Dynamic Verification of a Large Discrete System[1]

Johan Gunnarsson, Roger Germundsson
Division of Automatic Control
Department of Electrical Engineering
Linköping University, S-581 83 Linköping, Sweden
{johan,roger}@isy.liu.se
http://control.isy.liu.se

Accepted for the 35th Conference on Decision and Control

Abstract

Symbolic algebraic analysis techniques are applied to the landing gear subsystem in the new Swedish fighter aircraft, JAS 39 Gripen. Our methods are based on polynomials over finite fields (with Boolean algebra and propositional logic as special cases). Polynomials are used to represent the basic dynamic equations for the processes (controller and plant) as well as static properties of these. Temporal algebra (or temporal logic) is used to represent specifications of system behavior. These specifications are verified both on a model of the landing gear controller, and a model of the closed loop behavior of the landing gear controller connected to a plant. The model of the landing gear controller is made from the actual implementation in Pascal. The tools used are developed by the authors in *Mathematica* and uses an efficient implementation of binary decision diagrams (BDDs).

1 Introduction

We have modeled and analyzed an existing discrete subsystem of a modern fighter aircraft, the landing gear system on the JAS 39 Gripen. This system was designed and implemented without any formal methods or tools.We have built a mathematical model of this system and analyzed its behavior w.r.t. to its specification. The main focus has not been on the specific system, but rather on the general methods that can be applied to discrete dynamic systems of industrial size, e.g. the process is fairly complex, with some hundred variables, of which 66 are Boolean. This paper describes the second part of the project, where the focus is on analysis. The objective of the first part of this project was to build a mathematical model of the behavior of the landing gear controller (LGC). This was done by the development of a compiler that translates Pascal code to a model of polynomial relations. Further information of this work can be found in [4, 7, 9].

1.1 The Polynomial Framework

Quantities and relations in DES are of a finite nature and can therefore be represented by finite relations. These relations can in turn be represented mathematically by polynomials over finite fields $\mathbb{F}_q[Z]$, i.e. polynomials of variables in the set Z with coefficients from a finite field $\mathbb{F}_q$. By further restricting the class of polynomials we construct a quotient polynomial ring (see [3] or the tutorial [5]) that gives a one to one correspondence between polynomials and relations as well as a compact representation of the relations. (Similar results can be found in [8].) The computational framework used for manipulating polynomials is based on *binary decision diagrams* (BDD) [1], which give a powerful representation as well as fast computations which allow us to manipulate rather complex systems.

1.2 Modeling of the LGC

The purpose of the LGC is to perform maneuvers of the landing gears and the corresponding doors which enclose the gears in retracted position. The LGC is a software process that interacts with 5 binary actuators, 30 binary landing gear sensors, 2 binary pilot signals, and 5 integer mode signals from other subsystems in the aircraft. The state of the LGC is represented by 26 Boolean variables. The only formal description of the controller available to use was the actual implemented 1200 line Pascal code. See [4] for further details.

In the modeling part of the project the implemented Pascal code of the LGC was compiled to a polynomial model. The Pascal code is first parsed to a intermediate code called MPascal which essentially is the same Pascal code written as a *Mathematica* expression. This code is then processed by a compiler, also written in *Mathematica*. The result from the compiler is a poly-

[1] This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK), which is gratefully acknowledged.

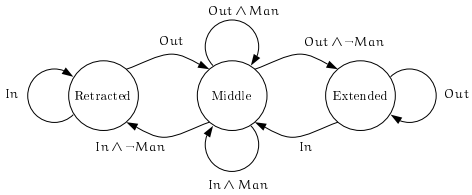

Figure 1: Landing gear model.

nomial model, denoted $C(z, z^+)$, represented as a BDD, where all static and dynamic relations between input variables and output variables are stored whereas temporary variables in the code are removed. See [4, 7] for details.

2 Closed Loop Verification

Having a polynomial model for the LGC, $C(z, z^+)$, we need a model for the plant, i.e., the physical landing gear. The sensors of the landing gear system determine if the gears are retracted, extended or in between. Therefore we use the three state automata in figure 1 as an illustration for the plant model $P(z, z^+)$. This model has two input signals In and Out that are controlled by the LGC when connected into closed loop. The input signal Man is an auxiliary signal for verification purpose. The plant model stays in the middle state until Man becomes `false`. The outputs in this model correspond to the sensors in the physical plant, but are not shown in figure 1.

