Mona: decidable arithmetic in practice

Morten Biehl[1], Nils Klarlund[2], and Theis Rauhe[3]

[1] BRICS, University of Aarhus (mbiehl@brics.dk)
[2] AT&T Research (klarlund@research.att.com)
[3] BRICS, University of Aarhus (theis@brics.dk)

Abstract. In this note, we describe how a fragment of arithmetic can be decided in practice. We follow essentially the ideas of [8], which we have embedded in the MONA tool.
MONA is an implementation of *Monadic Second-order Logic* on finite strings (and trees). The previous semantics used in MONA is the one provided in current literature [7, 9], where the meaning of first-order terms is restricted to being a position in the string over which the formula is interpreted.
In this note, we describe our new semantics, where terms are interpreted relative to *all* natural numbers. With this semantics MONA becomes a decision procedure for the calculus called WS1S, the Weak Second-order theory of 1 Successor.
We also show how the MONA representation of automata subsumes a recent proposal for representing queues. We exploit the natural semantics to carry out automated reasoning about queue operations.

In practice, the fundamental concept of *regularity* (finite-state acceptance of strings) is often exceedingly hard to express. For example, regular expressions (as used in text editors and UNIX shell programming) are elegant when expressing simple patterns, but often unreadable for patterns of even modest complexity.

The aim of the FIDO/MONA project pursued at the University of Aarhus is to devise new practical means of describing finite-state systems in formal logics that naturally capture informal requirements.

MONA is a tool that translates formulas to finite-state automata. The formulas may express search patterns, temporal properties of reactive systems, or parse tree constraints. MONA is based on Monadic Second-order Logic on finite strings. (FIDO is a high-level language, incorporating logic and many usual programming language concepts like recursive data types. A FIDO program is translated into MONA, which in turn is translated into an automaton.)

In this paper, we discuss a new semantics for MONA that we have recently implemented. We also show that a recently proposed data structure for representing queues [2] is but a special case of the BDD-represented automata of MONA.

Some Mona applications The previous MONA implementation for finite strings is described in [3]. We have applied MONA to hardware verification [1], verification of complicated behavioral descriptions of distributed systems [5], and

design constraints in software engineering [4]. (In the latter application, we used a version of MONA for finite trees.)

In addition, we have built prototypes of tools for graphical representation of search patterns (specified in an extension of regular expressions) and for the automatic verification of Hoare logic of pointers. (For publications, see our WEB page `http://www.brics.aau.dk/~mbiehl/Mona/main.html`.) The largest formula that has been decided by MONA contains half a million characters. This formula is a transcription of a FIDO description of behaviors in distributed memory systems [6, 5].

Semantics The *Weak Second-order theory of 1 Successor*, WS1S, is a variation of predicate logic interpreted over the natural numbers. A first-order term is either a variable p or of the form $t + 1$, where t is a first-order term and $+1$ is a function symbol interpreted as the successor function. A first-order term denotes a natural number. A second-order term is a set expression, made out of second-order variables and the usual set-theoretic operations (except complementation, but including difference). A second-order term denotes a finite subset of the natural numbers. Formulas relate first-order and second-order terms through equality and membership. Formulas may be formed also through use of the standard Boolean connectives and quantification over both first-order and second-order variables.

For example, if p is a first-order variable and Qe is a second-order variable, then the formula

$$p \in Qe \wedge (\forall^1 q' : q' \in Qe \Rightarrow q' \leq p)$$

expresses that p is in Qe and that all numbers in Qe are less or equal to p. This MONA formula may be used to define a predicate *isLast* by the definition:

$$\begin{aligned} &\mathit{isLast}(\textbf{var1}\ p, \textbf{var2}\ Q2) = \\ &\quad p \in Qe \wedge (\forall^1 q' : q' \in Qe \Rightarrow q' \leq p) \end{aligned}$$

Decision procedure To explain the decision procedure, we review our earlier semantics, which is simpler to implement. Previously, we followed [7, 9] in defining the meaning of a formula relative to a natural number n, called the *length*. Under this view, which we call the *bounded semantics*, all first-order terms take on values in $\{0, \ldots, n-1\}$. Similarly, second-order terms may only denote subsets of $\{0, \ldots, n-1\}$. Consequently, all quantifiers are bounded by n.

This semantics makes for a strange arithmetic, but it is useful for many purposes anyway. To see this, let us consider the formula $\phi \equiv \neg(P = Q)$. For $n = 3$, a possible interpretation $\mathcal{I}$ for this formula consists of assigning, say, $\{0, 2\}$ to P and $\{0, 1\}$ to Q. Alternatively, $\mathcal{I}$ can be regarded as a string w of length $n = 3$ over the alphabet $\mathbf{B} \times \mathbf{B}$, where $\mathbf{B} = \{0, 1\}$:

