# A Two-way Path between Formal and Informal Design of Embedded Systems

Mingshuai Chen[1], Anders P. Ravn[2], Shuling Wang[1], Mengfei Yang[3], Naijun Zhan[1]

[1] State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences
[2] Department of Computer Science, Aalborg University
[3] Chinese Academy of Space Technology

Reykjavík, June 2016

# Motivations

| **Simulation-Based Design** | **Formal Verification** |
|---|---|
| engineers | theorists |
| efficient | costly |
| incomplete | reliable |

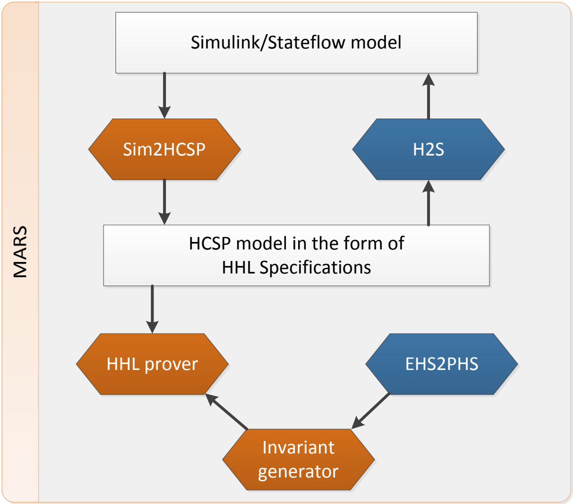
# Motivations

| Simulation-Based Design | | Formal Verification |
|---|---|---|
| engineers |  | theorists |
| efficient | | costly |
| incomplete | | reliable |

# Outline

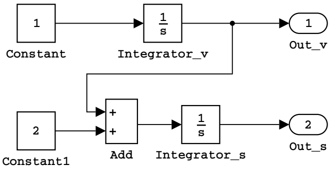
# Outline

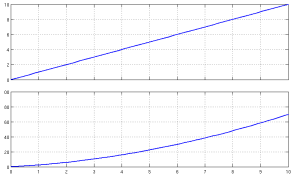
# Verification Architecture



# Simulink Diagrams

- A data flow diagram : blocks connected with wires.
- Example : $\dot{v} = 1, \dot{s} = v + 2$



- Blocks are running in parallel by receiving inputs and computing outputs.
- Sample time : 0/-1/positive value $t$.

# Hybrid CSP (HCSP)

- Syntax :

$$
\begin{array}{rcl}
P & ::= & \text{skip} \mid x := e \mid ch?x \mid ch!e \mid P; Q \mid B \to P \mid P \sqcup Q \mid P^* \\
 & & \mid \langle F(\dot{s}, s) = 0 \& B \rangle \mid \langle F(\dot{s}, s) = 0 \& B \rangle \trianglerighteq []_{i \in I}(io_i \to Q_i) \\
S & ::= & P \mid S \| S
\end{array}
$$

- Example : timeout $\langle F(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d Q$ can be defined by

$$t := 0; \langle F(\dot{s}, s) = 0 \wedge \dot{t} = 1 \& t < d \wedge B \rangle; t \geq d \to Q$$

# Outline

# Arithmetic Expressions

$$e \quad \widehat{=} \quad x \mid c \mid -e \mid (e) \mid e+e \mid e-e \mid e*e \mid e/e$$

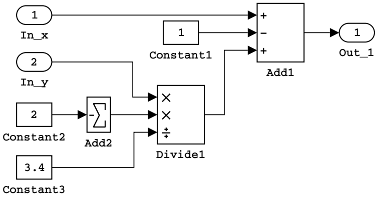



Figure : $x - 1 + y * ((-2)/3.4)$

# Boolean Expressions

$$B \quad \widehat{=} \quad \top \mid \bot \mid e \rhd e \mid \neg B \mid (B) \mid B \wedge B \mid B \vee B\,, \; \rhd \in \{<, \leq, >, \geq, =, \neq\}$$

