Skolem Functions and Herbrand Universes in a Tree Generalization of First Order Logic

Roger Villemaire Sylvain Hallé Omar Cherkaoui
Department of Computer Science, Université du Québec à Montréal
C.P. 8888, Succ. Centre-ville, Montréal (Canada) H3C 3P8
villemaire.roger@uqam.ca

Abstract

Skolem functions and Herbrand universes are fundamental concepts in first-order logic that form the basis of many works in artificial intelligence. In this paper, we study a fragment of the XML Query Language (XQuery) that generalizes first-order logic to a setting where variables form a forest instead of a set. A formal description of the logic and its semantics is given; Skolem functions and Herbrand universes are generalized to this setting.

1. Introduction

Trees, forests and hierarchical structures in general are fundamental in computer science. The advent of the Extensible Markup Language (XML) in recent years as the *de facto* standard for the representation and transmission of information over the Internet, especially in the area of web services, has further confirmed the importance of so-called *semi-structured* data. Consequently, logical formalisms such as XPath [8] and XQuery [4] have been developed to express properties and extract data from these structures.

Although considerable work has been carried around these logics from a database and computational complexity standpoint, fewer works have investigated them in an AI perspective. First-order logic (FOL) has long become a household name with the success of general purpose reasoning tools like Prolog; in this field, skolemization and Herbrand models are fundamental concepts that form the basis of a large corpus of works in artificial intelligence [12, 16], particularly in automated deduction and reasoning such as the Otter theorem prover [21], the SNARK [1] tool, the Protheo software verification project [2]; on the other hand, MACE [17] and SEM [22] are tools that find and generate models and counter-examples for first-order theories. However, the same methods remain yet to be developed for tree logics.

In this paper, we investigate a subset of the XML Query Language XQuery that we call Configuration Logic (CL). We show that CL is a pure generalization of FOL where the variables, instead of forming a "flat" set structure, are hierarchically organized into forests. Our aim is then to restore into this setting the Skolem functions and Herbrand universes similarly to classical FOL. Equipped with these constructions, we show how properties of FOL transfer directly to CL; in particular, tree

We gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada on this research.

structures can be automatically generated to fulfill some predefined constraints by using the same model-building procedure as for classical FOL.

The motivation for studying skolemization in CL stems from its use as a formalism expressing constraints in the configuration of network routers. The principles described in this paper are being integrated into an existing automated validation tool called ValidMaker [10].

The paper is structured as follows. Section 2 motivates by simple examples from network configuration the kind of Skolem functions we need and summarizes related work. The proposed logic is defined in section 3; Skolem functions are generalized to CL and Herbrand universes are presented in section 4. Finally, section 5 concludes the paper.

2. Examples and Motivation

The development and study of Configuration Logic as a subset of the well-known tree language XQuery originates from the representation of the configuration of network devices such as routers. All properties of a given router configuration are represented by attribute-value pairs. However, these pairs are organized in a hierarchy that takes the form of a tree structure. The tree representation is a natural choice, since it reflects dependencies among components, and it is in close relatioship to the XML format used by many network configuration management protocols such as Netconf [11].

2.1. Network Configuration Examples

Figure 1 depicts a portion of a configuration representing an IP address with its subnet mask. According to the IP addressing scheme, the declaration of the IP address of a device must always be accompanied by its mask. This example gives a first, basic tree property.

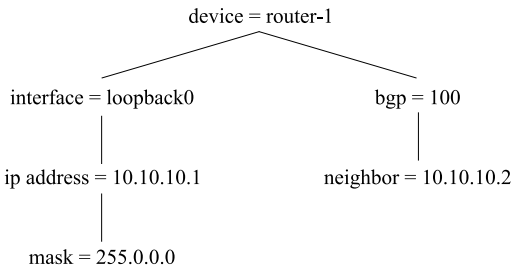

Figure 1. A portion of the configuration tree for a router

As a more complex example, a *Virtual Private Network* (VPN) service [19] is a private network constructed within a public network such as a service provider's network. Usually, the VPN is used to link together several geographically dispersed sites of a customer by a protected communication throughout the provider's network. 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).

