PiCore: A Rely-guarantee Framework for Event-based Systems

Yongwang Zhao
School of Computer Science and Engineering, Beihang University, China
zhaoyongwang@gmail.com, zhaoyw@buaa.edu.cn

March 17, 2019

Contents

1 Integrating the CSimpl language into Picore

theory *picore-CSimpl*
imports *CSimpl.LocalRG-HoareDef PiCore.PiCore-RG-Invariant*
begin

type-synonym $('s,'p,'f,'e)$ *configI* $= ('s,'p,'f,'e)com \times ('s,'f)$ *xstate*

type-synonym $('s,'p,'f,'e)$ *confsI* $= (('s,'p,'f,'e)$ *configI*$)$ *list*

type-synonym $('s,'p,'f,'e)$ *Env* $= ('s,'p,'f,'e)$ *body* $\times ('s,'p,'f,'e)$ *sextuple set*

type-synonym $('s,'p,'f,'e)$ *confs'* $= ('s,'p,'f,'e)$ *Env* $\times(('s,'p,'f,'e)$ *config*$)$ *list*

definition *ptranI* :: $('s,'p,'f,'e)$ *Env* $\Rightarrow (('s,'p,'f,'e)$ *configI* $\times ('s,'p,'f,'e)$ *configI*$)$ *set*
where *ptranI* $\Psi \equiv \{(P,Q).$ *fst* $\Psi \vdash_c P \rightarrow Q\}$

definition *ptranI'* :: $('s,'p,'f,'e)$ *Env* $\Rightarrow ('s,'p,'f,'e)$ *configI* $\Rightarrow ('s,'p,'f,'e)$ *configI* $\Rightarrow$ *bool*
$(\text{-} \vdash_{cI} \text{-} \rightarrow \text{-}\ [81,81]\ 80)$
where $\Psi \vdash_{cI} P \rightarrow Q \equiv (P,Q) \in$ *ptranI* Ψ

lemma *none-no-tranI'*: $((Q,\ s),(P,t)) \in$ *ptranI* $\Psi \Longrightarrow Q \neq$ *Skip*
apply (*simp add:ptranI-def*) **apply**(*rule stepc.cases*)
by *simp+*

lemma *none-no-tranI*: $((Skip,\ s),(P,t)) \notin ptranI\ \Psi$
using *none-no-tranI'* **by** *fast*

lemma *ptran-neq'*: $((P,\ s),(Q,t)) \in ptranI\ \Psi \Longrightarrow P \neq Q$
apply (*simp add*:*ptranI-def*)
apply(*rule stepc.cases*) **apply** *simp*
apply(*rule stepc.cases*) **apply** *simp+*
using *mod-env-not-component* **apply** *blast*
apply *simp+*
using *step-change-p-or-eq-s* **apply** *blast*
apply (*simp add*: *step-change-p-or-eq-s*)
apply *simp+*
done

lemma *ptran-neqI*: $((P,\ s),(P,t)) \notin ptranI\ \Psi$
using *ptran-neq'* **by** *fast*

inductive *petranI* :: ('*s*,'*p*,'*f*,'*e*) *Env* ⇒ ('*s*,'*p*,'*f*,'*e*) *configI* ⇒ ('*s*,'*p*,'*f*,'*e*) *configI* ⇒ *bool*
(⊢$_{cI}$ (- →$_e$/ -) [*81*,*81*,*81*] *100*)
for Ψ :: ('*s*,'*p*,'*f*,'*e*) *Env*
where *petranI-nor*: $\Psi \vdash_{cI} (P,s) \rightarrow_e (P,t)$

lemma *petran-simpsI*:
$\Psi \vdash_{cI} (a,\ b) \rightarrow_e (c,\ d) \Longrightarrow a = c$
apply(*rule petranI.cases*) **apply** *simp+*

done

inductive-set *cptn'* :: (('*s*,'*p*,'*f*,'*e*) *confs'*) *set*
where
CptnOne': $(\Psi,\ [(P,s)]) \in cptn'$
| *CptnEnv'*: $(\Psi,(P,\ t)\#xs) \in cptn' \Longrightarrow (\Psi,(P,s)\#(P,t)\#xs) \in cptn'$
| *CptnComp'*: $[\![\Psi\vdash_{cI}(P,s) \rightarrow (Q,t);\ (\Psi,(Q,\ t)\#xs) \in cptn'\ ]\!] \Longrightarrow$
$(\Psi,(P,s)\#(Q,t)\#xs) \in cptn'$

