CREDIBLE CODE GENERATION CHAIN FOR CONTROL SYSTEMS (AND OPTIMIZATION ALGORITHMS)

TIMOTHY WANG, ROMAIN JOBREDEAUX, ERIC FERON
MESSAGE:

• AFTER 120 YEARS OF LYAPUNOV-BASED METHODS.. WE ARE FINALLY GOING TO EXTEND IT DOWN TO THE SOURCE CODE LEVEL
CONTENTS

• PROBLEM MOTIVATION
• INTRODUCTION TO SOFTWARE CERTIFICATION
• CREDIBLE AUTOCODING
• VERIFICATION OF THE CREDIBLE CODE
• CONCLUSION
MOTIVATION

1. HIGH SOFTWARE CERTIFICATION COST! APPROACHING 50% OF TOTAL PROJECT DEVELOPMENT COSTS ~ BILLIONS!

2. MAY NOT BE PRACTICAL FOR CERTIFYING UNMANNED AUTONOMOUS VEHICLES OR UNMANNED AERIAL SYSTEMS
CONTENTS

• PROBLEM MOTIVATION
• INTRODUCTION TO SOFTWARE CERTIFICATION
• CREDIBLE AUTOCODING
• VERIFICATION OF THE CREDIBLE CODE
• CONCLUSION
CURRENT CERTIFICATION PROCESS

- V CYCLE
- SIMULATION-BASED
- TEST UNTIL YOU DROP
CREDIBLE AUTOCODING PROCESS
CONTENTS

• PROBLEM MOTIVATION
• INTRODUCTION TO SOFTWARE CERTIFICATION
• CREDIBLE AUTOCODING
• VERIFICATION OF THE CREDIBLE CODE
• CONCLUSION
CREDIBLE AUTOCODING PROTOTYPE

- Translates the property of stability into axiomatic semantics for the generated code
  - Open-loop
  - Closed-loop
  - Performance measures (work in progress)

- Axiomatic semantics are expressed using a code level specification language
  - Code comments i.e. does not change the generated code
  - Proof checking the comments \(\Rightarrow\) the code satisfies the specification of stability with respect to a plant model
• CONTROL ANALYSIS YIELDS AN ELLIPSOID INVARIANT SET
• VERSATILE I.E. ENCODE BOTH STABILITY AND PERFORMANCE MEASURES
• EFFICIENT COMPUTATIONALLY: INTERIOR POINT METHOD TO SOLVE THE APPROPRIATE LINEAR MATRIX INEQUALITY
TYPE OF SYSTEMS THAT COULD BE COVERED

Example: the Lur’e system

- LINEAR SYSTEMS
- QUASI-NONLINEAR SYSTEMS (LINEAR SYSTEMS WITH BOUNDED UNCERTAINTIES AND NONLINEARITIES IN THE LOOP)
- NONLINEARITIES: SATURATIONS, HYSTERESIS, DEAD ZONE, ETC
- UNCERTAINTIES: MODELLED UNCERTAINTIES OR PARAMETER VARYING MODELS
CREDIBLE AUTOCODING PROCESS

- TRANSFORMATION OF PROOFS AND PROPERTIES FROM HIGH LEVEL OF ABSTRACTION DOWN TO THE CODE LEVEL.
- LANGUAGES CHOSEN FOR THE PROTOTYPE: A SUBSET OF SIMULINK AND C
INITIAL LEVEL OF ABSTRACTION

1. LANGUAGE OF MATH
2. FORMALISM: STATE-SPACE SYSTEMS
3. CONTROL SYSTEM ANALYSIS SHOULD BE PERFORMED AT THIS LEVEL
NEXT LEVEL OF ABSTRACTION: MODEL LANGUAGE (SUBSET OF SIMULINK)

1. ALLOWED SIMULINK BLOCKS: DELAYS, GAINS, SUMS, INPUT, OUTPUT, ...

2. BASICS BLOCKS: WELL-DEFINED SEMANTICS..

3. COMPOSITE BLOCKS: VAGUE SEMANTICS

4. INSERT ELLIPSOID INVARIANTS SETS FROM ANALYSIS: ANNOTATED MODEL
AND ALL THE WAY DOWN TO THE CODE LEVEL

1. GENERATE ELLIPSOID SETS FOR EACH LINE OF CODE IF POSSIBLE.
   1. THE AFFINE STRUCTURE OF THE GENERATED CODE
   2. CONVEX COMBINATION OF ELLIPSOIDS WHENEVER NEEDED.. USING MULTIPLIERS FROM S PROCEDURE (TRANSLATED FROM THE ANNOTATED MODEL)

2. VERIFICATION CONDITIONS FROM LINE OF CODE: TO BE VERIFIED BY THE BACKEND

ELLIPSOID(Q2,X) → ELLIPSOID(Q,X)

// ELLIPSOID(Q,X)
WHILE (TRUE) {
// ELLIPSOID(Q,X)
.
// ELLIPSOID(Q2, X)
// ELLIPSOID(Q,X)
}
// ELLIPSOID(Q,X)
IF THIS VERIFICATION CONDITION DOES NOT HOLD, THEN WHAT?

• ERROR IN THE CONSTRUCTION MODEL

• USE ELLIPSOID(Q2,X) TO TRACE THE SOURCE THE ERROR, OR AT LEAST NARROW THE AREA OF MODEL THAT CAUSE THIS ERROR

• EXAMPLE: Q2 IS THE FOLLOWING MATRIX

\[
\begin{bmatrix}
76.748 & -1.991 & 5.3123 & -1166.4 \\
-1.991 & 3.8374 & -3.8232 & 21.056 \\
5.3123 & -3.8232 & 34.112 & -195.66 \\
-1166.4 & 21.056 & -195.66 & 38043
\end{bmatrix}
\]
ERROR ANALYSIS ON MODEL: LOCALIZES THE ERROR IN A SUBSET OF THE MODEL
ACSL

1. A specification language used to express invariants for C programs.
2. Contract-like structure \(\rightarrow\) Hoare triples

PVS

Proof assistants provide:

1. A language to express mathematical properties
2. An interactive mode where the user can provide a manual proof of every claim
3. Proof strategies
VERIFYING THE GENERATED CODE

• EACH LINE OF CODE IS ANNOTATED WITH A CONTRACT
• CONTRACT DESCRIBES HOW THE CODE TRANSFORMS THE INVARIANT SET
• FOR EACH CONTRACT, GENERATE A FIRST ORDER LOGIC PROPERTY TO BE PROVEN
• PROVE IT USING A PROOF ASSISTANT (WITH AUTOMATION)
CONTENTS

• PROBLEM MOTIVATION
• INTRODUCTION TO SOFTWARE CERTIFICATION
• CREDIBLE AUTOCODING
• VERIFICATION OF THE CREDIBLE CODE
• CONCLUSION
CREDIBLE AUTOCODING OF CONVEX OPTIMIZATION SYSTEMS (MPC FOR EXAMPLE)

IN Variant: constructed from a decreasing potential function

Other invariant properties include:

1. Positive-definiteness of the optimization variables
2. Small deviation from the central path
CONCLUSION

• A PRACTICAL CREDIBLE AUTOCODING FRAMEWORK

• A PRELIMINARY PROTOTYPE IS CAPABLE OF TRANSLATING THE SEMANTICS OF OPEN-LOOP AND CLOSED-LOOP STABILITY

• CREDIBLE CODE PRODUCED HAVE BEEN AUTOMATICALLY VERIFIED BY THE PROOF-CHECKING BACKEND