An important issue is to ensure the transmission of routing information between the sites forming 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 PEs [18]; this entails that one interface in each PE router must have its IP address declared as a BGP neighbor in each other PE router.

These are only two examples of network properties that must be validated on configurations of routers. Actually, each network service imposes dozens of such constraints, and in turn dozens of different services can coexist on networks formed of hundreds of devices.

2.2. A Case for Generalizing First-order Logic

It is important for the network engineer to validate a given configuration against a set of service rules; however, it is also equally important for a configuration tool to allow the suggestion of correct values to the user, as well as the automated generation of parts of the configuration that ensure no rule is violated. These two features require automated model construction, a property that is not readily available when using tree languages like CL.

To this end, Skolem functions and Herbrand universes are well-known model construction techniques. Let us examine skolemization with this simple network property: every IP address has a mask. In a classical first-order setting, one would introduce a Skolem function $f_{\text{mask}}(a)$ giving for an IP address a its mask $f_{\text{mask}}(a)$. But this is misleading: the mask is not really a function of the IP address. With network address translation (NAT), the same IP address can be used at different sites and there is no reason why the mask should always be the same. In fact, the mask is really a function of the *node's* IP address: it is the mask of the IP address of *this* specific interface in the router's configuration.

One could think that considering configuration nodes as being typed (e.g. a type for the nodes containing IP addresses) would be sufficient. This is not the case, for IP addresses appear as values in many different locations in a configuration. For instance, a BGP neighbor (right branch in Figure 1) is specified by giving the IP address of an interface of the neighbor, but not its mask. The mask is provided only when configuring the interface. Nevertheless, the IP address given for an interface or for a BGP neighbor configuration has the same structure and must be comparable. We need to check whether a device is indeed a BGP neighbor of some other by comparing an interface's IP address with an IP address appearing in a neighbor node of the BGP configuration. The issue is therefore that one must consider the hierarchical position of a parameter: the mask is present below the *ip address*, of the *interface*, of a *device*; it is not present in the *neighbor* declaration of the *bgp* configuration of a *device*.

Our approach is simply to consider a Skolem function to return not just a value but a descendant of a specific node. In the case of the mask, given a *device* x, an *interface* y of this device and the *ip address* z of this interface, the Skolem function returns a node containing the mask, the mask's node being at the end of the path

$$(\text{device} = x)(\text{interface} = y)(\text{ip address} = z)$$

We hence denote our Skolem function by

$$(\text{device} = x)(\text{interface} = y)(\text{ip address} = z)f_{\text{mask}}(\bar{x})$$

prefixing the function symbol by the path to the return node. Note that in this case $\bar{x}$ contains x, y, z but in general it could contain more variables, since the value of the returned node could be function of values on other branches.

Even if this approach seems quite different from the usual notion of Skolem functions, we show in this paper that simply adding a forest structure on the set of variables allows to give a precise and formal presentation following the lines of classical first-order logic.

2.3. Related Work

In the context of network configuration, quantifiers are needed to express properties of the form "for all routers r, there exists an interface i of r". However, quantification on *all* nodes (as in "for all parameters") is never necessary. The situation is therefore similar to multi-modal logics like

LTL and CTL [9] where the operators $\langle a \rangle$ and $[a]$ respectively denote "there exists an action a" and "for all actions a". Configuration Logic uses a similar syntax where a is a parameter. Modal ($\diamond, \Box$) and multi-modal ($\langle a \rangle$, $[a]$) logics trace a path and allow to refer to properties of nodes in the future. Hence classical modal and multi-modal logics can be seen as mono-site: one refers to properties of individual future states and basic relations are on the contents of a single node. On the other hand properties like those of the previous examples involve basic relations on tuples of nodes as in first-order logic. This has been explained in more detail in [20] and makes modal logics insufficient for the task.

