VARIABLE ELIMINATION FROM LINEAR MODULAR
CONSTRAINTS

Ajith K.J.
Reactor Control Division

Abstract

Application of formal verification and analysis tools is recommended in the development of hardware and software systems used in safety-critical applications. Formal verification tools for such systems generate constraints (formulae) that are solved by solving algorithms. Variable elimination (existential quantifier elimination) from these constraints facilitates the simplification of the constraints. This is crucial in improving the scalability of the verification tools to target systems of increased complexity. This paper presents efficient techniques for variable elimination from a class of such constraints called linear modular constraints, and an application of these techniques in the formal verification of FPGA based hardware designs.

Introduction

Ensuring the correctness of hardware and software systems used in safety-critical applications is extremely important, as bugs in such systems can have disastrous consequences. Hence, application of formal verification and analysis tools that use rigorous mathematical reasoning for functional verification is recommended in the development of systems used in safety-critical applications. However, in spite of the technological advancements made over years, there is an increasing gap between the sizes of the systems being designed and sizes of systems amenable to automatic formal verification. Moreover, many of the commercial formal verification tools have prohibitive prices and have restrictions on transfer of technology. This motivates development of efficient and indigenous formal verification techniques that can target systems of increased complexity.

Formal verification tools for hardware and software systems generate constraints (formulae) that are solved by solving algorithms. Variable elimination (existential quantifier elimination) involves elimination of variables from these constraints in order to compute another set of simplified constraints that depend only on variables relevant for the formal specification being verified. This facilitates significant simplification of the constraints, and thereby improves the scalability of the verification tools.

Linear modular constraints are a class of constraints extremely important in the context of formal verification and analysis tasks. Let \( p \) be a positive integer constant, \( x_1, \ldots, x_n \) be \( p \)-bit non-negative integer variables, and \( a_0, \ldots, a_n \) be integer constants in \( \{0, \ldots, 2^p-1\} \). A linear term over \( x_1, \ldots, x_n \) is a term of the form \( a_1 x_1 + \cdots + a_n x_n + a_0 \). A linear modular equality (LME) is a formula of the form \( t_1 = t_2 \mod(2^p) \), where \( t_1 \) and \( t_2 \) are linear terms over \( x_1, \ldots, x_n \). Similarly, a linear modular disequality (LMD) is a formula of the form \( t_1 \neq t_2 \mod(2^p) \), and a linear modular inequality (LMI) is a formula of the form \( t_1 \leq t_2 \mod(2^p) \). We use linear modular constraint (LMC) to denote an LME, LMD or

Shri Ajith K.J. is the recipient of the
DAE Young Engineer Award for the year 2012

Abstract

Application of formal verification and analysis tools is recommended in the development of hardware and software systems used in safety-critical applications. Formal verification tools for such systems generate constraints (formulae) that are solved by solving algorithms. Variable elimination (existential quantifier elimination) from these constraints facilitates the simplification of the constraints. This is crucial in improving the scalability of the verification tools to target systems of increased complexity. This paper presents efficient techniques for variable elimination from a class of such constraints called linear modular constraints, and an application of these techniques in the formal verification of FPGA based hardware designs.

Introduction

Ensuring the correctness of hardware and software systems used in safety-critical applications is extremely important, as bugs in such systems can have disastrous consequences. Hence, application of formal verification and analysis tools that use rigorous mathematical reasoning for functional verification is recommended in the development of systems used in safety-critical applications. However, in spite of the technological advancements made over years, there is an increasing gap between the sizes of the systems being designed and sizes of systems amenable to automatic formal verification. Moreover, many of the commercial formal verification tools have prohibitive prices and have restrictions on transfer of technology. This motivates development of efficient and indigenous formal verification techniques that can target systems of increased complexity.

Formal verification tools for hardware and software systems generate constraints (formulae) that are solved by solving algorithms. Variable elimination (existential quantifier elimination) involves elimination of variables from these constraints in order to compute another set of simplified constraints that depend only on variables relevant for the formal specification being verified. This facilitates significant simplification of the constraints, and thereby improves the scalability of the verification tools.

Linear modular constraints are a class of constraints extremely important in the context of formal verification and analysis tasks. Let \( p \) be a positive integer constant, \( x_1, \ldots, x_n \) be \( p \)-bit non-negative integer variables, and \( a_0, \ldots, a_n \) be integer constants in \( \{0, \ldots, 2^p-1\} \). A linear term over \( x_1, \ldots, x_n \) is a term of the form \( a_1 x_1 + \cdots + a_n x_n + a_0 \). A linear modular equality (LME) is a formula of the form \( t_1 = t_2 \mod(2^p) \), where \( t_1 \) and \( t_2 \) are linear terms over \( x_1, \ldots, x_n \). Similarly, a linear modular disequality (LMD) is a formula of the form \( t_1 \neq t_2 \mod(2^p) \), and a linear modular inequality (LMI) is a formula of the form \( t_1 \leq t_2 \mod(2^p) \). We use linear modular constraint (LMC) to denote an LME, LMD or
LMI. Conventionally $2^p$ is called the modulus of the LMC.

