model and the specification are inputs given to the model checker. The model checker uses an exhaustive search over all reachable states of the model to check whether the model satisfies the formula. In the end, it returns a result. The result may be that the model satisfies the formula or that the model does not satisfy the formula.
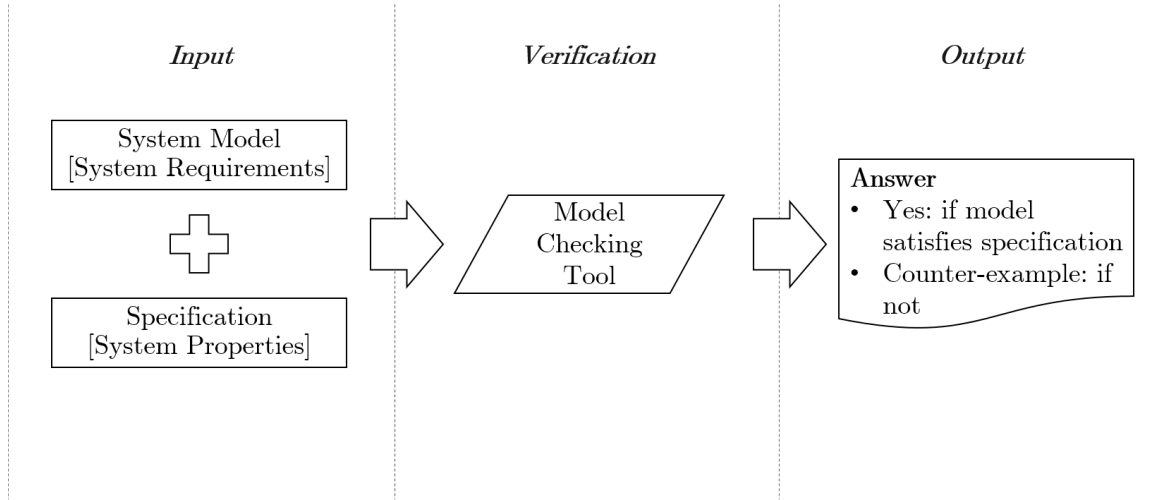
*Figure 2.2:* Model checking process

## 2.3.1 Spin Model Checker and PROMELA

Spin and most of its predecessors are implementations of standard automata-based reachability analyzers, with primary emphasis on performance. It provides an efficient implementation of a classic reachability analyzer with a firm and well-understood theory for LTL model checking [19] - or also known as the VardiWolper framework for automata-theoretic verification [20]. It supports the use of LTL (linear temporal logic) for the specification of correctness properties.

Spin accepts specifications written in a meta-language named Promela (Process Meta Language). Promela is a verification modeling language introduced by Gerard J. Holzmann. The language allows for the dynamic creation of concurrent processes to model, for example, concurrent or distributed systems. The three main types of objects that can be manipulated are:

- processes,

- channels, and

- variables.

Spin represents the system as a finite state machine. It visits each reachable state explicitly using Nested DFS and using partial order reduction. If a property is specified in LTL, Spin will first convert the LTL formula into an equivalent never claim, using the procedure outlined in [21]. Formally, the never claim specifies a Buchi automaton (a specific type of omega automaton) and the acceptance conditions from this automaton are evaluated on the global systems execution graph [19]. The efficiency of the Spin system comes from its on-the-fly procedure for performing this check that requires only small amounts of memory to be consumed for each state that is reached. The details of this so-called nested depth-first search algorithm are documented in [22].

**Data Types**

In Promela, there are seven predefined integer data types: bit , bool , byte , pid , short , int , and unsigned . There are also constructors for user-defined data types, and there is a separate predefined data type for message passing channels. Variables of type bit and bool are stored in a single bit of memory, which means that they can hold only binary, or boolean values.

Typedef declarations can be used to introduce user-defined data types. User-defined data types can use any predefined integer data types.

## 2.3.2 Linear Temporal Logic - LTL

Spin, as any other model checker, needs a number of system properties to verify the System Model (expressed in Promela). Without the system properties, the model checker cannot really tell whether or not the System Model is correct. In Promela, Linear Temporal Logic is used for specifying the correctness of requirements.

Linear temporal logic (LTL) is a modal temporal logic with modalities referring to time. In LTL, one can encode formulas about the future of paths, e.g. a condition will eventually be true, a condition will be true until another fact becomes true, etc. It is a fragment of the more complex CTL*, which additionally allows branching time and quantifiers.

LTL is built up from a finite set of propositional variables, logical operators (i.e. $\neg$, $\wedge$, $\vee$, $\implies$, $\iff$, *true* and *false*), and four additional temporal modal operators.

| LTL Formula | Description | Semantics Meaning |
|---|---|---|
| $\square\,\phi$ | Always | $\phi$ is satisfied at every state. |
| $\lozenge\,\phi$ | Eventually | $\phi$ is satisfied at some state. |
| $\psi$ U $\phi$ | Until | $\psi$ has to hold at least until $\phi$, which holds at the current or a future position. |
| X $\phi$ | Next | $\phi$ has to hold at the next state. |

**Table 2.2:** Semantics for the temporal operators

An important taxonomy of properties, as given below, is found in many books and papers [23]. Safety and liveness properties described below, are of our particular interest; as most of our specifications are safety and liveness properties.

*Safety properties* state that for all computations of the system, and for all instances of time, some property will invariantly hold. Colloquially, Safety properties assert that something "bad"never happens. For example, a safety property could be defined as: *Facebook traffic will never be blocked.*

*Liveness properties* state that some desired state of the system can eventually be reached. Colloquially, a Liveness property makes sure that something "good"eventually happens. An example of a liveness property could be: *If any other traffic is generated, it will eventually be blocked.*

*Persistence properties* are related to the stabilization of certain properties. In general, a persistence property describes that for all possible computations, there is a point of time when a certain property will always hold afterwards. For example: there is a point where rule A is always enabled.

*Fairness properties* state that some property will hold infinitely often. For example: No process is ignored infinitely often by an O.S.

## 2.4    Related Work

Model checking, or property checking, has been widely used in many other domains to exhaustively and automatically check whether a given model meets a given specification. Our primary interest is the use of Promela as the Model Checking tool to detect conflicts, via LTL formulas, in rules-based plans.

One of the main contributions for this Thesis is the research done by Zhiping Duan in 2003 [24]. In his work, he provides an approach to automatically detect feature interactions (i.e. conflicts) in telecommunications systems, which are not essentially rules-based. Even though, Duan uses LTL and FOL (First-Order-Logic) to prove their systems, he proposed an implementation of a FOL prover in $\lambda$Prolog. We think it is more intuitively to use Promela for our particular domain, as Promela is a verification modeling language which directly allows LTL formulas for its verification.

In 2007, Antoniou, et al, integrate several aspects of policy specification languages (i.e. rule-based reasoning), in a common framework. One key contribution from his research to our Thesis, is the use of priorities to resolve conflict among rules. In his research, the implementation of a conflict checker was not addressed.

Between 2010 and 2011, several requirements frameworks for business process compliance management have been proposed. In [25, 26], the authors formulate requirements for compliance rules. The requirements address the issues of lifetime compliance. The focus is also on the requirements to languages expressing compliance rules, on the rules priority, and on validation of process models against rules during design time and runtime.

In 2012, Gawanmeh, et al, proposed a novel algorithm for detecting conflicts in firewall rules [27]. Even though his novel approach, using domain restriction, is interesting; its proposed algorithm doesn't consider rules with multiple types of conditions.

The review of the related work above, illustrates that the problem of conflicts among rules exists in different domains and, more importantly, Model-Checking is a proven approach for detecting the conflicts efficiently. In this Thesis, we will use a novel tool, Promela, to detect conflicts in a particular area of telecommunications: rule-based Plans.