Guarded logic is a fragment of first-order logic allowing n-ary relations, that generalize multi-modal logic. Guarded logic and a further generalization called weakly-guarded logic have been showed to have the small model property [3, 15]. Unfortunately, it was shown in [20] that network properties have no guarded or loosely guarded equivalent sentences. Therefore, neither guarded nor weakly guarded logics are sufficient for describing properties of configurations.

Finally, the Tree Query Logic (TQL) [5, 6], a logic for querying XML documents, was used in [13, 14] to verify configuration properties of network services. However, TQL is a very powerful query logic that has undecidable model checking [7], and of which only a small fragment was actually needed.

3. Configuration Logic

In this section, we formally present CL as a fragment of XQuery and define the special concepts that generalize FOL.

3.1. Names and Forests

In CL, the names of the parameters are considered static and given beforehand. We fix a set $Names$ of *parameter names* (for short just names). In classical first-order logic, the semantics are defined on a set and the variables form a set. Similarly in CL, the semantics are defined on a *named forest*.

Definition 1. A named forest is a set of trees whose nodes are labeled by names. If N is a node of a forest, we write $label(N)$ for its label.

Since a configuration is a forest whose nodes contain name-value pairs, it is therefore a named forest. We will also make the set of variables of the logic to be a named forest.

Definition 2. A named path is a finite non-empty sequence of names. The named path of an element in a named forest is the sequence of names encountered on the path from a root to this element.

A valuation is a way to associate values to variables. Since variables and values form named forests, the valuations have to preserve this structure.

Definition 3. A named forest morphism (n.f.m. for short) $\alpha : F_1 \to F_2$ from a named forest F_1 to a named forest F_2 is a partial function from the nodes of F_1 to the nodes of F_2 (with domain $dom(\alpha)$) such that:

- $dom(\alpha)$ is a sub-forest (i.e. the parent of an element of $dom(\alpha)$ is also in $dom(\alpha)$).
- if N is a root, then $\alpha(N)$ is a root
- $label(\alpha(N)) = label(N)$
- if N_2 is a child of N_1, then $\alpha(N_2)$ is a child of $\alpha(N_1)$.

3.2. Syntax and Structures

CL is syntactically similar to first-order logic, but again the set of variables forms a named forest instead of a simple set. In this sub-section we formally define CL and its semantics.

Definition 4. A *Configuration Logic* language (for short a CL *language*) $\mathcal{CL}$ is formed of

- a named forest V of *variables* $v, w, v_1, v_2, w_1, \ldots$
- a set $\mathcal{R}$ of relation symbols $R_1(\bar{v}), R_2(\bar{w}), \ldots$ where $\bar{v}$, $\bar{w}$ are finite sequences of variables.
- a set $\mathcal{F}$ of partial function symbols $v.f(\bar{v}), w.g(\bar{w}), \ldots$ where $v, w, \ldots$ (called the *prefix*) are either variables or the empty string; in the latter case we usually just write $f(\bar{v}), g(\bar{w}), \ldots$ The $\bar{v}, \bar{w}, \ldots$ are finite sequences of variables. Similarly to variables, each of these partial function symbols also has a name attached to it.

In the above definition, the arity of $R(\bar{v})$ and of $f(\bar{v})$ is the length of $\bar{v}$. The arity of $v.f(\bar{v})$ is the length of $\bar{v}$ plus one.

Definition 5. Let $\mathcal{CL}$ be a CL language and F be a named forest. A $\mathcal{CL}$-valuation for F (if the context is clear we will just say valuation) is an n.f.m. $\rho : V \to F$, where V is the forest of variables of $\mathcal{CL}$.

In order to formally define configurations (the structures on which CL will be interpreted), we must define how relations and partial functions symbols are to be interpreted. Since in CL, variables form a named forest, the interpretation of a relation or partial function symbols must preserve this structure.

