Satisfying a Fragment of XQuery by Branching-Time Reduction

Sylvain Hallé and Roger Villemaire[*]
Université du Québec à Montréal
C.P. 8888, Succ. Centre-Ville, Montréal, Canada H3C 3P8

Abstract

Configuration Logic (CL) is a fragment of XQuery that allows first-order quantification over node labels. In this paper, we study CL satisfiability and seek a deterministic decision procedure to build models of satisfiable CL formulæ. To this end, we show how to revert CL satisfiability into an equivalent CTL satisfiability problem in order to leverage existing model construction algorithms for CTL formulæ.

1 Introduction

The similarity between tree languages and temporal logic has long been recognized. Recently, fragments of XPath have been interpreted in terms of CTL [3] and Propositional Dynamic Logic (PDL) [2]. Complexity and decidability results on semi-structured data are revisited from a modal logic perspective in [4, 15]. Finally, correspondences between XML Schemata and multi-modal hybrid logic formulæ have also been established [7]. The main interest of such connections is the "well-behaved" nature of modal logics, an observation that has also led to the development of guarded logic [5].

The satisfiability of a tree logic is an important question in a number of practical contexts [6]: database query optimization –where the unnecessary calculation of inconsistent queries can be avoided,– information leakage in security views and consistency of XML specifications.

The majority of works relating tree languages to modal logics are considered *navigational*: they are less flexible in stating relations between *values* of different tree nodes. In contrast, Configuration Logic (CL) includes first-order quantification over node labels in addition to the usual Boolean and parent-child connectives. In this paper, we attempt to solve the problem of CL satisfiability from a temporal logic perspective. We first recall the syntax and semantics of CL in Section 2 and show that its satisfiability is undecidable in general. We then seek ways of "taming" the full-blown language CL. This task is delicate, since adding even a simple hierarchical structure over the variables of a decidable first-order language is sometimes enough to render it undecidable. Nevertheless, it can be shown that satisfiability of CL with unary predicates is complete for non-deterministic exponential time.

However, it is important to note that all previously mentioned applications assume a *deterministic* decision procedure for the language in question. Therefore, from a practical point of view, we should not be content with the NEXPTIME complexity result for CL and look for a deterministic procedure. We show in Section 3 that given a CL formula φ, there exists CTL formulæ $\omega(\varphi)$ and ψ such that φ is satisfiable if and only if $\omega(\varphi) \wedge \psi$ is satisfiable. Our construction provides a 2EXPTIME decision procedure which, under widely-held complexity theoretic assumptions, is optimal for *deterministic* time and restates the problem in a pure CTL setting.

2 Satisfiability of Configuration Logic

CL is a logic over tree structures whose nodes are name-value pairs. It has been introduced as an appropriate language to express constraints on the configuration of network routers [18].

2.1 Syntax and Semantics

A *configuration* is a forest of *nodes*, each formed of two parts: a *name* and a *value*, represented in the form "name = value". We associate to each node of T a unique symbol that we call a *number*. This symbol does not play a role in the logic properly speaking, but will be used in the decision procedure we present in Section 3. We assume that each forest is topped with an additional source node and consider that a configuration is a tree.

The succession of name-value pairs from the source to an arbitrary node is called a *named path*; two nodes are considered identical if they have the same named path. A named

[*]E-mail: halle@info.uqam.ca, villemaire.roger@uqam.ca. The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada on this research.

path can also have one or more variables in place of the "value" part of some nodes. In this case, depending on the values taken by those variables, the named path will designate different nodes. Variables standing for the "name" part of nodes are not authorized.

In addition to the traditional Boolean connectives, CL allows quantification on the "value" part of "name = value" pairs. Existential quantification takes the form $\langle \overline{p}\,;\, n = x\rangle\, \varphi$, where $\overline{p}$ is a (possibly empty) named path, n is a name and x is a variable free in φ. In this quantification, only x is bound; the other variables possibly occurring in $\overline{p}$ are considered free. In line with the interpretation of CL as a tree generalization of first-order logic [19], the presence of named paths in a quantifier amounts to locating, within the hierarchy, the first-order variable to be quantified; this location can depend on previously quantified variables.