lemma *tl-in-cptn'*: $[\![\ (\Psi,a\#xs) \in cptn';\ xs\neq[]\ ]\!] \Longrightarrow (\Psi,xs)\in cptn'$
by (*force elim*: *cptn'.cases*)

lemma *ab-cptn'-c-or-eq*: $(\Psi,a\#b\#l) \in cptn' \Longrightarrow \Psi\vdash_{cI} a \rightarrow b \vee fst\ a = fst\ b$
by (*force elim*: *cptn'.cases*)

lemma *cptn-not-empty'*: $(\Psi,l) \in cptn \Longrightarrow l \neq []$
by(*force elim*:*cptn.cases*)

lemma *cptn'-not-empty'*: $(\Psi,l) \in cptn' \Longrightarrow l \neq []$
by(*force elim*:*cptn'.cases*)

lemma *cptn-in-cptn'-h*: $((fst\ \Psi,\ l) \in cptn \Longrightarrow (\Psi,\ l) \in cptn') \Longrightarrow (fst\ \Psi,\ a\ \#\ l) \in cptn \Longrightarrow (\Psi,\ a\ \#\ l) \in cptn'$
apply(*rule cptn.cases*[*of fst* Ψ *a#l*])
apply *simp*
using *CptnOne'* **apply** *fast*
using *CptnEnv'* **apply** *fast*
using *CptnComp'* **apply**(*simp add*:*ptranI'-def ptranI-def*) **apply** *fast*
done

lemma *cptn-in-cptn'*: $(fst\ \Psi,l) \in cptn \Longrightarrow (\Psi,l) \in cptn'$
apply(*induct l*) **using** *cptn-not-empty'* **apply** *fast*
using *cptn-in-cptn'-h* **apply** *fast*
done

definition *cpts-pI* :: $('s,'p,'f,'e)\ Env \Rightarrow (('s,'p,'f,'e)\ confsI)\ set$
where *cpts-pI* $\Psi \equiv \{l.\ \exists l'.\ (\Psi,l') \in cptn' \wedge l = l'\}$

definition *cpts-of-pI* :: $('s,'p,'f,'e)\ Env \Rightarrow (('s,'p,'f,'e)\ com) \Rightarrow ('s,'f)\ xstate \Rightarrow (('s,'p,'f,'e)\ confsI)\ set$ **where**
cpts-of-pI $\Psi\ P\ s \equiv \{l.\ l!0=(P,s) \wedge l \in cpts\text{-}pI\ \Psi\}$

lemma *cptn-in-cpts-pI*: $(\Psi,l') \in cptn' \wedge l = l'$
$\Longrightarrow l \in cpts\text{-}pI\ \Psi$
by (*simp add*:*cpts-pI-def*)

lemma *cptn-not-emptyI*:$[] \notin cpts\text{-}pI\ \Psi$
apply(*simp add*:*cpts-pI-def*) **apply**(*force elim*:*cptn'.cases*)
done

lemma *cpts-of-pI-emptyI*: $l \in cpts\text{-}of\text{-}pI\ \Psi\ P\ s \Longrightarrow l \neq []$
apply(*simp add*:*cpts-of-pI-def*) **using** *cptn-not-emptyI* **by** *fast*

lemma *cpts-of-p-defI*: $l!0=(P,s) \wedge l \in cpts\text{-}pI\ \Psi \Longrightarrow l \in cpts\text{-}of\text{-}pI\ \Psi\ P\ s$
by(*simp add*:*cpts-of-pI-def*)