This is not always the case. Consider for example a binary relation $R(x, y)$ where y is a child of x. This relation cannot be interpreted as a set of pairs of nodes, since this would not preserve the fact that y is a child of x. In fact, this is not surprising if one keeps in mind that variables cannot be mapped to arbitrary values but must be mapped using named forest morphisms. The interpretation of a relation R can hence be defined to be the set of n.f.m. making R hold.

Similarly, the interpretation of a partial function $w.f(\bar{w})$ will be a partial function sending a valuation ρ to the interpretation of $w.f(\bar{w})$ which will be a child of $\rho(w)$, in accordance with the intuition that $w.f(\bar{w})$ denotes a child of w.

For technical reasons, when considering a relation or a partial function symbol it is convenient to restrict ourselves to valuations defined only on variables relevant to the relation or the partial function symbols. This set of variables must obviously contain the variables appearing in the relation or partial function symbol but it must also contain their ancestors, since an n.f.m.'s domain is always a sub-forest.

Definition 6. Let F be a named forest and $\bar{m}$ be a finite sequence of elements of F. The sub-forest generated by $\bar{m}$, noted $sf(\bar{m})$, is the smallest sub-forest containing all the elements of the sequence $\bar{m}$ (i.e. the closure under parenthood).

Definition 7. Let $\mathcal{CL}$ be a CL language, F be a named forest and $\rho : V \to F$ be a $\mathcal{CL}$-valuation. Let also $\bar{v}$ be a finite sequence of variables. We will say that ρ is "a valuation on $\bar{v}$" if $dom(\rho) = sf(\bar{v})$.

Definition 8 (Configuration)**.** Let $\mathcal{CL}$ be a CL language. A $\mathcal{CL}$-configuration (if the context is clear we will often just speak of a configuration) is a structure

$$\mathcal{M} = \langle M; R_{\mathcal{M}}(\bar{v}), w.f_{\mathcal{M}}(\bar{w}) \rangle$$

where

- R ranges over $\mathcal{R}$
- $w.f(\bar{w})$ ranges over $\mathcal{F}$
- M is a named forest
- $R_{\mathcal{M}}(\bar{v})$ is a set of valuations for M on $\bar{v}$
- $w.f_{\mathcal{M}}(\bar{w})$ is a partial function sending a valuation ρ for M on $w, \bar{w}$ to a child of $\rho(w)$ in M or to a root of M, if w is empty. Furthermore this node must have the same name as $w.f(\bar{w})$.

We abuse notation and write $\rho \in R_{\mathcal{M}}(\bar{v})$ to mean that $dom(\rho) \supseteq sf(\bar{v})$ and that the restriction of ρ to $sf(\bar{v})$ is in $R_{\mathcal{M}}(\bar{v})$. Therefore, the interpretation of a relation symbol is seen as a relation on the set of valuations. Similarly, for partial function symbols, we consider that a valuation ρ is in the domain of the interpretation if $dom(\rho) \supseteq sf(\bar{w})$ and if the restriction of ρ to $sf(\bar{w})$ is in $dom(w.f_{\mathcal{M}}(\bar{w}))$. Therefore, we consider that the interpretation of a partial function symbol is a partial function from the set of all valuations to the set of nodes.

Finally, even if we write $w.f_{\mathcal{M}}(\bar{w})$ for the partial function interpretation, we write $w.f_{\mathcal{M}}(\rho)$ for the application of this function to the argument ρ (which is a valuation) in order to simplify notation.

3.3. Terms, Substitutions, Formulas and Sentences

To define the terms (composition of partial function symbols) of the language, we must define how substitutions of variables are done. Here again, a variable cannot be replaced by any term but only by terms that preserve the named forest structure. One must therefore define terms and substitution by a mutually recursive definition by giving both definitions simultaneously.

Note that terms will naturally form a named forest, taking $w.f(\bar{w})$ to be a child of w where $w, \bar{w}$ can either be variables or terms. For instance, if η is a substitution, we write $v.f(\bar{v})\eta$ for the term obtained by replacing the variables $v, \bar{v}$ with the terms given by η.

Definition 9 (Term and Substitution)**.** Let $\mathcal{CL}$ be a CL language.