## 2.5    Summary of the chapter

The literature review is summarized below:

- Network Operators are basically moving from unlimited or volume-based models to value-added data offerings. These value-added services comprise a number of custom rules to form a Plan.

- Because all of these services are built on top of technologies, there are a number of technical, commercial, regulatory or integration impacts, that needs to be considered when designing these value-added services.

- Model Checking can perfectly be used to exhaustively and automatically verify whether a model meets a given specification. In our case the rule-based plans will be modeled using Promela and the specification will be defined in LTL.

- The problem of conflicts among rules has been studied in many other domains and, more importantly, Model-Checking is a proven approach for detecting the conflicts efficiently.

- In this Thesis, Promela, a novel tool, will be used to detect conflicts in a particular area of telecommunications: rule-based Plans.

## CHAPTER 3

# Promela Model and LTL Properties

The purpose of this section is to introduce the model proposed for specifying the rule-based plans and to expand on the LTL properties to verify the model.

## 3.1    Rule-based Plans Model

### 3.1.1    Overview

The key element to model is the specification of the rule-based plans. For that purpose, we declared a number of user-defined data types (i.e. typedef data structures), which are shown in a typedef diagram below in figure 3.1.
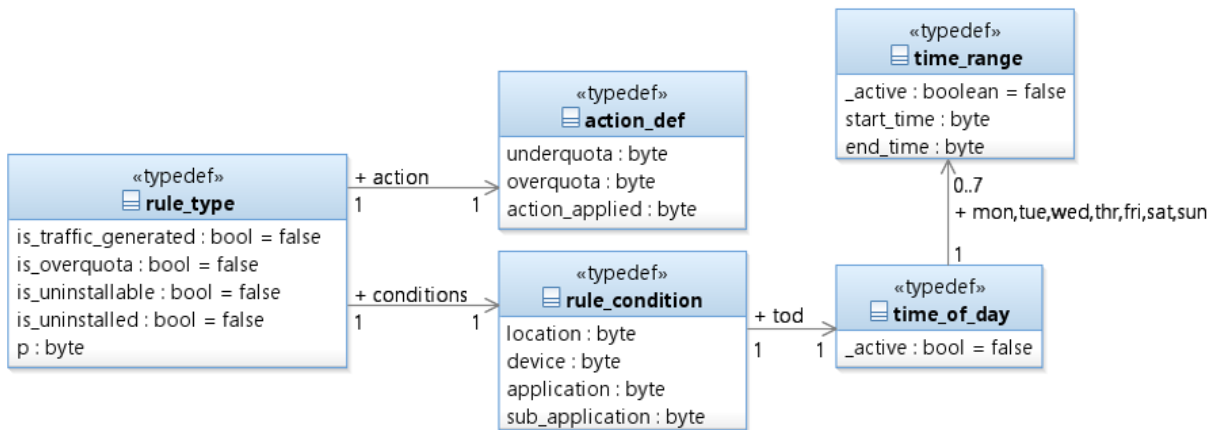


*Figure 3.1:* Typedef diagram

**Rule Type**

This data type allows us to specify a rule with its attributes described in table 3.1 below. Additionally, each rule, declares one *action_ def* variable and one *rule_ conditions* variable, described respectively in tables 3.2 and 3.3 below.

| Attribute | Type | Description | Example |
|---|---|---|---|
| *is_generated_traffic* | bool | If traffic is generated for this particular rule, *is_ generated_ traffic* attribute is set to true. | true \| false |
| *is_ overquota* | bool | If this attribute is set to true, the current state of the rule is *overquota*. If it is set to false, then it is *underquota*. | true \| false |
| *is_ uninstallable* | bool | If this is set to true, this rule will eventually be uninstalled when its quota is reached. | true \| false |
| *uninstalled* | bool | If this is set to true, this rule has been uninstalled and it is no longer active. | true \| false |
| *p* | byte | *p* stands for the *priority* of the rule. Valid values are from 0 to 255. | 5 |

**Table 3.1:** *Rule_ Type* Attribute definitions.

**Action Def**

This data type allows us to specify the actions to be applied for each state of the rule.

| Attribute | Type | Description | Example |
|---|---|---|---|
| *underquota* | byte | The action to be applied when the rule is in the underquota state. | 0:block<br>1:allow |
| *overquota* | byte | The action to be applied when the rule is in the overquota state. | 0:block<br>1:allow |
| *action_ applied* | byte | To store the current action applied. | 0:block<br>1:allow |

**Table 3.2:** *Action_ Def* Attribute definitions.

**Rule Conditions**

This data type allows us to specify the conditions for the rule described in table 3.3 below. Additionally, each rule condition, declares one *time_ of_ date* variable for specifying the different time conditions for each day. Whenever the conditions are met, the actions specified are performed.

| Attribute | Type | Description | Example |
|---|---|---|---|
| *location* | byte | The location condition for applying the rule actions. | 0:local <br> 1:roaming |
| *device* | byte | The device condition for applying the rule actions. | 0:Mobile <br> 1:Tablet <br> 2:Smart TV |
| *application* | byte | The application condition for applying the rule actions. | 0:Social Networks <br> 1:Video Streaming |
| *subapplication* | byte | The subapplication condition for applying the rule actions. | For Social Networks <br> 0:Facebook <br> 1:Twitter <br> 2:Instagram |

**Table 3.3:** *Rule_ Conditions* Attribute definitions.

**Time of Day and Time Ranges conditions**

As shown in table 3.4 below, the time_of_day data type allows us to specify the time conditions for each day.

| Attribute | Type | Description | Example |
|---|---|---|---|
| *_ active* | bool | Determines whether or not there is a time condition defined. | true \| false |
| *mon, tue, wed, thr, fri, sat, sun* | time_ range | Time conditions for each day. | See table 3.5 below |

**Table 3.4:** *Time_ of_ day* Attribute definitions.

As shown below in table 3.5, each time-day condition compromises a start time and end time.

| Attribute | Type | Description | Example |
|-----------|------|-------------|---------|
| _active_ | bool | Determines whether or not there is a time condition defined for this particular day. | true \| false |
| _start_time_ | byte | Start time condition. | 08 |
| _end_time_ | byte | End time condition. | 20 |

**Table 3.5:** _Time Range_ Attribute definitions.

### 3.1.2 Process Definition

Two processes have been defined to model rule-based plans:

- PriorityChecker: this process randomly chooses two rules, and executes the rule with higher priority.

- OverQuota: this process iterates through the defined rules, and change its state from underquota to overquota over time.

### 3.1.3 Example of rule-based plans

A mobile operator, would like to offer free facebook to its subscribers. The following Promela code would represent that rule in our model.

**Listing 3.1:** Free Facebook Rule.

```
1    rule_type freeFB
2    freeFB.conditions.application = 0;
3    freeFB.conditions.subapplication = 1;
4    freeFB.conditions.location = 0;
5    freeFB.action.underquota = 1;
6    freeFB.action.overquota = 1;
7    freeFB.p = 1;
```

Additionally, the same mobile operator may also need to offer a limited amount of bytes that could be consumed in a monthly basis. As shown in the listing 3.5 below, there is one 1GB allowed for any traffic generated. After 1GB is consumed, the action is blocked (0). 255 is used as the wildcard for specifying any application and any location.

*Listing 3.2:* Free Navigation Rule.

```
1    rule_type navRule
2    navRule.conditions.application = 255;
3    navRule.conditions.location = 255;
4    navRule.action.underquota = 1;
5    navRule.action.overquota = 0;
6    navRule.p = 0;
```

This simple rule-based plan example, would consist of an array of rule_type, comprising the two rules listed above.

## 3.2    LTL Properties

### 3.2.1    Macros Definitions

As seen before we will use Linear-Temporal-Logic formulas to verify if the model satisfies the specification. If it is not satisfied, then the system requirements, expressed in our Promela Model, are not well defined.