lemma *cpts-p-simpsI*:
$((\exists P\ s.\ aa = [(P,\ s)]) \vee$
$(\exists P\ t\ xs\ s.\ aa = (P,\ s)\ \#\ (P,\ t)\ \#\ xs \wedge (P,\ t)\ \#\ xs \in cpts\text{-}pI\ \Psi) \vee$
$(\exists P\ s\ Q\ t\ xs.\ aa = (P,\ s)\ \#\ (Q,\ t)\ \#\ xs \wedge \Psi \vdash_{cI} (P,\ s) \rightarrow (Q,\ t) \wedge (Q,\ t)\ \#\ xs \in cpts\text{-}pI\ \Psi)) \Longrightarrow (aa \in cpts\text{-}pI\ \Psi)$
apply(*simp add*:*cpts-pI-def*)
using *cptn'.simps* **by** *fast*

definition *rghoare-pI* :: $('s,'p,'f,'e)$ *Env* $\Rightarrow$ $[('s,'p,'f,'e)com,$ $('s,'f)$ *xstate set*, $(('s,'f)$ *xstate* $\times$ $('s,'f)$ *xstate*$)$ *set*,
$(('s,'f)$ *xstate* $\times$ $('s,'f)$ *xstate*$)$ *set*, $('s,'f)$ *xstate set*$] \Rightarrow$ *bool*
$(\text{-} \vdash_I \text{-}\ sat_p\ [\text{-}, \text{-}, \text{-}, \text{-}]\ [60,0,0,0,0]\ 45)$
where *rghoare-pI* Ψ *c p R G q*
$\equiv (p \subseteq Normal\ `\ UNIV) \wedge (q \subseteq Normal\ `\ UNIV)$
$\wedge\ (\forall (c,p,R,G,q,a) \in snd\ \Psi.\ fst\ \Psi \models_{/\{\}} (Call\ c)\ sat\ [p,\ R,\ G,\ q,a])$
$\wedge\ (fst\ \Psi, snd\ \Psi \vdash_{/\{\}} c\ sat\ [\{s.\ Normal\ s \in p\},\ R,\ G,\ \{s.\ Normal\ s \in q\},$
$UNIV]) (* this\ is\ the\ hoare\ rule\ in\ CSimpl\ *)$
$\wedge\ (\forall (s,t) \in R.\ s \notin Normal\ `\ UNIV \longrightarrow s = t)$

definition *assume-pI* :: $('s,'p,'f,'e)$ *Env* $\Rightarrow$ $(('s,'f)$ *xstate set* $\times$ $(('s,'f)$ *xstate* $\times$ $('s,'f)$ *xstate*$)$ *set*$)$
$\Rightarrow$ $(('s,'p,'f,'e)$ *confsI*$)$ *set*
where *assume-pI* $\Psi \equiv$
$\lambda(pre,\ rely).\ \{c.\ snd\ (c!0) \in pre \wedge (\forall i.\ Suc\ i < length\ c \longrightarrow \Psi \vdash_{cI} c!i \rightarrow_e c!(Suc\ i)$
$\longrightarrow (gets\text{-}p\ (c!i),\ gets\text{-}p\ (c!Suc\ i)) \in rely)\}$

definition *commit-pI* :: $('s,'p,'f,'e)$ *Env* $\Rightarrow$ $((('s,'f)$ *xstate* $\times$ $('s,'f)$ *xstate*$)$ *set* $\times$ $('s,'f)$ *xstate set*$)$
$\Rightarrow$ $(('s,'p,'f,'e)$ *confsI*$)$ *set*
where *commit-pI* $\Psi \equiv$
$\lambda(guar,\ post).\ \{c.\ (\forall i.\ Suc\ i < length\ c \longrightarrow fst\ \Psi \vdash_c c!i \rightarrow c!(Suc\ i) \longrightarrow (snd\ (c!i),\ snd\ (c!Suc\ i)) \in guar)$
$\wedge\ (fst\ (last\ c) = Skip \longrightarrow snd\ (last\ c) \in post)\}$

definition *prog-validityI* :: $('s,'p,'f,'e)$ *Env* $\Rightarrow$ $('s,'p,'f,'e)com \Rightarrow$ $('s,'f)$ *xstate set*
$\Rightarrow$ $(('s,'f)$ *xstate* $\times$ $('s,'f)$ *xstate*$)$ *set*
$\Rightarrow$ $(('s,'f)$ *xstate* $\times$ $('s,'f)$ *xstate*$)$ *set* $\Rightarrow$ $('s,'f)$ *xstate set* $\Rightarrow$ *bool*
$(\text{-} \models_I \text{-}\ sat_p\ [\text{-}, \text{-}, \text{-}, \text{-}]\ [60,60,0,0,0,0]\ 45)$
where *prog-validityI* Ψ *P pre rely guar post* $\equiv$
$\forall s.\ cpts\text{-}of\text{-}pI\ \Psi\ P\ s \cap assume\text{-}pI\ \Psi\ (pre,\ rely) \subseteq commit\text{-}pI\ \Psi\ (guar,\ post)$

