Extending Hoare Logic to Real-Time

Jozef Hooman

Outline

- Introduction
- Basic Framework
- Example Chemical Batch Processing
- Asynchronous Communication
- Introducing Sensor and Actuator in the Example
- Programming Language
- Example Chemical Batch-Final Implementation
- Concluding Remarks

Introduction

- Distributed real-time systems.
- The timing of observable actions (*now*).
- Termination and nonterminating computation.

Basic Framework

Parallel Process

- obs(P) be the set of (representations of) observable objects of process P.

$$obs(P_1 \| P_2) = obs(P_1) \cup obs(P_2)$$

The names of channels and shared variables contained in P.

- loc(P), describing local objects of P that are not observable by other parallel processes.

$$loc(P_1) \cap loc(P_2) = \emptyset$$

The names of local variables.

Specifications

Assertion A expresses assumptions:

- the values of local objects at the start of P,
- the starting time of P,
- the timed occurrence of observable events.

(timed occurrence function assigns to each point of time a set of records representing the observable events occurring at that time.)

Given assumption A, assertion C expresses a commitment of P,

- if P terminates, about the values of the local objects at termination,
- the termination time (∞ if P does not terminate),
- the timed occurrence of observable events.

Given $P\ at\ t$ and a set (usually an interval) $I \subseteq TIME$, we use

- P **during** $I \equiv \forall t \in I\ : P$ **at** t,
- P **in** $I \equiv \exists t \in I\ : P$ **at** t.

Examples of Specifications

$$\langle\langle x = 5 \wedge now = 6 \wedge O \textbf{ at } 3 \rangle\rangle \quad F \quad \langle\langle x = f(5) \wedge 15 < now < 23 \wedge O \textbf{ at } 3 \rangle\rangle.$$

$$\langle\langle x = v \wedge now = t < \infty \rangle\rangle \ FUN \ \langle\langle y = f(v) \wedge x = v \wedge t + 5 < now < t + 13 \rangle\rangle.$$

$$\langle\langle x = 0 \wedge now = 0 \rangle\rangle \quad L \quad \langle\langle now = \infty \wedge \forall i \in \mathbf{N} : (output, f(i)) \textbf{ at } \boxed{T(i)} \rangle\rangle.$$

$$\langle\langle now = 0 \rangle\rangle \quad REACT \quad \langle\langle (\forall t < now : (input, v) \textbf{ at } t \rightarrow (output, f(v)) \textbf{ in } [t + T_l, t + T_u]) \wedge (now < \infty \leftrightarrow \exists t_0 < now : \boxed{(input, 0)} \textbf{ at } t_0) \rangle\rangle.$$

$$\langle\langle p \wedge now < \infty \rangle\rangle \ P \ \langle\langle now < \infty \rightarrow q \rangle\rangle.$$

$$\langle\langle p \wedge now < \infty \rangle\rangle \ P \ \langle\langle now < \infty \wedge q \rangle\rangle.$$

Proof Rules

Rule 2.1. (Consequence)

$$\frac{\langle\!\langle A_0 \rangle\!\rangle \; P \; \langle\!\langle C_0 \rangle\!\rangle, A \rightarrow A_0, C_0 \rightarrow C}{\langle\!\langle A \rangle\!\rangle \; P \; \langle\!\langle C \rangle\!\rangle}$$

Rule 2.2. (Parallel Composition)

$$\frac{\langle\!\langle A_1 \rangle\!\rangle \; P_1 \; \langle\!\langle C_1 \rangle\!\rangle, \quad \langle\!\langle A_2 \rangle\!\rangle \; P_2 \; \langle\!\langle C_2 \rangle\!\rangle, \quad Comb(C_1, C_2) \rightarrow C}{\langle\!\langle A_1 \wedge A_2 \rangle\!\rangle \; P_1 \| P_2 \; \langle\!\langle C \rangle\!\rangle}$$

provided

- $\boxed{loc(C_1) \cap loc(P_2) = \emptyset \text{ and } loc(C_2) \cap loc(P_1) = \emptyset,}$ that is, the commitment of one process should not refer to local objects of the other.
- $\boxed{obs(A_1, C_1) \cap obs(P_2) \subseteq obs(P_1)}$ and $\boxed{obs(A_2, C_2) \cap obs(P_1) \subseteq obs(P_2)}$, i.e., if assertions in the specification of one process refer to the interface of another process then this concerns a joint interface.