Listing 3.3 below, enumerates the macros we have defined for each rule to be used in the LTL formulas to prove the predefined System properties.

*Listing 3.3:* Macros for each rule

```
1    #define generated_rule_X (Rules[X].is_traffic_generated)
2    #define allowed_rule_X (Rules[X].action.actionApplied==ALLOWED_ACTION)
3    #define blocked_rule_X (Rules[X].action.actionApplied==BLOCKED_ACTION)
4    #define overquota_rule_X (Rules[X].is_overquota)
```

A further description of each macro is presented in table 3.6 below.

| Macro | Description | Expression |
|---|---|---|
| *generated_rule_{index}* | Whenever the Priority Checker process generates traffic for this particular rule, this macro will be evaluated to true. | Rules[{index}].is_traffic_generated |
| *allowed_rule_{index}* | Whenever the Priority Checker process applies the allowed action ("1"), this macro evaluates to true. | Rules[{index}].action.action_applied== ALLOWED_ACTION |
| *blocked_rule_{index}* | Whenever the Priority Checker process applies the blocked action ("0"), this macro evaluates to true. | Rules[{index}].action.action_applied== BLOCKED_ACTION |
| *overquota_rule_{index}* | Whenever the Overquota process sets the current state of the rule as over-quota, this macro evaluates to true. | Rules[{index}].is_overquota |

**Table 3.6:** Macros Definitions

## 3.2.2 Properties Definitions

Based on the macros described above, the following LTL properties will be used to verify the different rule-based plans.

| Id | Property Description | LTL | Classification |
|---|---|---|---|
| *PT01* | Rule that always allows or blocks the traffic satisfied by the rule condition. | $\Box$(generated_rule_{index} $\implies$ [allowed\|blocked]_rule_{index} ) | Safety |
| *PT02* | Rule that allows the traffic satisfied by the rule condition and then eventually blocks it when all the quota is consumed. | $\Box($ (generated_rule_{index} $\implies$ allowed_rule_{index} ) $\implies$ $\Diamond$blocked_rule_{index} ) | Liveness |

**Table 3.7:** Properties Definitions

These two properties are very useful and, when used combined, will let us prove a number of rule-based plans. The LTL for RT01 above, can be translated to: *Every* time traffic is generated for rule X, then its traffic will be allowed. While the LTL for RT02 above, can be translated to: *Every* time traffic is generated for rule X, then its traffic will be allowed and then will *eventually* be blocked.

In addition to the properties described above, we defined other properties considering the state of the rule, or an interaction with other rules, using the until operator (U), as shown in table 3.8 below.

| Id | Property Description | LTL | Classification |
|---|---|---|---|
| *PT03* | Consider PT01 only until some new state is reached. | $\Box((\text{generated\_rule\_\{index\}} \implies$ $\text{allowed\_rule\_\{index\}}$ $) \text{ U overquota\_rule\_\{index\}})$ $— \text{ OR } —$ $\Box((\text{generated\_rule\_\{index\}} \implies$ $\text{allowed\_rule\_\{index\}}$ $) \text{ U !overquota\_rule\_\{other\_index\}})$ | Liveness |
| *PT04* | Consider PT02 only until some new state is reached. | $\Box((\$ $(\text{generated\_rule\_\{index\}} \implies$ $\text{allowed\_rule\_\{index\}}$ $) \implies \Diamond\text{blocked\_rule\_\{index\}}$ $) \text{ U overquota\_rule\_\{index\}})$ $— \text{ OR } —$ $\Box((\$ $(\text{generated\_rule\_\{index\}} \implies$ $\text{allowed\_rule\_\{index\}}$ $) \implies \Diamond\text{blocked\_rule\_\{index\}}$ $) \text{ U !overquota\_rule\_\{other\_index\}})$ | Liveness |

***Table 3.8:*** Extended Properties Definitions

### 3.2.3    Conflict verification through properties

The LTL formulas described above, provide a powerful set of properties that will be used to prove a number of desired behaviors on each rule-based plan. In addition to the verification of each property, there is an implicit property, not so intuitively, which is also verified when the properties are combined; this is the conflict-free property

The always operator ($\square$) surrounding each property guarantees that the property must be satisfied in all the possible cases – regardless the pair of rules chosen by the Priority-Checker process. That is, the Model Checker by definition, will explore every possible combination of pair of rules, to make sure that the property is satisfied.

For instance, if a wrong priority were assigned to a rule and the traffic that meets the condition of the rule was generated, the model checker will eventually pick the other rule with similar conditions but with a higher priority; in which case the action of the rule with the wrong priority will not be applied. This scenario would prevent the Model Checker to verify the system with the given LTL property. In other words, a conflict would have been detected by the Model Checker and the verification would have failed.

## CHAPTER 4

# MexCom - Use Cases

In this chapter, we present the rule-based plans, which will be verified using the Promela Model described in the previous chapter.

We begin this chapter by introducing "MexCom", a not real company we created to illustrate the number of use-cases described below. We build the use-cases, based on the rules-based plans found in the Public Domain. We also considered some not-yet-existing rules; which, based on the literature review, are quite reasonable to predict.

In the next chapter, MexCom plans will be modeled using our proposed Promela Model and verified through the LTL formulas previously studied.

## 4.1    MexCom Overview

MexCom is a new Mobile Network Operator, which is eager to start operations in Mexico soon. MexCom will target low-income subscribers, who cannot afford an unlimited mobile plan.

The next sections describe the evolution of the service plans MexCom will offer to gain market share. It is important to mention that a plan will consist of one or more rules described below.

## 4.2 Core Rules

MexCom wants to start operations in Mexico by offering an initial set of monthly limited usage rules, which are listed below in table 4.1

| Core Rule ID | Capacity | Monthly Cost (MXN) |
|---|---|---|
| CORE01 | 100MB | $75.00 |
| CORE02 | 200MB | $100.00 |
| CORE03 | 400MB | $150.00 |
| CORE04 | 1GB | $350.00 |
| CORE05 | 2GB | $600.00 |

***Table 4.1:*** Core Limited Usage Rules.

To make the initial offer more attractive to its future subscribers, MexCom decides to offer certain commonly-used applications without charging their subscribers to their limited usage rule. This is known as Zero-Rated applications, offered at a very low-monthly cost.

| Rule Bolt-on ID | Application(s) | Monthly Cost (MXN) |
|---|---|---|
| BOLT01 | Unlimited Local Facebook | $10.00 |
| BOLT02 | Unlimited Local Twitter | $10.00 |
| BOLT03 | Unlimited Local Whatsapp | $10.00 |

***Table 4.2:*** Bolt-on Rules

The bolt-on rules above, essentially allow subscribers to form a low-cost plan by acquiring a basic limited usage rule, with any bolt-on rule. For example, a subscriber can acquire a limited monthly usage plan of 100MB with unlimited access to Facebook for $85.00 MXN monthly (CORE01 + BOLT01).

## 4.3     Roaming Bolt-on Rules

MexCom now wants to offer roaming bolt-on rules to their current offer of limited usage rules, previously described.

| Rule Bolt-on ID | Application(s) | Monthly Cost (MXN) |
|---|---|---|
| BOLT04 | Unlimited Roaming Twitter | $40.00 |
| BOLT05 | Unlimited Roaming Whatsapp | $40.00 |
| BOLT06 | Unlimited Roaming Email | $40.00 |

**Table 4.3:** Roaming Bolt-on Rules

Roaming data rates are usually much more expensive than local traffic rates. Most of the mobile devices, run a number of background applications which constantly consume traffic without the subscriber noticing it. With roaming bolt-on rules shown in table 4.3 above, MexCom will allow subscribers to choose which applications shall only be allowed while roaming. This will prevent background applications from consuming part of a quota.