	0	1	2
P	1	0	1
Q	1	1	0

Here, the three letters in the string are numbered according to their position from 0 to 2. The bit pattern 101 defined in the row marked P is called the *track* for P. The set it defines is the set of positions where the track contains a 1; in this case $\{0, 2\}$. Similarly, Q is defined by this string to be $\{0, 1\}$. Thus the interpretation $\mathcal{I}(w)$ denoted by w is identical to $\mathcal{I}$ above. Since this interpretation satisfies ϕ, we write

$$\mathcal{I}(w) \models \phi.$$

In general, assume that ϕ has k free variables $P_1, \ldots, P_k$. (It can be shown that it is sufficient to consider second-order variables.) For each n, we may consider the strings of length n that satisfy a formula ϕ with free variables among $P_1, \ldots, P_k$. The union of all such strings for all lengths n constitutes a language $L(\phi)$ over $\mathbf{B}^k$. It can be shown by induction that for any formula the language of satisfying interpretations is accepted by a finite-state deterministic automaton. For example, if we have calculated automata A_ϕ such that $L(A_\phi) = L(\phi)$ and A_ψ such that $L(A_\psi) = L(\psi)$, then we can construct an automaton A recognizing $L(\phi \wedge \psi)$ by forming the cross product of A_ϕ and A_ψ. Similarly, it can be shown that existential quantification of a variable P in the formula $\exists P : \phi$ corresponds to the removal of the track P by a projection operation from the automaton corresponding to ϕ. The resulting automaton is non-deterministic and must be determinized by a subset construction.

The natural semantics Under the natural semantics, we can also use finite strings to denote interpretations. But note that $\mathcal{I}(w)$, regarded as a function from strings over $\mathbf{B}^k$ to assignments of finite sets to $P_1, \ldots, P_k$ is not injective. The connectives of the logic are treated in the same way as before. There is one complication, however: existential quantification. We proceed as follows.

Let the right-quotient of a language L with L', denoted by L/L', be defined as $\{w \mid \exists u \in L' : w \cdot u \in L\}$. Let the projection operation E^i be defined by $E^i(L) = \{w \mid \exists w' \in L : w \text{ is obtained from } w' \text{ by deleting the } P_i \text{ track}\}$. It can then be proven that if ϕ has k variables, then $L(\exists P_i : \phi) = E^i(L(\phi)/L^i)$, where $L^i = \{u \in \mathbf{B}^k \mid j \text{ track is all 0 for } j \neq i\}$. Fortunately, the right-quotient operation $L \mapsto L/L^i$ can be expressed as an efficient automata-theoretic algorithm that runs in linear time. The subsequent subset construction is potentially exponential. But preliminary evidence shows that just as for the bounded semantics this operation is usually benign—the new automaton being at most a couple of times bigger before minimization.

An extended version of this paper will discuss the new algorithms for BDD-represented automata.

A queue example In [2], BDD-like data structures, called QBDDs, are proposed to improve the representation of certain queues. For example, the set of all queues consisting of ordered subsets of $\{1, \ldots, k\}$ is shown to require approximately k^2 "nodes" with usual BDDs but only k "nodes" with the QBDD representation (the "nodes" referred to here are really subgraphs of size k).

In our demonstration, we show how a straightforward logical description in MONA of the set of queues for $n = 4$ yields the QBDD representation depicted in Figure 6 of [2].

We also show how reasoning about such queues can be formulated with MONA using the natural semantics. Specifically, we show that the set of queues for $n = 4$ is equivalent to the set of queues that arise from the lossy queue consisting of $0, 1, 2, 3$.

Other applications *Presburger arithmetic* is the first-order theory of natural numbers, where the functional symbol is + (a dyadic function) and the only relational symbol is =. It is not hard to see that with the natural semantics, statements of Presburger arithmetic can now be translated into MONA.

We also believe that our new semantics will make reasoning about timing behavior and hardware easier to formulate.

Our tool

The present version of MONA with natural semantics runs only on SUN Solaris workstations. However, we expect to release a more universal version, written fully in C and C++, later this summer.

References

1. D. Basin and N. Klarlund. Hardware verification using monadic second-order logic. In *Computer aided verification : 7th International Conference, CAV '95, LNCS 939*, 1995.
2. P. Godefroid and D.E. Long. Symbolic protocol verification with Queue BDDs. In *Proc. LICS' 96*, 1996.
3. J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic in practice. In *Tools and Algorithms for the Construction and Analysis of Systems, First International Workshop, TACAS '95, LNCS 1019*, 1996. Also available through http://www.brics.aau.dk/klarlund.
4. N. Klarlund, J. Koistinen, and M. Schwartzbach. Formal design constraints. In *Proc. OOPSLA '96*, 1996. to appear.
5. N. Klarlund, M. Nielsen, and K. Sunesen. Automated logical verification based on trace abstraction. Technical Report RS-95-53, BRICS, 1995. To appear in Proceedings of PODC '96.
6. N. Klarlund, M. Nielsen, and K. Sunesen. A case study in automated verification based on trace abstractions. Technical Report RS-96-?, BRICS, Aarhus University, 1996. In preparation. To appear in LNCS proceedings on Dagstuhl workshop.
7. Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, 1994.
8. J.W. Thatcher and J.B. Wright. Generalized finite automata with an application to a decision problem of second-order logic. *Math. Systems Theory*, 2:57–82, 1968.
9. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. MIT Press/Elsevier, 1990.