Comb

1. If *now* does not occur in C_1 and C_2 then define

$$Comb(C_1, C_2) \equiv C_1 \wedge C_2.$$

2 $Comb(C_1, C_2) \equiv C_1[t_1/now] \wedge C_2[t_2/now] \wedge now = max(t_1, t_2).$

3 $NoAct(oset)$ **at** $texp \equiv \bigwedge_{O \in oset} \neg O$ **at** $texp.$

$$\begin{aligned} Comb(C_1, C_2) \equiv\ & C_1[t_1/now] \wedge NoAct(obs(P_1)) \textbf{ during } [t_1, now) \wedge \\ & C_2[t_2/now] \wedge NoAct(obs(P_2)) \textbf{ during } [t_2, now) \wedge \\ & now = max(t_1, t_2). \end{aligned}$$

Example Chemical Batch Processing

Example Chemical Batch Processing

- **expl at** $texp$ $\qquad obs(\text{expl } \mathbf{at}\ texp) = \{\text{expl}\}.$

$$\langle\langle now = 0\rangle\rangle\ CBP\ \langle\langle \forall t < \infty : \neg\text{expl } \mathbf{at}\ t\rangle\rangle.$$

- **empty at** $texp$
- temp($texp$)

$obs(\text{empty } \mathbf{at}\ texp) = \{\text{empty}\}\quad obs(\text{temp}(texp)) = \{\text{temp}\}$

$$CV \equiv \forall t < \infty : \text{temp}(t) \le \text{ExpTemp} \vee \text{empty } \mathbf{at}\ t \rightarrow \neg\text{expl } \mathbf{at}\ t.$$

$$\langle\langle now = 0\rangle\rangle\ V\ \langle\langle CV\rangle\rangle \qquad \boxed{obs(V) = \{\text{expl}, \text{temp}, \text{empty}\}}$$

Example Chemical Batch Processing

$$CHL \equiv \forall t < \infty : \mathsf{temp}(t) > \mathsf{ExpTemp} \rightarrow \mathsf{empty} \textbf{ at } t.$$

$$\langle\!\langle now = 0 \rangle\!\rangle \; HLContr \; \langle\!\langle CHL \rangle\!\rangle$$

$$obs(HLContr) \supseteq \{\mathsf{temp}, \mathsf{empty}\}$$

$$obs(CV) \cap obs(HLContr) \subseteq obs(CV) = \{\mathsf{expl}, \mathsf{temp}, \mathsf{empty}\} = obs(V)$$

$$obs(CHL) \cap obs(V) = \{\mathsf{temp}, \mathsf{empty}\} \subseteq obs(HLContr).$$

$$\langle\!\langle now = 0 \rangle\!\rangle \; V \| HLContr \; \langle\!\langle CV \wedge CHL \rangle\!\rangle.$$

$$CV \wedge CHL \text{ implies } \forall t < \infty : \neg\mathsf{expl} \textbf{ at } t.$$

Asynchronous Communication

Asynchronous Communication

- $(c!!, exp)$ **at** $texp$ Start sending
- $c?$ **at** $texp$ Wait to receive
- (c, exp) **at** $texp$ Start to receive.

$$obs((c!!, exp) \textbf{ at } texp) = \{c!!\}, \quad obs(c? \textbf{ at } exp) = \{c?\}$$

$$obs((c, exp) \textbf{ at } texp) = \{c\}.$$

- $await\,(c?, v) \textbf{ at } t \equiv \boxed{c? \textbf{ during } [t, \infty)} \vee$ $\boxed{(\exists t_1 \in [t, \infty) : c? \textbf{ during } [t, t_1) \wedge (c, v) \textbf{ at } t_1)}$

We often abstract from the value that is transmitted, using

- c **at** $t \equiv \exists v : (c, v)$ **at** t
- $c!!$ **at** $t \equiv \exists v : (c!!, v)$ **at** t
- $await\ c?$ **at** $t \equiv \boxed{\exists v : await\ (c?, v) \textbf{ at } t}$

Start waiting to receive value v through channel c.

