Compositional Assume-Guarantee Reasoning of Control Law Diagrams using UTP

Kangfeng Ye, Simon Foster, Jim Woodcock

Department of Computer Science, University of York

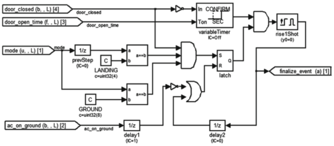
July 24, 2018

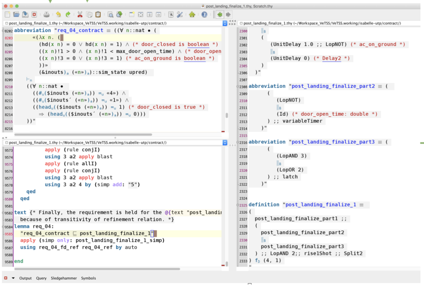
Outline

Outline

Objective

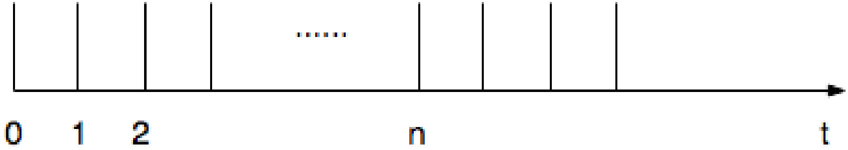
Requirement 1	A finalize event will be broadcast after the aircraft door has been open continuously for door_open_time seconds while the aircraft is on the ground after a successful landing.	Verified
Requirement 2	A finalize event is broadcast only once while the aircraft is on the ground.	Verified
Requirement 3	The finalize event will not occur during flight.	Verified
Requirement 4	The finalize event will not be enabled while the aircraft door is closed.	Verified





Understanding of Simulation in Simulink

- Based on an idealized timing model [MM07, CMW13]
  - All executions and updates of blocks are performed instantaneously (and infinitely fast) at exact simulation steps.
  - Between the simulation steps, the system is quiescent and all values held on lines and blocks are constant.
  - The inputs, states and outputs of a block can only be updated when there is a time hit for this block.



Assume-Guarantee Reasoning

- Based on the theory of designs in UTP

$$P \vdash Q \triangleq (P \wedge ok \Rightarrow Q \wedge ok')$$

- Compositional Assume-Guarantee reasoning
- Able to reason and resolve diagrams with algebraic loops
- Verified one subsystem (post landing finalize) in an aircraft cabin pressure control application

Outline

State Space

inouts from simulation time ($\mathbb{R}_{\geq 0}$) to a list of inputs (*inouts*) or outputs (*inouts*′).

$$inouts : \mathbb{R}_{\geq 0} \to \operatorname{seq} \mathbb{R} \qquad \text{[Dense time]}$$

According to the idealized timing model, abstracted to exact simulation steps ($t = n * T_b,\ T_b$ base rate)

$$inouts : \mathbb{N} \to \operatorname{seq} \mathbb{R} \qquad \text{[Step number]}$$

Healthiness Condition

Definition (***SimBlock***)

$$\textbf{\textit{SimBlock}}(m, n, P) \triangleq$$
$$\left(\begin{array}{l} (pre_D(P) \land post_D(P) \neq \textbf{\textit{false}}) \land \\ ((\forall\, l \bullet \#\,(inouts\ l) = m) \sqsubseteq Dom\ (pre_D(P) \land post_D(P))) \\ ((\forall\, l \bullet \#\,(inouts\ l) = n) \sqsubseteq Ran\ (pre_D(P) \land post_D(P))) \end{array}\right)$$

$$Dom(P) \triangleq (\exists\, inouts' \bullet P) \qquad Ran(P) \triangleq (\exists\, inouts \bullet P)$$

Definition (*inps* and *outps* - axiomatization)

$$\textbf{\textit{SimBlock}}(m, n, P) \Rightarrow (inps(P) = m \land outps(P) = n)$$