The formal semantics of CL is shown below, where T is a tree, ρ is a valuation, R_i is a predicate, $\overline{x}$ is a tuple of variables matching the arity of R_i, $\overline{p}$ is a (possibly empty) named path, and V is the set of node values. Disjunction and universal quantification are derived in the classical way.

$$\begin{aligned}
T, \rho \models R_i(\bar{x}) \quad &\Leftrightarrow \quad R_i(\rho(\bar{x})) \text{ holds} \\
T, \rho \models \varphi \wedge \psi \quad &\Leftrightarrow \quad T, \rho \models \varphi \text{ and } T, \rho \models \psi \\
T, \rho \models \neg\varphi \quad &\Leftrightarrow \quad T, \rho \not\models \varphi \\
T, \rho \models \langle \bar{p}; p = x\rangle\varphi \quad &\Leftrightarrow \quad \exists v \in V \,:\, \rho(\bar{p}), p = v \in T \text{ and} \\
&\qquad\ T, \rho[x/v] \models \varphi
\end{aligned}$$

2.2 Necessary Conditions for Decidability

In the following, we study the complexity of the decision problem for CL: given a formula φ with unary predicates, determining whether there exists a tree that satisfies φ. Trakhtenbrot showed that for any first-order language with n-ary relations ($n \geq 2$) other than equality, satisfiability for finite structures is undecidable [17]. CL being a generalization of first-order logic, this theorem also applies to it. However, while in the case of first-order logic, restricting the signature of the language to unary relations and equality is sufficient for ensuring decidability, this condition no longer holds with CL.

Theorem 1. *There exists a CL formula $\varphi(x, y)$ such that for every structure $\langle S, R\rangle$, where R is a binary relation over S, there exists a configuration T such that $R(s_1, s_2) \Leftrightarrow T \models \varphi(s_1, s_2)$.*

Proof. Let T be the tree constructed with dummy name p as follows: there exists a named path $p = u, p = v$ if and only if $R(u, v)$ holds. Then $R(u, v)$ is equivalent to $\varphi(u, v) = \langle\,;\, p = x\rangle\langle p = x\,;\, p = y\rangle\, x = u \wedge y = v$. $\square$

Using the fact that variables in CL are organized in a tree structure instead of a flat set, CL with unary relations and equality over infinite domains can simulate first-order logic with arbitrary binary relations and is therefore undecidable. To restore decidability, assuming no restriction on the structure of formulæ, CL must be restricted to unary predicates, or equivalently to equality over finite domains. Decidability then comes as no surprise:

Theorem 2. *The satisfiability problem for CL with equality is NEXPTIME-complete given fixed sets A and B of names and values.*

Proof. For the lower bound, it suffices to remark that CL with unary predicates is a generalization of monadic first-order logic, whose satisfiability is NEXPTIME-complete [8, section 6.2.1]. For the upper bound, let φ be a CL formula using only names and values in A and B. By the semantics of CL, the arity of trees is bounded by $C = |A||B|$ (since no two children of the same parent can have the same "name=value" pair), and the depth of the trees is bounded by $|\varphi|$ (since recursion on paths is not allowed). The maximum size of a tree is then bounded by $O(2^{|\varphi|})$; by [10], model checking of a single tree of size $|T|$ is in $O(|\varphi| \cdot |T|^{|\varphi|})$ and the problem is in NEXPTIME. $\square$

Numerous works have considered the satisfiability of related fragments of tree languages; each of them differs from the others in the set of features they support: first-order quantification on node values, "next-sibling" relation, number and type of Boolean connectives ($\vee$, $\neg$, $\wedge$), "child" relation, recursion or transitive closure, equality between node values. We refer the reader to [2, 11–13, 16] for complexity results on each of these fragments.