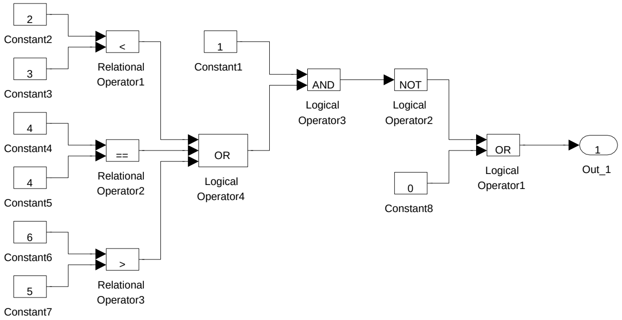



Figure : $\top \wedge (2 < 3 \vee 4 = 4 \vee 6 > 5) \Rightarrow \bot$

# Differential Equations

$$F \quad \widehat{=} \quad \dot{s} = e \mid F, F$$

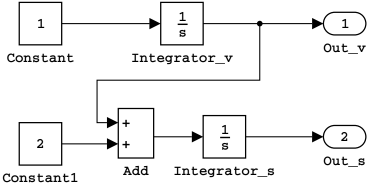
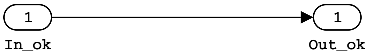



Figure : $\dot{v} = 1, \dot{s} = v + 2$

# Skip

skip




$$ok' = ok$$

# Assignment

$$x := e$$

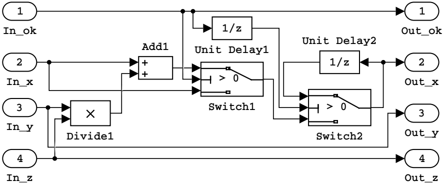
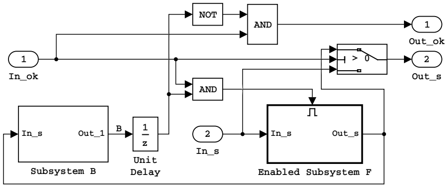



Figure : $x := x + y * z$

$$ok' = ok \qquad x' = \begin{cases} x'_{new}, & ok \wedge \neg d(ok) \\ x, & \neg ok \wedge \neg d(ok) \\ d(x'), & d(ok) \end{cases} \qquad \mathbf{u}' = \mathbf{u}$$

# Continuous Evolution

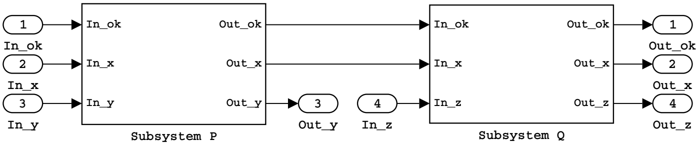
$$\langle F(\dot{s}, s) = 0 \& B \rangle$$




$$en = ok \wedge d(B) \qquad ok' = ok \wedge \neg d(B) \qquad s' = \begin{cases} s'_F, & ok \\ s, & \neg ok \end{cases}$$

# Sequential

$$P; Q$$

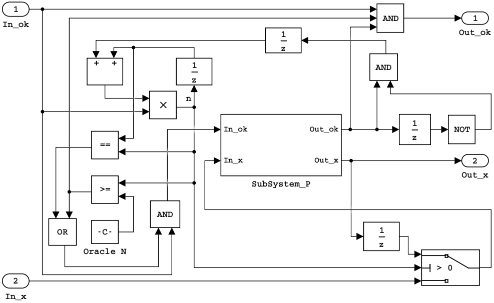



$$ok_P = ok \quad ok_Q = ok'_P \quad ok' = ok'_Q$$

$$x_P = x \quad x_Q = x'_P \quad x' = x'_Q$$

# Repetition

$P^*$

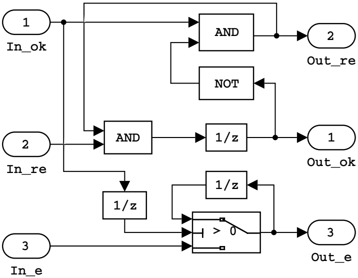



$$n = ok \times (d(n) + d(ok'_P \wedge \neg d(ok'_P))) \qquad ok' = ok \wedge ok'_P \wedge (n \geq N)$$