## 4.4     Family Rules

MexCom decides to extend their offer by including family plans. Family plans let a subscriber share a monthly quota between multiple family members or devices. A family plan consists of a quota, big enough to be shared amongst family members, listed in table 4.4, and individual bolt-on/constraint rules per device listed in table 4.5 below.

| Family Rule ID | Capacity | Monthly Cost (MXN) |
|---|---|---|
| FAM01 | 10GB | $450.00 |
| FAM02 | 15GB | $600.00 |
| FAM03 | 20GB | $800.00 |

**Table 4.4:** Family Shared Rules

| Family Rule Bolt-on/Constraint ID | Bolt-on/Constraint | Monthly Cost (MXN) |
|---|---|---|
| FAMC01 | Unlimited Facebook | $10.00 |
| FAMC02 | Video Streaming HD Blocked | $0.00 |
| FAMC03 | Social Network Blocked | $0.00 |
| FAMC04 | Video Streaming Optimized | $50.00 |

*Table 4.5:* Family Shared Rules - Bolt-ons/Constraints

The above set of rules let a subscriber form its own family plan. For instance, the head of a family can acquire the FAM01 rule for all its family members; and additionally, acquire a constraint rule for blocking Social Networks to their kids device (FAMC03) or acquire a Video-Streaming-Optimization rule for an Smart TV device (FAMC04).

## 4.5    Parental-Control Rules

Parental-Control let a subscriber add a number of constraint rules to a rule-based plan. The constraints are usually applied during the day time and are released at night. This type of plans, prevent the subscriber to access certain applications during the day, for example during class hours.

Table 4.6 below, describes the rule constraints that can be added to the initial set of plans previously listed in the table 4.1, to form a Parental Control plan.

| Rule Constraint ID | Traffic Condition | Time of Day | Action | Monthly Cost (MXN) |
|---|---|---|---|---|
| CON01 | Social Networking | Weekdays 0800-1800 | Block | $0.00 |
| CON02 | Video Streaming | Weekdays 0800-1800 | Block | $0.00 |
| CON03 | Gaming | Weekdays | Block | $0.00 |

*Table 4.6:* Rule Constraints

These plans, let a head of a family, acquire a Limited Usage rule and on top of that, add a rule constraint to block a set of applications, during a predefined time of a day.

## 4.6    Limited Bolt-on Rules

Limited bolt-on rules are similar to the bolt-on rules seen before, except that they are not unlimited. That is, there is a usage limit. For example, a subscriber may purchase a plan based on the unlimited usage rules previously described in table 4.1, and only enhance it during a particular weekend, when an special sport event would take place.

In order to satisfy the need described above, MexCom decides to offer the limited bolt-on rules below, so subscribers can add them to their current plans at any time.

| Limited Rule Bolt-on ID | Limited Bolt-on | Cost (MXN) |
|---|---|---|
| *LADD01* | 1GB Video Streaming Optimized | $15.00 |
| *LADD02* | 2GB Video Streaming Optimized | $25.00 |
| *LADD03* | 1GB Gaming Optimized | $10.00 |
| *LADD04* | 1GB Free Usage | $15.00 |

***Table 4.7:*** Limited Bolt-on Rules

At this point, the spectrum of possible combinations of rules to form a plan is huge. We will see in the next chapter the significance of determining the right priority for each of the above rules, and most importantly, the value of an automated system that verifies the rule-based plans against its specification and that finds any possible conflict among the rules.

# CHAPTER 5

# Results

In this chapter, we verify the the use-cases studied in the previous chapter using our Promela model introduced in Chapter 3. The main purpose of this chapter is to prove the hypotheses introduced in chapter 1:

- **H1:** Rule-based Internet plans can be abstracted and specified in a verification modeling language.

- **H2:** Model Checking can be used to verify whether the model meets the specification of the plan and to detect conflicts between the rules within the plans.

In the sections below, each use-case is modeled and verified using our proposed Promela Model.

## 5.1 Core Rules

### 5.1.1 Promela Model

The model of the Core Rules and unlimited bolt-ons is shown in Listing 5.1. Given that a plan could consist of one or more rule instances, the worst case scenario was modeled; that is, one instance of each rule was selected to form the plan below.

**Listing 5.1:** Core-Rules-based Plan Model

```
1    // CoreRule
2    Rules[0].conditions.application = 255;
3    Rules[0].conditions.location = 255;
4    Rules[0].action.underquota = 1; //Allow Traffic
5    Rules[0].action.overquota = 0; //Block Traffic
6    Rules[0].p = 0;
7
8    // Unlimited local Facebook Bolt-on
9    Rules[1].conditions.application = 0;
10   Rules[1].conditions.location = 0;
11   Rules[1].action.underquota = 1; //Allow Traffic
12   Rules[1].action.overquota = 1; //Allow Traffic
13   Rules[1].p = 1;
14
15   // Unlimited local Twitter Bolt-on
16   Rules[2].conditions.application = 1;
17   Rules[2].conditions.location = 0;
18   Rules[2].action.underquota = 1; //Allow Traffic
19   Rules[2].action.overquota = 1; //Allow Traffic
20   Rules[2].p = 1;
21
22   // Unlimited local Whatsapp Bolt-on
23   Rules[3].conditions.application = 2;
24   Rules[3].conditions.location = 0;
25   Rules[3].action.underquota = 1; //Allow Traffic
26   Rules[3].action.overquota = 1; //Allow Traffic
27   Rules[3].p = 1;
```

### 5.1.2  System Properties

The key rule of this plan, is the Rule[0] above; it dictates how most of the traffic should be handled. Basically, we are specifying no condition for that rule and indicating to apply the block action when the quota is reached. Please note that the amount of bytes of the quota of the Rule[0] is irrelevant as it is assumed that eventually the quota will be consumed regardless of the quota allowance.

Additionally there are three more rules, each one to always allow a different application. As it has been discussed, these additional rules shall have a higher priority, so the

action to allow these particular applications (i.e. facebook, twitter and whatsapp), takes precedence over the Rule[0].

If any of the additional rules has a lower priority assigned, the Rule[0] will take precedence over that rule and consequently, the desired behavior of allowing that particular application will not be met.
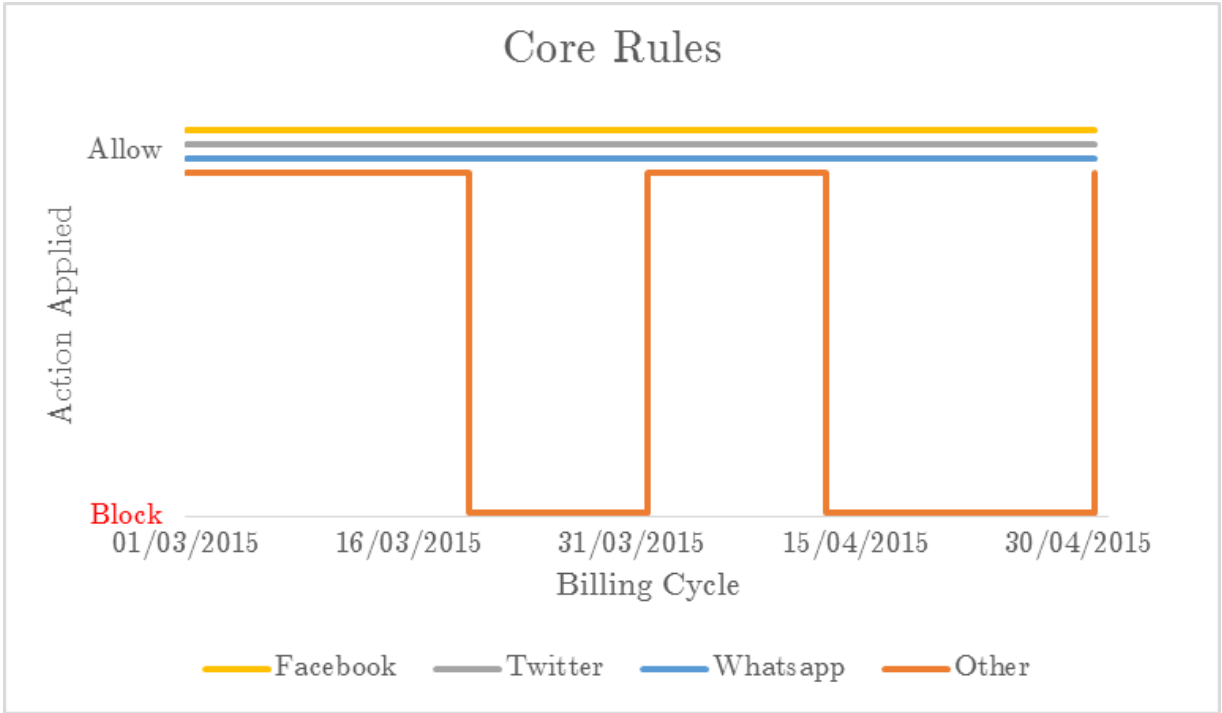


