Intuitions behind A-C reasoning

WHEN REASONING IN ISOLATION about a process P, communicating by synchr. message passing with its environment, one should ask the ff question:

ABOUT WHICH QUANTITIES DO I NEED INFO?

- The initial state of P, expressed by its PRECONDITION φ
- The inputs since initialization, expressed by its environmental ASSUMPTION A, which is a predicate on the computation history h

(- IF also props like blocking are needed, this format must be extended)

- So, if one knows φ, the precondition, and also that assumption A holds after every input, it is possible to express which properties hold for the outputs of a process P, expressed by its commitment C, and, in case P terminates, its postcondition ψ (both relative to φ and A).

- In fact, $\models SP(\varphi, A, P) \Rightarrow \psi$

  $\models SC(\varphi, A, P) \Rightarrow C$

  for appropriately def'd strongest postcondition operators $SP(\varphi, A, P)$ and strongest commitment $SC(\varphi, A, P)$ relative to φ, A, assuming $\models \langle A, C \rangle : \{\varphi\}\, P\, \{\psi\}$

- Notice that A,C reasoning applies to infinite computations as well. For even, if P doesn't terminate, at any stage of the comp., once φ holds for the initial state, and A holds for any input take place before that stage of comp, C (or rather $SC(\varphi, A, P)$) should hold $\Longrightarrow$

A-C reasoning specifies REACTIVE SYSTEMS (...)

- Therefore, we must reason now about
  FINITE AND INFINITE COMPUTATIONS

- We describe an infinite computation by means of all its prefixes – for ~~the~~ an infinite comp. is uniquely determined by this set.

  Also we are only interested, at the moment, in INVARIANCE properties, and to check these, only all finite prefixes need to be examined

- Since we have now finite sequences of two kinds
  – prefixes of inf. comps
  – (prefixes of) terminated computations

  we need to DISTINGUISH THESE :

  $\Rightarrow$

  done by introducing an extra termination flag τ, indicating whether a comp. has terminated, expressed by $\tau = \top$ (top),

  or not,

  exp'd by $\tau = \bot$ (bottom)

Definition 7.30 The compositional semantics $O[\![P]\!]$ of a program P is defined as follows. For $B \equiv (L,T,s,t)$,

- $O[\![B]\!] \stackrel{\text{def}}{=} \bigcup_{l\in L}\{(\sigma,\sigma',\theta,\bot)|(\sigma,\sigma',\theta) \in O_l(B)\}$
  $\cup\{(\sigma,\sigma',\theta,\top)|(\sigma,\sigma',\theta) \in O_t(B)\}$,
- $O[\![P_1;P_2]\!] \stackrel{\text{def}}{=} \{(\sigma,\sigma_1,\theta,\bot)|(\sigma,\sigma_1,\theta,\bot) \in O[\![P_1]\!]\}$
  $\cup\{(\sigma,\sigma_2,\theta,\tau)|\exists\sigma_1,\theta_1,\theta_2.(\sigma,\sigma_1,\theta_1,\top) \in O[\![P_1]\!]$
  $\wedge(\sigma_1,\sigma_2,\theta_2,\tau) \in O[\![P_2]\!] \wedge \theta = \theta_1 \cdot \theta_2\}$,
- $O[\![P_1||P_2]\!] \stackrel{\text{def}}{=} \{(\sigma,\sigma',\theta,\tau)|$for $i = 1,2$, $(\sigma,\sigma'_i,\theta{\downarrow}P_i,\tau_i) \in O[\![P_i]\!]$
  $\wedge\theta = \theta{\downarrow}Chan(P_1||P_2)$
  $\wedge(\tau = \top \leftrightarrow (\tau_1 = \top \wedge \tau_2 = \top))$,
  where
  $$\sigma'(x) = \begin{cases} \sigma'_1(x), & \text{if } x \in var(P_1), \\ \sigma'_2(x), & \text{if } x \in var(P_2), \\ \sigma(x), & \text{otherwise}\}. \end{cases}$$ □

Note that, due to the condition $\theta = \theta{\downarrow}Chan(P_1||P_2)$ in the definition of $O[\![P_1||P_2]\!]$, θ does not contain communications along channels not occurring in $P_1||P_2$.