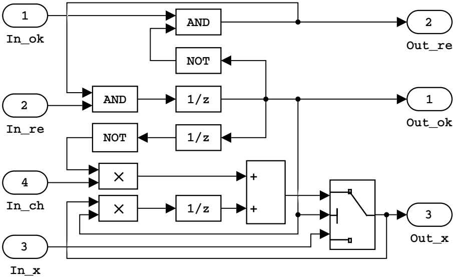
$$ok_P = ok \wedge (n == d(n) \vee n \geq N) \qquad x_P = \begin{cases} d(x'_P), n > 0 \\ x, \quad n == 0 \end{cases}$$

# Communication Events

$ch!e$




$ch?x$




$$re' = ok \wedge \neg ok' \quad ok' = f(d(re \wedge re'))$$

$$e' = \begin{cases} e, & \neg d(ok) \\ d(e'), & d(ok) \end{cases}$$

$$re' = ok \wedge \neg ok' \quad ok' = f(d(re \wedge re'))$$

$$x' = \begin{cases} x, & \neg ok' \\ \neg d(ok') \times ch + d(ok') \times d(x'), & ok' \end{cases}$$

# Parallel

$$P \| Q$$

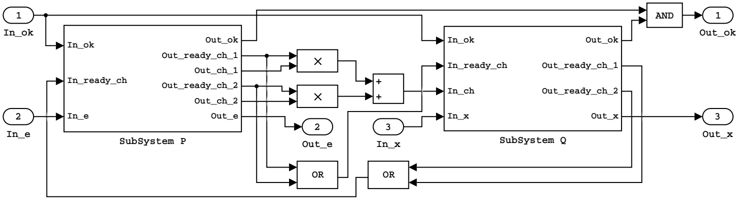


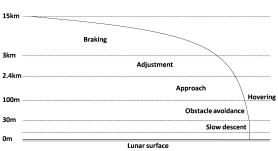

Figure : $e := 0; ch!e; < \dot{e} = 1 \& e < 2 >; ch!e \| x := 3; ch?x; ch?x$

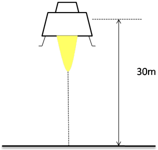
$$ok_P = ok_Q = ok \quad ok' = ok'_P \wedge ok'_Q \quad re_{ch\_P} = \bigvee_{i=1}^{n} re'_{ch\_i\_Q} \quad re_{ch\_Q} = \bigvee_{j=1}^{m} re'_{ch\_j\_P}$$

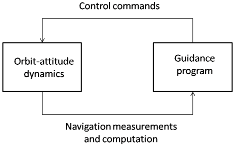
# Outline

# Design Problem

## Mission Description




## Design Objectives

(R1) $|v + 2| \leq 0.05$m/s during the slow descent phase and before touchdown ;

(R2) $|v| < 5$m/s at the time of touchdown ;