Communication Properties

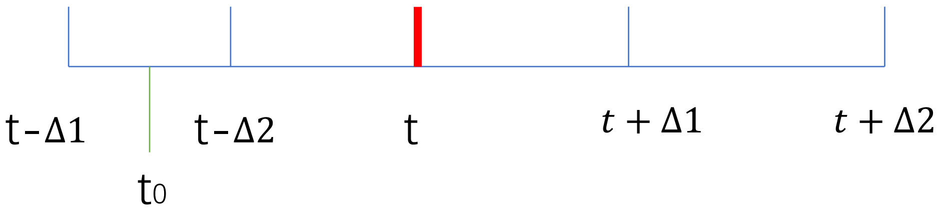
- $\forall t < \infty \, \forall v_1, v_2 : (c!!, v_1) \textbf{ at } t \wedge (c!!, v_2) \textbf{ at } t \rightarrow v_1 = v_2$
- $\forall t < \infty \, \forall v : (c, v) \textbf{ at } t \rightarrow (c!!, v) \textbf{ at } t$
- $\forall t < \infty : \neg(c!! \textbf{ at } t \wedge c? \textbf{ at } t)$

$$\forall t < \infty\, \forall v : (c, v) \text{ at } t \rightarrow t \geq \Delta \wedge (c!!, v) \text{ at } (t - \Delta)$$

$$\forall t < \infty : \neg(c!! \text{ at } t \wedge c? \text{ at } (t + \Delta)).$$

$$\forall t < \infty\, \forall v : (c, v) \text{ at } t \rightarrow$$
$$\exists t_0 \in [t - \Delta_1, t - \Delta_2] : (c!!, v) \text{ at } t_0 \wedge (\neg c!!) \text{ during } (t_0, t - \Delta_2]$$

$$\forall t < \infty : \neg(c!! \text{ at } t \wedge c? \text{ during } [t + \Delta_1, t + \Delta_2])$$



Syntax Programming Language

Atomic statements

- **skip** terminates immediately.
- Assignment $x := e$ assigns the value of expression e to the variable x.
- **delay** e suspends execution for (the value of) e time units. If e yields a negative value then **delay** e is equivalent to **skip**.
- Output statement $c!!e$ is used to send the value of expression e along channel c. It does not wait for a receiver but sends immediately.
- Input statement $c?x$ is used to receive a value along channel c and assign this value to the variable x. Such an input statement has to wait until a message is available.

Compound statements

- $S_1 ; S_2$ indicates sequential composition.
- **if** b **then** S_1 **else** S_2 **fi** denotes the usual conditional choice construct.
- **sel** $c?x$ **then** S_1 **or delay** e **then** S_2 **les** is a select statement. First wait to receive a message on channel c and, if a message is available within e time units, execute S_1. If no message is available during e time units, S_2 is executed.

- **while** b **do** S **od**,
- $S_1 \| S_2$

if b **then** S **fi** $\Rightarrow$ **if** b **then** S **else skip fi**

Basic Timing Assumptions

- The execution time of atomic statements. Here we use (nonnegative) parameters representing the duration of atomic statements. We assume that
  - there exists a parameter T_a such that each assignment of the form $x := e$ takes T_a time units;
  - **delay** e takes exactly e time units if e is positive and 0 time units otherwise;
  - there exist a parameter $T_{comm} > 0$ such that each communication takes T_{comm} time units.

T_w **while** b **do** S **od**

Proof System Programming Language

Axiom 6.1. (Initial Invariance) $\langle\langle p \rangle\rangle \; S \; \langle\langle p \rangle\rangle$

provided p does not refer to *now* or program variables ($loc(p) = \emptyset$).

Axiom 6.2. (Variable Invariance) $\langle\langle p \rangle\rangle \; S \; \langle\langle now < \infty \rightarrow p \rangle\rangle$

provided *now* does not occur in p and $loc(p) \cap loc(S) = \emptyset$.

Axiom 6.3. (Observables Invariance)

$$\langle\langle now = t_0 \rangle\rangle \; S \; \langle\langle NoAct(oset) \ \mathbf{during} \ [t_0, now) \rangle\rangle$$

provided *oset* is a finite set of observables with $oset \cap obs(S) = \emptyset$.