$\Longrightarrow$

7.5.4

Previously we had as format for $\Theta[\![P]\!]$:

$\{(\sigma, \sigma', \theta) / \ldots \}$

Now we get as format ... :

$\{(\sigma, \sigma', \theta, \tau) / \tau \in \{\bot, T\}, \ldots . \}$

Read def 7.30.

Note : • no "dirt" in θ allowed, i.e., $\theta\downarrow(P_1 \| P_2) = \theta$

• for basic (seq.) synchr. tds, only terminated comps which reach node t, the terminal node, are marked by $\tau = T$

• only when, in $P_1 \| P_2$, both P_1 and P_2 have reached their ~~exit~~ final locations, the resulting comp. of $P_1 \| P_2$ is marked as terminated

• observe that in def. 7.30 prefix closure is preserved (with pr. closure def'd next) explaining the need that $(\sigma, \sigma', \theta, \bot) \prec (\sigma, \sigma', \theta, T)$ (otherwise ";" doesn't preserve prefix closure)

Definition 7.30 The compositional semantics $O[\![P]\!]$ of a program P is defined as follows. For $B \equiv (L, T, s, t)$,

- $O[\![B]\!] \stackrel{\text{def}}{=} \bigcup_{l \in L} \{(\sigma, \sigma', \theta, \bot) | (\sigma, \sigma', \theta) \in O_l(B)\}$
  $\cup \{(\sigma, \sigma', \theta, \top) | (\sigma, \sigma', \theta) \in O_t(B)\}$,
- $O[\![P_1; P_2]\!] \stackrel{\text{def}}{=} \{(\sigma, \sigma_1, \theta, \bot) | (\sigma, \sigma_1, \theta, \bot) \in O[\![P_1]\!]\}$
  $\cup \{(\sigma, \sigma_2, \theta, \tau) | \exists \sigma_1, \theta_1, \theta_2 . (\sigma, \sigma_1, \theta_1, \top) \in O[\![P_1]\!]$
  $\wedge (\sigma_1, \sigma_2, \theta_2, \tau) \in O[\![P_2]\!] \wedge \theta = \theta_1 \cdot \theta_2\}$,
- $O[\![P_1 || P_2]\!] \stackrel{\text{def}}{=} \{(\sigma, \sigma', \theta, \tau) | \text{for } i = 1, 2, (\sigma, \sigma'_i, \theta{\downarrow}P_i, \tau_i) \in O[\![P_i]\!]$
  $\wedge \theta = \theta{\downarrow}Chan(P_1 || P_2)$
  $\wedge (\tau = \top \leftrightarrow (\tau_1 = \top \wedge \tau_2 = \top))$,
  where
  $$\sigma'(x) = \begin{cases} \sigma'_1(x), & \text{if } x \in var(P_1), \\ \sigma'_2(x), & \text{if } x \in var(P_2), \\ \sigma(x), & \text{otherwise}\}. \end{cases}$$ □

Note that, due to the condition $\theta = \theta{\downarrow}Chan(P_1 || P_2)$ in the definition of $O[\![P_1 || P_2]\!]$, θ does not contain communications along channels not occurring in $P_1 || P_2$.

"is a prefix of" - sign

$\preceq$

. HERE $\preceq$ def'd by:

$(\sigma, \sigma', \theta, \bot) \preceq (\sigma, \sigma', \theta, \top)$ for appr. σ, σ', θ

$(\sigma, \sigma', \theta, \bot) \preceq (\sigma, \sigma'', \theta\theta', \top)$ for appr. $\sigma, \sigma', \sigma'', \theta, \theta'$

With this def. of $\preceq$,

$O[\![P_1; P_2]\!]$ and $O[\![P_1 || P_2]\!]$ preserve prefix-closure.