Pattern to Define Blocks

Definition ($FBlock$)

$$
\begin{array}{l}
FBlock\left(f_1, m, n, f_2\right) \\
\triangleq \left(\begin{array}{l}
\forall\, nn \bullet f_1\left(inouts, nn\right) \\
\vdash \\
\forall\, nn \bullet \left(\begin{array}{l}
\#\left(inouts(nn)\right) = m \,\wedge \\
\#\left(inouts'(nn)\right) = n \,\wedge \\
\left(inouts'(nn) = f_2\left(inouts, nn\right)\right)
\end{array}\right)
\end{array}\right)
\end{array}
$$

Common Blocks

Definition (Unit Delay)

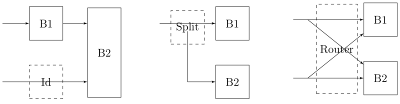
$$\begin{aligned} &\mathit{UnitDelay}\,(x_0) \triangleq \\ &\mathit{FBlock}\,(\mathit{true}_f, 1, 1, (\lambda\, x, n \bullet \langle x_0 \lhd n = 0 \rhd hd\,(x\,(n-1))\rangle)) \end{aligned}$$

where $\mathit{true}_f = (\lambda\, x, n \bullet \mathit{true})$

Definition (Product (Divide))

$$\begin{aligned} &\mathit{Div2} \triangleq \\ &\mathit{FBlock}\,((\lambda\, x, n \bullet hd(tl(x\ n)) \neq 0)\,, 2, 1, (\lambda\, x, n \bullet \langle hd(x\ n)/hd(tl(x\ n))\rangle)) \end{aligned}$$

Virtual Blocks



(a) Id (b) Split (c) Router

Definition (Id)

$$Id \triangleq FBlock\left(true_{\mathfrak{f}}, 1, 1, \left(\lambda\, x, n \bullet \langle hd\,(x\ n)\rangle\right)\right)$$

Definition (Split2)

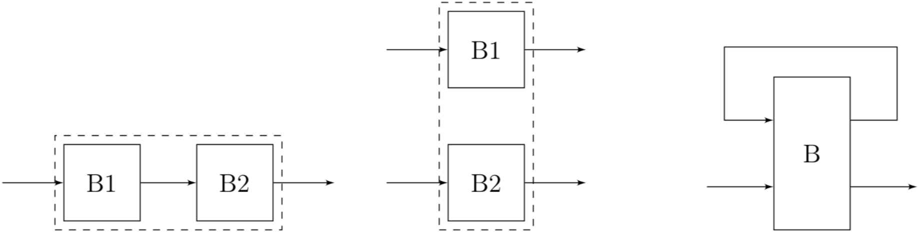
$$Split2 \triangleq FBlock\left(true_{\mathfrak{f}}, 1, 2, \left(\lambda\, x, n \bullet \langle hd\,(x\ n)\,, hd\,(x\ n)\rangle\right)\right)$$

Definition (Router)

$$Router\,(m, table) \triangleq FBlock\left(true_{\mathfrak{f}}, m, m, \left(\lambda\, x, n \bullet reorder\,((x\ n)\,, table)\right)\right)$$

Outline

Block Compositions



(d) Sequential Composition (e) Parallel Composition (f) Feedback

Sequential Composition

$$P = (FBlock\,(true_f, m_1, n_1, f_1)) \qquad \textbf{\textsf{SimBlock}}\,(m_1, n_1, P)$$
$$Q = (FBlock\,(true_f, n_1, n_2, f_2)) \qquad \textbf{\textsf{SimBlock}}\,(n_1, n_2, Q)$$

Theorem (Expansion)

$$(P;\ Q) = FBlock\,(true_f, m_1, n_2, (f_2 \circ f_1)) \qquad [\text{Expansion}]$$

Theorem (Closure)

$$\textbf{\textsf{SimBlock}}\,(m_1, n_2, (P;\ Q)) \qquad [\textbf{\textsf{SimBlock}}\ \text{Closure}]$$

