Configuration Logic: A multi-site Modal Logic

Roger Villemaire, Sylvain Hallé, Omar Cherkaoui
Université du Québec à Montréal
C. P. 8888, Succ. Centre-Ville
Montréal, Canada H3C 3P8
villemaire.roger@uqam.ca

Track 3: Temporal Logic in Computer Science
Specification and verification of systems
Temporal logics for distributed systems

Abstract

We introduce a logical formalism for describing properties of configurations of computing systems. This logic of trees allows quantification on nodes labels which are modalities containing variables. We explain the motivation behind our formalism and give a classical semantics and also a new equivalent one based on partial functions on variables.

1 Introduction

Managing computing equipment configurations is a central task in computer science. The increase in computational power and in number of computing devices makes this task error prone. It is important to be able to describe properties in a abstract way to help automatize the management of devices configuration.

We propose in this paper a simple extension to modal logic, aimed at describing properties of computing equipment. Even if our motivation is towards applications in computer networks, our logical system is applicable to any type of device.

Our goal is to develop a formalism to check, analyze and help debug complex configurations. In this paper we make a first step in this direction presenting the logical system, a classical semantics and also a new formal semantics in terms of partial functions on variables.

We have tried to keep our logical system as simple as possible and as close as possible to the goal of analyzing configurations.

The paper is structured as follows. We define our logic in section 2, we explain the motivation in the field of computer networks in section 3, we relate our logic to others computational logic in section 4. Finally we develop the new formal semantics in section 5 and we conclude with section 6.

2 Configuration logic

2.1 Syntax

A *Configuration Logic* language CL is formed of a set of *names* $Names = \{p, q, r, p_1, q_1, r_1, \ldots\}$, a set of *variables* $Variables = \{x, y, z, x_1, y_1, z_1, \ldots\}$ and a set of relations $R_1(\bar{x}), R_2(\bar{x}), \ldots$ (respectively of arity $arity(R_1), arity(R_2), \ldots$).

Formulas are built using the usual boolean connectives ($\wedge, \vee, \neg$) and the following quantifiers:

Existential quantifiers: There are two forms of existential quantifiers: $\langle p = x\rangle\varphi$ and $\langle \bar{p} = \bar{x}; p = x\rangle\varphi$, where p is a name, $\bar{p}$ a finite sequence of names, x a variable and $\bar{x}$ a finite sequence of variables of the same length as $\bar{p}$. Only the last variable x is considered bound as it will become clear in the classical semantics described further.

Universal quantifiers: There are also two forms of universal quantifiers: $[p = x]\varphi$ and $[\bar{p} = \bar{x}; p = x]\varphi$, where p, $\bar{p}$, x and $\bar{x}$ are the same as for existential quantifiers. Here again only the last variable x is considered bound.

If it is necessary to explicitly write the elements

$p_1, \dots, p_n$ of $\bar{p}$ and those $x_1, \dots, x_n$ of $\bar{x}$, we will write the quantifiers as $\langle p_1 = x_1, \dots, p_n = x_n; p = x\rangle\varphi$ and $[p_1 = x_1, \dots, p_n = x_n; p = x]\varphi$.

To simplify proofs and definitions, we will consider that $\langle p = x\rangle\varphi$ and $[p = x]\varphi$ are special cases of $[\bar{p} = \bar{x}; p = x]\varphi$ and $\langle \bar{p} = \bar{x}; p = x\rangle\varphi$, where $\bar{p}$ and $\bar{x}$ are empty.

Without loss of generality, we will restrict ourselves to sentences in which every variable is bound only once. By renaming, every sentence can be put into this form.

In fact, we want to limit the use of quantifiers in such a way that they are extensions of previous ones. To make this notion clear, let us introduce the following definition.

Definition 1 *A sentence is a formula such that every variable is bound and furthermore any sub-formula*

$$\langle p_1 = x_1, \dots, p_n = x_n; p = x\rangle\psi$$

is contained in a sub-formula that does not bind any of the $x_1, \dots, x_{n-1}$ and is of the form

$$\langle p_1 = x_1, \dots, p_{n-1} = x_{n-1}; p_n = x_n\rangle\varphi$$

or of the form

$$[p_1 = x_1, \dots, p_{n-1} = x_{n-1}; p_n = x_n]\varphi$$

The same must be true of sub-formulas $[p_1 = x_1, \dots, p_n = x_n; p = x]\psi$.