I.e., if $O[\![P_i]\!]$, $i = 1, 2$ is prefix-closed, so are

- Recall that (sim. as in section 6.4)
  process P satisfies (A, C)
  provided:

  ... at any stage of an ongoing computation,
  P's actions should satisfy C, as long as
  A has been satisfied before by P's environment,
  i.e., (A, C) must be satisfied by all prefixes
  of P's computations.

- Observe that in the def. of ~~bas~~ the sems of basic transition diagrams B, all prefixes of any comp. of B are considered
- That ";" and "//" preserve prefix closure

  ↑

  explaining $(\sigma, \sigma', \theta, \bot) \leq (\sigma, \sigma', \theta, \top)$

7.5.2 Validity

Recall that an A-C correctness formula has the form

$$\langle A, C\rangle : \{\varphi\}\ P\ \{\psi\},$$

where A and C are trace predicates (defined below) and φ and ψ ordinary predicates.

Definition 7.31 A trace predicate A is a predicate which involves no program variables $\bar{x} \subseteq Pvar$; its satisfaction depends only on the communication sequence which is recorded in the value of h and on its logical variables. For a trace predicate A we have that for all σ and σ',

$$\sigma \models A \Leftrightarrow \sigma' \models A \text{ iff } \sigma(x) = \sigma'(x), \text{ for } x \in Lvar \cup \{h\}. \qquad \square$$

Now validity of an A-C correctness formula $\langle A, C\rangle : \{\varphi\}P\{\psi\}$ has the following intuitive meaning:

If φ holds in the initial state, including the communication history, in which P starts its execution, then

(i) *C holds initially, and C holds after every communication provided A holds after all preceding communications, and*

(ii) *if P terminates and A holds after all previous communications (including the last one) then ψ holds in the final state, including the final communication history.*

Formally, validity of A-C formulae is defined as below.

Definition 7.32 (Validity)

$$\models \langle A, C\rangle : \{\varphi\}\ P\ \{\psi\} \text{ if}$$

$$\begin{array}{l}
\forall(\sigma, \sigma', \theta, \tau) \in O[\![P]\!]. \\
\quad \sigma \models \varphi \Rightarrow \\
\quad ((\forall \theta' \prec \theta.(\sigma : h \mapsto \sigma(h) \cdot \theta') \models A) \Rightarrow (\sigma : h \mapsto \sigma(h) \cdot \theta) \models C) \wedge \\
\quad ((\tau = \top \wedge (\forall \theta' \preceq \theta.(\sigma : h \mapsto \sigma(h) \cdot \theta') \models A)) \Rightarrow (\sigma' : h \mapsto \sigma(h) \cdot \theta) \models \psi).
\end{array}$$

$\square$

PROOF METHOD

Definition 7.33 (A-C-inductive assertion networks) Q is an A-C-inductive assertion network w.r.t. A and C for $B \equiv (L, T, s, t)$ if:

- $\models Q_s \to C$.
- In case of an internal transition $l \xrightarrow{a} l' \in T$, $a \equiv b \to f$, we require

$$\models Q_l \wedge b \wedge A \to Q_{l'} \circ f.$$

- In case of an output transition $l \xrightarrow{a} l' \in T$, $a \equiv b; D!e \to f$, we require, for $v = e(\sigma)$,

$$\models Q_l \wedge A \wedge b \to ((A \to Q_{l'}) \wedge C) \circ (f \circ g),$$

where $g(\sigma) \stackrel{\text{def}}{=} (\sigma : h \mapsto \sigma(h) \cdot (D, v))$.

