A DECISION PROCEDURE FOR THE WS*K*S LOGIC

Tomáš Fiedor
Master Degree Programme 2, FIT BUT
E-mail: xfiedo01@stud.fit.vutbr.cz

Supervised by: Ondřej Lengál
E-mail: ilengal@fit.vutbr.cz

Abstract: Various types of logics are often used as a means for a formal specification of systems. The weak monadic second-order logic with k successors (WSkS) is one of these logics with quite high expressivity, yet still decidable. Although the complexity of checking satisfiability of a WSkS formula is not even in the ELEMENTARY class, there are some approaches to this problem that perform well in practice. The currently existing implementations of a WSkS decision procedures are based on the use of deterministic tree automata. The aim of this work is to exploit the recently developed techniques for efficient manipulation of non-deterministic tree automata and implement an efficient WSkS decision procedure based on those.

Keywords: formal verification, tree automata, WSkS, decision procedures

1 INTRODUCTION

In formal verification, logics are often used for a specification of the verified systems in a very natural and intuitive way. The *weak monadic second-order logic of k successors* (WSkS) [4] is a fragment of the second order monadic logic that allows to quantify over finite set variables where every element from the universe of discourse has k successors. This way various k-ary tree structures, e.g. heaps or binary trees, and linear structures, e.g. linked lists, can be expressed.

There has been established a one-to-one correspondence between WSkS formulae and tree automata. Based on this there has already been several attempts to implement a decision procedure for WSkS using deterministic tree automata [2]. However, with the recent development in algorithms for manipulation of non-deterministic tree automata, the aim of this work is to exploit these ideas in an implementation of a WSkS decision procedure based on non-deterministic tree automata.

2 WS*K*S SYNTAX

A WSkS *term* is either the empty constant ε, a first-order variable symbol written in lower-case letters (e.g. x, y, ...) or an unary symbol from $\{1,\ldots,k\}$ written in postfix notation. For example $x1123$ or $\varepsilon 2111$ are terms. An *atomic formula*, for some second-order variables X and Y, is either the subset predicate $X \subseteq Y$, the singleton predicate $Sing(X)$, the successor predicate $X = Yi$, for some $1 \leq i \leq k$, or predicate $X = \varepsilon$ denoting that X is a singleton set $\{\varepsilon\}$. A WSkS *formula* is then built out of these atomic formulae using only the logical connectives $\wedge, \vee, \neg$ and the existential quantifier $\exists X$ for quantification over second-order variables.

Example 2.1 *The following example WSkS formula* ψ *denotes that there does not exist a singleton set which is not a subset of the set* X.

$$\psi \stackrel{def}{=} \neg\exists P : Sing(P) \wedge P \not\subseteq X \tag{1}$$

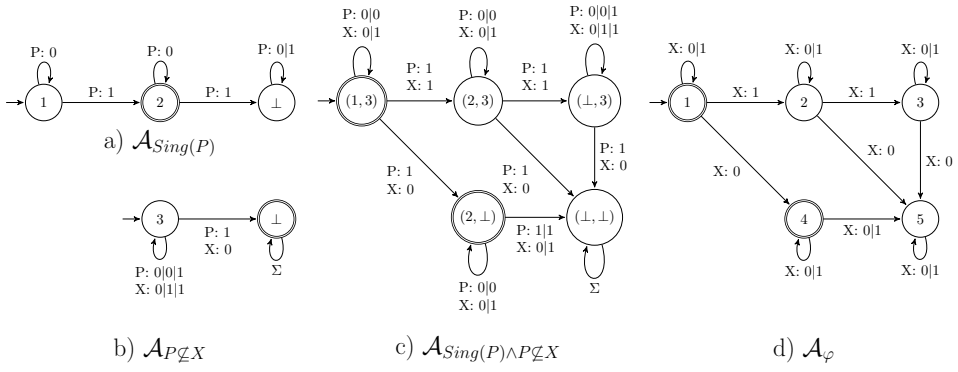

Figure 1: Finite automata corresponding to the subformulae of the formula $\varphi \stackrel{def}{=} \exists P : Sing(P) \wedge P \not\subseteq X$

3 DECIDING WS*K*S

Many decision procedures for a wide range of logics are based on the use of some kind of finite-state automata. WS*k*S is no exception and its currently most often used decision procedure is based on constructing a *k*-ary tree automaton and examining its language.

3.1 DECIDING WS*k*S USING DETERMINISTIC AUTOMATA

One of the tools for deciding WS*k*S, MONA [2], constructs a deterministic tree automaton $\mathcal{A}_\varphi$ for the given formula φ recursively on the structure of the formula. As a base, each atomic subformula is transformed to a corresponding automaton. Further it constructs for connectives $\phi \vee \psi$, $\phi \wedge \psi$, $\neg\phi$ and $\exists X.\phi$, union of $\mathcal{A}_\phi$ and $\mathcal{A}_\psi$, intersection of $\mathcal{A}_\phi$ and $\mathcal{A}_\psi$, complement of $\mathcal{A}_\phi$ and projection on the track of X of $\mathcal{A}_\phi$ respectively, such an automaton then represents all models of formula φ.