We will introduce some definitions before giving the classical semantics.

Definition 2 *A Path is a finite non-empty word on the alphabet formed of all $(p = x)$ where p is a name and x a variable.*

Definition 3 *A Name-Path is a finite non-empty word on the alphabet formed of all names.*

If $\bar{p} = p_1 \dots p_n$ and $\bar{x} = x_1 \dots x_n$ we will usually write $(\bar{p} = \bar{x})$ for the path $(p_1 = x_1) \cdots (p_n = x_n)$.

2.2 Configurations

A configuration is a forest (set of trees) where every node is labeled by a name and a value. Furthermore, no two roots (top level nodes) can have the same name and value. Similarly, every node has no more than one child having the same name and value. Formally, we introduce the following definition.

Definition 4 *A Configuration is a structure of the form $\langle V, N, \tilde{R}_1, ..., \tilde{R}_n\rangle$ where:*

- *V is the set of values;*
- *N the set of nodes, is a set of words closed under prefix, on the alphabet formed of $(p = v)$, with p a name and $v \in V$;*
- *$\tilde{R}_1, \dots, \tilde{R}_n$ are relations on V (i.e. subsets of $V^{arity(R_1)}, \dots, V^{arity(R_n)}$ respectively).*

A configuration represents a hierarchical set of parameters configuring some computing equipment. The nodes representing the parameters have a name and a value.

To introduce the classical semantics we need the following definition.

Definition 5 *A Valuation for a configuration is a function $\rho : Variables \to V$.*

We denote by $\rho[x/v]$ the valuation that agrees with ρ on every variable but x, in which case it returns v. We will also write $\rho(\bar{x})$ for $\rho(x_1) \cdots \rho(x_n)$ where $\bar{x} = x_1 \cdots x_n$.

We can now give the classical semantics for the configuration logic.

Definition 6 *Let $\mathcal{C}$ be a configuration and ρ be a valuation for this configuration. We say that $\mathcal{C}$ satisfies a configuration logic formula φ under valuation ρ (in notation $\mathcal{C}, \rho \models \varphi$), if recursively:*

- *$\mathcal{C}, \rho \models R_i(\bar{x})$ if $\tilde{R}_i(\rho(\bar{x}))$ holds;*
- *$\mathcal{C}, \rho \models \varphi \wedge \psi$ if $\mathcal{C}, \rho \models \varphi$ and $\mathcal{C}, \rho \models \psi$;*
- *$\mathcal{C}, \rho \models \varphi \vee \psi$ if $\mathcal{C}, \rho \models \varphi$ or $\mathcal{C}, \rho \models \psi$;*
- *$\mathcal{C}, \rho \models \neg\varphi$ if $\mathcal{C}, \rho \not\models \varphi$;*

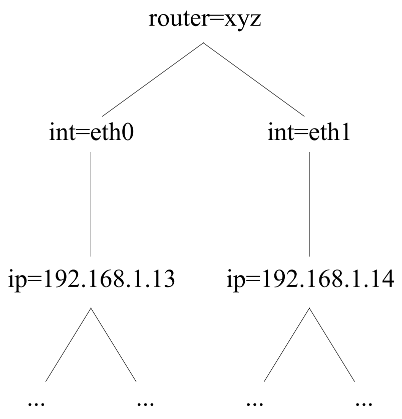

Figure 1. A simple configuration for IP address

- $\mathcal{C}, \rho \models \langle \bar{p} = \bar{x}; p = x \rangle \varphi$ *if there exists a* $v \in V$ *such that* $(\bar{p} = \rho(\bar{x}))(p = v) \in N$ *and* $\mathcal{C}, \rho[x/v] \models \varphi$*;*
- $\mathcal{C}, \rho \models [\bar{p} = \bar{x}; p = x]\varphi$ *if for all* $v \in V$ *such that* $(\bar{p} = \rho(\bar{x}))(p = v) \in N$ *it holds that* $\mathcal{C}, \rho[x/v] \models \varphi$.

Remind that as stated before only the last variable of a quantifier is bound.

3 Motivation

The goal behind the development of CL is to describe and verify properties on configurations of computing equipment. In particular, we are interested in validating the configuration of network routers, which are the equipment responsible for forwarding packets towards their destination.