- In case of an input transition $l \xrightarrow{a} l' \in T$, with $a \equiv b; D?x \to f$, we require, for an *arbitrary* value $v \in VAL$,

$$\models Q_l \wedge A \wedge b \to ((A \to Q_{l'}) \wedge C) \circ (f \circ g),$$

where $g(\sigma) \stackrel{\text{def}}{=} (\sigma : x, h \mapsto v, \sigma(h) \cdot (D, v))$. □

Rule 7.11 (Basic diagram rule) For $B \equiv \langle L, T, s, t \rangle$:

$$\frac{Q(A,C) \vdash B}{\langle A,C \rangle : \{Q_s\}\ B\ \{Q_t\}}.$$

Example 7.35 (Even number generator) We demonstrate the application of the method by verifying the first specification of Example 6.1, i.e., for the program P of Figure 6.3 one has

$$\models \langle true, \#D = \#A = \#B \geq 1 \rightarrow last(D) = last(A) + last(B) \rangle : \{\#D = \#A = \#B = 0\}\ P\ \{false\}.$$

We have the following assertion network for P:

$$\begin{aligned}
&Q_s \stackrel{\text{def}}{=} \#D = \#A = \#B = 0,\\
&Q_{l_1} \stackrel{\text{def}}{=} \#D = \#A = \#B,\\
&Q_{l_2} \stackrel{\text{def}}{=} \#B = \#D = (\#A - 1) \wedge last(A) = x,\\
&Q_{l_3} \stackrel{\text{def}}{=} \#A = \#B = (\#D + 1) \wedge last(A) = x \wedge last(B) = y, \text{ and}\\
&Q_t \stackrel{\text{def}}{=} false.
\end{aligned}$$

Example 6.1 We continue the example from Figure 6.1 and propose an implementation of P using distributed communication in Figure 6.3.

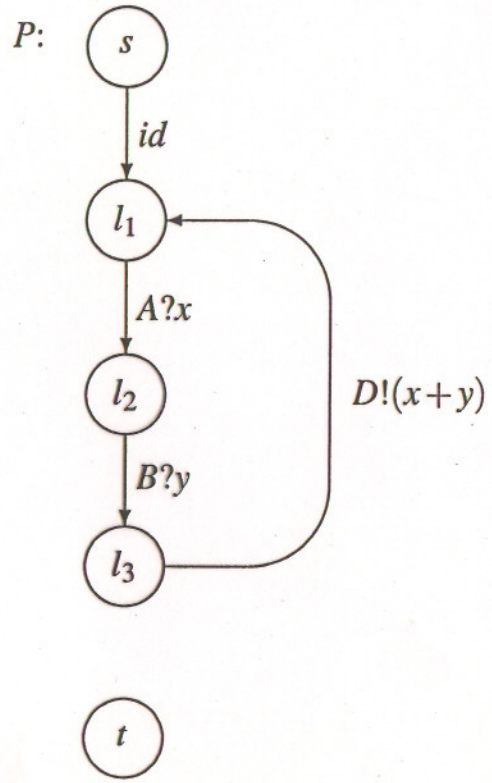


Fig. 6.3. Structure of adder P.

Ex. 7.35 To prove : $\vdash \langle true, \#D = \#A - \#B \geq 1 \rightarrow last(D) = last(A) + last(B) \rangle$ ::

$\{\#D = \#A = \#B = 0\}\ P\ \{false\}$

Example 6.1 We continue the example from Figure 6.1 and propose an implementation of P using distributed communication in Figure 6.3.