3 Satisfiability as a Temporal Logic Problem

The decision procedure presented in Theorem 2 is nondeterministic. A straightforward determinization would consist of generating all possible finite trees in a sequence until one that satisfies the desired formula is found. Termination is guaranteed by the fact that the number of candidates to search is finite, due to the semantics of CL with unary predicates. In this section, CL satisfiability is rather translated into a CTL decision problem. Since CTL is decidable in simple deterministic exponential time [9], it provides an alternate, deterministic CL decision procedure. The advantage of using CTL decision procedures is twofold: it leverages algorithms for a well-studied logic, and the decision algorithms (especially [14]) create a model by decomposing the original formula, therefore making better "educated guesses" on the candidate model than a brute-force enumeration.

$$\begin{aligned}
\omega(\langle n = x_i\,,\, \overline{p}\,;\, m = x_i\rangle\varphi) &= \mathbf{EX}\,(\alpha = n \wedge \beta = x_i \wedge \omega(\langle\overline{p}\,;\, m = x_i\rangle\varphi)) \\
\omega([n = x_i\,,\, \overline{p}\,;\, m = x_i]\,\varphi) &= \mathbf{EX}\,(\alpha = n \wedge \beta = x_i \wedge \omega([\overline{p}\,;\, m = x_i]\,\varphi)) \\
\omega(\langle\,;\, n = x_i\rangle\varphi) &= \mathbf{EX}\,((\alpha = n \wedge x_i = \#) \wedge \mathbf{EX}\,(x_i \neq \# \wedge \omega(\varphi))) \\
\omega([\,;\, n = x_i]\,\varphi) &= \mathbf{AX}\,((\alpha = n \wedge x_i = \#) \rightarrow \mathbf{EX}\,(x_i \neq \# \wedge \omega(\varphi)))
\end{aligned}$$

Table 1. Embedding of CL quantifiers into CTL

3.1 Reducing CL to CTL

The first part of the translation consists in converting a configuration T and a CL formula φ with n quantified variables to a Kripke structure $K_{T,\varphi} = (S, I, R, L)$, where S is a set of states, $I \subseteq S$ is a set of initial states, $R \subseteq S^2$ is a transition relation and L is a labelling function. Then φ is translated into a CTL formula $\omega(\varphi)$ such that that $T \models_{\text{CL}} \varphi$ if and only if $K_{T,\varphi} \models_{\text{CTL}} \omega(\varphi)$. The construction we use is based on the reduction of CL model checking to CTL model checking presented in [10]. We briefly summarize this construction.

A state $s \in S$ of $K_{T,\varphi}$ corresponds to a node of the original tree and a valuation of the quantified variables $x_1, \ldots, x_n$ of φ. Let A, B and Γ be respectively the sets of all names, values and numbers appearing in T. From these sets, we create the sets A', B' and Γ' by adding to each a special, unused symbol # that will stand for "undefined". For each state variable x, $L_x : S \rightarrow Dom(x)$ assigns a unique value to every variable in every state. Formally, let $a \in A'$, $b \in B'$, $c \in \Gamma'$ and $(b_1, \ldots, b_n) \in B'^n$. Let s be a state such that $L_\alpha(s) = a$, $L_\beta(s) = b$, $L_\gamma(s) = c$ and for all $1 \leq i \leq n$, $L_{x_i} = b_i$. Then s is a state in S if and only if there exists a node in T with name a, value b and number c. Hence, for each node in T, there are multiple states in $K_{T,\varphi}$ associated to it, namely one for each possible valuation of the $x_1, \ldots, x_k$; we call these the *copies* of T. We extend the term and call a copy the sub-Kripke structure obtained by taking the restriction of R to $S_{(d_1,\ldots,d_n)}$. Variable γ is then called a *representative* of α and β: for every $d \in \Gamma'$, there exist values $a \in N'$ and $b \in V'$ such that for every state $s \in S$, $L_\gamma(s) = d$ implies that $L_\alpha(s) = a$ and $L_\beta(s) = b$.

The transition relation R is composed of two kinds of transitions. *Tree* transitions link the states belonging to the same copy of T. A tuple $(s_1, s_2) \in S^2$ is a *tree transition* if and only if s_1 and s_2 belong to the same copy. For each tree transition $t = (s_1, s_2)$ such that $L_\gamma(s_1) = c_1$ and $L_\gamma(s_2) = c_2$ and for each $1 \leq i \leq n$, $L_{x_1}(s_1) = \#$, $t \in R$ if and only if node numbered c_2 is the child of node numbered c_1 in T. A transition $(s_1, s_2) \in R$ is a *freeze transition* if and only if there exists $1 \leq j \leq n$ such that $L_{x_j}(s_1) \neq L_{x_j}(s_2)$ and for every $1 \leq i \leq n$ $(i \neq j)$, $L_{x_i}(s_1) = L_{x_i}(s_2)$. A freeze transition $(s_1, s_2) \in R$ is *resetting* if and only if $L_\gamma(s_2) = \#$.