**Figure 5.1:** Core Rules - Linear Temporal Logic

Figure 5.1 above, represent the expected behavior of this rule-based plan. Facebook, Twitter and Whatapp traffic shall always be allowed; while the rest of the traffic will be allowed and then eventually be blocked.

In order to verify the expected behavior expressed above, the LTL below will be used. As seen in chapter 3, Linear-Temporal-Logic formulas are used to prove the property desired.

*Listing 5.2:* Core-Rules LTL

```
1    ltl Core_Rules_LTL {
2      [](
3          ((generated_rule_0 -> allowed_rule_0) -> <>blocked_rule_0) &&
4          (generated_rule_1->allowed_rule_1) &&
5          (generated_rule_2->allowed_rule_2) &&
6          (generated_rule_3->allowed_rule_3)
7        )
8    }
```

Listing 5.2 above can be translated to natural language as *every* time traffic that meets the conditions of rules number 1, 2 and 3 is generated, then it will be allowed. Additionally, *every* time traffic that meets the condition of rule number 0 is generated, then it will be allowed and *eventually* will be blocked.

### 5.1.3 SPIN Verification

At this point we have intuitively specified the first rule-based plan with our Promela model and we have determined which LTL formulas will be used to verify the model. Listing 5.3 below, presents the result obtained in Spin, with 0 errors shown in line 7.

*Listing 5.3:* Core-Rules Plan Verification

```
1    (Spin Version 6.3.2 -- 17 May 2014) + Partial Order Reduction
2    Full statespace search for:
3            never claim  + (Core_Rules_LTL)
4            assertion violations + (if within scope of claim)
5            acceptance cycles  + (fairness enabled)
6            invalid end states - (disabled by never claim)
7    State-vector 324 byte, depth reached 55, *** errors: 0 ***
8            60 states, stored
9            19 states, matched
10           79 transitions (= stored+matched)
11           48 atomic steps
12   hash conflicts: 0 (resolved)
13   Stats on memory usage (in Megabytes):
14      0.019 equivalent memory usage for states (stored*(State-vector + overhead))
15      0.143 actual memory usage for states
16     64.000 memory used for hash table (-w24)
17      0.069 memory used for DFS stack (-m2000)
18     64.195 total actual memory usage
```

**Exceptions Verification**

Our first results were expected, as the right priorities were assigned to each rule. For the next test, the priority of the key rule will be increased from 0 to 2, as shown below:

*Listing 5.4:* Increased priority of the Core Rule

```
1    // CoreRule
2    Rules[0].conditions.application = 255;
3    Rules[0].conditions.location = 255;
4    Rules[0].action.underquota = 1; //Allow Traffic
5    Rules[0].action.overquota = 0; //Block Traffic
6    Rules[0].p = 2;
7    ...
```

The updated model above, will not meet the LTL specification. The rules number 1, 2 and 3 will not be executed, because facebook, whatsapp and twitter traffic will be handled by the rule 0, which has a higher priority assigned. SPIN model basically identifies that when facebook traffic is generated, it is not always allowed.

*Listing 5.5:* Invalid Core-Rules Plan - Verification

```
1  (Spin Version 6.3.2 -- 17 May 2014) + Partial Order Reduction
2  Warning: Search not completed
3  Full statespace search for:
4        never claim  + (Core_Rules_LTL)
5        assertion violations + (if within scope of claim)
6        acceptance cycles  + (fairness enabled)
7        invalid end states - (disabled by never claim)
8  State-vector 324 byte, depth reached 57, *** errors: 1 ***
9      162 states, stored (244 visited)
10      100 states, matched
11      344 transitions (= visited+matched)
12      202 atomic steps
13 hash conflicts: 0 (resolved)
14 Stats on memory usage (in Megabytes):
15    0.053 equivalent memory usage for states (stored*(State-vector + overhead))
16    0.143 actual memory usage for states
17   64.000 memory used for hash table (-w24)
18    0.069 memory used for DFS stack (-m2000)
19   64.195 total actual memory usage
```

## 5.2    Roaming Bolt-on Rules

### 5.2.1    Promela Model

The model of the Roaming Bolt-on Rules is shown in Listing 5.6. Given that a plan could consist of one or more rule instances, the worst case scenario was modeled; that is, one instance of each rule was selected to form the plan below, including all the roaming bolt-on.

*Listing 5.6:* Core-Rules-based Roaming Plan Model

```
1    // CoreRule
2    Rules[0].conditions.application = 255;
3    Rules[0].conditions.location = 255;
4    Rules[0].action.underquota = 1; //Allow Traffic
5    Rules[0].action.overquota = 0; //Block Traffic
6    Rules[0].p = 0;
7
8    // Unlimited local Facebook Bolt-on
9    Rules[1].conditions.application = 0;
10   Rules[1].conditions.location = 0;
11   Rules[1].action.underquota = 1; //Allow Traffic
12   Rules[1].action.overquota = 1; //Allow Traffic
13   Rules[1].p = 1;
14
15   // Unlimited local Twitter Bolt-on
16   Rules[2].conditions.application = 1;
17   Rules[2].conditions.location = 0;
18   Rules[2].action.underquota = 1; //Allow Traffic
19   Rules[2].action.overquota = 1; //Allow Traffic
20   Rules[2].p = 1;
21
22   // Unlimited local Whatsapp Bolt-on
23   Rules[3].conditions.application = 2;
24   Rules[3].conditions.location = 0;
25   Rules[3].action.underquota = 1; //Allow Traffic
26   Rules[3].action.overquota = 1; //Allow Traffic
27   Rules[3].p = 1;
28
29   // Unlimited Email Roaming Bolt-on
30   Rules[4].conditions.application = 3;
31   Rules[4].conditions.location = 1;
32   Rules[4].action.underquota = 1;
33   Rules[4].action.overquota = 1;
```

```
34      Rules[4].p = 1;
35
36      // Unlimited Twitter Roaming Bolt-on
37      Rules[5].conditions.application = 1;
38      Rules[5].conditions.location = 1;
39      Rules[5].action.underquota = 1;
40      Rules[5].action.overquota = 1;
41      Rules[5].p = 1;
42
43      // Unlimited Whatsapp Roaming Bolt-on
44      Rules[6].conditions.application = 2;
45      Rules[6].conditions.location = 1;
46      Rules[6].action.underquota = 1;
47      Rules[6].action.overquota = 1;
48      Rules[6].p = 1;
```

As it was seen in the first test result, rules that have exclusive conditions may have the same priority as there is no conflict between them. However, and again, the Rule[0] above, shall have a higher priority assigned to meet the specification.

