On the Verification of Synthesized Kalman Filters

Ruben Gamboa, John Cowles, Jeff Van Baalen
University of Wyoming

ACL2 Workshop 2003

Supported by NASA grant NAG 2-1570

The General Challenge

- Consider the automatic generation of software
  - ⋆ customized for a particular use
  - ⋆ optimized, taking advantage of domain knowledge
  - ⋆ based on theorem proving technology
- How can we verify the resulting software is correct?

Verifying the Process

- certify the software generator
  - ⋆ . . . may much more complex than the software it generates
- problems: customizations, optimizations, complexity of the generator, etc. make this a daunting challenge
- the same problem applies to theorem provers

Verifying the Product

- certify the software that is generated, regardless of the generation process
- problems: software may be hard to read or understand
- solution: annotate generated software with a correctness argument
- software can be inspected manually (or mechanically)

The Specific Challenge

- Verify the correctness of automatically generated Kalman Filters
- Use “hints” in the generated code to guide the proof
- Process should be 100% automatic

Our Approach

- Separate the correctness of the program
  - ⋆ correctness of Kalman Filters
  - ⋆ correctness of the implementation
- Use as much manual intervention as necessary in the first part
- The second part must be automatic

The Kalman Filter

The roots of the Kalman Filter are in estimation theory. How can we predict the next value of the time-series x_1, x_2, . . . , x_n? This is especially important when the x_i can not be measured directly.

The Kalman Filter Conditions

$$z_k = H_k x_k + v_k$$

The Kalman Filter Conditions

$$
\begin{aligned}
z_k &= H_k x_k + v_k \\
x_{k+1} &= \Phi_k x_k + w_k
\end{aligned}
$$

The Kalman Filter Conditions

$$
\begin{aligned}
z_k &= H_k x_k + v_k \\
x_{k+1} &= \Phi_k x_k + w_k
\end{aligned}
$$

$$
\begin{array}{rl}
E[v_k] = 0 & \qquad E[w_k] = 0 \\
E[v_k {v_i}^{\mathrm{T}}] = \delta_{k-i} R_k & \qquad E[w_k {w_i}^{\mathrm{T}}] = \delta_{k-i} Q_k
\end{array}
$$

The Kalman Filter Conditions

$$
\begin{aligned}
z_k &= H_k x_k + v_k \\
x_{k+1} &= \Phi_k x_k + w_k \\
E[v_k] = 0 & \qquad E[w_k] = 0 \\
E[v_k {v_i}^{\mathrm{T}}] = \delta_{k-i} R_k & \qquad E[w_k {w_i}^{\mathrm{T}}] = \delta_{k-i} Q_k \\
E[v_k {w_i}^{\mathrm{T}}] &= 0
\end{aligned}
$$

The Kalman Filter Conditions

$$
\begin{aligned}
z_k &= H_k x_k + v_k \\
x_{k+1} &= \Phi_k x_k + w_k \\
E[v_k] = 0 & \qquad E[w_k] = 0 \\
E[v_k {v_i}^{\mathrm{T}}] = \delta_{k-i} R_k & \qquad E[w_k {w_i}^{\mathrm{T}}] = \delta_{k-i} Q_k \\
E[v_k {w_i}^{\mathrm{T}}] &= 0 \\
E[x_0 {v_k}^{\mathrm{T}}] = 0 & \qquad E[x_0 {w_k}^{\mathrm{T}}] = 0
\end{aligned}
$$

The Kalman Filter

The estimate $\hat{x}_k$ that minimizes $E[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^{\mathrm{T}}]$ is

$$\begin{aligned} \hat{x}_k &= \overline{x}_k + K_k(z_k - H_k\overline{x}_k) \\ \overline{x}_k &= \Phi_{k-1}\hat{x}_{k-1} \end{aligned}$$