Proof System Programming Language

Axiom 6.4. (Nontermination) $\langle\langle p \wedge now = \infty \rangle\rangle \ S \ \langle\langle p \wedge now = \infty \rangle\rangle$

Rule 6.1. (Substitution)

$$\frac{\langle\langle p \rangle\rangle \ S \ \langle\langle q \rangle\rangle}{\langle\langle p[exp/t] \rangle\rangle \ S \ \langle\langle q \rangle\rangle}$$

provided t does not occur free in q.

Rule 6.2. (Conjunction)

$$\frac{\langle\langle p_1 \rangle\rangle \ S \ \langle\langle q_1 \rangle\rangle, \langle\langle p_2 \rangle\rangle \ S \ \langle\langle q_2 \rangle\rangle}{\langle\langle p_1 \wedge p_2 \rangle\rangle \ S \ \langle\langle q_1 \wedge q_2 \rangle\rangle}$$

Rule 6.3. (Disjunction)

$$\frac{\langle\langle p_1 \rangle\rangle \ S \ \langle\langle q_1 \rangle\rangle, \langle\langle p_2 \rangle\rangle \ S \ \langle\langle q_2 \rangle\rangle}{\langle\langle p_1 \vee p_2 \rangle\rangle \ S \ \langle\langle q_1 \vee q_2 \rangle\rangle}$$

Axiomatization of Programming Constructs

Axiom 6.5. (Skip) $\langle\langle p \rangle\rangle$ **skip** $\langle\langle p \rangle\rangle$

Axiom 6.6. (Assignment) $\langle\langle q[e/x, now + T_a/now] \wedge now < \infty \rangle\rangle\ x := e\ \langle\langle q \rangle\rangle$

Axiom 6.7. (Delay) $\langle\langle q[now + max(0, e)/now] \wedge now < \infty \rangle\rangle$ **delay** $e\ \langle\langle q \rangle\rangle$

Rule 6.4. (Output)

Send immediately

$$\frac{\boxed{\begin{array}{l}(p \wedge now < \infty)[t_0/now] \wedge (c!!, e) \textbf{ at } t_0 \wedge (\neg c!!) \textbf{ during } (t_0, now) \wedge \\ \quad now = t_0 + T_{comm} \rightarrow q\end{array}}}{\langle\langle p \wedge now < \infty \rangle\rangle\ c!!e\ \langle\langle q \rangle\rangle}$$

Axiomatization of Programming Constructs

$$comm(c, v)(t_0, t) \equiv c? \textbf{ during } [t_0, t) \wedge (c, v) \textbf{ at } t \wedge (\neg c? \wedge \neg c) \textbf{ during } (t, now).$$

Rule 6.5. (Input)

$$\frac{\begin{array}{l}(p \wedge now < \infty)[t_0/now] \wedge c? \textbf{ during } [t_0, \infty) \wedge now = \infty \rightarrow q_{nt} \\ (p \wedge now < \infty)[t_0/now] \wedge \exists t \in [t_0, \infty) : comm(c, v)(t_0, t) \wedge now = t + T_{comm} \\ \qquad \rightarrow q[v/x]\end{array}}{\langle\langle p \wedge now < \infty \rangle\rangle \; c?x \; \langle\langle q_{nt} \vee q \rangle\rangle}$$

provided $loc(q_{nt}) = \emptyset$.

Rule 6.6. (Sequential Composition) $$\frac{\langle\langle p \rangle\rangle \; S_1 \; \langle\langle r \rangle\rangle, \quad \langle\langle r \rangle\rangle \; S_2 \; \langle\langle q \rangle\rangle}{\langle\langle p \rangle\rangle \; S_1 ; S_2 \; \langle\langle q \rangle\rangle}$$

Axiomatization of Programming Constructs

Rule 6.7. (Choice)

$$\frac{\langle\langle p \wedge b \rangle\rangle\ S_1\ \langle\langle q \rangle\rangle, \quad \langle\langle p \wedge \neg b \rangle\rangle\ S_2\ \langle\langle q \rangle\rangle}{\langle\langle p \rangle\rangle\ \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\ \langle\langle q \rangle\rangle}$$

