Complement

1 Trace-based HHL

1.1 Trace Synchronization

The full definition of trace synchronization function is defined as following:

$$\frac{ch \in cs \quad tr_1 \|_{cs} tr_2 \Downarrow tr}{\langle ch!, v\rangle^\frown tr_1 \|_{cs} \langle ch?, v\rangle^\frown tr_2 \Downarrow \langle ch, v\rangle^\frown tr}\;\text{SyncIO}$$

$$\frac{ch \notin cs \quad tr_1 \|_{cs} tr_2 \Downarrow tr}{\langle ch\triangleright, v\rangle^\frown tr_1 \|_{cs} tr_2 \Downarrow \langle ch\triangleright, v\rangle^\frown tr}\;\text{NoSyncIO} \qquad \frac{ch \in cs}{\langle ch\triangleright, v\rangle^\frown tr_1 \|_{cs} \epsilon \Downarrow \delta}\;\text{SyncEmpty1}$$

$$\frac{tr_1 \|_{cs} \epsilon \Downarrow tr}{\langle d, \overrightarrow{p}_1, rdy_1\rangle^\frown tr_1 \|_{cs} \epsilon \Downarrow \delta}\;\text{SyncEmpty2} \qquad \frac{}{\epsilon \|_{cs} \epsilon \Downarrow \epsilon}\;\text{SyncEmpty3}$$

$$\frac{tr_1 \|_{cs} tr_2 \Downarrow tr \quad \text{compat}(rdy_1, rdy_2) \quad d > 0}{\begin{array}{c}\langle d, \overrightarrow{p}_1, rdy_1\rangle^\frown tr_1 \|_{cs} \langle d, \overrightarrow{p}_2, rdy_2\rangle^\frown tr_2 \Downarrow \\ \langle d, \overrightarrow{p}_1 \uplus \overrightarrow{p}_2, (rdy_1 \cup rdy_2) - cs\rangle^\frown tr\end{array}}\;\text{SyncWait1}$$

$$\frac{\begin{array}{c}d_1 > d_2 > 0 \quad \text{compat}(rdy_1, rdy_2) \\ \langle d_1 - d_2, \overrightarrow{p}_1(\cdot + d_2), rdy_1\rangle^\frown tr_1 \|_{cs} tr_2 \Downarrow tr\end{array}}{\begin{array}{c}\langle d_1, \overrightarrow{p}_1, rdy_1\rangle^\frown tr_1 \|_{cs} \langle d_2, \overrightarrow{p}_2, rdy_2\rangle^\frown tr_2 \Downarrow \\ \langle d_2, \overrightarrow{p}_1 \uplus \overrightarrow{p}_2, (rdy_1 \cup rdy_2) - cs\rangle^\frown tr\end{array}}\;\text{SyncWait2}$$

1.2 Big-step Semantics

The full big-step semantics of HCSP process is defined by the following rules:

$$\frac{}{(\text{skip}, s) \Rightarrow (s, \epsilon)}\;\text{SkipB} \qquad \frac{}{(x := e, s) \Rightarrow (s[x \mapsto e], \epsilon)}\;\text{AssignB}$$

$$\frac{}{(ch!e, s) \Rightarrow (s, \langle ch!, s(e)\rangle)}\;\text{OutB1} \qquad \frac{}{(ch!e, s) \Rightarrow (s, \langle d, I_s, \{ch!\}\rangle^\frown \langle ch!, s(e)\rangle)}\;\text{OutB2}$$

$$\frac{}{(ch?x, s) \Rightarrow (s[x \mapsto v], \langle ch?, v\rangle)}\;\text{InB1} \qquad \frac{}{(ch?x, s) \Rightarrow (s[x \mapsto v], \langle d, I_s, \{ch?\}\rangle^\frown \langle ch?, v\rangle)}\;\text{InB2}$$

$$\frac{}{(c^*, s) \Rightarrow (s, \epsilon)}\;\text{RepB1} \qquad \frac{(c, s) \Rightarrow (s_1, tr_1) \quad (c^*, s_1) \Rightarrow (s_2, tr_2)}{(c^*, s) \Rightarrow (s_2, tr_1{}^\frown tr_2)}\;\text{RepB2}$$

$$\frac{}{(\text{wait}e, s) \Rightarrow (s, \langle s(e), I_s, \{\}\rangle)}\;\text{WaitB} \qquad \frac{(c, s_1) \Rightarrow (s_2, tr_1) \quad (c_2, s_2) \Rightarrow (s_3, tr_2)}{(c_1; c_2, s_1) \Rightarrow (s_3, tr_1{}^\frown tr_2)}\;\text{SeqB}$$

$$\frac{s_1(B) \quad (c_1, s_1) \Rightarrow (s_2, \mathit{tr})}{(\text{if } B \text{ then } c_1 \text{ else } c_2, s_1) \Rightarrow (s_2, \mathit{tr})}\ \text{CondB1} \qquad \frac{(c_1, s_1) \Rightarrow (s_2, \mathit{tr})}{(c_1 \sqcup c_2, s_1) \Rightarrow (s_2, \mathit{tr})}\ \text{IChoiceB1}$$

$$\frac{\neg s_1(B) \quad (c_2, s_1) \Rightarrow (s_2, \mathit{tr})}{(\text{if } B \text{ then } c_1 \text{ else } c_2, s_1) \Rightarrow (s_2, \mathit{tr})}\ \text{CondB2} \qquad \frac{(c_2, s_1) \Rightarrow (s_2, \mathit{tr})}{(c_1 \sqcup c_2, s_1) \Rightarrow (s_2, \mathit{tr})}\ \text{IChoiceB2}$$