By connecting the LGC model, $C(z, z^+)$ and the plant model $P(z, z^+)$ we get the closed loop model

$$G(z, z^+) := C(z, z^+) \wedge P(z, z^+).$$

To verify the behavior of the closed loop model we use *temporal logic* (CTL) [2] to formally represent the specification of the behavior. See table 1 for a subset of temporal operators. If we want to verify the following specification: *"The gear should always reach the extended state* $\mathtt{Gear(ext)}$ *in finite time, when pilot command is extension* $\mathtt{Pilot(ext)}$*."* we can search if there exists behavior not fulfilling the statement above by using the temporal expression

$$F(z) := \mathrm{EG}[\neg(\mathtt{Pilot(ext)} \rightarrow \mathtt{Gear(ext)}) \wedge \neg \mathtt{Man}].$$

By adding $\neg$Man to the temporal expression we specify that the plant model will reach the extended state in arbitrary but finite time if the LGC command Out is `true` long enough. This shows that temporal logic can be powerful for modeling complex behavior in a compact way.

Temporal Algebra	Natural Language
$Q(z)$	$Q(z)$ holds in the initial state.
$\mathrm{EX}[Q(z)]$	$Q(z)$ can hold in the next time step.
$\mathrm{EU}[Q_1(z), Q_2(z)]$	$Q_1(z)$ will hold for finitely many steps and then $Q_2(z)$ can hold.
$\mathrm{EF}[Q(z)]$	$Q(z)$ can hold at some future time.
$\mathrm{EG}[Q(z)]$	$Q(z)$ can hold at all future times, i.e. from this point onwards.

Table 1: Temporal algebra constructs.

The verification is performed by tools developed by the authors in *Mathematica* as

$$S(z) := \mathtt{BDDTLEvaluate}[G(z, z^+), F(z)].$$

The result is $S(z) \neq$ `false` which means that there exists behaviors where the specification above is not true. From $S(z)$ we can analyze why this is the case and build a more detailed specification, i.e., we get a more complete $F(z)$. We have also used more complex plant models where sensor errors are added to the behavior. The result of the verification proves that the behavior of the controller code is correct even for sensor failures of the plant. See [6] for details.

The project has showed that it is possible to do dynamic analysis of complex systems (>100 boolean variables for $G(z, z^+)$) by using formal symbolic techniques.

References

[1] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a BDD package. In *27th ACM/IEEE Design Automation Conference*, pages 40–45, 1990.

[2] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic. *ACM Transactions on Programming Languages and Systems*, 8(2):244–63, April 1986.

[3] Roger Germundsson. *Symbolic Systems - Theory, Computation and Applications.* PhD thesis, Linköping University, September 1995.

[4] Johan Gunnarsson. On modeling of discrete event dynamic systems, using symbolic algebraic methods. Technical Report LiU-TEK-LIC-1995:34, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, June 1995.

[5] Johan Gunnarsson. Algebraic methods for discrete event systems - a tutorial. In *Workshop on Discrete Event Systems.* IEE, August 1996.

[6] Johan Gunnarsson. Symbolic algebraic discrete systems - applied to the JAS 39 fighter aircraft, part ii. Technical Report LiTH-ISY-R-1873, Department of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, August 1996. Available through ftp at ftp://ftp.control.isy.liu.se-/pub/Reports/1996/1873.ps.Z.

[7] Johan Gunnarsson, Jonas Plantin, and Roger Germundsson. Verification of a large discrete system using algebraic methods. In *Workshop on Discrete Event Systems*. IEE, August 1996.

[8] M. Le Borgne, A. Benveniste, and P. Le Guernic. Polynomial ideal theoretic methods in discrete events and hybrid dynamical systems. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 2695–2700, 1989.

[9] Jonas Plantin, Johan Gunnarsson, and Roger Germundsson. Symbolic algebraic discrete systems theory - applied to a fighter aircraft. In *34th IEEE Conference on Decision and Control*, pages 1863–1864, 1995.