- A term of $\mathcal{CL}$ is either a variable or the result of applying a substitution on a partial function symbol (in notation $v.f(\bar{v})\eta$). We will denote the set of terms by $Terms(\mathcal{CL})$ or just $Terms$ if the context is clear.
- A substitution of $\mathcal{CL}$ is a n.f.m $\eta : V \to Terms(\mathcal{CL})$.

The notion of formula now becomes similar to FOL.

Definition 10 (Formula)**.** Let $\mathcal{CL}$ be a CL language. A formula of $\mathcal{CL}$ is built from atomic formulas (which are $R\eta$ where $R \in \mathcal{R}$ and η is a substitution), using the usual Boolean connectives $\wedge, \vee, \neg$ as well as the following two quantifiers:

- Existential quantifier: $\langle v \rangle \varphi$, where v is a variable and φ a formula.
- Universal quantifier: $[v]\varphi$, where v is a variable and φ a formula.

We say that a variable is free in a formula φ if it has a descendant which is not bounded by a quantifier. $Free(\varphi)$ denotes the set of free variables of φ. As usual, a *sentence* is a formula having no free variable. In order to simplify the presentation, we will consider that every variable is quantified at most once.

3.4. Semantics

To give the formal semantics for interpreting formulas on configurations, we first need to give the interpretation of terms.

Consider a partial function symbol $w.f(\bar{w})$ and its interpretation $w.f_{\mathcal{M}}$ in a configuration $\mathcal{M}$. If a valuation ρ is defined on the variables of $sf(w, \bar{w})$, then ρ can be extended to $w.f(\bar{w})$ by setting $\rho(w.f(\bar{w}))$ to be $w.f(\rho)$. This extension is an n.f.m. by Definition 8. The following definition extends valuations to arbitrary terms.

Definition 11. Let $\mathcal{M} = \langle M; R_{\mathcal{M}}(\bar{v}), w.f_{\mathcal{M}}(\bar{w})\rangle$ be a $\mathcal{CL}$-configuration for the CL language $\mathcal{CL}$. Let $v.f(\bar{v})\eta$ be a term. A valuation ρ for M can be extended to $v.f(\bar{v})\eta$ by recursively defining $\rho(v.f\eta) = (v.f_{\mathcal{M}})(\rho \circ \eta)$.

This definition makes sense if $\rho \circ \eta$ is defined (i.e. ρ is defined for all terms in the range of η) and $\rho \circ \eta \in dom(v.f_{\mathcal{M}})$.

Since n.f.m.'s are closed under composition, $\rho \circ \eta$ is indeed a valuation. Therefore, a valuation ρ defined on the variables of a term can be recursively extended by the above method to the term itself. In the following we will not distinguish between the valuation and this extension. Similarly, we say that $\rho \circ \eta$ is defined if ρ can be extended to all terms in the range of η.

Definition 12. Let $\alpha : F_1 \to F_2$ be an n.f.m. and let a be an element of F_1. We say that $\alpha' : F_1 \to F_2$ extends α to a or that it is an extension of α to a if $dom(\alpha) \subseteq dom(\alpha')$, $a \in dom(\alpha')$ and α and α' agree on $dom(\alpha)$.

Definition 13 (Semantics)**.** Let $\mathcal{CL}$ be a CL language and $\mathcal{M}$ be a configuration. Let φ be a CL formula and ρ be a valuation for M on $Free(\varphi)$ (if $dom(\rho)$ is greater than $Free(\varphi)$ we consider, by notation abuse, that ρ is replaced by its restriction to $Free(\varphi)$).