# From Simulink to HCSP

$$
\begin{array}{rcl}
P & \widehat{=} & PC \,\|\, PD \\
PC & \widehat{=} & v := -2;\; m := 1250;\; r := 30; \\
 & & (\, \langle Sys_1 \,\&f > 3000\rangle \trianglerighteq CommI; \\
 & & \langle Sys_2 \,\&f \leq 3000\rangle \trianglerighteq CommI\,)^* \\
PD & \widehat{=} & t := 0;\; g := 1.622;\; vslw := -2;\; f_1 = 2027.5; \\
 & & (\, ch_v?v_1;\; ch_m?m_1;\; f_1 := m_1 * aIC;\; ch_f!f_1; \\
 & & temp := t;\; \langle \dot{t} = 1 \& t < temp + 0.128\rangle\,)^* \\
aIC & \widehat{=} & g - 0.01 * (f_1 / m_1 - g) - 0.6 * (v_1 - vslw) \\
Sys_1 & \widehat{=} & \dot{m} = -f/2548,\; \dot{v} = f/m - 1.622,\; \dot{r} = v \\
Sys_2 & \widehat{=} & \dot{m} = -f/2842,\; \dot{v} = f/m - 1.622,\; \dot{r} = v \\
CommI & \widehat{=} & ch_f?f \rightarrow \text{skip} \;[]\; ch_v!v \rightarrow \text{skip} \;[]\; ch_m!m \rightarrow \text{skip}
\end{array}
$$

# From HCSP to Simulink

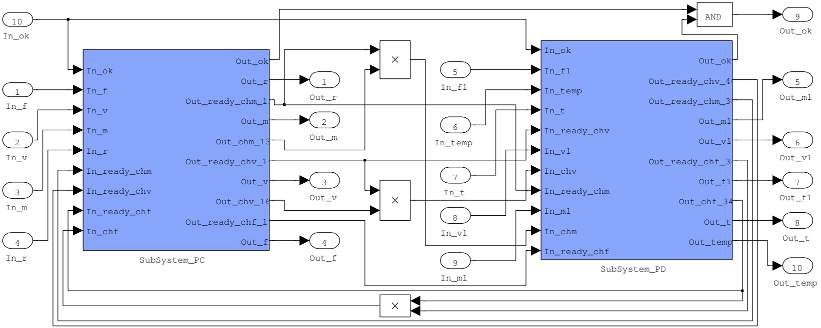



Figure : The top-level view of the translated Simulink model

# Simulation Results

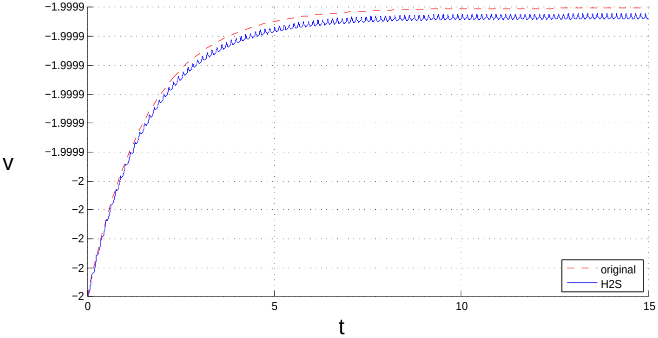



Figure : The evolution of velocity $v$ in physical plant $PC$

# Outline

# Proving Target

$$[\![P]\!] \Leftrightarrow [\![\mathsf{H2S}(P)]\!] \quad \textbf{?}$$

# Reactive Design

- A *sequential program* is represented by a design $D = (\alpha, P)$, where
  - $\alpha$ : the set of state variables (observables), $\{x, x', ok, ok'\}$ ;
  - $P$ : a predicate, denoted by $p(x) \vdash R(x, x')$, and defined as

$$(ok \wedge p(x)) \Rightarrow (ok' \wedge R(x, x')).$$

- The domain of designs forms a complete lattice with the refinement partial order, and this lattice is closed under the classical programming constructs.

- A *concurrent and reactive program* is defined by a reactive design $P$,

$$\mathcal{H}'(P) = P \qquad (\textit{Healthiness condition})$$

with $\mathcal{H}'(P) \mathrel{\widehat{=}} (\vdash \wedge_{x \in \alpha(P)} x' = x \wedge wait' = wait) \lhd wait \rhd P$.

# Hybrid Design

## Hybrid Design

A design is called a hybrid design if it meets the healthiness condition

$$\mathcal{H}(S) = S, \quad \text{where}$$
$$\mathcal{H}(S) \mathrel{\widehat=} (\vdash \mathbf{x}' = \mathbf{x} \wedge wait' = wait \wedge S_C) \lhd wait \rhd S.$$

with $S_C \mathrel{\widehat=} \langle F(\dot{\mathrm{s}}, \mathrm{s}) = 0 \& B \rangle$.