$$\frac{\neg B(s)}{(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B\rangle, s) \Rightarrow (s, \epsilon)}\ \text{ContB1}$$

$$\frac{\begin{array}{c}\overrightarrow{p} \text{ is a solution of the ODE } \dot{\overrightarrow{x}} = \overrightarrow{e} \\ \overrightarrow{p}(0) = s(\overrightarrow{x}) \quad \forall t \in [0, d).\, s[\overrightarrow{x} \mapsto \overrightarrow{p}(t)](B) \quad \neg s[\overrightarrow{x} \mapsto \overrightarrow{p}(d)](B)\end{array}}{(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B\rangle, s) \Rightarrow (s[\overrightarrow{x} \mapsto \overrightarrow{p}(d)], \langle d, \overrightarrow{p}, \{\}\rangle)}\ \text{ContB2}$$

$$\frac{i \in L \quad ch_i* = ch!e \quad (c_i, s_1) \Rightarrow (s_2, \mathit{tr})}{(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B \propto c\rangle \trianglerighteq []_{i\in L}(ch_i* \rightarrow c_i), s_1) \Rightarrow (s_2, \langle ch!, s_1(e)\rangle^\frown \mathit{tr})}\ \text{IntB1}$$

$$\frac{\begin{array}{c}\overrightarrow{p} \text{ is a solution of the ODE } \dot{\overrightarrow{x}} = \overrightarrow{e} \quad \overrightarrow{p}(0) = s_1(\overrightarrow{x}) \\ \forall t \in [0, d).\, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(t)](B) \\ i \in L \quad ch_i* = ch!e \quad (c_i, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(d)]) \Rightarrow (s_2, \mathit{tr})\end{array}}{\begin{array}{c}(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B \propto c\rangle \trianglerighteq []_{i\in L}(ch_i* \rightarrow c_i), s_1) \Rightarrow \\ (s_2, \langle d, \overrightarrow{p}, rdy(\cup_{i\in L} ch_i*)\rangle^\frown \langle ch!, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(d)](e)\rangle^\frown \mathit{tr})\end{array}}\ \text{IntB2}$$

$$\frac{i \in L \quad ch_i* = ch?y \quad (c_i, s_1[y \mapsto v]) \Rightarrow (s_2, \mathit{tr})}{(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B \propto c\rangle \trianglerighteq []_{i\in L}(ch_i* \rightarrow c_i), s_1) \Rightarrow (s_2, \langle ch?, v\rangle^\frown \mathit{tr})}\ \text{IntB3}$$

$$\frac{\begin{array}{c}\overrightarrow{p} \text{ is a solution of the ODE } \dot{\overrightarrow{x}} = \overrightarrow{e} \quad \overrightarrow{p}(0) = s_1(\overrightarrow{x}) \\ \forall t \in [0, d).\, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(t)](B) \\ i \in L \quad ch_i* = ch?y \quad (c_i, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(d), y \mapsto v]) \Rightarrow (s_2, \mathit{tr})\end{array}}{\begin{array}{c}(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B \propto c\rangle \trianglerighteq []_{i\in L}(ch_i* \rightarrow c_i), s_1) \Rightarrow \\ (s_2, \langle d, \overrightarrow{p}, rdy(\cup_{i\in L} ch_i*)\rangle^\frown \langle ch?, v\rangle^\frown \mathit{tr})\end{array}}\ \text{IntB4}$$

$$\frac{\neg s(B) \quad (c, s_1) \Rightarrow (s_2, \mathit{tr})}{(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B \propto c\rangle \trianglerighteq []_{i\in L}(ch_i* \rightarrow c_i), s1) \Rightarrow (s2, \mathit{tr})}\ \text{IntB5}$$

$$\frac{\begin{array}{c}\overrightarrow{p} \text{ is a solution of the ODE } \dot{\overrightarrow{x}} = \overrightarrow{e} \quad \overrightarrow{p}(0) = s_1(\overrightarrow{x}) \\ \forall t \in [0, d).\, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(t)](B) \quad \neg s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(d)](B) \\ (c, s_1[\overrightarrow{x} \mapsto \overrightarrow{p}(d)]) \Rightarrow (s_2, \mathit{tr})\end{array}}{\begin{array}{c}(\langle \dot{\overrightarrow{x}} = \overrightarrow{e} \& B \propto c\rangle \trianglerighteq []_{i\in L}(ch_i* \rightarrow c_i), s_1) \Rightarrow \\ (s_2, \langle d, \overrightarrow{p}, rdy(\cup_{i\in L} ch_i*)\rangle^\frown \mathit{tr})\end{array}}\ \text{IntB6}$$

$$\frac{(c_1, s_1) \Rightarrow (s_1', tr_1) \quad (c_2, s_2) \Rightarrow (s_2', tr_2) \quad tr_1 \|_{cs} tr_2 \Downarrow \mathit{tr}}{(c_1 \|_{cs} c_2, s_1 \uplus s_2) \Rightarrow (s_1' \uplus s_2', \mathit{tr})}\ \text{ParB}$$

2 Complement Sequential Rules

2.1

In this section, we first explain the rules for interrupt command with explicit solution in detail.