We say that $\mathcal{M}, \rho$ satisfies φ (in notation $\mathcal{M}, \rho \models \varphi$), if we have recursively:

- if $\varphi \equiv R\eta$, then $\rho \circ \eta$ is defined and is in $R_{\mathcal{M}}$
- if $\varphi \equiv \psi_1 \wedge \psi_2$, then $\mathcal{M}, \rho \models \psi_1$ and $\mathcal{M}, \rho \models \psi_2$
- if $\varphi \equiv \psi_1 \vee \psi_2$, then $\mathcal{M}, \rho \models \psi_1$ or $\mathcal{M}, \rho \models \psi_2$
- if $\varphi \equiv \neg\psi_1$, then $\mathcal{M}, \rho \not\models \psi_1$
- if $\varphi \equiv \langle v\rangle\psi$, then there exists an extension ρ' of ρ to v such that $\mathcal{M}, \rho' \models \psi$
- if $\varphi \equiv [v]\psi_1$, then for all extensions ρ' of ρ to v, $\mathcal{M}, \rho' \models \psi_1$ holds.

The above recursive semantics gives a model checking algorithm for CL. Note that if one considers the configuration size to be constant, then this algorithm is linear in the size of the formula.

4. Skolem Functions and Herbrand Configurations

We show in this section how Skolem functions and Herbrand universes can be generalized to CL. Herbrand universes give a configuration construction method.

4.1. Skolem Functions

As for classical first-order logic, skolemization is the process of expanding the language by adding function symbols, in order to replace a sentence by an equivalent universal sentence. To simplify the presentation of Skolem functions, we consider that every formula has been converted to negative normal form (NNF) by pushing negations down to atomic formulas.

Definition 14 (Skolemization)**.** Let $\mathcal{CL}$ be a CL language and φ be a sentence in this language. We define recursively a skolemization of φ as:

- an extension SL of the language $\mathcal{CL}$ by adding new partial functions symbols

- a universal sentence $Skolem(\varphi)$ in SL

in the following way:

- $Skolem(R\eta) = R\eta$
- $Skolem(\neg R\eta) = \neg R\eta$
- $Skolem(\varphi \wedge \psi) = Skolem(\varphi) \wedge Skolem(\psi)$
- $Skolem(\varphi \vee \psi) = Skolem(\varphi) \vee Skolem(\psi)$
- $Skolem(\langle v \rangle \varphi) = Skolem(\varphi \{v/w.f(\bar{w})\})$ where w is the parent of v, the variables $\bar{w}$ are the free variables of $\langle v \rangle \varphi$ and $w.f(\bar{w})$ is a new partial function symbol added to SL having the same name as v
- $Skolem([v]\varphi) = [v]Skolem(\varphi)$

As for classical first-order logic, one can for a configuration $\mathcal{M}$ on $\mathcal{CL}$ create a configuration on SL called *Skolem Extension* $\mathcal{SM}$ to SL, in the following way:

- $\mathcal{SM}$ agrees with $\mathcal{M}$ for all relations and partial functions of $\mathcal{CL}$
- for new partial functions, one defines recursively (i.e. during the recursion of definition 14): If $w.f(\bar{w})$ is introduced for $\langle v \rangle \varphi$, let the domain of $w.f_{\mathcal{SM}}(\bar{w})$ be the set of ρ such that $\mathcal{SM}, \rho \models \langle v \rangle \varphi$ and pick ρ', an extension of ρ to v, such that $\mathcal{SM}, \rho' \models \varphi$. Finally, define $w.f_{\mathcal{SM}}(\rho)$ to be $\rho'(v)$.

We can now show that as for first-order logic, the skolemization of a CL formula preserves its validity.

Proposition 1. *Let $\mathcal{CL}$ be a CL language, φ be a sentence in this language and SL and $Skolem(\varphi)$ be a skolemization of φ. Let also $\mathcal{M}$ be a $\mathcal{CL}$-configuration and $\mathcal{SM}$ be a* Skolem Extension *of $\mathcal{M}$ to SL.*

We have that for any valuation ρ and formula ψ of SL met during skolemization, $\mathcal{SM}, \rho \models \psi$ holds if and only if $\mathcal{SM}, \rho \models Skolem(\psi)$ holds.

Proof. By induction on the number of connectors and quantifiers in ψ.

For ψ an atomic formula, the negation of an atomic formula, a conjunction, a disjunction or a universal quantified formula the result easily holds.