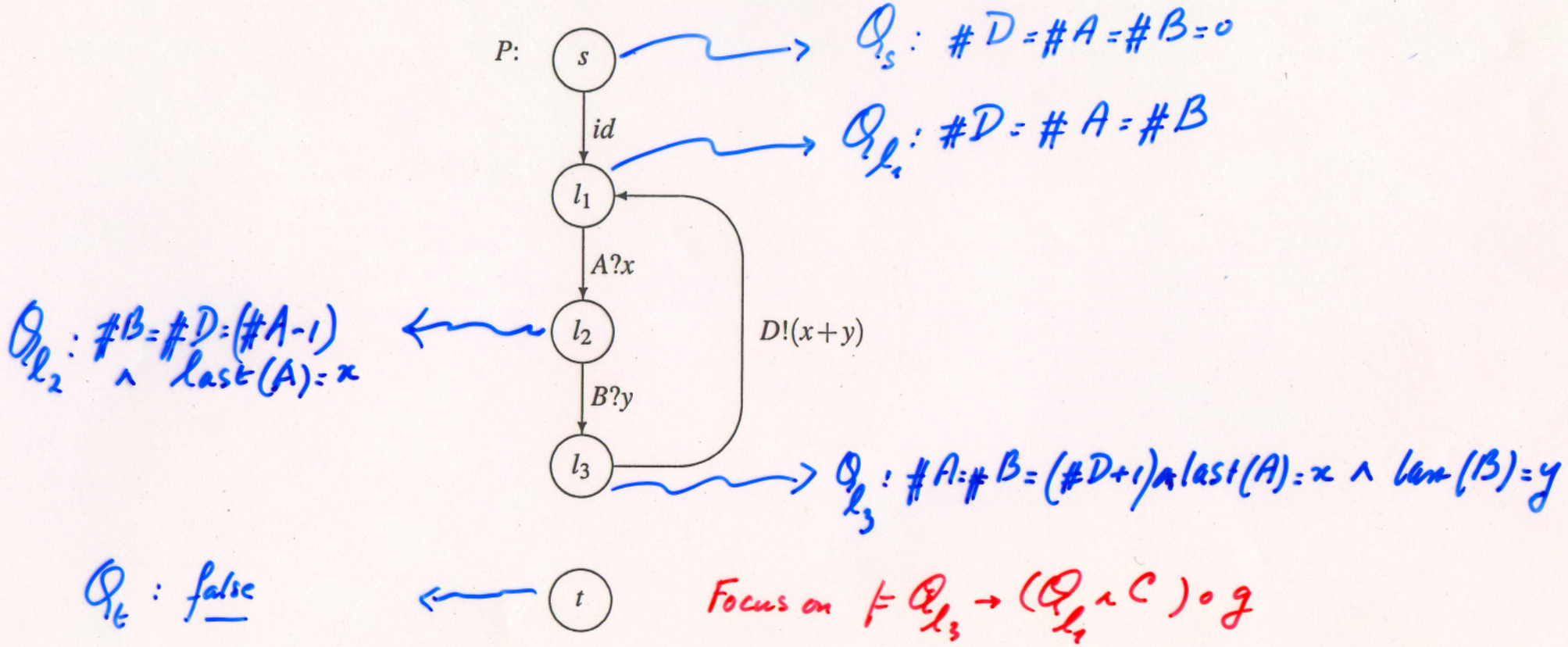


Fig. 6.3. Structure of adder P.

Focus on $\vdash Q_{l_3} \rightarrow (Q_{l_1} \wedge C) \circ g$

with $g(\sigma) = (\sigma : h \mapsto \sigma(h).(D, \sigma(x) + \sigma(y)))$

Application of the Basic Diagram rule : For $B \equiv \langle L, T, s, t \rangle$:

$$\frac{Q(A,C) \vdash B}{\langle A, C \rangle : \{Q_s\}\, B\, \{Q_t\}}$$

establishes the proof

Next, prove the same but with $Ass_1 \stackrel{def}{=} (\#A>0 \rightarrow odd(last(A)) \wedge (\#B>0 \rightarrow odd(last(B))$ 6.7

and $C_1 \stackrel{def}{=} (\#D=\#A=\#B \geq 1 \rightarrow even(last(D))$

the environment always supplies odd numbers

Example 6.1 We continue the example from Figure 6.1 and propose an implementation of P using distributed communication in Figure 6.3.