The configuration of a router is the set of parameter-value pairs that describe the state of the device at a given moment. These parameter-value pairs are organized in a hierarchical fashion: for example, each router may have multiple interfaces, and each interface has its IP address. Figure 1 shows a portion of the configuration of a router containing two interfaces, called `eth0` and `eth1`, whose IP addresses are respectively `192.168.1.13` and `192.168.1.14`.

In a parameter-value pair, the parameter is a static name, while the value is configurable. It is important to note that the parameter-value pair is unique among its siblings. In our example the node `int=eth0` represents the unique interface with name `eth0`.

3.1 Network Management

The global configuration of a network is formed of the configuration of its routers. To ensure proper functioning of the network, specific relations must be satisfied on the values of the parameters, which may span multiple devices.

When new network services are added, some parameters of the configuration must be changed. In order to assure that all services still function properly, these changes must be made in such a way that existing relations are still fulfilled.

Due to the size of present networks and the complexity of services, it is of prime importance to develop formalisms and methods to help manage complex configurations in order to ensure that the network stays in a consistent state.

To illustrate the kind of applications we have in mind, we will give two simple, but still representative examples.

3.2 Example 1: IP addresses

As has been explained earlier, the parameters of the configuration affected by a service must verify some specific relations.

The simplest example of such relation can be seen in an IP address following the Classless Inter-Domain Routing (CIDR) scheme [5], [11], whose two components, the *value* and the *subnet mask*, are linked by a simple relationship: an address like `206.13.01.48/25`, having a network prefix of 25 bits, must carry a mask of at least `255.255.255.128`, while the same address with a network prefix of 27 bits must not have a subnet mask under `255.255.255.224`.

Figure 2 depicts a portion of a configuration representing an IP address (ip) with its subnet mask (mask) and network prefix (pref).

Let $R(m, p)$ be a relation which holds if m is an acceptable mask for the prefix p. The previous property can be expressed by the CL formula of Figure 3, stating that all addresses a must have a mask m and a prefix p satisfying $R(m, p)$.

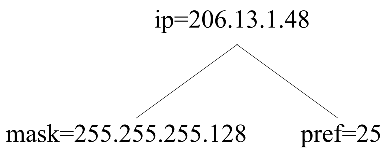

Figure 2. A simple configuration for IP address

$$[\text{ip} = a]\langle \text{ip} = a; \text{mask} = m\rangle$$
$$\langle \text{ip} = a; \text{pref} = p\rangle\ R(m,p)$$

Figure 3. A formula for the correct specification of IP addresses

3.3 Example 2: Virtual Private Networks

More complex situations can be encountered, in which the parameters of several devices supporting the same service are interdependent. An example is provided by the configuration of a *Virtual Private Network* (VPN) service [10], [12], [13].

A VPN is a private network constructed within a public network such as a service provider's network. A customer might have several sites geographically dispersed, and would like to link them together by a protected communication.

Most of the configuration of a VPN is realized in routers placed at the border between the client's and the provider's networks. On the client side, these routers are called *customer edge* (CE) routers, and on the provider side, they are called *provider edges* (PE).

Many techniques have been developed to ensure the transmission of routing information inside a VPN without making this information accessible from the outside. One frequently used method consists in using the Border Gateway Protocol (BGP). This method involves the configuration of each PE to make it a "BGP neighbor" of the other PE's [10].

Without getting into the details, it suffices to know that one interface in each PE router must have its IP address present as a BGP neighbor of each other PE router.

Let $PE(r)$ be a relation satisfied by the PE routers, and $Neighbor(a,r)$ be a relation which holds when

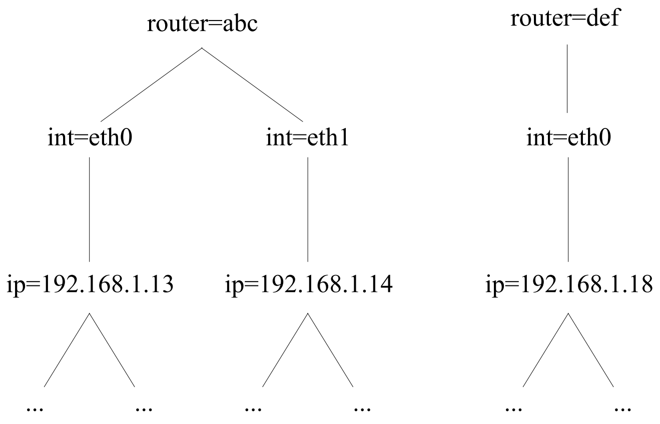

Figure 4. An excerpt from a configuration of a VPN

$$[\text{router} = r_1][\text{r} = r_2]$$
$$(PE(r_1) \wedge PE(r_2) \wedge r_1 \neq r_2 \rightarrow$$
$$\langle \text{router} = r_1; \text{int} = i\rangle Neighbor(i, r_2))$$