- allowing function variables and quantifications over functions ;
- continuous dynamics $S_C$ is not blockable by communications ;
- $now$, $now'$ ;
- $\text{Periodic}(ch^*, st) \mathrel{\widehat=} \forall n \in \mathbb{N}.\ t = n * st \Rightarrow ch^*(t)$.

# Blocks

$$
\begin{aligned}
& [\![ B(ps, in, out) ]\!] \\
\widehat{=} \quad & \mathcal{H}(Ass \vdash out(0) = ps.init \wedge \bigwedge_{k=1}^{m} (B_k(ps, in) \Rightarrow P_k(ps, in, out)))
\end{aligned}
$$

# Continuous Blocks

$$[\![ \mathcal{CB}(ps, in, out) ]\!] \quad \widehat{=} \quad \mathcal{H}(in! \vdash out(0) = ps.init \wedge \left( \begin{pmatrix} B_1(in, ps) & \Rightarrow & F_1(\dot{out}, out, in, ps) = 0 \wedge \cdots \wedge \\ B_m(in, ps) & \Rightarrow & F_m(\dot{out}, out, in, ps) = 0 \end{pmatrix} \wedge out! \right)),$$

with $wait \ \widehat{=}\ \neg out?$.

# Continuous Blocks

$$[\![\mathcal{CB}(ps, in, out)]\!] \;\widehat{=}\; \mathcal{H}(in! \vdash\; out(0) = ps.init \wedge \left(\begin{pmatrix} B_1(in, ps) & \Rightarrow & F_1(\dot{out}, out, in, ps) = 0 \wedge \cdots \wedge \\ B_m(in, ps) & \Rightarrow & F_m(\dot{out}, out, in, ps) = 0 \end{pmatrix} \wedge out!\right)),$$

with $wait \;\widehat{=}\; \neg out?$.

## Example

- A `Constant` block generates a scalar constant value :

$$[\![\texttt{Constant}(ps.c, out)]\!] \;\widehat{=}\; \mathcal{H}(\vdash\; out(0) = c \wedge \dot{out} = 0 \wedge out!).$$

- A `Delay` block holds and delays its input by one sample period :

$$\begin{aligned} & [\![\texttt{Delay}(ps, in, out)]\!] \\ \widehat{=}\;\; & \mathcal{H}(in! \vdash \begin{pmatrix} cnow < ps.st \Rightarrow out(cnow) = ps.init \wedge \\ cnow \geq ps.st \Rightarrow out(cnow) = in(cnow - ps.st) \end{pmatrix} \wedge out!). \end{aligned}$$

- The `Integrator` block outputs the value of the integral of its input signal :

$$[\![\texttt{Integrator}(ps, in, out)]\!] \;\widehat{=}\; \mathcal{H}(in! \vdash out(0) = ps.init \wedge (\dot{out} = in \wedge out!)).$$

# Discrete Blocks

$$
\begin{aligned}
[\![ DB(ps, in, out) ]\!] \quad \widehat{=} \quad & \mathcal{H}(\text{Periodic}(in!, ps.st) \wedge \text{Periodic}(out?, ps.st) \vdash \; out(0) = ps.init \wedge \\
& \text{Periodic}(out!, ps.st) \wedge (\exists n \in \mathbb{N}.\ cnow = n * st) \Rightarrow \\
& \qquad \begin{pmatrix} B_1(in, ps) & \Rightarrow & [\![ P_{comp_1}(in, out, ps) ]\!] \wedge \cdots \wedge \\ B_m(in, ps) & \Rightarrow & [\![ P_{comp_m}(in, out, ps) ]\!] \end{pmatrix}),
\end{aligned}
$$