lemma *prog-validity-defI*: *prog-validityI* Ψ *P pre rely guar post* $\Longrightarrow$
$\forall s.\ cpts\text{-}of\text{-}pI\ \Psi\ P\ s \cap assume\text{-}pI\ \Psi\ (pre,\ rely) \subseteq commit\text{-}pI\ \Psi\ (guar,\ post)$
by$(simp\ add{:}prog\text{-}validityI\text{-}def)$

lemma *assume-p-defI*: *gets-p* $(c!0) \in pre \wedge (\forall i.\ Suc\ i < length\ c \longrightarrow$

$\Psi \vdash_{cI} c!i \rightarrow_e c!(Suc\ i) \longrightarrow (gets\text{-}p\ (c!i), gets\text{-}p\ (c!Suc\ i)) \in rely) \Longrightarrow$
$c \in assume\text{-}pI\ \Psi\ (pre, rely)$
by (*simp add:assume-pI-def gets-p-def*)

lemma *commit-p-defI*: $commit\text{-}pI\ \Psi \equiv \lambda(guar, post).\ \{c.\ (\forall i.\ Suc\ i<length\ c \longrightarrow$

$\Psi \vdash_{cI} c!i \rightarrow c!(Suc\ i) \longrightarrow (gets\text{-}p\ (c!i), gets\text{-}p\ (c!Suc\ i)) \in guar) \wedge$
$(getspc\text{-}p\ (last\ c) = Skip \longrightarrow gets\text{-}p\ (last\ c) \in post)\}$
by (*simp add:commit-pI-def getspc-p-def gets-p-def ptranI'-def ptranI-def*)

lemma *assume-p-defI2*: $gets\text{-}p\ (c!0) \in pre \wedge (\forall i.\ Suc\ i<length\ c \longrightarrow$
$\Psi \vdash_{cI} c!i \rightarrow_e c!(Suc\ i) \longrightarrow (gets\text{-}p\ (c!i), gets\text{-}p\ (c!Suc\ i)) \in rely)$
$\Longrightarrow c \in assume\text{-}pI\ \Psi\ (pre, rely)$
by (*simp add*: *assume-p-defI*)

lemma *commit-p-defI2*: $c \in commit\text{-}pI\ \Psi\ (guar, post) \Longrightarrow (\forall i.\ Suc\ i<length\ c$
$\longrightarrow$
$\Psi \vdash_{cI} c!i \rightarrow c!(Suc\ i) \longrightarrow (gets\text{-}p\ (c!i), gets\text{-}p\ (c!Suc\ i)) \in guar) \wedge$
$(getspc\text{-}p\ (last\ c) = Skip \longrightarrow gets\text{-}p\ (last\ c) \in post)$
using *commit-p-defI*[*of* Ψ] **by** *simp*

lemma *stepe-imp-petranI*: $Suc\ i < length\ l \Longrightarrow fst\ \Psi \vdash_c (l\ !\ i) \rightarrow_e (l\ !\ Suc\ i)$
$\Longrightarrow \Psi \vdash_{cI} (l!i) \rightarrow_e (l!(Suc\ i))$
by (*metis etranE petranI-nor*)

lemma *CptnComp-h*: $[\![\Psi \vdash_c a \rightarrow b;\ (\Psi, b\#xs) \in cptn\ ]\!] \Longrightarrow (\Psi, a\#b\#xs) \in cptn$
using *CptnComp* **by** (*metis prod.collapse*)

