TaPAS: The Talence Presburger Arithmetic Suite

Jérôme Leroux and Gérald Point

LaBRI, Université Bordeaux 1, CNRS
{leroux,point}@labri.fr

Abstract. TAPAS is a suite of libraries dedicated to FO $(\mathbb{R}, \mathbb{Z}, +, \leq)$. The suite provides (1) the application programming interface GENEPI for this logic with encapsulations of many classical solvers, (2) the BDD-like library SATAF used for encoding Presburger formulae to automata, and (3) the very first implementation of an algorithm decoding automata to Presburger formulae.

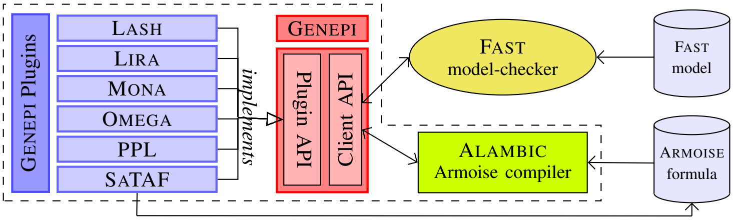
1 The Mixed Additive Theory

The automatic verification of reactive systems is a major field of research. These systems are usually modeled with variables taking values in some infinite domains. Popular approaches for analyzing these models are based on decision procedures adapted to the variables domains. For numerical variables the mixed additive theory FO $(\mathbb{R}, \mathbb{Z}, +, \leq)$ provides a natural logic to express linear constraints between integral variables and real variables. This logic has positive aspects: it is decidable and actually many solvers implement decision procedures for the full logic or some sub-logics like the Presburger arithmetic. TAPAS is a suite of libraries dedicated to these logics.

The sequel is organized as follows. In Section 2 the architecture of TAPAS is presented. Section 3 presents the BDD-like library SATAF used for encoding Presburger formulae to automata and for decoding automata to Presburger formulae. Finally, Section 4 provides some benchmarks.

2 TAPAS at a Glance

The following figure shows the architecture of TAPAS (enclosed by the dotted line). The FAST model-checker is also depicted but it does not actually belong to the suite; it is just a client application.

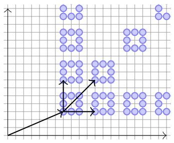


Selecting the most efficient solver for a given application can be difficult. This choice can change dramatically the practical performance of the application. Since solvers have

S. Kowalewski and A. Philippou (Eds.): TACAS 2009, LNCS 5505, pp. 182–185, 2009.

incompatible application programming interfaces (API), a particular one must be first selected prior the implementation of the application; change afterward require additional development effort. The Genepi library[BLP06] addresses this issue by offering small APIs between, on one side, applications requiring solvers and, on the other side, solvers implementing decision procedures. The connection between applications and solvers is realized transparently using dynamically loaded modules (*plugins*) specified at the execution time. This way the choice of a solver can be postponed after the implementation of an application based on Genepi; the best one can be selected by performing benchmarks. Since [BLP06], Genepi has been enhanced with new features: support for $\mathbb{R}$, new plugins (Lira and PPL), ...

In general, specifying sets with low level logics is difficult and fastidious. The same problem appears with FO $(\mathbb{R}, \mathbb{Z}, +, \leq)$. The Armoise language allows to describe concisely sets of vectors (in $\mathbb{Z}$ and/or $\mathbb{R}$). The succinctness of formulae is achieved using, among other tricks, hierarchical definitions and arithmetic over sets (which hides numerous and ugly quantifications). Below is an example of an Armoise formula which specifies a repeated pattern in $\mathbb{N}^2$. Note that Armoise formulae may define sets which are not necessarily definable in FO $(\mathbb{R}, \mathbb{Z}, +, \leq)$.



```
let
   origin := (7,3);
   directions := { (4,0), (0,4), (4,4) };
   pattern := ([0...2], [0...2]) \ (1,1);
in
   origin + nat * directions + pattern;
```

TaPAS provides tools to support the Armoise language. The Alambic library permits to compile Armoise formulae into calls to Genepi functions. An important step in this compilation process is the translation of high-level constructions into the mixed additive theory; due to the expressive power of Armoise this transformation is not always possible.

3 SATAF: Shared-Automata and the Synthesis of Formulae

Many solvers like Lash, Lira and Mona are based on automata packages. Intuitively, mappings from words to numerical vectors are used to associate to automata potentially infinite sets of numerical vectors (one vector per accepted word). Usually, the minimal deterministic automata are proved canonically associated to the represented sets and not on the way they have been computed. In particular these solvers are well adapted to sets obtained after long chains of operations like in symbolic model checking. In [Cou04], Couvreur introduced a new data-structure called *shared-automata*. Intuitively, the sharing of a finite set of automata is performed by merging every states equivalent with respect to the Nérode relation. TaPAS contains the SATAF library that implements this data structure. In SATAF, deciding if two automata recognize the same language reduces to check the equality of their pointers. Similarly to BDD packages, SATAF uses a pointer-based hash-table cache algorithm. Up to our knowledge, no other automata package implements these features. Since the first version of SATAF written in Java by Couvreur [Cou04], a new version written in C is now maintained in TaPAS. TaPAS also provides an implementation of the Genepi API based on SATAF.

Extracting geometrical properties (for instance linear constraints) from automata is a challenging problem. From a theoretical point of view, this problem has been solved in [Ler05]. This article provides a polynomial time algorithm for computing Presburger formulae from automata representing Presburger sets. The algorithm first extracts the "necessary" linear constraints from the automaton. Then, it computes an unquantified Presburger formula using only these constraints, Boolean operations, translations by integer vectors, and scaling by integer values. An implementation of this algorithm is provided with SaTAF. The generated formulae follows the Armoise syntax. Note that SaTAF, Genepi and Alambic provide together the very first implementation of an algorithm that normalizes Presburger formulae into unique canonical forms that only depend on the denoted sets. Intuitively in the normalization process, the minimization procedure for automata acts like a simplification procedure for the Presburger arithmetic.