Rule 6.8. (Select)

$$\frac{\begin{array}{l}(p \wedge now < \infty)[t_0/now] \wedge \exists t \in [t_0, t_0 + e) : comm(c, v)(t_0, t) \wedge \\ \qquad now = t + T_{comm} \rightarrow p_1[v/x] \\ (p \wedge now < \infty)[t_0/now] \wedge c?\ \textbf{during}\ [t_0, t_0 + e) \wedge now = t_0 + max(0, e) \rightarrow p_2 \\ \langle\langle p_i \rangle\rangle\ S_i\ \langle\langle q_i \rangle\rangle, \text{for } i = 1, 2\end{array}}{\langle\langle p \wedge now < \infty \rangle\rangle\ \textbf{sel}\ c?x\ \textbf{then}\ S_1\ \textbf{or delay}\ e\ \textbf{then}\ S_2\ \textbf{les}\ \langle\langle q_1 \vee q_2 \rangle\rangle}$$

Axiomatization of Programming Constructs

Rule 6.9. (While)

$$\frac{\begin{array}{l}\langle\langle I \wedge b \wedge now < \infty\rangle\rangle \ \mathbf{delay}\ T_w\ ;\ S\ \langle\langle I\rangle\rangle \\ \langle\langle I \wedge \neg b \wedge now < \infty\rangle\rangle \ \mathbf{delay}\ T_w\ \langle\langle q\rangle\rangle \\ I \rightarrow I_0, \quad loc(I_0) = \emptyset \\ (\forall t_1 < \infty\, \exists t_2 > t_1 : I_0[t_2/now]) \rightarrow q_{nt}\end{array}}{\langle\langle I\rangle\rangle\ \mathbf{while}\ b\ \mathbf{do}\ S\ \mathbf{od}\ \langle\langle (q_{nt} \wedge now = \infty) \vee q\rangle\rangle}$$

While

Axiom 6.4. (Nontermination) $\langle\langle p \wedge now = \infty \rangle\rangle \ S \ \langle\langle p \wedge now = \infty \rangle\rangle$

- The initial model, which satisfies I, is nonterminating, i.e., has a state σ_0 with $\sigma_0(now) = \infty$. Then a model of **while** b **do** S **od** by definition equals this model (this property is represented by the Nontermination Axiom). Since $now = \infty$ and $I \rightarrow I_0$ hold in this model, it satisfies $\forall t_1 < \infty \, \exists t_2 > t_1 : I_0[t_2/now]$, and then the third condition leads to q_{nt}. Thus the model satisfies $q_{nt} \wedge now = \infty$.

- It represents a terminating computation, obtained from a finite number of terminating computations of S. For all these computations of S, except for the last one, b is true initially.

While

- It represents a nonterminating computation obtained from a nonterminating computation of S. Then, as in the previous case, we have $I \wedge b$ in the initial state of this last computation. Thus, using the first condition and the fact that it is a nonterminating computation, $I \wedge now = \infty$ holds for this model. Hence, since $I \rightarrow I_0$, we obtain $\forall t_1 < \infty\, \exists t_2 > t_1 : I_0[t_2/now]$, and then the third condition leads to q_{nt}.
- It represents a nonterminating computation obtained from an infinite sequence of terminating computations of S.

$$\textbf{while } x \neq 0 \textbf{ do } in?x \; ; \; out!!f(x) \textbf{ od}$$

$$\langle\!\langle now = 0 \wedge x \neq 0 \rangle\!\rangle$$

$$\textbf{while } x \neq 0 \textbf{ do } in?x \; ; \; out!!f(x) \textbf{ od}$$

$$\langle\!\langle\ (now = \infty \wedge \exists t < \infty : in? \textbf{ during } [t, \infty)) \vee (now = \infty \wedge \forall t < \infty : \neg(in, 0) \textbf{ at } t) \vee$$
$$(now < \infty \wedge \exists t < \infty : (in, 0) \textbf{ at } t) \rangle\!\rangle.$$