Given an interrupt command $\langle \overrightarrow{x} = \overrightarrow{e} \& B \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i)$, where we use es to denote the list of communications in the form $(ch?x \to c_i)$ or $(ch!e \to c_i)$, and f is a solution to $\overrightarrow{x} = \overrightarrow{e}$, the branches of assertions corresponding to the communication list is computed by $\mathsf{rel\_cm}(es, c, f)$, if for each $es[i] = (ch?y \to c_i)$, we have $\mathsf{spec\_of}(c_i; c, Q_i)$ then

$$\mathsf{rel\_cm}(es, c, f)[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow Q_i[y := \mathsf{v}][\overrightarrow{x} := f(s_0(\overrightarrow{x}), \mathsf{d})]\}\rangle$$

and for each $es[i] = (ch!e \to c_i)$, we have $\mathsf{spec\_of}(c_i; c, Q_i)$ then

$$\mathsf{rel\_cm}(es, c, f)[i] = \langle ch!, \{\mathsf{d} \Rightarrow e(p(s_0, \mathsf{d}))\}, \{\mathsf{d} \Rightarrow Q_i[\overrightarrow{x} := f(s_0(\overrightarrow{x}), \mathsf{d})]\}\rangle$$

Then the inference rule for interrupt is:

$$\frac{\begin{array}{c}\mathsf{paramODEsol}(\overrightarrow{x} = \overrightarrow{e}, B, f, e) \qquad \mathsf{lipschitz}(\overrightarrow{x} = \overrightarrow{e}) \\ \mathsf{spec\_of}(c'; c, P) \qquad \forall\, i \in L,\ \mathsf{spec\_of}(c_i; c, Q_i)\end{array}}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{x} = \overrightarrow{e} \& B \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i); c, \mathsf{interrupt}(\overrightarrow{x} \rightarrowtail f(\overrightarrow{x}, t), \\ e, \{\mathsf{d} \Rightarrow P[\overrightarrow{x} := f(s_0(\overrightarrow{x}), \mathsf{d})]\}, \mathsf{rel\_cm}(es, c, f))\end{array}}$$

The meaning of this rule is as follows: the specification of the interrupt first evolves along the path $p(t) = s_0[\overrightarrow{x} \mapsto f(s_0(\overrightarrow{x}), t)]$, and one of the following three situations occurs:

– If the evolution is interrupted by an input communication $(ch?x \to c_i)$ at time d and with value v, then update the state to $s_0[\overrightarrow{x} \mapsto f(s_0(\overrightarrow{x}), d)][x \mapsto v]$, followed by the behavior of $c_i; c$ as specified by Q_i.
– If the evolution is interrupted by an output communication $(ch!e \to c_i)$ at time d and with value $v = e(s_0[\overrightarrow{x} \mapsto f(s_0(\overrightarrow{x}), d)])$, and then update the state to $s_0[\overrightarrow{x} \mapsto f(s_0(\overrightarrow{x}), d)]$, followed by the behavior of $c_i; c$ as specified by Q_i.
– If no interrupt occurs before time $d = s_0(e)$, then update the state to $s_0[\overrightarrow{x} \mapsto f(s_0(\overrightarrow{x}), d)]$, followed by the behavior of $c'; c$ as specified by P.

The above assumes that the ODE with boundary condition has a solution of finite length for any starting state. Another important case is when the ODE has a solution of infinite length, in particular when the boundary condition is true. In this case, the appropriate assertion is $\mathsf{interrupt}_\infty$. We first define predicate $\mathsf{paramODEsolInf}(\overrightarrow{x} = \overrightarrow{e}, f)$, meaning that f is the (infinite length) solution to $\overrightarrow{x} = \overrightarrow{e}$, then the corresponding rule is:

$$\frac{\mathsf{paramODEsolInf}(\overrightarrow{x} = \overrightarrow{e}, \overrightarrow{p}) \quad \mathsf{lipschitz}(\overrightarrow{x} = \overrightarrow{e}) \quad \forall\, i \in L,\ \mathsf{spec\_of}(c_i; c, Q_i)}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{x} = \overrightarrow{e} \& \mathsf{true} \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i); c, \\ \mathsf{interrupt}_\infty(\overrightarrow{x} \rightarrowtail f(\overrightarrow{x}, t), \mathsf{rel\_cm}(es, c, f))\end{array}}$$

Next, we introduce the rules for interrupt with differential invariants.

Similarly, we define the branches of assertions corresponding to the communication list, denoted by $\mathsf{relinv\_cm}(es, c, inv)$, if for each $es[i] = (ch?y \to c_i)$, we have $\mathsf{spec\_of}(c_i; c, Q_i)$ then

$$\mathsf{relinv\_cm}(es, c, inv)[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow (\uparrow inv \wedge Q_i[y := \mathsf{v}])[\overrightarrow{x} := \overrightarrow{nx_i}]\}\rangle$$

and for each $es[i] = (ch!e \to c_i)$, we have $\mathsf{spec\_of}(c_i; c, Q_i)$ then

$$\mathsf{relinv\_cm}(es, c, inv)[i] = \langle ch!, \{\mathsf{d} \Rightarrow e(s_0[\overrightarrow{x} := \overrightarrow{nx_i}])\}, \{\mathsf{d} \Rightarrow (\uparrow inv \wedge Q_i)[\overrightarrow{x} := \overrightarrow{nx_i}]\}\rangle$$

And then, we have the following rule:

$$\frac{\begin{array}{c}\mathsf{paramODEInv}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, inv, pp) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e}) \\ \mathsf{spec\_of}(c'; c, P) \quad \forall\, i \in L,\ \mathsf{spec\_of}(c_i; c, Q_i)\end{array}}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& B \propto c' \rangle \trianglerighteq []_{i\in L}(ch_i{*} \to c_i); c, (\uparrow (\neg B) \bar{\wedge} P) \bar{\vee} \uparrow (\neg pp \wedge B) \bar{\vee} \\ \exists\, T\, \overrightarrow{nx}\, \overrightarrow{nx_i}_{i\in L}.\, (\uparrow (pp \wedge B) \bar{\wedge} \mathsf{interrupt}(inv, T, \\ \{d \Rightarrow (\uparrow (inv \wedge bound(B)) \bar{\wedge} P)[\overrightarrow{x} := \overrightarrow{nx}]\}, \mathsf{relinv\_cm}(es, c, inv))))\end{array}}$$