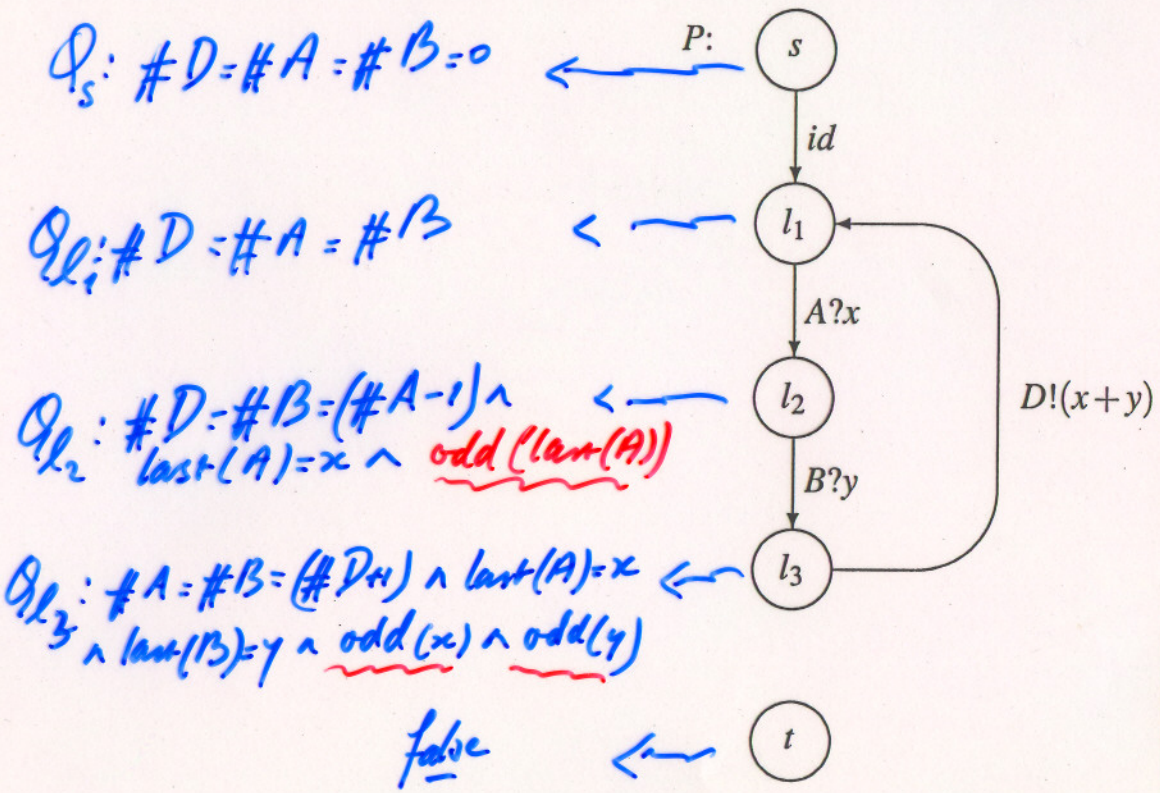


Fig. 6.3. Structure of adder P.

Consider the VC for $(l_2, B?y, l_3)$

$VC: F \varphi_{l_2} \wedge Ass_1 \rightarrow (Ass_1' \rightarrow \varphi_{l_3}') \wedge C_1) \circ g$

with $g(\sigma) \stackrel{def}{=} (\sigma: h \mapsto val).(B, \sigma(y))$

Whereas we can conclude $y = last(B)$ as before, the claim of φ_{l_3} that $odd(last(B))$ holds, cannot be proved without using the info of Ass_1 that also for the last comm. via B

$odd(last(B))$

can be assumed.

$\Rightarrow$

Explains $(Ass_1 \rightarrow \varphi_{l_3})$ clause in VC

Example 7.36 (Continuation of the previous example) The assumption that the environment always provides odd numbers via channels A and B is formalised by

$$Ass_1 \stackrel{\text{def}}{=} (\#A > 0 \to odd(last(A))) \wedge (\#B > 0 \to odd(last(B))).$$

Using this assumption and the following assertion network:

$$\begin{aligned}
Q_s &\stackrel{\text{def}}{=} \quad \#D = \#A = \#B = 0,\\
Q_{l_1} &\stackrel{\text{def}}{=} \quad \#D = \#A = \#B,\\
Q_{l_2} &\stackrel{\text{def}}{=} \quad \#B = \#D = (\#A - 1) \wedge last(A) = x \wedge odd(last(A)),\\
Q_{l_3} &\stackrel{\text{def}}{=} \quad \#A = \#B = (\#D + 1) \wedge last(A) = x \wedge last(B) = y \wedge\\
&\qquad odd(last(A)) \wedge odd(last(B)), \text{ and}\\
Q_t &\stackrel{\text{def}}{=} \quad false,
\end{aligned}$$