Figure 5. A formula for the configuration of a Virtual Private Network

the address a is a BGP neighbor of router r. This property can be expressed by the CL formula of figure 5, stating that for each pair of different routers r_1 and r_2 that are both PE's, some interface of r_1 must be a BGP neighbor of r_2.

4 Related Logics

In this section, we provide a comparison of CL to other related logics.

4.1 Modal Logics

Modal ($\diamond, \Box$) and multi-modal ($\langle a\rangle$, $[a]$) modalities trace a path and allow to refer to properties of nodes in the future. While in modal and multi-modal logic one refers to properties of individual future states, in CL the quantifiers allow to reach different nodes and then refer to a property involving many nodes. For instance the following CL sentence

$$\langle \text{p} = x\rangle\langle \text{p} = y\rangle x \neq y$$

could at best be expressible in multi-modal logic by

$$\bigvee_{a\neq b} \langle \text{p} = a\rangle T \ \wedge \langle \text{p} = b\rangle T$$

where a, b range over the domain of x and y.

Hence classical modal and multi-modal logics can be seen as mono-site: basic relations are on the contents of nodes. On the other hand CL can be seen as a multi-site modal logic: basic relations can involve many nodes.

Of course the presence of variables in modalities will come at a price as we will show below.

4.2 TQL

The logic that mainly inspired us is the Tree Query Logic (TQL) [2, 3]. TQL has been developed as the spatial fragment of a more general logic called the ambient logic. It is a logic which not only allows formulation of sentences that can be model checked against a given tree, but also queries that extract data from those same trees. The main application of TQL is targeted towards the extraction of data from databases modeled by XML files.

Using TQL prefix operator and its quantification on arbitrary labels of nodes, one gets the CL quantifiers. CL is therefore a fragment of TQL. Moreover, TQL provides fix-point operators for expressing recursive properties. Hence, TQL is much more expressive than CL: It allows more flexible quantifications and recursion by fix-points.

It has been shown that TQL is an undecidable logic: there is no algorithm to decide if there exists a finite structure satisfying a TQL sentence [4].

We have used TQL as a tool for the validation of device configurations [7, 8]. This motivates us to investigate fragments of a logic which would be suitable for describing configurations. Our goal is to tailor a logic for configuration purpose and to avoid non necessary constructs like fix-points. Even if our logic is still undecidable as we show below, its simplicity makes easier its integration in a tool. Our team is actually working on its integration in a network configuration tool.

4.3 Guarded Logics

Guarded logic is a fragment of first-order logic which generalize modal logic. In guarded logic all quantifiers must be relativized by atomic formulas. Therefore, quantifiers in the guarded fragment of first-order logic appear only in the form

$$\exists \bar{y}(\alpha(\bar{x}, \bar{y}, \bar{z}) \wedge \psi(\bar{x}, \bar{y}))$$

or

$$\forall \bar{y}(\alpha(\bar{x}, \bar{y}, \bar{z}) \rightarrow \psi(\bar{x}, \bar{y}))$$

The atom α, called the *guard*, must contain all free variables of ψ [6].

The loosely guarded logic is a generalization of guarded logic where the condition on the guard is relaxed. In this case the guard must be a conjunction of atomic formulas such that if x is a free variable of α, and y is a variable from $\bar{y}$, then there is a conjunct in the guard where x and y both occur [9].

These fragments of first-order logic have a number of interesting properties. It has been shown [1] that the satisfiability problem for the guarded fragment is decidable, and, moreover, that it has the finite model property (every satisfiable formula in the guarded fragment has a finite model). The loosely guarded fragment has been shown to have the small model property [9].