3.2 DECIDING WS*k*S USING NON-DETERMINISTIC AUTOMATA

Although this approach yields good results in many practical examples, every time non-determinism is introduced the automaton is determinised and the information about the original states is forgotten. Therefore such an approach has issues with extensive use of automaton complementation and since currently there is no known tree automaton complementation technique better than bottom-up determinization of automaton with complementation of the set of final states, heavy optimizations and heuristics had to be used in MONA to achieve good results.

We propose that it is not necessary to construct the automaton representing all models of φ. Instead construction of the automaton and the search for an accepting or a non-accepting state can be done *on-the-fly* and we can further exploit recent development in algorithms for manipulating non-deterministic tree automata. Given a WS*k*S formula φ we transform it to the formula in the *existentially-quantified prenex normal form* $\psi \stackrel{def}{=} \exists \mathcal{X}_{m+1} \neg\exists \mathcal{X}_m \ldots \neg\exists \mathcal{X}_2 \neg\exists \mathcal{X}_1 . \pi(\mathbb{X})$, where $\mathcal{X}_i$ denotes a set of second order variables. We then create a hierarchical family of formulae $\Phi = \{\varphi_0, \ldots, \varphi_m\}$ where $\varphi_0 \stackrel{def}{=} \pi$ and for all $0 \leq i \leq m-1$ it holds that $\varphi_{i+1} \stackrel{def}{=} \neg\exists \mathcal{X}_{i+1} . \varphi_i$.

Further, using the operations of complementation γ and projection $\omega_\mathcal{X}$ over the set of variables $\mathcal{X}$, we define the family of automata $\mathbb{A} = \{\mathcal{A}_0, \ldots, \mathcal{A}_m\}$ as follows:

$$\mathcal{A}_0 = \mathcal{A}_\pi \tag{2}$$

$$\mathcal{A}_{i+1} = \gamma(\omega_{\mathcal{X}_{i+1}}(\mathcal{A}_i)) \tag{3}$$

such that there is a correspondence between $\mathcal{A}_i$ and φ_i for all $0 \leq i \leq m$.

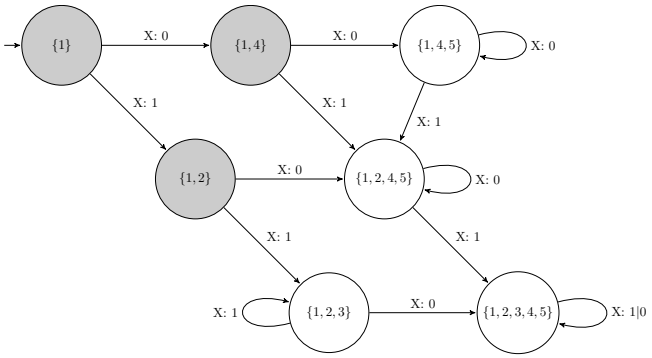

Figure 2: Comparison of the antichain-based (grey nodes) and the classical approach to universality checking of the automaton $\mathcal{A}_\psi$ corresponding to the formula $\psi \stackrel{def}{=} \neg\exists P : Sing(P) \wedge P \not\subseteq X$

The Language of the automaton $\mathcal{A}_\psi$ is further tested for universality. The classical naive approach performs determinization using subset construction to obtain a deterministic automaton, which is further complemented by swapping final and non-final states and finally checked whether it contains reachable final state. In our approach we wish to exploit the ideas from the Antichains algorithm [1] and check universality of $\mathcal{A}_\psi$ without explicitly constructing it, but rather using a search of the state space over powers of the state set of $\mathcal{A}_0$. To illustrate this on an example, consider the example formula $\psi \stackrel{def}{=} \neg\exists P : Sing(P) \wedge P \not\subseteq X$ and the automaton $\mathcal{A}_\varphi$ from Figure 1 corresponding to the formula $\varphi \stackrel{def}{=} \exists P : Sing(P) \wedge P \not\subseteq X$, i.e. $\psi \stackrel{def}{=} \neg\varphi$. We can search for a non-accepting state of $\mathcal{A}_\psi$ by searching for a reachable set of states of $\mathcal{A}_\varphi$ that does not contain a final state. Further, by using the antichains principle, if we encounter a set of states P and later a set of states R, s.t. $P \subseteq R$, we do not need to explore R as P is smaller and therefore has a bigger chance of reaching a non-accepting set of states.

4 CONCLUSION

We proposed a new decision procedure of WSkS logic that uses non-deterministic automata instead of deterministic ones used, e.g. in tool MONA [2]. This different approach makes use of recent developments in the field of non-deterministic automata algorithms such as universality checking or language inclusion checking, allowing us to search for a rejecting or accepting states on-the-fly without constructing the automaton corresponding to the given formula at all, possibly yielding a faster decision procedure for some class of formulae.

REFERENCES

[1] Abdulla, Parosh Aziz et al.: When simulation meets antichains (on checking language inclusion on nondeterministic finite (tree) automata). In *Tools and Algorithms for Construction and Analysis of Systems*, LNCS 6015, pages 158–174, Springer Verlag, 2010.

[2] MONA: Web pages of MONA. [online] Available on: http://www.brics.dk/mona/.

[3] Comon, H. et al.: Tree automata techniques and applications. 2007, release October 12th, 2007.

[4] Büchi, J. R.: Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1–6):66–92, 1960.