with $wait \;\widehat{=}\; \neg \exists n \in \mathbb{N}.\ cnow = n * st.$

# Discrete Blocks

$$[\![\mathit{DB}(ps, in, out)]\!] \;\widehat{=}\; \mathcal{H}(\mathrm{Periodic}(in!, ps.st) \wedge \mathrm{Periodic}(out?, ps.st) \vdash out(0) = ps.init \wedge \mathrm{Periodic}(out!, ps.st) \wedge (\exists n \in \mathbb{N}.\ cnow = n * st) \Rightarrow \begin{pmatrix} B_1(in, ps) & \Rightarrow & [\![P_{comp_1}(in, out, ps)]\!] \wedge \cdots \wedge \\ B_m(in, ps) & \Rightarrow & [\![P_{comp_m}(in, out, ps)]\!] \end{pmatrix}),$$

with $wait \;\widehat{=}\; \neg\exists n \in \mathbb{N}.\ cnow = n * st.$

## Example

- The logical operator And performs conjunction on its inputs :

$$[\![\mathrm{And}(ps.I, \{in_i\}_{i\in I}, out)]\!] \widehat{=} \mathcal{H}(\wedge_{i\in I}\mathrm{Periodic}(in_i!, ps.st) \wedge \mathrm{Periodic}(out?, ps.st) \vdash \mathrm{Periodic}(out!, ps.st) \wedge \exists n \in \mathbb{N}.\ cnow = n * ps.st \Rightarrow out = \bigwedge_{i\in I} in_i).$$

- The Switch block passes through the first or the third input :

$$[\![\mathtt{Switch}(ps, in_1, in_2, in_3, out)]\!] \;\widehat{=}\; \mathcal{H}(\wedge_{i=1}^{3}\mathrm{Periodic}(in_i!, ps.st) \wedge \mathrm{Periodic}(out?, ps.st) \vdash \mathrm{Periodic}(out!, ps.st) \wedge (\exists n \in \mathbb{N}.\ cnow = n * ps.st) \Rightarrow \begin{pmatrix} ps.op(in_2, ps.c) \Rightarrow out = in_1 \wedge \\ \neg ps.op(in_2, ps.c) \Rightarrow out = in_3 \end{pmatrix}).$$

# Diagrams

## Example

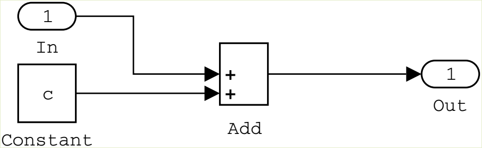



Figure : A diagram performing $out = in + c$

$$[\![\mathtt{Diag}(ps, in, out)]\!] \mathrel{\widehat{=}} \exists out'.\mathcal{H}(\mathrm{Periodic}(in!, ps.st) \wedge \mathrm{Periodic}(out?, ps.st) \vdash ([\![\mathtt{Constant}(ps, out')]\!] \wedge [\![\mathtt{Add}(ps, \{+1, +1\}, \{in_1, in_2\}, out)]\!][in/in_1, out'/in_2]).$$

# Subsystems

- Normal subsystem :

$$[\![ \mathsf{NSub}(ps, \{in_i\}_{i\in I}, \{out_j\}_{j\in J} ]\!] \mathrel{\widehat{=}} [\![ \mathtt{Diag}(ps, \{in'_i\}_{i\in I'}, \{out'_j\}_{j\in J'} ]\!][\sigma].$$

- Enabled subsystem :

$$\begin{aligned}[\![ \mathsf{ESub}(ps, \{in_i\}_{i\in I}, en, \{out_j\}_{j\in J}) ]\!] \mathrel{\widehat{=}} \; & en(now) > 0 \Rightarrow [\![ \mathsf{NSub}(ps, \{in_i\}_{i\in I}, en, \{out_j\}_{j\in J}) ]\!] \wedge \\ & en(now) \leq 0 \Rightarrow out(now) = out(now - ps.st).\end{aligned}$$

# Timed Observation

- Alphabet of a hybrid system :
  1. $\mathcal{V}(P)$ : the set of variable names, arranged as a vector $\mathbf{v}$.
  2. $i\Sigma(P)$ : the set of input channel names.
  3. $o\Sigma(P)$ : the set of output channel names. $\Sigma(P) \widehat{=} i\Sigma(P) \cup o\Sigma(P)$ is put in a vector $ch_P$.