4 Experimenting the Automata to Formulae Algorithm

Fast [BLP06] is a tool for verifying reachability properties of infinite-state systems. The tool is implemented over TaPAS. Benchmarks of Fast on various Genepi implementations are available in [BLP06, BDEK07]. We experimented the computation of Armoise specifications from SaTAF automata denoting the reachability sets of 40 systems. Some results are presented in the following table. Column "$\rightarrow \mathcal{A}$" is the time in seconds spent for computing an automaton $\mathcal{A}$ representing the reachability set, "$|\mathcal{A}|$" is the number of states of $\mathcal{A}$, "$\mathcal{A} \rightarrow S$" is the time in seconds to produce an Armoise specification S from $\mathcal{A}$, $|S|$ is the number of characters of S, "n" is the number of variables of S, and "l" is the number of linear constraints generated in S.

system	$\rightarrow \mathcal{A}$	$\|\mathcal{A}\|$	$\mathcal{A} \rightarrow S$	$\|S\|$	n	l
`Central Server system`	9.57	75	0.07	3716	12	10
`Consistency Protocol`	194.96	90	0.06	3811	11	11
`CSM - N`	24.38	66	0.05	3330	14	11
`Dekker ME`	17.73	200	0.01	13811	22	0
`Multipoll`	11.38	612	2.54	60995	18	19
`SWIMMING POOL`	71.03	553	0.14	1803	9	18
`Time-Triggered Protocol`	116.89	17971	90.53	984047	11	44

We also experimented the normalization of Armoise formulae on various examples. Due to space limitation, only four representative examples are presented: `modulo`, `large-coef`, `large-var` and `unsimplified`. Example `modulo` denotes the Cartesian product $(11\mathbb{N}) \times (7\mathbb{N}) \times (5\mathbb{N}) \times (3\mathbb{N})$ where $m\mathbb{N}$ is the set of non-negative integers multiple of m, `large-coef` is the conjunction of three linear constraints with coefficients larger than 20, `large-var` is a linear constraint defined over 36 variables, and `unsimplified` is a disjunction of two sets of linear constraints with redundant constraints. Benchmark results are summarized in the following table. Columns have the following meaning: "example" provides the name of the Armoise specification S_0, "$|S_0|$" is the number of characters of S_0, "$S_0 \rightarrow \mathcal{A}$" is the time in seconds to produce a SaTAF automaton $\mathcal{A}$ from S_0. The other columns have been defined previously.

example	$\|S_0\|$	$S_0 \rightarrow \mathcal{A}$	$\|\mathcal{A}\|$	$\mathcal{A} \rightarrow S$	$\|S\|$	n	l
`modulo`	40	0.1	4620	35.8	335	4	0
`large-coef`	167	1.7	147378	25.1	957	4	3
`large-var`	543	0.2	4320	12.7	2220	36	1
`unsimplified`	529	1.1	16530	1.3	1026	5	2

We observe that the computation of SATAF automata $\mathcal{A}$ from ARMOISE specifications S_0 takes less than 2 seconds. Moreover, the computation of ARMOISE specifications S from $\mathcal{A}$ takes less than half a minute even on automata with more than 10^5 states. In practice the implementation (quadratically) slows down in presence of modular constraints. Note that $|S_0| < |S|$ in all our examples due to non-optimized outputs. However, even if $|S_0| < |S|$ in the last example `unsimplified`, the redundant linear constraints of S_0 no longer appear in S.

Contrary to previous results, we observe that $|S|$ is quite smaller than $|\mathcal{A}|$. Recall that the generated ARMOISE specifications are combinations of linear constraints. In both series of benchmarks formulae contain a few number of constraints (see the "l" column). The complexity of the combinations explains the differences in the results. In practice, we observe that small automata can encode complex combinations but only a small number of linear constraints.

5 Conclusion and Future Work

TAPAS is distributed at `http://altarica.labri.fr/wiki/tools:tapas:` under GPLv2. Thanks to GENEPI, this is the first tool to our knowledge which offers (1) a simple framework for the development of applications requiring solvers for the mixed additive theory and (2) an easy way to benchmark solvers. TAPAS also provides the SATAF solver based on shared-automata and the first implementation of the algorithm translating them into equivalent ARMOISE formulae.

Future Work. Predicate abstraction methods are limited by the number of considered predicates. Usually some of these predicates are redundant. More precisely, some predicates are not semantically used even if they syntactically appear. In future work, we are interested in using the normalization procedure of TAPAS in order to remove redundant predicates. In fact, the set of extracted linear constraints from an automaton is known minimal : these constraints syntactically appears in any unquantified Presburger formula that denotes the set represented by the automaton.

References

[BDEK07] Becker, B., Dax, C., Eisinger, J., Klaedtke, F.: LIRA: Handling constraints of linear arithmetics over the integers and the reals. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 307–310. Springer, Heidelberg (2007)

[BLP06] Bardin, S., Leroux, J., Point, G.: FAST extended release. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 63–66. Springer, Heidelberg (2006)

[Cou04] Couvreur, J.-M.: A BDD-like implementation of an automata package. In: Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S. (eds.) CIAA 2004. LNCS, vol. 3317, pp. 310–311. Springer, Heidelberg (2005)

[Ler05] Leroux, J.: A polynomial time presburger criterion and synthesis for number decision diagrams. In: LICS, pp. 147–156. IEEE Comp. Soc., Los Alamitos (2005)