Freeze transitions restrict the behaviour of variables $x_1, \ldots, x_n$; such variables are aptly called *freeze* variables. A freeze variable acts as some kind of permanent "memory device": it is originally set to some undefined value, and once it takes a definite value, it keeps ("freezes") it for the remainder of the execution of the system. A variable x_i in $K_{T,\varphi}$ is a *freeze variable* if and only if whenever $L_s(x_i) = \#$, for every $s \in I$, and for every $(s_1, s_2) \in R$, either $L_{s_1}(x_i) = \#$ or $L_{s_1}(x_i) = L_{s_2}(x_i)$. A freeze variable x_i is a *memory* of state variable β if and only if for every transition $(s_1, s_2) \in R$, either $L_{s_1}(x_i) = L_{s_2}(x_i)$ or $L_{s_2}(x_i) = L_{s_1}(\beta)$.

Finally, since each copy in K is meant to be an image of the original tree T, we must ensure that all such images are identical with the concept of *symmetry*: K is symmetrical if any tree transition $(s_1, s_2) \in R$ such that $L_{s_1}(\gamma) = c_1$ and $L_{s_2}(\gamma) = c_2$ either exists in all copies or in no copy of K.

No constraint is imposed yet on the actual paths that the algorithm takes on each traversal: these constraints are enforced by the translation of the CL formula into CTL. Once the Kripke structure $K_{T,\varphi}$ is obtained, the translation function ω of the CL formula φ into CTL becomes simple. It is defined recursively on the structure of the formula; ground equality testing and Boolean connectives are translated in the traditional way; the quantifiers are translated as shown in Table 1.

Theorem 3 (from [10])**.** *Let T be a tree, φ be a CL formula and ω be the embedding defined previously. Let $K_{T,\varphi}$ be the Kripke structure built as described above. Then $T \models_{CL} \varphi$ if and only if $K_{T,\varphi} \models_{CTL} \omega(\varphi)$.*

3.2 Bad Models of $\omega(\varphi)$

The mapping produced by ω produces a CTL formula linear in the size of the original CL formula. By the EXPTIME-completeness of CTL satisfiability and Theorem 2, providing a model of $\omega(\varphi)$ is not sufficient to show there exists a tree T that satisfies φ unless EXPTIME = NEXPTIME. For example, consider the unsatisfiable CL formula $\langle\,;\, \text{a} = x_1\rangle\ [\,;\, \text{a} = x_2]\, x_1 \neq x_2$ whose translation $\omega(\varphi)$ is the following:

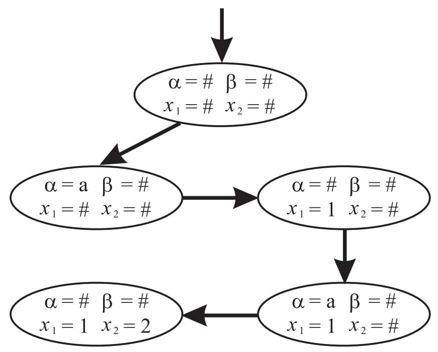

Figure 1. A Kripke structure satisfying formula (1). Values of state variable γ are not shown.

$$\mathbf{EX}\,((\alpha = \mathrm{a} \wedge x_1 = \#) \wedge \mathbf{EX}\,(x_1 \neq \# \wedge \mathbf{AX} \\ ((\alpha = \mathrm{a} \wedge x_2 = \#) \rightarrow \mathbf{EX}\,(x_2 \neq \# \wedge x_1 \neq x_2)))) \tag{1}$$

One can easily check that this CTL formula is satisfiable; Figure 1 shows a model of this formula. However, this model is not a structure of the form $K_{T,\varphi}$ for some tree T. We give below a property ψ such that $\omega(\varphi) \wedge \psi$ is satisfiable in CTL if and only if φ is satisfiable in CL, thus giving a sufficient condition for a Kripke structure to be $K_{T,\varphi}$ for some tree T. Moreover, we shall see that ψ only depends on the number of variables in φ.