The most dominant technique used in practice for eliminating variables from LMCs is conversion to bit-level constraints\(^1\), followed by bit-level variable elimination. Since LMCs can be expressed as formulae in integer arithmetic, variable elimination techniques for integer arithmetic such as OmegaTest\(^1\) can also be used to eliminate variables from LMCs. However these techniques scale poorly as the modulus of the LMCs increases. In this paper, we present efficient techniques for variable elimination from LMCs that significantly outperform variable elimination techniques based on bit-level reasoning and integer arithmetic.

**Algorithms for Variable Elimination from Linear Modular Constraints**

In this section, we present an efficient algorithm for variable elimination from conjunctions of LMCs called \textit{Project}\(^3\). Later on, we present an algorithm \textit{QE\textunderscore LMDD} – an extension of \textit{Project} for variable elimination from Boolean combinations of LMCs.

\textit{Project} uses a layered approach to eliminate variables from a conjunction of LMCs. Sound but incomplete, cheaper layers are invoked first, and expensive but complete layers are called only when required. The first layer of \textit{Project} (Layer1) involves simplification of the given conjunction of LMCs using the LMEs present in the conjunction. For example, consider the problem of eliminating the variable $x$ from the conjunction of LMCs $(6x+y = 4)$ and $(2x+z \neq 0)$ with modulus 8. Note that $(6x+y = 4)$ can be equivalently expressed $(2x = 5y+4)$ in modulo 8 using modular arithmetic operations\(^1\). The variable $x$ can be eliminated from the conjunction by replacing the occurrences of $2x$ in the conjunction by $5y+4$. Simplifications as above using LMEs present in the conjunction forms the crux of Layer1.

The second layer of \textit{Project} (Layer2) makes use of an efficient combinatorial heuristic to identify unconstraining LMI s and LMDs that can be dropped from the conjunction of LMCs without changing the set of satisfying solutions of the conjunction. For example, consider the problem of eliminating the variable $x$ from the conjunction of LMCs $(2x = 5y+4)$ and $(x+y \leq 3)$ with modulus 8. Note that $x$, $y$ are 3-bit variables here. It can be observed that $(2x = 5y+4)$ is independent of the most significant bit of $x$, denoted as $x[2]$. It can be observed that every solution of $(2x = 5y+4)$ can be engineered to become a solution of the conjunction of $(2x = 5y+4)$ and $(x+y \leq 3)$ by choosing $x[2]$ appropriately. This means that $(x+y \leq 3)$ is irrelevant (unconstraining) as far as the elimination of the variable $x$ from the conjunction of $(2x = 5y+4)$ and $(x+y \leq 3)$ is concerned. Layer2 drops such unconstraining LMDs and LMI s from the conjunction of LMCs, and thereby simplifies the problem instance.

The cases that are not computed by the application of the above computationally cheap layers are handled by expensive but more complete techniques. The third layer of Project (Layer3) involves a variant of the classical Fourier-Motzkin\(^2\) variable elimination algorithm for reals adapted to work for LMCs. The final layer (Layer4) involves model enumeration that eliminates the variable by enumeration of all possible values in the domain of the variable.

The algorithm \textit{QE\textunderscore LMDD} extends \textit{Project} for variable elimination from Boolean combinations of LMCs using a data-structure called Linear Modular Decision Diagram (LMDD)\(^2\). LMDDs are decision diagrams with nodes labelled with LMEs or LMI s. Given a Boolean combination of LMCs represented as an LMDD, \textit{QE\textunderscore LMDD} traverses the LMDD in a top-down manner converting the problem of variable elimination from Boolean combination of LMCs into a set of simpler sub-problems each of which involves variable elimination from a conjunction of LMCs.
Applications

The primary motivation for the development of our techniques comes from a formal verification tool for FPGA based hardware designs, Vhdl Bounded Model Checker (VBMC), developed in Reactor Control Division, BARC. VBMC accepts a hardware design in VHDL, functional property in PSL, and verification bound (number of cycles of operation) as inputs. It either proves that the design satisfies the functional property for the given verification bound or generates a counterexample providing the reason of violation. Suppose we wish to use VBMC for proving that a given VHDL design satisfies a formal specification for N cycles of operation. VBMC performs this by deriving the transition relation of the VHDL design (relation between the present and next states of the design), unrolling the transition relation N times, conjoining the unrolled relation with the negation of the property, and then checking for the satisfiability of the resulting constraint using a constraint solver. However, unrolling the transition relation a large number of times can give a constraint with a large number of variables, which can crucially affect the performance of the constraint solver.

VBMC alleviates this problem by computing a simplified (abstract) transition relation that relates only a chosen subset of variables in the VHDL design relevant to the property being checked. As the transition relations of real-life VHDL designs largely belongs to the theory of LMCs, VBMC makes use of QE_LMDD for abstract transition relation computation. We have experimentally found that using QE_LMDD for abstract transition relation generation significantly outperforms the use of bit-level or integer arithmetic based techniques.

Conclusions

The application of formal verification and analysis tools in the development of hardware and software systems is extremely important, particularly when they are used in safety-critical applications. Our experiments demonstrate that efficient variable elimination techniques are crucial in enhancing the scalability of such formal verification and analysis tools to target systems of real-life complexity.

References