Unfortunately, CL configuration properties are neither guarded nor loosely guarded. For instance, let us consider the sub-formula

$$\langle \text{router} = r_1; \text{int} = i\rangle Neighbor(i, r_2)$$

of the sentence of figure 5. It can be translated in first-order terms to

$$\exists_i I(r_1, i) \wedge Neighbor(i, r_2)$$

where $I(r, i)$ holds if i is a interface of router r. Since r_2 is a free variable of $Neighbor(i, r_2)$ which is not in the guard $I(r_1, i)$, this formula is not guarded.

Furthermore, in general, there is no guarded or loosely guarded equivalent to a CL sentence. This follows from the fact that one can define an infinite total order in CL by the following sentences on one binary relation R. The conjunction of these sentences is consistent, but it has no finite model, hence the finite model property does not hold for configuration logic.

$[p = x]\neg R(x, x)$

$[p = x][p = y][p = z]R(x, y) \wedge R(y, z) \to R(x, z)$

$[p = x]\langle p = y\rangle R(x, y)$

4.4 From classical first-order logic to CL

In fact, classical first-order logic can be interpreted in CL by replacing existential quantifiers $\exists x$ by $\langle p = x\rangle$ and universal quantifiers $\forall x$ by $[p = x]$, for some fixed name p.

By Trakhtenbrot's result [14] which states that for a first-order language including a relation symbol that is not unary, satisfiability over finite structures is undecidable, we have the analog for CL.

Therefore there can be no effective way to find a bound on the size of the smallest finite model of a CL formula, since enumerating the structures of this size would give decidability for the existence of a finite structure satisfying the sentence.

5 Adapted Valuations

As shown above CL does not have nice computational properties like decidability and small model property. Nevertheless the simplicity of CL allows the investigation of fragments which would be expressible enough for the applications we have in mind, but would still satisfy these properties. Our investigation of fragments of CL has shown us that the above classical semantic is not appropriate for CL. CL being a logic about path in trees, it is difficult to work with valuations which are not constrained by the tree structure. For instance, our existential quantifiers mean the existence of a value on a path an not merely of some value.

Therefore we now give a semantics in terms of partial functions on variables.

The idea is that in order to check a sentence one has to recursively check sub-formulas. In turn, to check a sub-formula, one has to consider valuations. We show in this section that instead of considering general valuations, it is sufficient to restrict ourselves to functions sending variables to values that satisfy the hierarchical structure of the variables in the sentence. This allows to integrate the hierarchical condition on the values of variables into the definition of these new kinds of valuations.

We propose this new semantics in this section and show that it is equivalent to the previous classical semantics.

Definition 7 *The Path of a sub-formula of the form $\langle p = x\rangle\psi$ or $\langle p = x\rangle\psi$ is $(p = x)$. Similarly, the Path of a sub-formula $\langle \bar{p} = \bar{x}; q = y\rangle\psi$ and $\langle \bar{p} = \bar{x}; q = y\rangle\psi$ is $(\bar{p} = \bar{x})(q = y)$.*

Since in a specific sentence a variable is bound only once, we will speak of the *Path* of a bound variable which is the path of the quantifier binding this variable in the sentence.

From definition 1 one can show by induction that the following result holds.

Proposition 1 *Let φ be a sentence and x a variable of φ of path $(p_n = x_1), \ldots, (p_n, x_n)(p, x)$. For all $i = 1, \ldots, n$ we have that the path of x_i is $(p_n = x_1), \ldots, (p_i, x_i)$.*

We will say that $f : A \to B$ is a *partial function* if f is a function sending elements of its domain $dom(f)$ to elements of B.

Let φ be a CL formula. We denote by $Variables(\varphi)$ the set of variables (bound or free) of φ.

We can now give the definition of our restricted form of valuation.

Definition 8 *Let $\mathcal{C} = \langle V, N, \tilde{R}_1, ..., \tilde{R}_n\rangle$ be a configuration and φ be a sentence. A partial function $\rho : Variables(\varphi) \to V$ is said to be adapted (or φ-adapted) for $\mathcal{C}$ if for every variable $y \in dom(\varphi)$ of φ of path $(p_1 = y_1) \cdots (p_n = y_n)(p = y)$, the following conditions holds:*

1. *$\{y_1, \ldots, y_n\} \subseteq dom(\rho)$*
2. *$(p_1 = \rho(y_1)) \cdots (p_n = \rho(y_n))(p = \rho(y)) \in N$.*

We now have the following fact.