### 5.2.2 System Properties

As with the previous test, the key rule of this plan, is the Rule[0] above; it dictates how most of the traffic should be handled. Basically, we are specifying no condition for that rule and indicating to apply the block action when the quota is reached.

In addition to the local bolt-on rules seen before, there are three more roaming bolt-on rules, each one to always allow a different application while the subscriber is roaming. As it has been discussed, these additional rules shall have a higher priority, so the action to allow these particular applications (i.e. email, twitter and whatsapp roaming), takes precedence over the Rule[0].

If any of the bolt-on rules has a lower priority assigned, the Rule[0] will take precedence over that particular bolt-on rule, and consequently the desired behavior will not be met.
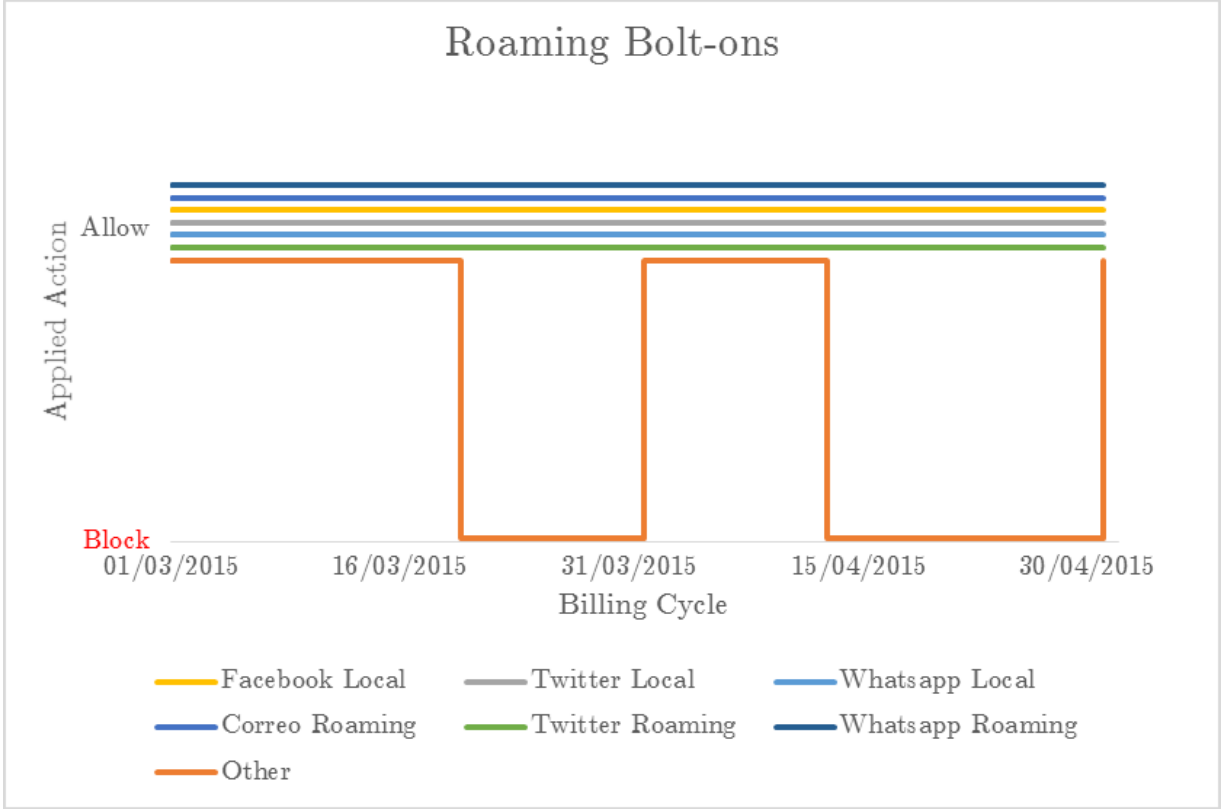
**Figure 5.2:** Roaming Rules - Linear Temporal Logic

Figure 5.2 above, represent the Model shown in Listing 5.6. As it is, local traffic of Facebook, Twitter and Whatapp applications would always be allowed. Additionally, roaming traffic of Email, Twitter and Whatsapp applications would always be allowed too. The rest of the traffic would have being allowed and then eventually would have being blocked, when the quota is reached.

The model in Listing 5.6, as it is, has a flaw. Other roaming traffic would eventually consume usage from the rule[0]. This is a problem for the subscribers, as they generally cannot control which applications should be allowed and which should not from the mobile devices. That is, background applications running on their mobiles, would consume their main usage while they are on roaming without subscribers even noticing it. To solve this flaw, one additional rule needs to be considered, as shown in Listing 5.7 below.

**Listing 5.7:** Core-Rules-based Roaming Plan Model - II

```
1    ...
2
3    // Unlimited Email Roaming Bolt-on
4    Rules[4].conditions.application = 3;
5    Rules[4].conditions.location = 1;
6    Rules[4].action.underquota = 1;
7    Rules[4].action.overquota = 1;
8    Rules[4].p = 2;
9
10   // Unlimited Twitter Roaming Bolt-on
11   Rules[5].conditions.application = 1;
12   Rules[5].conditions.location = 1;
13   Rules[5].action.underquota = 1;
14   Rules[5].action.overquota = 1;
15   Rules[5].p = 2;
16
17   // Unlimited Whatsapp Roaming Bolt-on
18   Rules[6].conditions.application = 2;
19   Rules[6].conditions.location = 1;
20   Rules[6].action.underquota = 1;
21   Rules[6].action.overquota = 1;
22   Rules[6].p = 2;
23
24   // Block other Roaming
25   Rules[7].conditions.application = 255;
26   Rules[7].conditions.location = 1;
27   Rules[7].action.underquota = 0;
28   Rules[7].action.overquota = 0;
29   Rules[7].p = 1;
```

Rule[7] was added in order to block always the traffic from the roaming location, specified as location 1. Note that in order to meet the desired behavior, rules [4], [5] and [6] require a higher priority than the priority of the rule [7]. When this model is verified, the priorities of the rule [7] will be updated, to show that model allow to detect that kind of wrong priority specification.