For $\psi \equiv \langle v \rangle \varphi$, we have that since $\mathcal{SM}$ is a Skolem extension of $\mathcal{M}$, it follows that $\mathcal{SM}, \rho \models \langle v \rangle \varphi$ if and only if $\mathcal{SM}, \rho \models \varphi \{v/w.f(\bar{w})\}$.

By induction hypothesis, we have that

$$\mathcal{SM}, \rho \models \varphi \{v/w.f(\bar{w})\} \text{ if and only if } \mathcal{SM}, \rho \models Skolem(\varphi \{v/w.f(\bar{w})\})$$

By definition of skolemization, $Skolem(\langle v \rangle \varphi) = Skolem(\varphi \{v/w.f(\bar{w})\})$. Therefore, the claim holds. □

4.2. Herbrand Configurations

Now that we have generalized Skolem functions to CL, we can build configurations using Herbrand universes as we show in this section.

Definition 15. A term is said to be closed if it contains no variables. We denote by $CT(\mathcal{CL})$ the closed terms of a configuration language $\mathcal{CL}$.

As for classical first-order logic, the value of a closed term does not depends on the chosen valuation.

Proposition 2. *Let t be a closed term t and ρ be a valuation. The node $\rho(t)$ is independent of the valuation chosen.*

Proof. The proof goes by induction. If the term is a function symbol, then it is a term of the form $f()$, with empty prefix and no variables. Therefore $dom(f())$ contains a unique valuation (the empty valuation).

If the result holds for the terms of the range of η, then since $\rho(v.f\eta) = (v.f_{\mathcal{M}})(\rho \circ \eta)$ and since $\rho \circ \eta$ does not depend on ρ by induction hypothesis, we have that the value of the term does not depend on ρ. □

Herbrand configurations are, as for classical first-order logic, the configurations built using closed terms.

Definition 16. Let $\mathcal{CL}$ be a CL language and $\mathcal{M}$ be a $\mathcal{CL}$-configuration. The Herbrand configuration of $\mathcal{M}$ is the configuration

$$\mathcal{HM} = \langle HM; R_{\mathcal{HM}}(\bar{v}), w.f_{\mathcal{HM}}(\bar{w}) \rangle$$

- R ranges over $\mathcal{R}$
- $w.f(\bar{w})$ ranges over $\mathcal{F}$
- $HM = \{\rho(t); t \in CT(\mathcal{CL})\}$, where ρ is a valuation
- $R_{\mathcal{HM}} = R_{\mathcal{M}}$ for $R \in \mathcal{R}$
- $v.f_{\mathcal{HM}}(\bar{v}) = v.f_{\mathcal{M}}(\bar{v})$, for $v.f(\bar{v}) \in \mathcal{F}$

Proposition 2 shows that HM is well defined. Finally, as for classical first-order logic, satisfiability of a universal sentence is preserved by sub-structure.

Proposition 3. *Let $\mathcal{CL}$ be a CL language, $\mathcal{M}$ be a $\mathcal{CL}$-configuration and $\mathcal{HM}$ be the Herbrand configuration of $\mathcal{M}$. If φ is a universal sentence satisfied by $\mathcal{M}$, then φ is also satisfied by $\mathcal{HM}$.*

Proof. A valuation ρ of $\mathcal{HM}$ is also a valuation of $\mathcal{M}$. One can show the result by induction on the structure of the formula. □

As in classical first-order logic, Herbrand configurations are a way to build a configuration satisfying a given sentence. Take φ such a sentence. Take $\mathcal{M}$ a configuration satisfying φ, $\mathcal{SM}$ be its Skolem extension and $\mathcal{HSM}$ be the Herbrand universe of $\mathcal{SM}$. Since $\mathcal{SM}$ agrees with $\mathcal{M}$ for all relations and partial functions of the original language, $\mathcal{SM} \models \varphi$ holds. By Proposition 1, $\mathcal{SM} \models Skolem(\varphi)$ holds. Finally by Proposition 3 $\mathcal{HSM} \models Skolem(\varphi)$ holds, thus reducing satisfiability to building $\mathcal{HSM}$.