If the ODE in interrupt command has infinite length, we have:

$$\frac{\mathsf{paramODEInv}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, inv, pp) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e}) \quad \forall\, i \in L,\ \mathsf{spec\_of}(c_i; c, Q_i)}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& \mathsf{true} \propto c' \rangle \trianglerighteq []_{i\in L}(ch_i{*} \to c_i); c, \uparrow (\neg pp) \bar{\vee} \\ \exists\, \overrightarrow{nx_i}_{i\in L}.\, (\uparrow pp \bar{\wedge} \mathsf{interrupt}_\infty(inv, \mathsf{relinv\_cm}(es, c, inv))))\end{array}}$$

2.2

In this section we give the all the sequential rules without subsequent process.

$$\frac{}{\mathsf{spec\_of}(x := e, \mathsf{init}[x := e])}$$

$$\frac{\mathsf{spec\_of}(c_1, P) \quad \mathsf{spec\_of}(c_2, Q)}{\mathsf{spec\_of}(\text{if } B \text{ then } c_1 \text{ else } c_2, (\uparrow (B) \bar{\wedge} P) \bar{\vee} (\uparrow (\neg B) \bar{\wedge} Q))}$$

$$\frac{}{\mathsf{spec\_of}(ch?x, \mathsf{wait\_in}(\mathsf{id\_inv}, ch, \{\mathsf{d}, \mathsf{v} \Rightarrow \mathsf{init}[x := \mathsf{v}]\}))}$$

$$\frac{}{\mathsf{spec\_of}(ch!e, \mathsf{wait\_outv}(\mathsf{id\_inv}, ch, e, \{\mathsf{d} \Rightarrow \mathsf{init}\}))}$$

$$\frac{}{\mathsf{spec\_of}(\text{wait } e, \mathsf{wait}(\mathsf{id}, e, \{\mathsf{d} \Rightarrow \mathsf{init}\}))}$$

$$\frac{\mathsf{paramODEsol}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, f, e) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e})}{\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& B \rangle, \mathsf{wait}(\overrightarrow{x} \rightarrowtail f(\overrightarrow{x}, t), e, \{\mathsf{d} \Rightarrow \mathsf{init}[\overrightarrow{x} := f(s_0(\overrightarrow{x}), \mathsf{d})]\}))}$$

$$\frac{\forall\, d\, Q.\ \mathsf{spec\_of}(d, Q) \longrightarrow \mathsf{spec\_of}(c; d, F(Q))}{\mathsf{spec\_of}(c^*, \mathsf{Rec}\ R.\ \mathsf{init} \bar{\vee} F(R))}$$

$$\frac{\begin{array}{c}\mathsf{paramODEsol}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, f, e) \qquad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e}) \\ \mathsf{spec\_of}(c', P) \qquad \forall\, i \in L.\, \mathsf{spec\_of}(c_i, Q_i)\end{array}}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& B \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i), \mathsf{interrupt}(\overrightarrow{x} \rightarrowtail f(\overrightarrow{x}, t), \\ e, \{\mathsf{d} \Rightarrow \mathsf{P}[\overrightarrow{x} := f(s_0(\overrightarrow{x}), \mathsf{d})]\}, \mathsf{rel\_cm}(es, \text{skip}, f))\end{array}}$$

$$\frac{\mathsf{paramODEsolInf}(\overrightarrow{\dot{x}} = \overrightarrow{e}, f) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e})}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& true \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i), \mathsf{interrupt}_\infty(\overrightarrow{x} \rightarrowtail f(\overrightarrow{x}, t), \\ \mathsf{rel\_cm}(es, \text{skip}, f))\end{array}}$$

$$\frac{\mathsf{paramODEInv}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, inv, pp) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e})}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& B \rangle; c, (\uparrow (\neg B) \bar{\wedge} \mathsf{init}) \bar{\vee} \uparrow (\neg pp \wedge B) \bar{\vee} \\ \exists\, T\, \overrightarrow{nx}.\ (\uparrow (pp \wedge B) \bar{\wedge} \mathsf{wait}(inv, T, \{\mathsf{d} \Rightarrow (\uparrow (inv \wedge bound(B)) \bar{\wedge} \mathsf{init})[\overrightarrow{x} := \overrightarrow{nx}]\})))\end{array}}$$

$$\frac{\begin{array}{c}\mathsf{paramODEInv}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, inv, pp) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e}) \\ \mathsf{spec\_of}(c', P) \quad \forall\, i \in L,\ \mathsf{spec\_of}(c_i, Q_i)\end{array}}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& B \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i), (\uparrow (\neg B) \bar{\wedge} P) \bar{\vee} \uparrow (\neg pp \wedge B) \bar{\vee} \\ \exists\, T\, \overrightarrow{nx}\, \overrightarrow{nx_i}_{i \in L}.\ (\uparrow (pp \wedge B) \bar{\wedge} \mathsf{interrupt}(inv, T, \\ \{\mathsf{d} \Rightarrow (\uparrow (inv \wedge bound(B)) \bar{\wedge} P)[\overrightarrow{x} := \overrightarrow{nx}]\}, \mathsf{relinv\_cm}(es, \text{skip}, inv))))\end{array}}$$

$$\frac{\mathsf{paramODEInv}(\overrightarrow{\dot{x}} = \overrightarrow{e}, B, inv, pp) \quad \mathsf{lipschitz}(\overrightarrow{\dot{x}} = \overrightarrow{e}) \quad \forall\, i \in L,\ \mathsf{spec\_of}(c_i, Q_i)}{\begin{array}{c}\mathsf{spec\_of}(\langle \overrightarrow{\dot{x}} = \overrightarrow{e} \& \mathsf{true} \propto c' \rangle \trianglerighteq []_{i \in L}(ch_i* \to c_i), \uparrow (\neg pp) \bar{\vee} \\ \exists\, \overrightarrow{nx_i}_{i \in L}.\ (\uparrow pp \bar{\wedge} \mathsf{interrupt}_\infty(inv, \mathsf{relinv\_cm}(es, c, inv))))\end{array}}$$

3 Complement Synchronization Rules

In this section we show the other synchronization rules.