Proposition 2 *Let $\mathcal{C} = \langle V, N, \tilde{R}_1, ..., \tilde{R}_n\rangle$ be a configuration, φ be a sentence, and ρ be a valuation for $\mathcal{C}$ adapted to φ.*

Let also $(p_1 = x_1)\cdots(p_r = x_r)(q = y)$ be the path of y in φ for some $y \notin dom(\rho)$.

We have that if $\{x_1,\dots,x_n\} \in dom(\rho)$ and if $v \in V$ is such that $(p_1 = \rho(x_1))\cdots(p_r = \rho(x_r))(q = v) \in N$ then $\rho' = \rho[y/v]$ is adapted to φ.

Proof 1 *To prove that $\rho' = \rho[y/v]$ is adapted, we must show that for $y' \in dom(\rho')$ of path $(q_1 = y_1)\cdots(q_m = y_m)(q = y')$ it holds that*

1. *$\{y_1,\dots,y_n\} \subseteq dom(\rho')$*
2. *$(q_1 = \rho'(y_1))\cdots(q_m = \rho'(y_m))(q = \rho'(y')) \in N$*

As $y \notin dom(\rho)$, y cannot appears in any path of a variable of $dom(\rho')$ except its own. Therefore the claim must be shown only for $y' = y$.

For $y = y'$ the claim follows from the hypothesis.

By the previous result we have that if ρ is a valuation satisfying definition 6 is adapted and if its domain contains all free variables of the formula under consideration but not y, then the valuations $\rho[y/v]$ considered in this definition are again adapted.

This fact makes it possible to precise the relationship between adapted and "regular" valuations.

Lemma 1 *Let φ be a sub-formula of some sentence φ', let $\mathcal{C} = \langle V, N, \tilde{R}_1, ..., \tilde{R}_n\rangle$ be a configuration and ρ be a valuation for $\mathcal{C}$ whose domain contains all free variables of φ.*

If F is a set of variables containing all free variables of φ but none of its bound variables, and if $\rho|_F$, the restriction of ρ to the domain F, is φ'-adapted, then the following conditions are equivalent:

1. *$\mathcal{C}, \rho \models \varphi$*
2. *$\mathcal{C}, \rho|_F \models \varphi$*

Proof 2 *The proof goes by induction on the structure of φ.*

The cases of an atomic formula, conjunction, disjunction and negation are clear.

All cases of existential and universal quantifiers are similar, so we give details only for $\varphi = \langle \bar{p} = \bar{x}; q = y\rangle\psi$.

If $\mathcal{C}, \rho \models \langle \bar{p} = \bar{x}; q = y\rangle\psi$, then by definition 6, we have that there exists a $v \in V$ such that $(\bar{p} = \rho(\bar{x}))(q = v) \in N$ and $\mathcal{C}, \rho[y/v] \models \psi$.

Now since $y \notin F$ it follows that $y \notin dom(\rho|_F)$. Therefore by Proposition 2 it follows that $(\rho|_F)[y/v]$ is φ'-adapted.

Let $F' = F \cup \{y\}$. We have that $(\rho|_F)[y/v] = \rho[y/v]|_{F'}$ since they both agree on F and on y. So $\rho[y/v]|_{F'}$ is φ'-adapted.

Since F' contains all free and no bound variable of ψ, it follows by induction hypothesis that $\mathcal{C}, \rho[y/v]|_{F'} \models \psi$ holds. Again by equality $\rho|_F[y/v] = \rho[y/v]|_{F'}$ and by definition 6 we have that $\mathcal{C}, \rho|_F \models \langle \bar{p} = \bar{x}; q = y\rangle\psi$ holds.

Conversely if $\mathcal{C}, \rho|_F \models \langle \bar{p} = \bar{x}; q = y\rangle\psi$, by definition 6, we have that there exists a $v \in V$ such that $(\bar{p} = \rho(\bar{x}))(q = v) \in N$ and $\mathcal{C}, \rho|_F[y/v] \models \psi$.

As before, it follows from Proposition 2 that $(\rho|_F)[y/v]$ is φ'-adapted. Furthermore $(\rho|_F)[y/v] = \rho[y/v]|_{F'}$ holds.

Therefore we have by induction hypothesis that $\mathcal{C}, \rho[y/v] \models \psi$ and hence $\mathcal{C}, \rho \models \langle \bar{p} = \bar{x}; q = y\rangle\psi$.