lemma *rgsound-pI-h*:
$(\Psi, l) \in cptn' \Longrightarrow$
$\forall (s, t) \in rely.\ s \notin range\ Normal \longrightarrow s = t \Longrightarrow$
$x = l \Longrightarrow$
$\forall i.\ Suc\ i < length\ x \longrightarrow (\Psi \vdash_{cI} x\ !\ i \rightarrow_e x\ !\ Suc\ i) \longrightarrow (gets\text{-}p\ (x\ !\ i), gets\text{-}p$
$(x\ !\ Suc\ i)) \in rely \Longrightarrow$
$(*\forall i.\ Suc\ i < length\ l \longrightarrow \Psi \vdash_c (l\ !\ i) \rightarrow_e (l\ !\ Suc\ i) \longrightarrow (snd\ (l\ !\ i), snd\ (l\ !$
$Suc\ i)) \in rely \Longrightarrow *)$
$(fst\ \Psi, l) \in cptn$
proof(*induct l arbitrary*: *x*)
case *Nil*
then show *?case* **using** *cptn'-not-empty'* **by** *fast*
next
case (*Cons a l*)
assume *p0*: $\bigwedge x.\ (\Psi, l) \in cptn' \Longrightarrow$
$\forall (s, t) \in rely.\ s \notin range\ Normal \longrightarrow s = t \Longrightarrow$
$x = l \Longrightarrow$
$\forall i.\ Suc\ i < length\ x \longrightarrow \Psi \vdash_{cI} x\ !\ i \rightarrow_e x\ !\ Suc\ i \longrightarrow (gets\text{-}p\ (x\ !$
$i), gets\text{-}p\ (x\ !\ Suc\ i)) \in rely \Longrightarrow (fst\ \Psi, l) \in cptn$
and *p1*: $(\Psi, a\ \#\ l) \in cptn'$
and *p2*: $\forall (s, t) \in rely.\ s \notin range\ Normal \longrightarrow s = t$

and *p4*: $x = (a \# l)$
and *p3*: $\forall i.\ Suc\ i < length\ x \longrightarrow \Psi \vdash_{cI} x\ !\ i \rightarrow_e x\ !\ Suc\ i \longrightarrow (gets\text{-}p\ (x\ !\ i), gets\text{-}p\ (x\ !\ Suc\ i)) \in rely$
show *?case*
proof(*cases* $l = []$)
assume *l*: $l = []$
thus *?thesis* **using** *CptnOne* **by** (*metis prod.collapse*)
next
assume *l-ne-empty*: $l \neq []$
then obtain *b* **and** l' **where** *l*: $l = b \# l'$
using *list.exhaust* **by** *blast*

with *p1* **have** *l-in-cptn'*: $(\Psi, l) \in cptn'$ **using** *tl-in-cptn'* **by** *blast*
from *p4* **have** *len-x-l*: $length\ x = length\ (a \# l)$ **by** *simp*

from *p4 len-x-l* **obtain** *y* **where** *y*: $y = tl\ x \wedge y = l$
by *simp*

from *p3 y* **have** *y-pe-rely*: $\forall i.\ Suc\ i < length\ y \longrightarrow \Psi \vdash_{cI} y\ !\ i \rightarrow_e y\ !\ Suc\ i \longrightarrow (gets\text{-}p\ (y\ !\ i), gets\text{-}p\ (y\ !\ Suc\ i)) \in rely$
by (*metis* (*no-types*, *lifting*) *List.nth-tl Suc-lessD Suc-mono len-x-l length-Cons*)

from *y-pe-rely y l-in-cptn' p0*[*of y*] *p2* **have** *l-in-cptn*: $(fst\ \Psi, l) \in cptn$ **by** *fast*

from *p1 l* **have** $\Psi \vdash_{cI} a \rightarrow b \vee fst\ a = fst\ b$ **using** *ab-cptn'-c-or-eq* **by** *simp*

thus *?thesis*
proof
assume $\Psi \vdash_{cI} a \rightarrow b$
with *l-in-cptn l* **show** *?thesis* **using** *CptnComp-h*[*of fst* Ψ *a b l'*] *ptranI'-def*[*of* Ψ *a b*] *ptranI-def*[*of* Ψ] **by** *simp*
next
assume *ab-spec*: $fst\ a = fst\ b$
with *l p4 len-x-l* **have** $fst\ (x\ !\ 0) = fst\ (x\ !\ Suc\ 0)$ **by** *simp*
hence $\Psi \vdash_{cI} x\ !\ 0 \rightarrow_e x\ !\ Suc\ 0$ **using** *petranI-nor* **by** (*metis prod.collapse*)

