Verifying Digital Systems with MATLAB (Tool Demo)

I. Motivation

Digital Controller

Finite word-length (FWL) effects

Microprocessor

ensuring the correctness of cyber-physical systems (CPS) remains a daunting challenge

Xi Zheng et al., 2014.

"Simulation alone is not sufficient to support verification and validation of CPS."

Sayan Mitra et al., 2013.

Step A

DSVerifier builds an ANSI-C code representation of the digital system based on the specification.

Step B

DSVerifier formulates a FWL model based on fixed-point arithmetic:

\[ \text{FWL [•]} : \mathbb{R} \rightarrow \mathbb{Q}[\mathbb{R}] \]

Step C

DSVerifier checks a property \( \Phi \) up to a bound \( k \):

\[ \Phi \text{ Bits } < I, F > \text{ Result} \]

Quantization error

32-bits <15,16> >1%

16-bits <7,8> >1%

8-bits <3,4> <1%

Stability

8-bits <3,4> Unstable

II. Approach and Uniqueness

Bounded model checking

\( \psi \) is satisfiable iff \( \phi \) has counterexample of max. depth \( k \)

Step 1

User →

Step A: Digital controller design

Step 2: Define numerical representation

Step 3: Configure verification

Input file (ANSI-C)

DSVerifier

Step A: Parser

Step B: Compute a FWL controller model

Step C: Verify using a BMC tool

Counterexample

Verification Successful

\[ \begin{align*}
    x(k+1) &= \begin{bmatrix} 0.9669 & 0.05649 & 0 \end{bmatrix} + \begin{bmatrix} -0.0001 & 0.9570 & 0.5658 \end{bmatrix} x(k) + \begin{bmatrix} 0.0024 & 0.0002 \end{bmatrix} u(k) \\
    y(k) &= \begin{bmatrix} 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \end{bmatrix} u(k)
\end{align*} \]

Step 2

Numerical representation \( < I, F > \):

- \( I \) is the integer part and
- \( F \) is the fractional part

Step 3

Setup verification:

- choose a property \( \Phi \);
- a maximum verification time;
- a bound \( k \);
- a BMC tool

Properties:

- stability;
- quantization error;
- controllability;
- observability;

III. DSVerifier Toolbox

Step 1

Digital controller design

Step 2

Define numerical representation

Step 3

Configure verification

Input file (ANSI-C)

DSVerifier

Step A: Parser

Step B: Compute a FWL controller model

Step C: Verify using a BMC tool

Counterexample

Verification Successful

\[ \begin{align*}
    x(k+1) &= \begin{bmatrix} 0.9669 & 0.05649 & 0 \end{bmatrix} + \begin{bmatrix} -0.0001 & 0.9570 & 0.5658 \end{bmatrix} x(k) + \begin{bmatrix} 0.0024 & 0.0002 \end{bmatrix} u(k) \\
    y(k) &= \begin{bmatrix} 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \end{bmatrix} u(k)
\end{align*} \]

Step 2

Numerical representation \( < I, F > \):

- \( I \) is the integer part and
- \( F \) is the fractional part

Step 3

Setup verification:

- choose a property \( \Phi \);
- a maximum verification time;
- a bound \( k \);
- a BMC tool

Properties:

- stability;
- quantization error;
- controllability;
- observability;

IV. Contributions

1. support for transfer-function and state-space representations in open- and closed-loop form;
2. verify different numerical representations and realization forms of digital systems;
3. provide a MATLAB toolbox to check specific properties of digital systems while taking into account FWL;

As future work:

- verify uncertainties in digital systems represented by state-space;
- integrate counterexample reproducibility for digital systems

For further information, publications, and downloads, see: http://www.dsverifier.org/