- Timed observation :

$$\langle now, \mathbf{v}, \mathbf{f_v}, re_{ch*}, msg_{ch} \rangle.$$

- Constant notations :

$$const(\mathbf{f}, \mathbf{b}, t_1, t_2) \ \widehat{=} \ \forall t \in [t_1, t_2].\ \mathbf{f}(t) = \mathbf{b},$$
$$const^l(\mathbf{f}, \mathbf{b}, t_1, t_2) \ \widehat{=} \ \forall t \in [t_1, t_2).\ \mathbf{f}(t) = \mathbf{b},$$
$$const^r(\mathbf{f}, \mathbf{b}, t_1, t_2) \ \widehat{=} \ \forall t \in (t_1, t_2].\ \mathbf{f}(t) = \mathbf{b}.$$

# UTP Semantics for HCSP

## Example

$$[\![\text{skip}]\!] \mathrel{\widehat{=}} \mathcal{H}(\vdash now' = now \wedge \mathbf{v}' = \mathbf{v} \wedge const(\mathbf{f_v}, \mathbf{v}, now, now') \wedge \underbrace{const(re_{ch*}, \mathbf{0}, now, now') \wedge const(msg_{ch}, msg_{ch}(now), now, now')}_{RE}).$$

$$[\![x := e]\!] \mathrel{\widehat{=}} \mathcal{H}(\vdash now' = now \wedge x' = e \wedge \mathbf{u}' = \mathbf{u} \wedge const(\mathbf{f}_x, e, now, now') \wedge const(\mathbf{f_u}, \mathbf{u}, now, now') \wedge RE).$$

$$[\![\langle F(\dot{s}, s) = 0 \& B\rangle]\!] \mathrel{\widehat{=}} (\vdash F(\dot{s}, s = 0) \wedge \dot{t} = 1) \lhd B \rhd [\![\text{skip}]\!].$$

# UTP Semantics for HCSP

## Example (Closed under Sequential Composition)

$$[\![P; Q]\!] \mathrel{\widehat{=}} [\![P]\!] \mathbin{;} [\![Q]\!],$$

where for

$$H_1 \mathrel{\widehat{=}} (\vdash \wedge_{x \in \mathcal{V}(H_1)} x' = x \wedge wait'_{H_1} = wait_{H_1} \wedge S_{H_1}) \lhd wait_{H_1} \rhd (p_{H_1} \vdash R_{H_1}),$$

$$H_2 \mathrel{\widehat{=}} (\vdash \wedge_{x \in \mathcal{V}(H_2)} x' = x \wedge wait'_{H_2} = wait_{H_2} \wedge S_{H_2}) \lhd wait_{H_2} \rhd (p_{H_2} \vdash R_{H_2}),$$