Sequential Composition

Theorem (Expansion)

$$\begin{aligned}&(FBlock\,(p_1, m_1, n_1, f_1))\,;\;(FBlock\,(p_2, n_1, n_2, f_2))\\ &= FBlock\left(\begin{array}{l}(\lambda\, x\; n \bullet (p_1\; x\; n) \wedge ((p_2 \circ f_1)\; x\; n) \wedge \#(x\; n) = m_1)\\ , m_1, n_2, (f_2 \circ f_1)\end{array}\right)\end{aligned}$$

[Expansion]

Theorem (Closure)

$$\textbf{\textit{SimBlock}}\,(m_1, n_2, ((FBlock\,(p_1, m_1, n_1, f_1))\,;\;(FBlock\,(p_2, n_1, n_2, f_2))))$$

[***SimBlock*** Closure]

Parallel Composition I

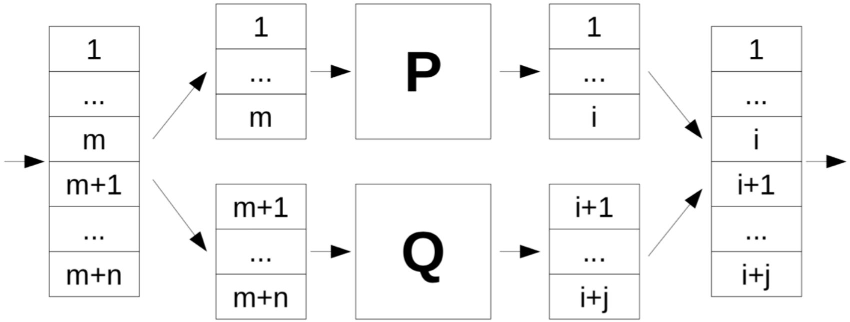
Definition (Parallel Composition)

$$P \parallel_B Q \triangleq \left(\begin{array}{l} (takem(inps(P) + inps(Q))\; inps(P);\; P) \\ \parallel_{B_M} \\ (dropm(inps(P) + inps(Q))\; inps(P);\; Q) \end{array} \right)$$

Definition (B_M)

$$B_M \triangleq \left(ok' = 0.ok \wedge 1.ok\right) \wedge \left(inouts' = 0.inouts \frown 1.inouts\right)$$

Parallel Composition II



Parallel Composition III

Theorem (Associativity, Monotonicity, and ***SimBlock*** Closure)

$$P_1 \parallel_B (P_2 \parallel_B P_3) = (P_1 \parallel_B P_2) \parallel_B P_3 \qquad \text{[Associativity]}$$

$$(P_1 \parallel_B Q_1) \sqsubseteq (P_2 \parallel_B Q_2) \qquad \text{[Monotonicity]}$$

$$\textbf{\textit{SimBlock}}\,(m1 + m2, n1 + n2, (P_1 \parallel_B P_2)) \qquad [\textbf{\textit{SimBlock}}\ \text{Closure}]$$

$$inps\,(P_1 \parallel_B P_2) = m_1 + m_2$$

$$outps\,(P_1 \parallel_B P_2) = n_1 + n_2$$

Parallel Composition IV

Theorem (Parallel Operator Expansion)

Provided

$$
\begin{array}{ll}
P = (FBlock\,(true_f, m_1, n_1, f_1)) & \textbf{\textit{SimBlock}}\,(m_1, n_1, P) \\
Q = (FBlock\,(true_f, m_2, n_2, f_2)) & \textbf{\textit{SimBlock}}\,(m_2, n_2, Q)
\end{array}
$$

then,

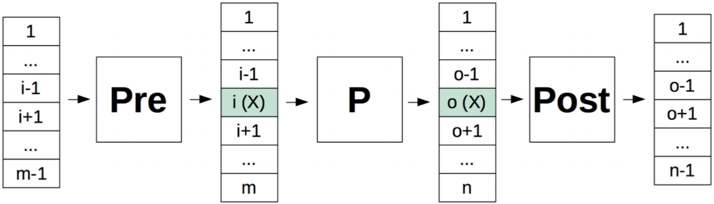
$$
(P \parallel_B Q) = FBlock \left(\begin{array}{l} true_f, m_1 + m_2, n_1 + n_2, \\ \left(\lambda\, x, n \bullet \left(\begin{array}{l} (f_1 \circ (\lambda\, x, n \bullet take\,(m_1, x\, n))) \\ {}^\frown (f_2 \circ (\lambda\, x, n \bullet drop\,(m_1, x\, n))) \end{array} \right) \right) \end{array} \right) \quad \text{[Expansion]}
$$

$$
\textbf{\textit{SimBlock}}\,(m_1 + m_2, n_1 + n_2, (P \parallel_B Q)) \quad [\textbf{\textit{SimBlock}} \text{ Closure}]
$$