First, we introduce the rules involving the common operators of assertions.

$$\frac{}{\mathsf{sync}(chs, \mathsf{false}, P)(s_0) \Longrightarrow_a \mathsf{false}(s_0)}\ \text{False}$$

if one side is a false assertion, we obtain a result of false.

$$\frac{\mathsf{sync}(chs, P_1, Q)(s_0) \Longrightarrow_a R_1(s_0) \quad \mathsf{sync}(chs, P_2, Q)(s_0) \Longrightarrow_a R_2(s_0)}{\mathsf{sync}(chs, P_1 \bar{\vee} P_2, Q)(s_0) \Longrightarrow_a (R_1 \bar{\vee} R_2)(s_0)}\ \text{Disj}$$

if one side is a disjunction, we can eliminate this to its components.

$$\frac{b(s_1) \longrightarrow \mathsf{sync}(chs, P, Q)(s_0) \Longrightarrow_a R(s_0)}{\mathsf{sync}(chs, \uparrow b \bar{\wedge} P, Q)(s_0) \Longrightarrow_a (\uparrow b \bar{\wedge} R)(s_0)}\ \text{Bool}$$

if one side is a conjunction with a boolean expression b, we perform synchronization on the rest part under b and pull out b lifted on a parallel state as a new condition.

$$\frac{}{\mathsf{sync}(chs, P[x := e], Q)(s_0) \Longrightarrow_a \mathsf{sync}(chs, P, Q)[x := e](s_0)}\ \text{Subst}$$

if one side is a substitution assertion, the substitution can be pulled out after lifting.

In principle, wait_out, wait_in and wait are all special cases of interrupt (including $\mathsf{interrupt}_\infty$, by viewing $\mathsf{interrupt}_\infty$(I,cm) as $\mathsf{interrupt}(I, \infty, \{\mathsf{d} \Rightarrow \mathsf{false}\}, cm)$). Thus, the synchronization rule for interrupt assertion is complex and contains all the potential situations. We will first give some simple cases, and then introduce the rule for interrupt as a complete form.

While synchronizing two init assertions, we can easily infer that the state of each part remains the same and the traces on both sides are empty lists. Naturally, we have

$$\frac{}{\mathsf{sync}(chs, \mathsf{init}, \mathsf{init})(s_0) \Longrightarrow_a \mathsf{init}(s_0)}\ \text{InitInit}$$

While synchronizing an init assertion and an wait assertion, if the wait time is greater than 0, we directly obtain a false assertion. Otherwise, if the wait time is Less than or equal to 0, the wait assertion turns to its tail by the definition.

$$\frac{}{\mathsf{sync}(chs, \mathsf{wait}(I, e, \{\mathsf{d} \Rightarrow P\}), \mathsf{init})(s_0) \Longrightarrow_a \uparrow (e \leq 0) \bar{\wedge} \mathsf{sync}(chs, P|_{\mathsf{d}=0}, \mathsf{init})(s_0)}\ \text{WaitInit}$$

While synchronizing an init assertion and an input assertion, if the communication channel belongs to the common channel set, we directly obtain a false assertion, Otherwise, this external communication must occur at once, since the init assertion does not support any waiting time. Thus, we have:

$$\frac{ch \in chs}{\mathsf{sync}(chs, \mathsf{wait\_in}(I, ch, \{\mathsf{d} \Rightarrow P\}), \mathsf{init})(s_0) \Longrightarrow_a \mathsf{false}(s_0)}\ \text{InInit1}$$

$$\frac{ch \notin chs}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{wait\_in}(I, ch, \{\mathsf{d}, \mathsf{v} \Rightarrow P\}), \mathsf{init})(s_0) \Longrightarrow_a \mathsf{interrupt}(I \uplus \mathsf{id}, 0, \\ \{\mathsf{d} \Rightarrow \mathsf{false}\}, [\langle ch?, \mathsf{d}, \mathsf{v} \Rightarrow \mathsf{sync}(chs, P, \mathsf{init})\rangle])(s_0)\end{array}}\ \text{InInit2}$$

The rules for synchronizing an init assertion and an output assertion are similar.

While synchronizing an output assertion and an input assertion, we need to consider the different cases of whether their channels belong to the common channel set. If they are both in the set and have the same name, then the handshake occurs at once. while if they have different names which means both sides are waiting for a handshake, but they don't match and this lead to a deadlock represented by a false assertion. So we have the following rules:

$$\frac{ch_1 \in chs \quad ch_2 \in chs \quad ch_1 = ch_2}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{wait\_in}(I_1, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1\}), \mathsf{wait\_outv}(I_2, ch_2, e, \{\mathsf{d}_2 \Rightarrow P_2\}))(s_0) \\ \Longrightarrow_a \mathsf{sync}(chs, P_1|_{\mathsf{d}_1=0,\mathsf{v}_1=s_0(e)}, P_2|_{\mathsf{d}_2=0})(s_0)\end{array}}\ \text{InOut1}$$

$$\frac{ch_1 \in chs \quad ch_2 \in chs \quad ch_1 \neq ch_2}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{wait\_in}(I_1, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1\}), \mathsf{wait\_outv}(I_2, ch_2, e, \{\mathsf{d}_2 \Rightarrow P_2\}))(s_0) \\ \Longrightarrow_a \mathsf{false}(s_0)\end{array}}\ \text{InOut2}$$