The Kalman Filter

The estimate $\hat{x}_k$ that minimizes $E[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^{\mathrm{T}}]$ is

$$
\begin{aligned}
\hat{x}_k &= \overline{x}_k + K_k(z_k - H_k \overline{x}_k) \\
\overline{x}_k &= \Phi_{k-1} \hat{x}_{k-1} \\
K_k &= \overline{P}_k {H_k}^{\mathrm{T}} \left(H_k \overline{P}_k {H_k}^{\mathrm{T}} + R_k \right)^{-1}
\end{aligned}
$$

The Kalman Filter

The estimate $\hat{x}_k$ that minimizes $E[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^{\mathrm{T}}]$ is

$$\begin{aligned}
\hat{x}_k &= \overline{x}_k + K_k(z_k - H_k\overline{x}_k) \\
\overline{x}_k &= \Phi_{k-1}\hat{x}_{k-1} \\
K_k &= \overline{P}_k H_k{}^{\mathrm{T}}\left(H_k\overline{P}_k H_k{}^{\mathrm{T}} + R_k\right)^{-1} \\
\overline{P}_k &= \Phi_{k-1}P_{k-1}\Phi_{k-1}{}^{\mathrm{T}} + Q_{k-1}
\end{aligned}$$

The Kalman Filter

The estimate $\hat{x}_k$ that minimizes $E[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^{\mathrm{T}}]$ is

$$
\begin{aligned}
\hat{x}_k &= \overline{x}_k + K_k(z_k - H_k\overline{x}_k) \\
\overline{x}_k &= \Phi_{k-1}\hat{x}_{k-1} \\
K_k &= \overline{P}_k {H_k}^{\mathrm{T}}\left(H_k\overline{P}_k {H_k}^{\mathrm{T}} + R_k\right)^{-1} \\
\overline{P}_k &= \Phi_{k-1}P_{k-1}{\Phi_{k-1}}^{\mathrm{T}} + Q_{k-1} \\
P_k &= (I - K_kH_k)\,\overline{P}_k
\end{aligned}
$$

The Proof Outline

- Assumptions
  - ⋆ initial estimates of $\overline{x}_0$ and its error covariance $\overline{P}_0$ are known
  - ⋆ best estimate is a linear combination of the best prior estimate and the measurement error

The Proof Outline

- Claims
  - ⋆ $P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^{\mathrm{T}}]$
  - ⋆ $\overline{P}_k = E[(x_k - \overline{x}_k)(x_k - \overline{x}_k)^{\mathrm{T}}]$
  - ⋆ $\hat{x}_k$ is the best possible (linear) estimate of x_k

Comments on the Proof

- Mathematics involves linear algebra, matrix calculus, and multivariate probability theory
- Only linear algebra portion is formalized in ACL2
- Assuming some key facts from the other branches of mathematics, the proof becomes an algebraic reduction

Taming Induction

- All functions we use are mutually recursive
- The proofs involve complex induction
- Our approach
  - ⋆ Avoid mutually recursive definitions
  - ⋆ Break complex (mutual) inductions into simpler inductions by (temporarily) assuming the needed instances of the mutual induction hypothesis

Matrix Inverses

- Matrix inverses appear in the computation of K_k
- How do we know these inverses exist?
  - ⋆ Currently, we are simply assuming they do
  - ⋆ In reality, they really do (matrices are pos. def.)
- In practice, if the algorithm fails to find an inverse, it can report the failure and reinitialize the filter — how can we capture this idea in ACL2?

Optimality Criterion

- Requires using matrix derivatives
- Currently, we are assuming the facts we need
- In principle, this could be formalized in ACL2(r)

Random Variables

- Proof uses several facts from multivariate probability
- Some of these are hard to formalize in ACL2
- In principle, we can formalize probability theory in ACL2(r)

Verifying Generated Software

- Annotate software with mapping from software entities to mathematical entities
- We verified a sample file — verification was fully automatic
- Open question: will it be as easy to verify other generated Kalman filters?