Theorem 4. *Let φ be a CL formula with $n \geq 1$ variables, and $K = (S, I, R, L)$ be a Kripke structure with state variables α, β, γ and $x_1, \ldots, x_n$ that respects the following conditions: 1. In the initial state, all state variables take value $\#$; 2. The graph $\langle S, R\rangle$ is a tree; 3. γ is a representative of α and β; 4. The x_i are freeze variables that memorize β; 5. All freeze transitions are resetting; 6. K is symmetrical. Then K is a* tree encoding*: there exists a configuration tree T such that K is $K_{T,\varphi}$.*

Proof. Due to lack of space, we only sketch the proof. It suffices to remark that a tree T can easily be extracted from the copy of K where all $x_i = \#$. From that tree, we build $K_{T,\varphi}$ as in Section 3.1 and observe that $K = K_{T,\varphi}$. □

3.3 Tree Encodings

In order to infer CL's satisfiability from CTL's, we must arrange for the CTL decision procedure to return only tree encodings. The following lemmas show that each of the conditions in Theorem 4 can actually be expressed as CTL formulæ, so that the property of being a tree encoding is by itself a CTL formula ψ. Therefore, searching for a model of $\omega(\varphi)$ that is a tree encoding simply amounts to finding an arbitrary model of $\omega(\varphi) \wedge \psi$.

All the following lemmas apply to a Kripke structure K defined as in the previous section. The first three take care of representatives and freeze variables.

Lemma 1. *State variable γ is a representative if and only if the following holds for K:*

$$\bigwedge_{d\in\Gamma'} \bigvee_{a\in A', b\in B'} \mathbf{AG}\,(\gamma = d \rightarrow (\alpha = a \wedge \beta = b)) \tag{2}$$

Lemma 2. *State variable x_i is a freeze variable if and only if the following holds for K:*

$$x_i = \# \wedge \bigwedge_{k\in B} \mathbf{AG}\,(x_i = \# \vee (x_i = k \wedge \mathbf{AX}\,x_i = k)) \tag{3}$$

Lemma 3. *State variable x_i is a memory of state variable β if and only if the following holds for K:*

$$\bigwedge_{k\in B} \mathbf{AG}\,((x_i = \# \wedge \beta = k) \rightarrow \mathbf{AX}\,(x_i \neq \# \rightarrow x_i = k)) \tag{4}$$

As for the existence of a representative, symmetry can be imposed by a CTL formula. Special care must be taken, since "terminal" copies (where all freeze variables are initialized) do not have successors.

Lemma 4. *K is symmetrical if and only if it respects the following, with state variable γ as a representative:*

$$\bigwedge_{d_1,d_2\in\Gamma'} ((\bigvee_{i=1}^{n} x_i \neq \#) \rightarrow ((\mathbf{AG}\,(\gamma = d_1 \rightarrow \mathbf{EX}\,\gamma = d_2)) \\ \vee (\mathbf{AG}\,(\gamma = d_1 \rightarrow \neg(\mathbf{EX}\,\gamma = d_2))))) \tag{5}$$

Lemma 5. *All freeze transitions are resetting if and only if the following holds for K with state variable γ as a representative:*

$$\bigwedge_{i=1}^{n} \mathbf{AG}\,((x_i = \# \rightarrow \mathbf{AX}\,(x_i \neq \# \rightarrow \gamma = \#)) \tag{6}$$

Since the Kripke structure constructed by the CTL decision procedure can be arbitrary, it is not excluded that such structure contains cycles. We must therefore impose two additional conditions restricting the structure to trees.

Lemma 6. *K is acyclic if and only if the following holds:*

$$\bigwedge_{d\in\Gamma'} \bigwedge_{k_1\in B} \cdots \bigwedge_{k_n\in B} \\ \mathbf{AG}\,((\gamma = d \wedge x_1 = k_1 \wedge \cdots \wedge x_n = k_n) \rightarrow \\ \neg\mathbf{EF}\,(\gamma = d \wedge x_1 = k_1 \wedge \cdots \wedge x_n = k_n)) \tag{7}$$

Lemma 7. *If K is acyclic, then K is a tree if and only if the following holds:*

$$\bigwedge_{d_1,d_2\in\Gamma'} \mathbf{EF}\,(\gamma = d_1 \wedge \mathbf{EX}\,\gamma = d_2) \rightarrow \\ (\mathbf{AG}\,(\gamma \neq d_1 \rightarrow \neg\mathbf{EX}\,\gamma = d_2)) \quad (8)$$

The combination of Lemmas 1–7 and Theorems 3 and 4 yields the desired result.