5. Conclusion and Future Work

This paper gave a presentation of a suitably chosen subset of the XML Query Language XQuery called Configuration Logic as a generalization of first-order logic in which the set of variables forms a forest. It also provided a formal semantics and generalizations of Skolem functions and Herbrand universes to this setting.

Support for defining and verifying network properties in Configuration Logic has already been integrated into the network configuration tool ValidMaker [10]. Current work is ongoing to further extend the tool by adding a configuration construction functionality based on the results of this paper as well as to give a precise computational complexity analysis of the logic.

More generally for logic in AI, this work shows that adding more structure to the set of variables of a first order logic can keep the validity of classical constructions while giving a new range of applicability. We considered a forest of variables but more generally, it should be interesting to investigate the range of applicability one could obtain using alternative structures such as graphs.

References

[1] A guide to SNARK. http://www.ai.sri.com/snark/tutorial.

[2] The PROTHEO project home page. http://protheo.loria.fr.

[3] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragment of predicate logic. Technical Report ML-96-03, ILLC Research Report, 1996.

[4] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML query language, W3C working draft, 2005.

[5] L. Cardelli. Describing semistructured data. *SIGMOD Rec.*, 30(4):80–85, 2001.

[6] L. Cardelli and G. Ghelli. Tql: a query language for semistructured data based on the ambient logic. *Mathematical. Structures in Comp. Sci.*, 14(3):285–327, 2004.

[7] W. Charatonik and J.-M. Talbot. The decidability of model checking mobile ambients. *Lecture Notes in Computer Science*, 2142:339–354, 2001.

[8] J. Clark and S. DeRose. XML path language (XPath) version 1.0, W3C recommendation, 1999.

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 2000.

[10] R. Deca, O. Cherkaoui, and D. Puche. A validation solution for network configuration. In *CNSR*, pages 273–275. IEEE Computer Society, 2004.

[11] R. Enns. Netconf configuration protocol, IETF Internet draft, February 2006.

[12] M. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

[13] S. Hallé, R. Deca, O. Cherkaoui, and R. Villemaire. Automated validation of service configuration on network devices. In J. B. Vicente and D. Hutchison, editors, *MMNS*, volume 3271 of *Lecture Notes in Computer Science*, pages 176–188. Springer, 2004.

[14] S. Hallé, R. Deca, O. Cherkaoui, R. Villemaire, and D. Puche. A formal validation model for the netconf protocol. In A. Sahai and F. Wu, editors, *DSOM*, volume 3278 of *Lecture Notes in Computer Science*, pages 147–158. Springer, 2004.

[15] I. M. Hodkinson. Loosely guarded fragment of first-order logic has the finite model property. *Studia Logica*, 70(2):205–240, 2002.

[16] W. McCune. Skolem functions and equality in automated deduction. In *AAAI*, pages 246–251, 1990.

[17] W. McCune. MACE 2.0 reference manual and guide. Technical report, Argonne National Laboratory, May 2001. Technical Memorandum No. 249.

[18] I. Pepelnjak and J. Guichard. *MPLS VPN Architectures*. Cisco Press, 2001.

[19] E. Rosen and Y. Rekhter. BGP/MPLS VPNs. Technical Report RFC 2547, IETF, 1999.

[20] R. Villemaire, S. Hallé, and O. Cherkaoui. Configuration logic: A multi-site modal logic. In *TIME*, pages 131–137. IEEE Computer Society, 2005.

[21] L. Wos and G. W. Pieper. *Automated Reasoning and the Discovery of Missing and Elegant Proofs*. Rinton Press, 2003.

[22] J. Zhang and H. Zhang. System description: Generating models by SEM. In M. A. McRobbie and J. K. Slaney, editors, *CADE*, volume 1104 of *Lecture Notes in Computer Science*, pages 308–312. Springer, 1996.