Figure 5.3 below, represent the expected behavior of this rule-based plan described previously in Listing 5.6 and 5.7. Note that other Roaming traffic shall always be blocked.
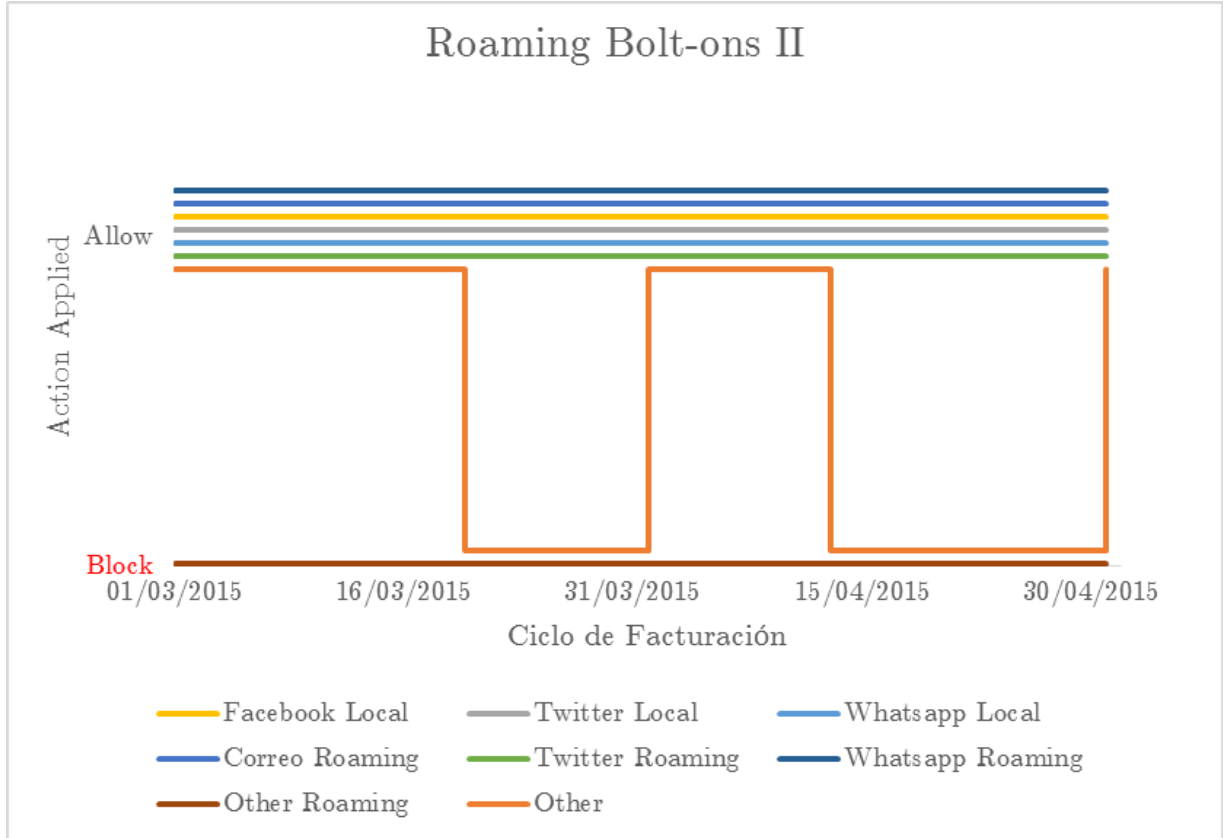
***Figure 5.3:*** Roaming Rules - Linear Temporal Logic II

In order to verify the expected behavior expressed above, the LTL below will be used. As seen in chapter 3, Linear-Temporal-Logic formulas are used to prove the property desired.

***Listing 5.8:*** Roaming-Rules LTL

```
1   ltl Romaing_Rules_LTL {
2     [](
3       ((generated_rule_0 -> allowed_rule_0) -> <>blocked_rule_0) &&
4       (generated_rule_1->allowed_rule_1) &&
5       (generated_rule_2->allowed_rule_2) &&
6       (generated_rule_3->allowed_rule_3) &&
7       (generated_rule_4->allowed_rule_4) &&
8       (generated_rule_5->allowed_rule_5) &&
9       (generated_rule_6->allowed_rule_6) &&
10      (generated_rule_7->blocked_rule_7)
11    )
12  }
```

Listing 5.12 above can be translated to natural language as *every* time traffic that meets the conditions of rules number 1 to 6 is generated, then it will be allowed. Additionally, *every* time traffic that meets rule number 7 is generated, it will be blocked. Finally, *every* time traffic that meets the condition of rule number 0 is generated, then it will be allowed and *eventually* will be blocked.

## 5.2.3    SPIN Verification

At this point we have intuitively specified the roaming rule-based plan with our Promela model (Listing 5.6 and 5.7 above) and we have determined which LTL formulas will be used to verify the model (Listing 5.12 above). Below it is presented the result obtained in Spin, with 0 errors shown in line 7.

*Listing 5.9:* Core-Rules Plan Verification

```
1  (Spin Version 6.3.2 -- 17 May 2014) + Partial Order Reduction
2  Full statespace search for:
3          never claim  + (Romaing_Rules_LTL)
4          assertion violations + (if within scope of claim)
5          acceptance cycles  + (fairness enabled)
6          invalid end states - (disabled by never claim)
7  State-vector 612 byte, depth reached 94, *** errors: 0 ***
8      2924 states, stored (3106 visited)
9      1299 states, matched
10      4405 transitions (= visited+matched)
11      3703 atomic steps
12  hash conflicts: 0 (resolved)
13  Stats on memory usage (in Megabytes):
14     1.751 equivalent memory usage for states (stored*(State-vector + overhead))
15     1.800 actual memory usage for states
16    64.000 memory used for hash table (-w24)
17     0.069 memory used for DFS stack (-m2000)
18    65.855 total actual memory usage
```

The memory used is also relevant to note. In this test execution, the number of rules was doubled from 4 to 8. The memory required for verifying the model increased in 1.66Mb, from 64.195 Mb to 65.855 Mb. The memory used is found in the last line of each execution.

**Exceptions Verification**

The results obtained previously were expected as the right priorities were assigned to verify the model. Empirical tests are performed below to validate invalid models, with wrong priorities assigned.

*Listing 5.10:* Increased priority of the Roaming Rule

```
1    ...
2    // Unlimited Whatsapp Roaming Bolt-on
3    Rules[6].conditions.application = 2;
4    Rules[6].conditions.location = 1;
5    Rules[6].action.underquota = 1;
6    Rules[6].action.overquota = 1;
7    Rules[6].p = 2;
8
9    // Block other Roaming
10   Rules[7].conditions.application = 255;
11   Rules[7].conditions.location = 1;
12   Rules[7].action.underquota = 0;
13   Rules[7].action.overquota = 0;
14   Rules[7].p = 3;
```

In the updated model above, the priority of roaming rule[7] was increased from 1 to 3. The action of the rule[7], is to block all the roaming traffic. With that priority, the other roaming applications that are supposed to be always allowed, will eventually be blocked, and consequently the model will not meet the LTL specification. That is, the rules [4], [5] and [6] will not be executed, all the roaming traffic will be handled by the rule 7, which has a higher priority assigned. SPIN model basically identifies that when whatsapp roaming traffic is generated, it is not always allowed.

As seen in Listing 5.11 below, an error is shown by the SPIN model checker. Also note that the number of states reached by the Model Checker is usually much lower when an error is found, as SPIN stops the state exploration as soon as an error is found.

*Listing 5.11:* Invalid Roaming-Rules Plan - Verification

```
1  (Spin Version 6.3.2 -- 17 May 2014) + Partial Order Reduction
2  Warning: Search not completed
3  Full statespace search for:
4          never claim  + (Romaing_Rules_LTL)
5          assertion violations + (if within scope of claim)
6          acceptance cycles  + (fairness enabled)
7          invalid end states - (disabled by never claim)
8  State-vector 612 byte, depth reached 88, *** errors: 1 ***
9      1861 states, stored (2043 visited)
10      865 states, matched
11     2908 transitions (= visited+matched)
12     2348 atomic steps
13  hash conflicts: 0 (resolved)
14  Stats on memory usage (in Megabytes):
15     1.115 equivalent memory usage for states (stored*(State-vector + overhead))
16     1.213 actual memory usage for states
17    64.000 memory used for hash table (-w24)
18     0.069 memory used for DFS stack (-m2000)
19    65.270 total actual memory usage
```

Another way for finding an error is having the right Model but using the wrong properties to verify it. Listing 5.13 below shows the result of verifying the original Roaming Model found in Listing 5.6 and 5.7, with the invalid LTL below.

*Listing 5.12:* Roaming-Rules LTL Invalidad

```
1    ltl Romaing_Rules_LTL_Invalid {
2      []( 
3        ((generated_rule_0 -> allowed_rule_0) -> <>blocked_rule_0) &&
4        (generated_rule_1->allowed_rule_1) &&
5        (generated_rule_2->allowed_rule_2) &&
6        (generated_rule_3->allowed_rule_3) &&
7        (generated_rule_4->allowed_rule_4) &&
8        (generated_rule_5->allowed_rule_5) &&
9        (generated_rule_6->allowed_rule_6) &&
10        (generated_rule_7->allowed_rule_7)
11      )
12    }
```

The LTL expression in line 10 above, was changed. Now the roaming traffic is expected to be always allowed. Since that is not the case, an error is shown per the output below.