If at least one of them is an external communication, then it must happen before the internal communication, because the condition for the internal handshake to occur are not met. Thus, we have:

$$\frac{ch_1 \in chs \quad ch_2 \notin chs}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{wait\_in}(I_1, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1\}), \mathsf{wait\_outv}(I_2, ch_2, e, \{\mathsf{d}_2 \Rightarrow P_2\}))(s_0) \\ \Longrightarrow_a \mathsf{wait\_outv}(I_1 \uplus I_2, ch_2, e, \{\mathsf{d}_2 \Rightarrow \\ \mathsf{sync}(chs, \mathsf{wait\_in}(I_1|_{t=t+\mathsf{d}_2}, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1|_{\mathsf{d}_1=\mathsf{d}_1+\mathsf{d}_2}\}), P_2))\})(s_0)\end{array}}\ \text{InOut3}$$

$$\frac{ch_1 \notin chs \quad ch_2 \in chs}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{wait\_in}(I_1, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1\}), \mathsf{wait\_outv}(I_2, ch_2, e\{\mathsf{d}_2 \Rightarrow P_2\}))(s_0) \\ \Longrightarrow_a \mathsf{wait\_in}(I_1 \uplus I_2, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow \\ \mathsf{sync}(chs, P_1, \mathsf{wait\_outv}(I_2|_{t=t+\mathsf{d}_1}, ch_2, e, \{\mathsf{d}_2 \Rightarrow P_2|_{\mathsf{d}_2=\mathsf{d}_2+\mathsf{d}_1}\}))\})(s_0)\end{array}}\ \text{InOut4}$$

$$\frac{ch_1 \notin chs \quad ch_2 \notin chs}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{wait\_in}(I_1, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1\}), \mathsf{wait\_outv}(I_2, ch_2, e, \{\mathsf{d}_2 \Rightarrow P_2\}))(s_0) \\ \Longrightarrow_a \mathsf{interrupt}_\infty(I_1 \uplus I_2, [\langle ch_1?, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow \mathsf{sync}(chs, \\ P_1, \mathsf{wait\_outv}(I_2|_{:=t+\mathsf{d}_1}, ch_2, e, \{\mathsf{d}_2 \Rightarrow P_2|_{\mathsf{d}_1=\mathsf{d}_1+\mathsf{d}_2}\}))\}\rangle, \\ \langle ch2!, \{\mathsf{d}_2 \Rightarrow e\}, \{\mathsf{d}_2 \Rightarrow \mathsf{sync}(chs, \\ \mathsf{wait\_in}(I_1|_{t=t+\mathsf{d}_2}, ch_1, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow P_1|_{\mathsf{d}_1=\mathsf{d}_1+\mathsf{d}_2}\}), P_2)\}\rangle])(s_0)\end{array}}\ \text{InOut5}$$

Next, we consider synchronizing two interrupt assertions $\mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1)$ and $\mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2)$. First, we need to determine whether there is a communication between two sides. The method of judgement is to check if there exists a channel name in the set chs, where its input is in the rdy set on one side and its output is in the rdy set on the other side. Define predicate compat to be the negation of this condition:

$$\begin{aligned}\mathsf{compat}(rdy(cm_1), rdy(cm_2)) \triangleq \neg\ (&\exists ch \in chs.(ch! \in rdy(cm_1) \wedge ch? \in rdy(cm_2)) \\ &\vee(ch? \in rdy(cm_1) \wedge ch! \in rdy(cm_2)))\end{aligned}$$

In the case where this predicate holds true, both sides are waiting to be interrupted by external communication, thus its synchronization result should still be in the form of interrupt assertion, and its maximum waiting time is the smaller of e_1 and e_2. While reaching the maximum waiting time, the shorter one will behave as the tail part and the longer one stays in an incomplete interrupt assertion denoted as $\mathsf{delay}(h, \mathsf{interrupt}(I, e, \{\mathsf{d} \Rightarrow P\}, cm))$:

$$\begin{aligned}\mathsf{delay}(h, \mathsf{interrupt}(I, e, \{\mathsf{d} \Rightarrow P\}, cm)) \triangleq\ & \mathsf{interrupt}(I|_{t=t+h}, e - h, \\ & \{\mathsf{d} \Rightarrow P|_{\mathsf{d}=\mathsf{d}+h}\}, \mathsf{delay\_cm}(cm, h))\end{aligned}$$

where for input $cm[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow Q_1\}\rangle$ or output $cm[i] = \langle ch!, g, \{\mathsf{d} \Rightarrow Q_2\}\rangle$, we have:

$$\mathsf{delay\_cm}(cm, h)[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow Q_1|_{\mathsf{d}=\mathsf{d}+h}\}\rangle$$
$$\mathsf{delay\_cm}(cm, h)[i] = \langle ch!, \{\mathsf{d} \Rightarrow g(\mathsf{d}+h)\}, \{\mathsf{d} \Rightarrow Q_2|_{\mathsf{d}=\mathsf{d}+h}\}\rangle$$

we can easily find that $\mathsf{delay}(0, \mathsf{interrupt}(I, e, \{\mathsf{d} \Rightarrow P\}, cm)) = \mathsf{interrupt}(I, e, \{\mathsf{d} \Rightarrow P\}, cm)$. By performing synchronization on them, we get the new tail assertion. A potential external interruption from cm_1 or cm_2 that does not belong to the shared set chs may occur during the waiting. Then, one side will behave as the corresponding assertion recorded in cm_1 or cm_2, the other side will remain its incomplete interrupt assertion. For this case, the synchronization produces the new communication list composed of two parts: $\mathsf{rel1}(cm_1|_{chs^c}, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2))$ and $\mathsf{rel2}(cm_2|_{chs^c}, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1))$ where $cm_1|_{chs^c}$ and $cm_2|_{chs^c}$ are lists of communications not in chs extracted from cm_1 and cm_2. The list functions rel1 and rel2 are set as: if $cm[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow Q_1\}\rangle$,