It now follows that:

Theorem 1 *Let $\mathcal{C} = \langle V, N, \tilde{R}_1, ..., \tilde{R}_n\rangle$ be a configuration, φ be a sentence and ρ be a valuation for $\mathcal{C}$. Let $\emptyset$ be the empty φ-adapted valuation (its domain is the empty set). We have that $\mathcal{C}, \rho \models \varphi$ if and only if $\mathcal{C}, \emptyset \models \varphi$.*

From the previous result we get the following equivalence.

Theorem 2 *Let $\mathcal{C} =< V, N, \tilde{R}_1, ..., \tilde{R}_n >$ be a configuration and φ be a sentence. The following condition are equivalent.*

1. *$\mathcal{C}, \rho \models \varphi$ for all valuations ρ;*
2. *$\mathcal{C}, \rho \models \varphi$ for some valuation ρ;*
3. *$\mathcal{C}, \rho \models \varphi$ for some φ-adapted valuation ρ;*
4. *$\mathcal{C}, \rho \models \varphi$ for all φ-adapted valuations ρ.*

Proof 3 *The result follows directly from Theorem 1, since to check that $\mathcal{C}, \rho \models \varphi$ for some valuation ρ, it suffices to check that $\mathcal{C}, \emptyset \models \varphi$ holds.*

Remark 1 *It is important to note that the hierarchical structure of variables constrains the possible adapted valuations. Therefore even if the empty valuation is always an adapted valuation, there is not always an adapted valuation whose domain contains all free variables, as the following example shows.*

Example 1 *If $\mathcal{C} = \langle V, N, \tilde{R}_1, ..., \tilde{R}_n \rangle$ is such that N contains no $(p = v)$ for some name p then there is no φ-adapted valuation on $\mathcal{C}$ for $\varphi = \langle p = x \rangle x = x$, whose domain contains x.*

6 Conclusion

We proposed a new logic for describing the configuration of computing equipment and motivated it with examples from the field of network configuration. We also gave a classical and a new equivalent semantics.

Since we are interested in applications, we are working at integrating CL in a network configuration tool. We are also working on using our new semantics to investigate a fragment of CL which would be sufficient to express the properties needed in practice, while having better theoretical properties like decidability and small model property.

References

[1] H. Andréka, J. van Benthem, I. Németi: Modal Languages and Bounded Fragment of Predicate Logic, ILLC Research Report ML-96-03 (1996), 59 pages.

[2] Cardelli, L.: Describing semistructured data. SIGMOD Record, 30(4), 80–85. (2001)

[3] Cardelli, L., Ghelli, G.: TQL: A query language for semistructured data based on the ambient logic. To appear in Mathematical Structures in Computer Science.

[4] Giovanni Conforti, Giorgio Ghelli: Decidability of Freshness, Undecidability of Revelation. FoSSaCS 2004: 105-120

[5] Fuller, V., Li, T., Yu, J., Varadhan, K.: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519 (1993)

[6] Grädel, E.: On The Restraining Power of Guards. J. Symb. Log. 64(4): 1719-1742 (1999)

[7] Sylvain Hallé, Rudy Deca, Omar Cherkaoui, Roger Villemaire: Automated Validation of Service Configuration on Network Devices. MMNS 2004: 176-188

[8] Sylvain Hallé, Rudy Deca, Omar Cherkaoui, Roger Villemaire, Daniel Puche: A Formal Validation Model for the Netconf Protocol. DSOM 2004: 147-158

[9] Hodkinson, I. M.: Loosely Guarded Fragment of First-Order Logic has the Finite Model Property. Studia Logica 70(2): 205-240 (2002)

[10] Pepelnjak, I., Guichard, J.: MPLS VPN Architectures, Cisco Press (2001)

[11] Rekhter, Y., Li, T.: An Architecture for IP Address Allocation with CIDR. RFC 1518 (1993)

[12] Rosen, E., Rekhter, Y.: BGP/MPLS VPNs. RFC 2547 (1999)

[13] Scott, C., Wolfe, P. Erwin, M.: Virtual Private Networks, O'Reilly (1998)

[14] B.A. Trakhtenbrot, Impossibility of an algorithm for the decision problem in finite classes, Dok. Akad. Nauk SSSR 70 (1950) 569-572.