$$\begin{aligned} H_1 \mathbin{;} H_2 \mathrel{\widehat{=}}\ & \exists wait_{H_1}, wait_{H_2}. \exists \mathbf{v}_{H_1}, now_{H_1}, ok_{H_1}. \\ & \exists \mathbf{f_v}_{H_1}, re_{ch_{H_1}*}, msg_{ch_{H_1}}, f_{\mathbf{v}_{H_2}}, re_{ch_{H_2}*}, msg_{ch_{H_2}}. \\ & (\vdash (wait_{H_1} \Rightarrow \Pi_{H_1}) \wedge (wait_{H_2} \Rightarrow \Pi_{H_2}) \wedge wait' = wait) \lhd wait \rhd \\ & \quad (\neg wait_{H_1} \wedge wait_{H_2} \wedge r_{H_1} \vdash R_{H_1})\sigma_{H_1} \wedge \\ & \quad (\neg wait_{H_1} \wedge \neg wait_{H_2} \wedge r_{H_2} \vdash R_{H_2})\sigma_{H_2} \wedge \\ & \qquad \forall t \in [now, now_{H_1}). wait(t) = wait_{H_1}(t) \wedge \\ & \qquad\quad \mathbf{f_v}(t) = \mathbf{f_v}_{H_1}(t) \wedge re_{ch*}(t) = re_{ch_{H_1}*}(t) \wedge msg_{ch}(t) = msg_{ch_{H_1}}(t) \wedge \\ & \qquad \forall t \in [now_{H_1}, now']. wait(t) = wait_{H_2}(t) \wedge \\ & \qquad\quad \mathbf{f_v}(t) = \mathbf{f_v}_{H_2}(t) \wedge re_{ch*}(t) = re_{ch_{H_2}*}(t) \wedge msg_{ch}(t) = msg_{ch_{H_2}}(t). \end{aligned}$$

# UTP Semantics for HCSP

## Example (Repetition)

$$[\![P^*]\!] \;\Leftrightarrow\; [\![\mathrm{rec}\, X.(\mathsf{skip} \sqcup (P;\ X)]\!] \;\Leftrightarrow\; \exists N.[\![P^N]\!],$$

with $P^0 \;\widehat{=}\; \mathsf{skip}$.

## Example (Receiving Event)

$$[\![ch?x]\!] \;\widehat{=}\; \vdash LHS \lhd re_{ch?} \wedge \neg re_{ch!} \rhd RHS,$$

where

$$LHS \;\widehat{=}\; \dot{t} = 1 \wedge x' = x \wedge \mathbf{u}' = \mathbf{u},$$

$$RHS \;\widehat{=}\; now' = now + d \wedge re'_{ch?} = 0 \wedge re'_{ch!} = 0 \wedge \mathbf{u}' = \mathbf{u} \wedge x' = msg_{ch}(now') \wedge$$
$$const^l(re_{ch?}, 1, now, now') \wedge const^l(re_{ch!}, 0, now, now').$$

# UTP Semantics for HCSP

**Example (Closed under Parallel Composition)**

$$[\![P \parallel Q]\!] \mathrel{\widehat{=}} [\![P]\!] \parallel [\![Q]\!] \,.$$

# Correctness

**Theorem (Correctness)**

Given an HCSP process $P$, denote the translated Simulink diagram by $\texttt{H2S}(P)$. Suppose there is a correspondence (denoted by $EA$) between $[\![P]\!]$ and $[\![\texttt{H2S}(P)]\!]$, i.e., $now = gst$, $now' = \tau$, $ok = In\_ok(gst) = \top$, $ok' = Out\_ok(\tau)$, $\mathbf{v} = In\_\mathbf{v}(gst)$, $\mathbf{v}' = Out\_\mathbf{v}(\tau)$, $f_{\mathbf{v}} = Out\_\mathbf{v}|_{[gst,\tau]}$, $re_{ch*} = Out\_re_{ch*}|_{[gst,\tau]}$, and $msg_{ch} = Out\_re_{ch}|_{[gst,\tau]}$, then we have

$$\text{Periodic}(in!, ps.gst) \wedge \text{Periodic}(out?, ps.gst) \Rightarrow \left([\![P]\!] \Leftrightarrow [\![\texttt{H2S}(P)]\!]|_{[gst,\tau]}\right)$$

as $gst \rightarrow 0$.

# Outline

# Concluding Remarks

1. A translator from HCSP formal models into Simulink graphical models :
   - simulating and testing HCSP formal models using the MATLAB platform ;
   - flexibly shifting between formal and informal models according to a desired trade-off.
2. A UTP semantics for both simulink and HCSP.
3. A UTP based semantical foundation to justify that the translation preserves semantics.