Feedback I

Definition (f_D)

$$P\ f_D\ (i, o)$$
$$\triangleq (\exists\ sig \bullet (PreFD(sig, inps(P), i);\ P;\ PostFD(sig, outps(P), o)))$$



Feedback II

Definition ($PreFD$)

$$
\begin{aligned}
&PreFD(sig, m, idx)\\
&\triangleq FBlock\,(true_f, m-1, m, (f\_PreFD(sig, idx)))\\
&f\_PreFD(sig, idx)\\
&\triangleq \lambda\, x, n \bullet (take(idx, (x\ n)) \frown \langle (sig\ n) \rangle \frown drop(idx, (x\ n)))
\end{aligned}
$$

Feedback III

Definition (*PostFD*)

$$PostFD(sig, n, idx)$$

$$\triangleq \left(\begin{array}{l} \textbf{\textit{true}} \\ \vdash \\ \forall\, nn \bullet \left(\begin{array}{l} \#\,(inouts(nn)) = n \,\land \\ \#\,(inouts'(nn)) = n - 1 \,\land \\ (inouts'(nn) = (f\_PostFD(idx, inouts, nn)) \,\land \\ sig(nn) = inouts(nn)!idx \end{array} \right) \end{array} \right)$$

$$f\_PostFD(idx)$$

$$\triangleq \lambda\, x, n \bullet (take(idx, (x\ n)) \frown drop(idx + 1, (x\ n)))$$

Feedback IV

Theorem (Monotonicity)

Provided

$$\textbf{\textsf{SimBlock}}(m_1, n_1, P_1) \qquad \textbf{\textsf{SimBlock}}(m_1, n_1, P_2)$$

$$P_1 \sqsubseteq P_2 \qquad i_1 < m_1 \wedge o_1 < n_1$$

then,

$$(P_1\, f_D\, (i_1, o_1)) \sqsubseteq (P_2\, f_D\, (i_1, o_1))$$

Feedback V

Theorem (Expansion)

Provided

$$P = FBlock\,(true_{\mathsf{f}}, m, n, f) \qquad \textbf{\textsf{SimBlock}}\,(m, n, P)$$
$$Solvable\_unique(i, o, m, n, f) \qquad is\_Solution(i, o, m, n, f, sig)$$

then,

$$(P\, f_D\, (i, o))$$
$$= FBlock \left(\begin{array}{l} true_{\mathsf{f}}, m - 1, n - 1, \\ (\lambda\, x, n \bullet (f\_PostFD(o) \circ f \circ f\_PreFD(sig, x, i))\ x\ n) \end{array} \right)$$

[Expansion]

$$\textbf{\textsf{SimBlock}}\,(m - 1, n - 1, (P\, f_D\, (i, o)))$$

[**SimBlock** Closure]

Feedback I

Definition (*Solvable_unique*)

$$Solvable\_unique\,(i, o, m, n, f) \triangleq$$

$$\left(\begin{array}{l} (i < m \land o < n) \land \\ \left(\forall\, sigs \bullet \left(\begin{array}{l} (\forall\, nn \bullet \#\,(sigs\; nn) = (m-1)) \Rightarrow \\ \left(\begin{array}{l} \exists_1\, sig \bullet \\ \left(\begin{array}{l} \forall\, nn \bullet \\ \left(\begin{array}{l} sig\; nn = \\ (f\;(\lambda\, n_1 \bullet f\_PreFD\,(sig, i, sigs, n_1)\,, nn))!o \end{array}\right) \end{array}\right) \end{array}\right) \end{array}\right)\right) \end{array}\right)$$