*Listing 5.13:* Invalid Roaming-Rules Plan - Verification II

```
1  (Spin Version 6.3.2 -- 17 May 2014)  + Partial Order Reduction
2  Warning: Search not completed
3  Full statespace search for:
4          never claim  + (Romaing_Rules_LTL_Invalid)
5          assertion violations + (if within scope of claim)
6          acceptance cycles  + (fairness enabled)
7          invalid end states - (disabled by never claim)
8  State-vector 616 byte, depth reached 93, *** errors: 1 ***
9       2592 states, stored (2774 visited)
10      1163 states, matched
11      3937 transitions (= visited+matched)
12      3283 atomic steps
13  hash conflicts: 0 (resolved)
14  Stats on memory usage (in Megabytes):
15      1.562 equivalent memory usage for states (stored*(State-vector + overhead))
16      1.596 actual memory usage for states
17     64.000 memory used for hash table (-w24)
18      0.069 memory used for DFS stack (-m2000)
19     65.660 total actual memory usage
```

## 5.3    Family Rules

### 5.3.1    Promela Model

The models of the Family Plans are shown below.

**Family Member #1**

The family plan specified in the Model below, can be used by a Family member, who wants to prevent its Smartphone to consume HD streaming videos. It consists on the Family Rule, which is shared by all the family members, and two additional Familiy Bolt-ons rules.

*Listing 5.14:* Family Member#1 Model

```
1    // Family Rule
```

```
2    Rules[0].conditions.application = 255;
3    Rules[0].conditions.location = 255;
4    Rules[0].action.underquota = 1;
5    Rules[0].action.overquota = 0;
6    Rules[0].p = 0;
7
8    // Unlimited Social Networks
9    Rules[1].conditions.application = 2; // Social Networks
10   Rules[1].conditions.sub_application = 255; // All Social Networks
11   Rules[1].conditions.location = 0;
12   Rules[1].action.underquota = 1;
13   Rules[1].action.overquota = 1;
14   Rules[1].p = 1;
15
16   // HD Blocked
17   Rules[2].conditions.application = 1; // Video Streaming
18   Rules[2].conditions.sub_application = 1; // Only HD Streaming
19   Rules[2].conditions.location = 0;
20   Rules[2].action.underquota = 0;
21   Rules[2].action.overquota = 0;
22   Rules[2].p = 1;
```

# References

[1] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "A survey of smart data pricing: Past proposals, current plans, and future trends," *ACM Comput. Surv.*, vol. 46, no. 2, pp. 15:1–15:37, Nov. 2013. [Online]. Available: http://doi.acm.org/10.1145/2543581.2543582

[2] D. D. Clark, "Policy routing in internet protocols," M.I.T. Laboratory for Computer Science, RFC 1102, 1989. [Online]. Available: https://tools.ietf.org/rfc/rfc1102.txt

[3] M. Sloman and E. Lupu, "Policy specification for programmable networks," in *Active Networks*, ser. Lecture Notes in Computer Science, S. Covaci, Ed. Springer Berlin Heidelberg, 1999, vol. 1653, pp. 73–84. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-48507-0_7

[4] G. Stone, B. Lundy, and G. Xie, "Network policy languages: a survey and a new approach," *Network, IEEE*, vol. 15, no. 1, pp. 10–21, Jan 2001.

[5] A. Albaladejo, F. de Gouveia, M. Corici, and T. Magedanz, "The pcc rule in the 3gpp ims policy and charging control architecture," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, Nov 2008, pp. 1–5.

[6] T. Grgic and M. Matijasevic, "An overview of online charging in 3gpp networks: new ways of utilizing user, network, and service-related information," *International Journal of Network Management*, vol. 23, no. 2, pp. 81–100, 2013. [Online]. Available: http://dx.doi.org/10.1002/nem.1816

[7] D. Evans, "The internet of things. how the next evolution of the internet is changing everything," 2011. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[8] M. Cheboldaeff, "Service charging challenges in converged networks," *Communications Magazine, IEEE*, vol. 49, no. 1, pp. 118–123, January 2011.

[9] K. Kimbler and M. Taylor, "Value added mobile broadband services innovation driven transformation of the smart pipe," in *Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on*, Oct 2012, pp. 30–34.

[10] Sandvine, "Success story: A million reasons to select vox telecom," 2014. [Online]. Available: https://www.sandvine.com/downloads/general/success-stories/a-million-reasons-to-select-vox-telecom.pdf

[11] Hk.chinamobile.com, "Lite data service plans," 2015. [Online]. Available: https://www.hk.chinamobile.com/en/services_plan/Lite_Data/

[12] Z. Ezziane, "Charging and pricing challenges for 3g systems," *Communications Surveys Tutorials, IEEE*, vol. 7, no. 4, pp. 58–68, Fourth 2005.

[13] J. Hart and R. Brown, "What lte policy control features can operators execute to differentiate themselves from ott players?" in *Intelligence in Next Generation Networks (ICIN), 2013 17th International Conference on*, Oct 2013, pp. 16–22.

[14] E. Technologies, *Whitepaper: A Hand-book of emerging Policy Management use cases*, 2012. [Online]. Available: http://techlibrary.informationweek.com/whitepaper/asset/detail/110443

[15] Sandvine, "Increasing revenue with innovative consumer services: Overview," 2014. [Online]. Available: https://www.sandvine.com/downloads/general/solutions/revenue-generation/consumer-services-overview.pdf

[16] Alcatel, "Monetize over-the-top mobile applications," 2014. [Online]. Available: http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2014/10718-monetize-over-the-top-mobile-applications.pdf

[17] A. Communications, "Delivering digital lifestyle services," 2014. [Online]. Available: http://47maz93acasm1hwicm2d3sw7.wpengine.netdna-cdn.com/wp-content/uploads/Service_Provider_Use_Case_Booklet_12-2014.pdf

[18] W. Chan, R. J. Anderson, P. Beame, S. Burns, F. Modugno, D. Notkin, and J. D. Reese, "Model checking large software specifications," *IEEE Trans. Softw. Eng.*, vol. 24, no. 7, pp. 498–520, Jul. 1998. [Online]. Available: http://dx.doi.org/10.1109/32.708566

[19] G. Holzmann, E. Najm, and A. Serrhrouchni, "Spin model checking: an introduction," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 2, no. 4, pp. 321–327, 2000.

[20] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," *1st Symposium in Logic in Computer Science (LICS)*, pp. 322–331, 1986.

[21] R. Gerth, D. Peled, M. Y. Vardi, R. Gerth, D. D. Eindhoven, D. Peled, M. Y. Vardi, and P. Wolper, "Simple on-the-fly automatic verification of linear temporal logic," in *In Protocol Specification Testing and Verification*.  Chapman and Hall, 1995, pp. 3–18.

[22] G. J. Holzmann, D. Peled, and M. Yannakakis, "On nested depth first search," 1996.

[23] K. Schneider, *Verification of reactive systems*.  Springer-Verlag, 2004.

[24] Z. Duan, "Proof search algorithms for detecting interactions in telecommunication features," 2003.

[25] A. Awad, M. Weidlich, and M. Weske, "Consistency checking of compliance rules," in *Business Information Systems*, ser. Lecture Notes in Business Information Processing, W. Abramowicz and R. Tolksdorf, Eds.  Springer Berlin Heidelberg, 2010, vol. 47, pp. 106–118. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12814-1_10

[26] F. Hantry, M.-S. Hacid, and R. Thion, "Detection of conflicting compliance rules," in *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International*, Aug 2011, pp. 419–428.

[27] A. Gawanmeh and S. Tahar, "Novel algorithm for detecting conflicts in firewall rules," in *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, April 2012, pp. 1–4.