Theorem 5. *Let φ be a CL formula with k quantified variables. Let $\varphi' = \omega(\varphi)$ be its CTL translation. Let ψ be the conjunction of (2)–(8). φ is satisfiable in CL if and only if $\varphi' \wedge \psi$ is satisfiable in CTL.*

By this theorem, a model for a CL formula can be constructed directly using decision procedures for CTL. It should be noted that in the expression of ψ, the conjunction of (2)–(8) is exponential in n. But since $n \leq |\varphi|$, then $|\omega(\varphi) \wedge \psi| \in O(2^{|\varphi|})$. In turn, satisfiability for CTL is EXPTIME-complete; hence the resulting decision procedure for CL is in deterministic double exponential time. Unless EXPTIME = NEXPTIME, this result is optimal for a *deterministic* CL model construction procedure.

4 Conclusion

In this paper, we have shown how to reduce the satisfiability of CL with unary predicates to the satisfiability of CTL. This leads to a 2EXPTIME upper bound on the *deterministic* decision procedure. Moreover, the reduction of CL satisfiability to CTL satisfiability, although it requires an exponential translation of the original formula, still yields a decision procedure which, under widely-held complexity theoretic assumptions, is optimal for deterministic time.

References

[1] *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*. IEEE Computer Society, 2006.

[2] L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Marx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classical Logics*, 15(2):115–135, 2005.

[3] L. Afanasiev, M. Franceschet, M. Marx, and M. de Rijke. CTL model checking for processing simple XPath queries. In *TIME*, pages 117–124. IEEE Computer Society, 2004.

[4] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *J. Log. Comput.*, 13(6):939–956, 2003.

[5] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, (27):217–274, 1998.

[6] M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of DTDs. In C. Li, editor, *PODS*, pages 25–36. ACM, 2005.

[7] N. Bidoit, S. Cerrito, and V. Thion. A first step towards modeling semistructured data in hybrid multimodal logic. *Journal of Applied Non-Classical Logics*, (4):447–476, 2004.

[8] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

[9] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982.

[10] S. Hallé, R. Villemaire, and O. Cherkaoui. CTL model checking for labelled tree queries. In *TIME*, pages 27–35. IEEE Computer Society, 2006.

[11] J. Hidders. Satisfiability of XPath expressions. In G. Lausen and D. Suciu, editors, *DBPL*, volume 2921 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2003.

[12] O. Kupferman and M. Y. Vardi. Memoryful branching-time logic. In *LICS* [1], pages 265–274.

[13] L. V. S. Lakshmanan, G. Ramesh, H. Wang, and Z. J. Zhao. On testing satisfiability of tree pattern queries. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 120–131. Morgan Kaufmann, 2004.

[14] W. Marrero. Using BDDs to decide CTL. In N. Halbwachs and L. D. Zuck, editors, *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2005.

[15] M. Marx. XPath and modal logics of finite DAG's. In M. C. Mayer and F. Pirri, editors, *TABLEAUX*, volume 2796 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2003.

[16] M. Marx. XPath with conditional axis relations. In E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, and E. Ferrari, editors, *EDBT*, volume 2992 of *Lecture Notes in Computer Science*, pages 477–494. Springer, 2004.

[17] B. Trakhtenbrot. Impossibility of an algorithm for the decision problem in finite classes. *Dok. Akad. Nauk SSSR*, 70:569–572, 1950.

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

[19] R. Villemaire, S. Hallé, R. Deca, and O. Cherkaoui. Skolem functions and Herbrand universes for a tree generalization of first-order logic. In A. Gelbukh and C. A. Reyes-García, editors, *Fifth Mexican International Conference on Artificial Intelligence, Special Session, November 13-17, Apizaco, Mexico*, pages 22–31. IEEE Computer Society, November 2006.