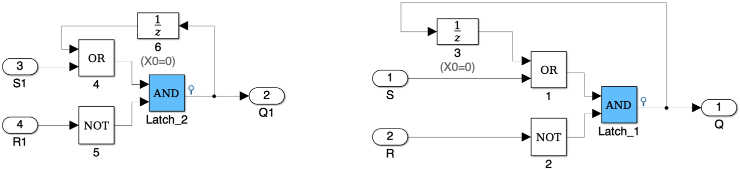
Feedback II

Definition ($is\_Solution$)

$$
is\_Solution\,(i, o, m, n, f, sig) \triangleq
\left(\left(\forall\, sigs \bullet \left(\begin{array}{l}
(\forall\, nn \bullet \#\,(sigs\; nn) = (m-1)) \Rightarrow \\
\left(\begin{array}{l}
\forall\, nn \bullet \\
\left(\begin{array}{l}
sig\; nn = \\
(f\,(\lambda\, n1 \bullet f\_PreFD\,(sig, i, sigs, n1)\,, nn))!o
\end{array}\right)
\end{array}\right)
\end{array}\right)\right)\right)
$$

Outline

Latch I




```
definition "latch ≡
  ((((UnitDelay 0 (*3*) ∥B Id) ;; (LopOR 2 (*1*)))
    ∥B
    (Id ;; LopNOT (*2*))
   ) ;; (LopAND 2) (*Latch_1*) ;; Split2
  ) fD (0,0)"
```

Latch II

```
fun latch_rec_calc_output:: "(nat ⇒ real) ⇒ (nat ⇒ real) ⇒ nat ⇒ real" where
"latch_rec_calc_output S R 0 =
    (if R 0 = 0 then (if S 0 = 0 then 0 else 1.0) else 0 )" |
"latch_rec_calc_output S R (Suc n) =
    (if R (Suc n) = 0 then (if S (Suc n) = 0 then (latch_rec_calc_output S R (n)) else 1.0) else 0)"
```

```
abbreviation "latch_simp_pat_f' ≡ (λx na. [
              latch_rec_calc_output (λn1. hd (x n1)) (λn1. x n1!(Suc 0)) (na)])"
```

```
abbreviation "latch_simp_pat' ≡ FBlock (λx n. True) 2 1 latch_simp_pat_f'"
```

```
lemma SimBlock_latch_simp':
   "SimBlock 2 1 latch_simp_pat'"
  using SimBlock_latch_simp latch_simp_pat_f_eq
  by simp

lemma latch_simp:
  "latch = latch_simp_pat'"
  proof - [325 lines]
  qed
```

Latch III

```
(* A SR AND-OR latch:
S   R   Action
0   0   No change
1   0   1
x   1   0
*)
text {* @{text "latch_req_00"}: if R is true, then the output is always false. *}
lemma latch_req_00:
  " ((∀ n::nat • (
      «(λx n. ((hd(x n) = 0 ∨ hd(x n) = 1) ∧ (hd(tl(x n)) = 0 ∨ hd(tl(x n)) = 1)))»
        (&inouts)ₐ («n»)ₐ)::sim_state upred)
    ⊢ₙ
    ((∀ n::nat •
      ((#ᵤ($inouts («n»)ₐ)) =ᵤ «2») ∧
      ((#ᵤ($inouts´ («n»)ₐ)) =ᵤ «1») ∧
      (headᵤ(tailᵤ($inouts («n»)ₐ)) ≠ᵤ 0) ⇒ (headᵤ(($inouts´ («n»)ₐ)) =ᵤ 0))
    )) ⊑ latch"
```

[CMW13] Ana Cavalcanti, Alexandre Mota, and Jim Woodcock.
Simulink timed models for program verification.
In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, volume 8051 of *Lecture Notes in Computer Science*, pages 82–99. Springer, 2013.

[MM07] Nicolae Marian and Yue Ma.
Translation of Simulink Models to Component-based Software Models, pages 274–280.
Forlag uden navn, 2007.