$$\mathsf{rel1}(cm, R)[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow \mathsf{sync}(chs, Q_1, \mathsf{delay}(\mathsf{d}, R))\}$$
$$\mathsf{rel2}(cm, R)[i] = \langle ch?, \{\mathsf{d}, \mathsf{v} \Rightarrow \mathsf{sync}(chs, \mathsf{delay}(\mathsf{d}, R), Q_1)\}$$

if $cm[i] = \langle ch!, g, \{\mathsf{d} \Rightarrow Q_2\}\rangle$,

$$\mathsf{rel1}(cm, R)[i] = \langle ch!, \{\mathsf{d} \Rightarrow g(\mathsf{d})\}, \{\mathsf{d} \Rightarrow \mathsf{sync}(chs, Q_2, \mathsf{delay}(\mathsf{d}, R))\}$$
$$\mathsf{rel2}(cm, R)[i] = \langle ch!, \{\mathsf{d} \Rightarrow g(\mathsf{d})\}, \{\mathsf{d} \Rightarrow \mathsf{sync}(chs, \mathsf{delay}(\mathsf{d}, R), Q_2)\}$$

So far we can obtain the following rules:

$$\frac{e1(s_1) < e2(s_2) \wedge e_2(s_2) > 0 \quad \mathsf{compat}(rdy(cm_1), rdy(cm_2))}{\begin{gathered}\mathsf{sync}(chs, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1), \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2))\\ (s_1 \uplus s_2) \Longrightarrow_a \mathsf{interrupt}(I_1 \uplus I_2, e_1, \{\mathsf{d}_1 \Rightarrow\\ \mathsf{sync}(chs, P_1, \mathsf{delay}(\mathsf{d}_1, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2)))\},\\ \mathsf{rel1}(cm_1|_{chs^c}, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2))@\\ \mathsf{rel2}(cm_2|_{chs^c}, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1)))(s_1 \uplus s_2)\end{gathered}} \text{IntInt1}$$

$$\frac{e1(s_1) = e2(s_2) \vee (e_1(s_1) \le 0 \wedge e_2(s_2) \le 0) \quad \mathsf{compat}(rdy(cm_1), rdy(cm_2))}{\begin{gathered}\mathsf{sync}(chs, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1), \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2))\\ (s_1 \uplus s_2) \Longrightarrow_a \mathsf{interrupt}(I_1 \uplus I_2, e_1, \{\mathsf{d} \Rightarrow\\ \mathsf{sync}(chs, P_1|_{\mathsf{d}_1=\mathsf{d}}, \mathsf{delay}(\mathsf{d}, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2)))\\ \bar{\vee}\ \mathsf{sync}(chs, \mathsf{delay}(\mathsf{d}, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1)), P_2|_{\mathsf{d}_2=\mathsf{d}})\},\\ \mathsf{rel1}(cm_1|_{chs^c}, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2))@\\ \mathsf{rel2}(cm_2|_{chs^c}, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1)))(s_1 \uplus s_2)\end{gathered}} \text{IntInt2}$$

Note that in the definition of interrupt assertion, if the expression of waiting time calculated as a negative value then it has equivalent meaning with 0. That is why we need to compare the expression with 0.

In the case when the compat function is false, there are three possible scenarios. The first is nondeterministicly executing one of the possible handshakes among all that could occur which we represent as $\mathsf{comm}(cm_1, cm_2)$. It is a disjunction of $\mathsf{sync}(chs, Q_1|_{\mathsf{d}_1=0,\mathsf{v}_1=g(0)}, Q_2|_{\mathsf{d}_2=0})$ and $\mathsf{sync}(chs, Q_1|_{\mathsf{d}_1=0}, Q_2|_{\mathsf{d}_2=0,\mathsf{v}_2=g(0)}))$ for all the pairs satisfying one of the following conditions:

$$ch \in chs \wedge cm_1[i] = \langle ch?, \{\mathsf{d}_1, \mathsf{v}_1 \Rightarrow Q_1\}\rangle \wedge cm_2[j] = \langle ch!, g, \{\mathsf{d}_2 \Rightarrow Q_2\}\rangle$$
$$ch \in chs \wedge cm_1[i] = \langle ch!, g, \{\mathsf{d}_1 \Rightarrow Q_1\}\rangle \wedge cm_2[j] = \langle ch?, \{\mathsf{d}_2, \mathsf{v}_2 \Rightarrow Q_2\}\rangle$$

The second is that if the maximum waiting time e_1 or e_2 is less than 0, then the corresponding side may immediately transit to the tail assertion. The last one is there is an external interrupt occurring at time 0. We obtain the following rule:

$$\frac{\neg\mathsf{compat}(rdy(cm_1), rdy(cm_2))}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1), \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2)) \\ (s_1 \uplus s_2) \Longrightarrow_a \mathsf{interrupt}(I_1 \uplus I_2, 0, \{\mathsf{d} \Rightarrow \mathsf{comm}(cm_1, cm_2)\bar{\vee} \\ (\uparrow (e_1 \leq 0)\bar{\wedge}\mathsf{sync}(chs, P_1|_{\mathsf{d}_1=0}, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2)))\bar{\vee} \\ (\uparrow (e_2 \leq 0)\bar{\wedge}\mathsf{sync}(chs, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1), P_2|_{\mathsf{d}_2=0}))\}, \\ \mathsf{rel1}(cm_1|_{chs^c}, \mathsf{interrupt}(I_2, e_2, \{\mathsf{d}_2 \Rightarrow P_2\}, cm_2))@ \\ \mathsf{rel2}(cm_2|_{chs^c}, \mathsf{interrupt}(I_1, e_1, \{\mathsf{d}_1 \Rightarrow P_1\}, cm_1)))(s_1 \uplus s_2)\end{array}} \text{IntInt3}$$

.