We use the iteration rule with

$$q_{nt} \equiv (\exists t < \infty : in? \textbf{ during } [t, \infty)) \vee (\forall t < \infty : \neg(in, 0) \textbf{ at } t)$$
$$q \equiv now < \infty \wedge \exists t < \infty : (in, 0) \textbf{ at } t$$
$$I \equiv (now = \infty \wedge \exists t < \infty : in? \textbf{ during } [t, \infty)) \vee$$
$$(now < \infty \wedge \forall t < now, t \neq now - 2T_{comm} : \neg(in, 0) \textbf{ at } t \wedge$$
$$(x = 0 \leftrightarrow (in, 0) \textbf{ at } now - 2T_{comm}))$$
$$I_0 \equiv (\exists t < \infty : in? \textbf{ during } [t, \infty)) \vee (\forall t < now - 2T_{comm} : \neg(in, 0) \textbf{ at } t)$$

Rule 6.9. (While)

$$\frac{\begin{array}{l}\langle\langle I \wedge b \wedge now < \infty\rangle\rangle \ \mathbf{delay}\ T_w\ ;\ S\ \langle\langle I\rangle\rangle \\ \langle\langle I \wedge \neg b \wedge now < \infty\rangle\rangle \ \mathbf{delay}\ T_w\ \langle\langle q\rangle\rangle \\ I \rightarrow I_0, \quad loc(I_0) = \varnothing \\ (\forall t_1 < \infty\ \exists t_2 > t_1 : I_0[t_2/now]) \rightarrow q_{nt}\end{array}}{\langle\langle I\rangle\rangle\ \mathbf{while}\ b\ \mathbf{do}\ S\ \mathbf{od}\ \langle\langle (q_{nt} \wedge now = \infty) \vee q\rangle\rangle}$$

- $\langle\langle I \wedge x \neq 0 \wedge now < \infty\rangle\rangle$ **delay** T_w ; $in?x$; $out!!f(x)$ $\langle\langle I\rangle\rangle$.
- $\langle\langle I \wedge x = 0 \wedge now < \infty\rangle\rangle$ **delay** T_w $\langle\langle q\rangle\rangle$.
- $I \rightarrow I_0$, which holds trivially. Further note that $loc(I_0) = \varnothing$.
- $(\forall t_1 < \infty\ \exists t_2 > t_1 : I_0[t_2/now]) \rightarrow q_{nt}$.
  Observe that $\forall t_1 < \infty\ \exists t_2 > t_1 : I_0[t_2/now]$ is equivalent to $\forall t_1 < \infty\ \exists t_2 > t_1$:

$(\exists t < \infty : in?$ **during** $[t, \infty)) \vee (\forall t < t_2 - 2T_{comm} : \neg(in, 0)$ **at** $t)$, which implies
$(\exists t < \infty : in?$ **during** $[t, \infty)) \vee (\forall t < \infty : \neg(in, 0)$ **at** $t)$, i.e., $\boxed{q_{nt}.}$

$\langle\langle I\rangle\rangle$ **while** $x \neq 0$ **do** $in?x$; $out!!f(x)$ **od** $\langle\langle (q_{nt} \wedge now = \infty) \vee q\rangle\rangle$.

Note that $now = 0 \wedge x \neq 0 \rightarrow I$. Further, $(q_{nt} \wedge now = \infty) \vee q$ is equivalent to
$((\exists t < \infty : in?$ **during** $[t, \infty \vee \forall t < \infty : \neg(in, 0)$ **at** $t) \wedge now = \infty) \vee$
$(now < \infty \wedge \exists t < \infty : (in, 0)$ **at** $t)$ which implies
$(now = \infty \wedge \exists t < \infty : in?$ **during** $[t, \infty)) \vee (now = \infty \wedge \forall t < \infty : \neg(in, 0)$ **at** $t) \vee$
$(now < \infty \wedge \exists t < \infty : (in, 0)$ **at** $t)$.

$$\langle\!\langle now = 0 \wedge x \neq 0 \rangle\!\rangle$$
$$\textbf{while } x \neq 0 \textbf{ do } in?x \; ; \; out!!f(x) \textbf{ od}$$
$$\langle\!\langle \; (now = \infty \wedge \exists t < \infty : in? \textbf{ during } [t, \infty)) \vee (now = \infty \wedge \forall t < \infty : \neg(in, 0) \textbf{ at } t) \vee$$
$$(now < \infty \wedge \exists t < \infty : (in, 0) \textbf{ at } t) \rangle\!\rangle.$$

Thanks!