with *p3 len-x-l l* **have** $(gets\text{-}p\ (x\ !\ 0), gets\text{-}p\ (x\ !\ Suc\ 0)) \in rely$ **by** *simp*
moreover
from *p4 len-x-l* **have** $gets\text{-}p\ (x\ !\ 0) = snd\ a$
by (*simp add*: *gets-p-def*)
moreover
from *l-ne-empty p4 l len-x-l* **have** $gets\text{-}p\ (x\ !\ Suc\ 0) = snd\ b$
by (*simp add*: *gets-p-def*)
ultimately have $fst\ \Psi \vdash_c a \rightarrow_e b$
apply(*case-tac* $\forall t'.\ snd\ a \neq Normal\ t'$)
using *Env-n*[*of snd a fst* Ψ *fst a*] *p2 ab-spec surjective-pairing*[*of a*] *surjective-pairing*[*of b*] **apply** *auto*[*1*]

using *Env*[*of fst* Ψ *fst a - snd b*] *p2 ab-spec surjective-pairing*[*of a*] *surjective-pairing*[*of b*] **apply** *auto*[*1*]
done
thus *?thesis* **using** *CptnEnv*[*of fst* Ψ *fst a snd a snd b l'*] *l l-in-cptn ab-spec surjective-pairing*[*of a*] *surjective-pairing*[*of b*]
by *auto*
qed
qed
qed

lemma *rgsound-pI*: *rghoare-pI* Ψ *P pre rely guar post* ⟶ *prog-validityI* Ψ *P pre rely guar post*
proof
assume *rghoare-pI* Ψ *P pre rely guar post*
hence *a10*: *pre* ⊆ *range Normal* ∧ *post* ⊆ *range Normal*
and *a11*: *fst* Ψ,*snd* Ψ ⊢$_{/\{\}}$ *P sat* [{*s*. *Normal s* ∈ *pre*}, *rely*, *guar*, {*s*. *Normal s* ∈ *post*},*UNIV*]
and *a12*: (∀ (*s*,*t*)∈*rely*. *s* ∉ *Normal* ‘ *UNIV* ⟶ *s* = *t*)
and *a13*: ∀ (*c*,*p*,*R*,*G*,*q*,*a*)∈ *snd* Ψ. *fst* Ψ ⊨$_{/\{\}}$ (*Call c*) *sat* [*p*, *R*, *G*, *q*,*a*] **apply** *auto*
apply (*simp add*:*rghoare-pI-def*, *fast*)+ **apply** (*simp add*:*rghoare-pI-def*)
apply (*simp add*:*rghoare-pI-def*) **apply** *auto*[*1*] **apply** (*simp add*:*rghoare-pI-def*)
by *auto*

hence *fst* Ψ,*snd* Ψ ⊨$_{/\{\}}$ *P sat* [{*s*. *Normal s* ∈ *pre*},*rely*, *guar*, {*s*. *Normal s* ∈ *post*},*UNIV*]
using *localRG-sound* **by** *fast*
with *a13* **have** ∀ *s*. *cp* (*fst* Ψ) *P s* ∩ *assum* ({*s*. *Normal s* ∈ *pre*}, *rely*) ⊆ *comm* (*guar*, {*s*. *Normal s* ∈ *post*}, *UNIV*) {}
by (*simp add*:*com-cvalidity-def com-validity-def*)
hence *a2*: ∀ *s*. {(Ψ*1*, *l*). *l* ! *0* = (*P*, *s*) ∧ (*fst* Ψ, *l*) ∈ *cptn* ∧ Ψ*1* = *fst* Ψ} ∩
{*c*. *snd* (*snd c* ! *0*) ∈ *Normal* ‘ {*s*. *Normal s* ∈ *pre*} ∧
(∀ *i*. *Suc i* < *length* (*snd c*) ⟶
fst c⊢$_c$ *snd c* ! *i* →$_e$ *snd c* ! *Suc i* ⟶ (*snd* (*snd c* ! *i*), *snd* (*snd c* ! *Suc i*)) ∈ *rely*)}
⊆ {*c*. (∀ *i*. *Suc i* < *length* (*snd c*) ⟶
fst c⊢$_c$ *snd c* ! *i* → *snd c* ! *Suc i* ⟶ (*snd* (*snd c* ! *i*), *snd* (*snd c* ! *Suc i*)) ∈ *guar*) ∧
(*final* (*last* (*snd c*)) ⟶
fst (*last* (*snd c*)) = *Skip* ∧ *snd* (*last* (*snd c*)) ∈ *Normal* ‘ {*s*. *Normal s* ∈ *post*} ∨
fst (*last* (*snd c*)) = *Throw* ∧ *snd* (*last* (*snd c*)) ∈ *range Normal*)}
by (*simp add*: *assum-def comm-def cp-def*)