one can prove similarly, as above,

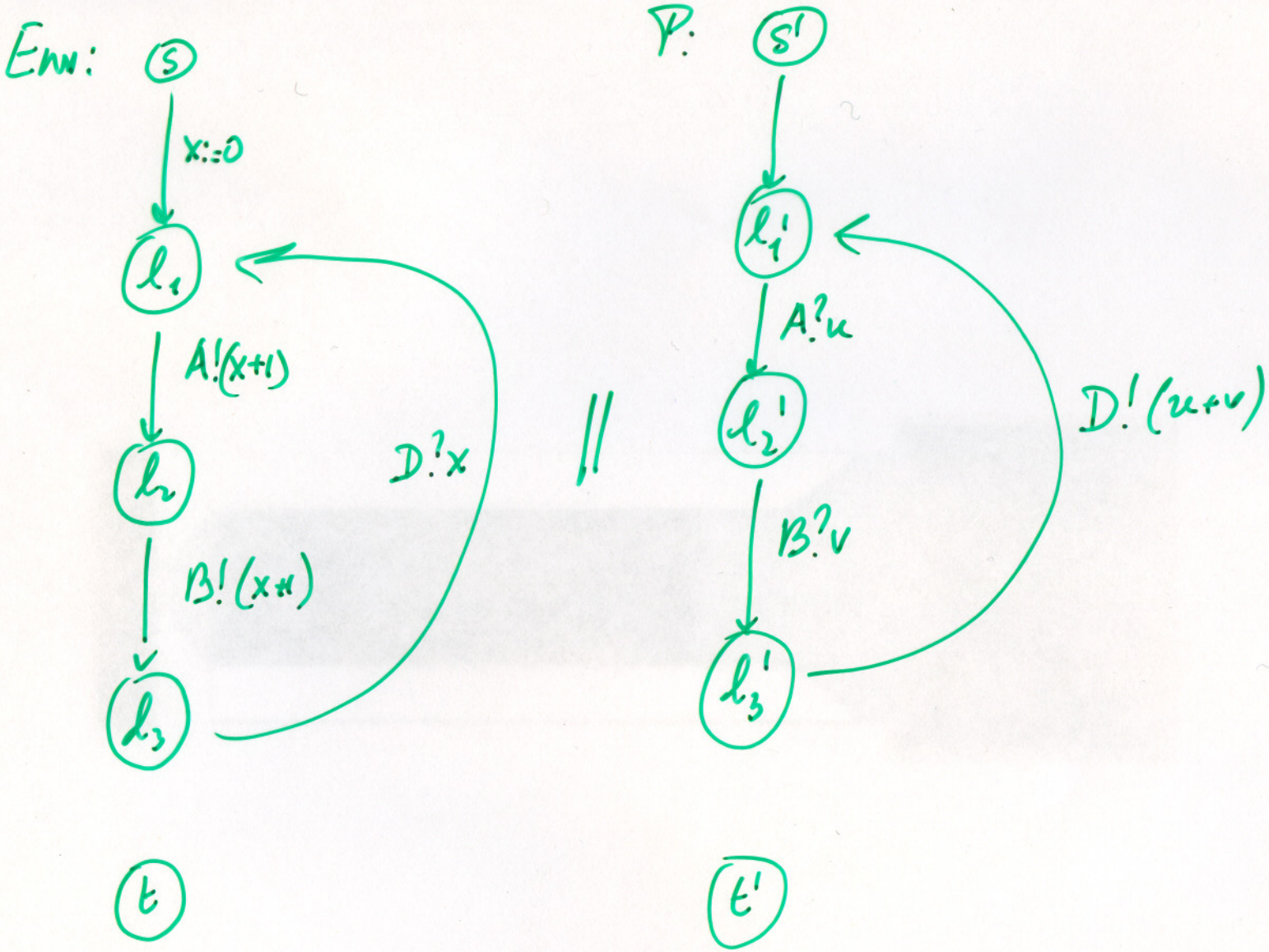
$$\models \langle Ass_1, C_1\rangle : \{\#D = \#A = \#B = 0\} P \{false\},$$