While synchronizing an interrupt assertion and an init assertion (representing the termination of one side), we have to consider whether there is an external interrupt occurring at time 0 and whether the interrupt assertion turns into the tail assertion at once. Thus, we have the rule:

$$\frac{}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{interrupt}(I, e, \{\mathsf{d} \Rightarrow P\}, cm), \mathsf{init})(s_1 \uplus s_2) \Longrightarrow_a \\ \mathsf{interrupt}(I \uplus \mathsf{id\_inv}, 0, \{\mathsf{d} \Rightarrow \uparrow (e \leq 0)\bar{\wedge}\mathsf{sync}(chs, P, \mathsf{init})\} \\ \mathsf{rel\_init1}(cm|_{chs^c}, \mathsf{init}))(s_1 \uplus s_2)\end{array}} \text{IntInit}$$

The list function rel_init1 is obtained from rel1 by replacing $\mathsf{delay}(d, R)$ by init

Synchronization involving recursive assertions is typically very complex, often requiring inductive analysis tailored to specific cases. As such, here we only provide the rule for a specific scenario to facilitate automated implementation.

$$\frac{\begin{array}{c}\forall\, s_0\, Q.\ \mathsf{sync}(chs, P_1, F_2(Q))(s_0) \Longrightarrow_a \mathsf{false}(s_0) \\ \forall\, s_0\, Q.\ \mathsf{sync}(chs, F_1(Q), P_2)(s_0) \Longrightarrow_a \mathsf{false}(s_0) \\ \forall\, s_0.\ \mathsf{sync}(chs, P_1, P_2)(s_0) \Longrightarrow_a P(s_0) \\ \forall\, s_0\, Q_1\, Q_2.\ \mathsf{sync}(chs, F_1(Q_1), F_2(Q_2))(s_0) \Longrightarrow_a F(\mathsf{sync}(chs, Q_1, Q_2))(s_0)\end{array}}{\begin{array}{c}\mathsf{sync}(chs, \mathsf{Rec}\ R_1.\ P_1\bar{\vee}F_1(R_1), \mathsf{Rec}\ R_2.\ P_2\bar{\vee}F_2(R_2))(s_0) \\ \Longrightarrow_a \mathsf{Rec}\ R.\ P\bar{\vee}F(R)(s0)\end{array}} \text{Rec}$$

The first two conditions state that if one side loops while the other doesn't, synchronization results in false. The third condition specifies that when both sides don't loop, synchronization is achieved. The last condition states that if both sides loop, synchronization depends on their outermost loops finishing together. Meeting all four conditions results in a new recursive assertion. This requires consistent recursion counts and simultaneous start and end of each iteration for both sides.

4 Proof Details of Example 3

In this section we give the details of the proof procedure of the example in the section of property verification.

According to the rule for recursion assertion, there are three premises to be checked:

$$\forall s.\, p(s) \longrightarrow loop(s) \tag{1}$$

$$\forall s_0\, s\, tr.\, loop(s_0) \longrightarrow (s_0, s, tr) \models \mathsf{init} \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2) \tag{2}$$

$$\forall Q\, s_0\, s\, tr.\, (\forall s_0\, s\, tr.\, loop(s_0) \longrightarrow (s_0, s, tr) \models Q \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2)) \longrightarrow$$
$$loop(s_0) \longrightarrow (s_0, s, tr) \models \mathsf{wait}(\mathsf{id}, 1, \{\mathsf{d} \Rightarrow Q[x := x+1]\}) \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2) \quad (3)$$

(1) is obvious and (2) is proved by the rule for init and the trivial fact $\forall s.\, loop(s) \longrightarrow q_1(s)$. To prove (3), we view (3a) as the assumption and we need to deduce (3b)

$$\forall s_0\, s\, tr.\, loop(s_0) \longrightarrow (s_0, s, tr) \models Q \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2) \quad (3a)$$

$$loop(s_0) \longrightarrow (s_0, s, tr) \models \mathsf{wait}(\mathsf{id}, 1, \{\mathsf{d} \Rightarrow Q[x := x+1]\}) \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2) \quad (3b)$$

By applying the rule for wait to (3b), we obtain the following two conditions. The first one:

$$loop(s_0) \wedge 1 > 0 \wedge t \geq 0 \wedge t \leq 1 \longrightarrow s = s_0 \longrightarrow q_2(s)$$

is also obvious. The second one is:

$$loop(s_0) \wedge 1 > 0 \longrightarrow (s_0, s, tr) \models Q[x := x+1] \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2)$$

By the rule for substitution, we have to prove for all s_0, s, tr:

$$(\exists v.\, loop[v/x] \wedge 1 > 0 \wedge x = v+1)(s_0) \longrightarrow (s_0, s, tr) \models Q \longrightarrow q_1(s) \wedge \mathsf{trl}(tr, q_2)$$

To make use of the assumption (3a), we need to prove:

$$\forall s.\, (\exists v.\, loop[v/x] \wedge 1 > 0 \wedge x = v+1)(s) \longrightarrow loop(s)$$

which is similar to that the loop invariant is still satisfied after executing one-round loop. Also this logic formula is obviously sound. So far, we have proved the property of this process.