{
fix *s x*
assume *b0*: *x* ∈ *cpts-of-pI* Ψ *P s* ∩ *assume-pI* Ψ (*pre*, *rely*)
hence *b1*: *x* ! *0* = (*P*, *s*) ∧

$(\exists\, l'.\ (\Psi,\ l') \in cptn' \wedge x = l')$
by(*simp add*:*cpts-of-pI-def cpts-pI-def*)
then obtain *l* **where** *b2*:
$(\Psi,\ l) \in cptn' \wedge\ \ x = l$
by *auto*
from *b0* **have** *x-not-empty*: $x \neq []$ **using** *cpts-of-pI-emptyI* **by** *fast*
from *x-not-empty b2* **have** *l-not-empty*: $l \neq []$ **by** *fast*
from *b0* **have** *b3*: $snd(x!0) \in pre \wedge (\forall\, i.\ Suc\ i < length\ x \longrightarrow \Psi \vdash_{cI} x\ !\ i \rightarrow_e x\ !\ Suc\ i$
$\longrightarrow (gets\text{-}p\ (x\ !\ i),\ gets\text{-}p\ (x\ !\ Suc\ i)) \in rely)$
by(*simp add*:*assume-pI-def assum-def*)

from *b1 b2 l-not-empty x-not-empty* **have** *l0*: $l\ !\ 0 = (P,\ s)$ **by** *fast*
from *x-not-empty b2 l0* **have** *x0-s*: $snd\ (x!0) = s$ **by** *simp*

from *l0 a10 b3 x0-s* **have** *l0-s*: $snd\ (l\ !\ 0) \in Normal\ `\ \{s.\ Normal\ s \in pre\}$ **by** *auto*

from *b2* **have** *len-x-l*: $length\ x = length\ l$ **by** *simp*

have *l-rely*: $\forall\, i.\ Suc\ i < length\ l \longrightarrow$
$(fst\ \Psi \vdash_c l\ !\ i \rightarrow_e l\ !\ Suc\ i) \longrightarrow (snd\ (l\ !\ i),\ snd\ (l\ !\ Suc\ i)) \in rely$
proof −
{
fix *i*
assume *c0*: $Suc\ i < length\ l$
and *c1*: $fst\ \Psi \vdash_c l\ !\ i \rightarrow_e l\ !\ Suc\ i$
with *b2* **have** $\Psi \vdash_{cI} x\ !\ i \rightarrow_e x\ !\ Suc\ i$ **using** *stepe-imp-petranI* **by** *fast*

with *b2 c0 b3* **have** $(gets\text{-}p\ (x\ !\ i),\ gets\text{-}p\ (x\ !\ Suc\ i)) \in rely$
by *auto*
moreover
have $snd\ (l\ !\ i) = gets\text{-}p\ (x\ !\ i)$ **using** *b2 c0 gets-p-def* **by** *metis*

moreover
have $snd\ (l\ !\ Suc\ i) = gets\text{-}p\ (x\ !\ Suc\ i)$ **using** *b2 c0 gets-p-def*
by (*metis* (*mono-tags*, *lifting*) *length-map*)
ultimately have $(snd\ (l\ !\ i),\ snd\ (l\ !\ Suc\ i)) \in rely$ **by** *simp*
}
then show *?thesis* **by** *auto* **qed**

from *a12 b2 b3* **have** *l-in-cptn*: $(fst\ \Psi,\ l) \in cptn$ **using** *rgsound-pI-h* **by** *blast*

from *a11 b2 b3 l0 a2*[*rule-format*, *of s*] *l0-s l-rely* **have** *g0*: $(\forall\, i.\ Suc\ i < length\ (l) \longrightarrow$
$fst\ \Psi \vdash_c l\ !\ i \rightarrow l\ !\ Suc\ i \longrightarrow (snd\ (l\ !\ i),\ snd\ (l\ !\ Suc\ i)) \in guar) \wedge$

```
                (SmallStepCon.final (last l) ⟶
                  fst (last l) = Skip ∧ snd (last l) ∈ Normal ‘ {s. Normal s ∈
post} ∨
                   fst (last l) = Throw ∧ snd (last l) ∈ range Normal) using
l-in-cptn by auto

  {
   fix i
   assume c0: Suc i < length x
     and  c1: fst Ψ ⊢c x!i → x!(Suc i)
   with b2 have fst Ψ ⊢c l ! i → l ! Suc i by simp
   moreover have snd (l ! i) = snd (x ! i) using b2 c0 by metis
   moreover
   have snd (l ! Suc i) = snd (x ! Suc i) using b2 c0 by metis
   ultimately  have (snd (x!i), snd (x!Suc i)) ∈ guar using g0 c0 len-x-l by
auto
  }
  moreover
  {
   assume fst (last x) = Skip
   with b2 have c1: fst (last l) = Skip using cptn'-not-empty'[of Ψ l] len-x-l
by fast
   moreover
   from c1 have final (last l) by (simp add:final-def)
   moreover
   have snd (last l) = snd (last x) using b2 gets-p-def
     by (simp add: case-prod-unfold l-not-empty last-map)
  ultimately have snd (last x) ∈ post using cptn-not-empty'[of fst Ψ l] using
b2 g0 by auto
  }
  ultimately have x ∈ commit-pI Ψ (guar, post) by (simp add:commit-pI-def)
 }
 hence ∀ s. cpts-of-pI Ψ P s ∩ assume-pI Ψ (pre, rely) ⊆ commit-pI Ψ (guar,
post) by auto

 thus prog-validityI Ψ P pre rely guar post by (simp add:prog-validityI-def)
qed

interpretation event ptranI petranI Skip
 using petran-simpsI none-no-tranI ptran-neqI
       event.intro[of petranI Skip ptranI] by blast

interpretation event-comp ptranI petranI Skip cpts-pI cpts-of-pI
 using cpts-p-simpsI cpts-of-p-defI petran-simpsI none-no-tranI ptran-neqI cptn-not-emptyI
         event-comp.intro[of ptranI petranI Skip cpts-pI cpts-of-pI] event.intro[of
petranI Skip ptranI]
       event-comp-axioms.intro[of cpts-pI ptranI cpts-of-pI]
 apply(simp add:ptranI-def ptranI'-def ptran'-def) by blast
```

interpretation *event-hoare ptranI petranI Skip cpts-pI cpts-of-pI prog-validityI assume-pI commit-pI rghoare-pI*
using *cpts-p-simpsI cpts-of-p-defI none-no-tranI ptran-neqI cptn-not-emptyI petran-simpsI*
prog-validity-defI assume-p-defI2 commit-p-defI2 rgsound-pI
event-comp-axioms.intro[*of cpts-pI ptranI cpts-of-pI*]
event-comp.intro[*of ptranI petranI Skip cpts-pI cpts-of-pI*] *event.intro*[*of petranI Skip ptranI*]
event-validity-axioms.intro[*of prog-validityI cpts-of-pI assume-pI commit-pI petranI ptranI Skip*]
event-validity.intro[*of ptranI petranI Skip cpts-pI cpts-of-pI prog-validityI assume-pI commit-pI*]
event-hoare.intro[*of ptranI petranI Skip cpts-pI cpts-of-pI prog-validityI assume-pI commit-pI rghoare-pI*]
event-hoare-axioms.intro[*of rghoare-pI prog-validityI*]
apply(*simp add*:*ptranI-def ptranI'-def ptran'-def*) **by** *blast*

end