where $C_1 \stackrel{\text{def}}{=} (\#D = \#A = \#B \geq 1 \to even(last(D)))$. Here we observe the use of assumption Ass_1 for proving the verification conditions. Consider, e.g., the transition $(l_2, B?y, l_3)$. The associated verification condition is

$$\models Q_{l_2} \wedge Ass_1 \to ((Ass_1 \to Q_{l_3}) \wedge C_1) \circ g,$$

where $g(\sigma) \stackrel{\text{def}}{=} (\sigma : h \mapsto \sigma(h) \cdot (B, \sigma(y)))$.

While we can conclude that $y = last(B)$ holds as before, the claim of Q_{l_3} that $odd(last(B))$ holds, i.e., that y is an odd number, is not provable without the information of assumption Ass_1 that also for this last communication via channel B one can assume that $odd(last(B))$ holds. □


Env:
s
x:=0
l_1
A!(x+1)
l_2
B!(x+1)
l_3
D?x
t
||
P:
s'
l_1'
A?u
l_2'
B?v
l_3'
D!(u+v)
t'

P has to wait for input via A and B.
First, Env sends 1 via both A and B
Then, P adds these values and sends back 2 via D

Now, Env receives an even value, and sends back odd values.
P receives these odd values a sends their EVEN back, ...
...

This operational explanation why Env||P behaves correctly contains explicitely an induction argument. We call this SPIRAL REASONING (in this context).
Our formal proof using the par. comps. rule DOES NOT DISPLAY THIS IND. ARG.
⇒ it is hidden in the SOUNDNESS PROOF of the par. comp. rule

(Similarly as for Hoare's proof rule for loops)

Rule 7.17 (Parallel Composition)

$$\frac{\begin{array}{c}\langle A_1, C_1\rangle : \{\varphi_1\}\ P_1\ \{\psi_1\},\\ \langle A_2, C_2\rangle : \{\varphi_2\}\ P_2\ \{\psi_2\},\\ A \wedge C_1 \rightarrow A_2, A \wedge C_2 \rightarrow A_1\end{array}}{\langle A, C_1 \wedge C_2\rangle : \{\varphi_1 \wedge \varphi_2\}\ P_1 \| P_2\ \{\psi_1 \wedge \psi_2\}}$$

provided

(i) $var(A_1, C_1, \psi_1) \cap var(P_2) = \emptyset$, $var(A_2, C_2, \psi_2) \cap var(P_1) = \emptyset$, and
(ii) $Chan(A_1, C_1, \psi_1) \cap Chan(P_2) \subseteq Chan(P_1)$,
$Chan(A_2, C_2, \psi_2) \cap Chan(P_1) \subseteq Chan(P_2)$.

Consider the parallel composition $P_1 \| P_2$, and assume we have assumption-commitment pairs (A_1, C_1) satisfied by P_1 and (A_2, C_2) satisfied by P_2. Which conditions have to be verified to obtain a pair (A, C) satisfied by $P_1 \| P_2$? Consider first assumption A_2 of P_2:

- If A_2 contains assumptions about joint channels of P_1 and P_2 which connect these two processes, these assumptions should be justified by the commitment C_1 of P_1.
- If A_2 contains assumptions about external channels of P_2, i.e., channels that are not connected with P_1, these assumptions should be justified by the new network assumption A for $P_1 \| P_2$.

Imposing both these conditions leads to requiring validity of the following verification condition:

$$A \wedge C_1 \rightarrow A_2.$$

Validity of $A \wedge C_2 \rightarrow A_1$ is argued similarly.

The soundness of a parallel composition rule with these implications depends heavily on the definition of validity of the formula $\langle A_i, C_i \rangle : \{\varphi_i\}\ P_i\ \{\psi_i\}$, for $i = 1,2$. Observe that if in this definition one had chosen a simple implication between A_i and C_i to hold instead of $\langle A_i, C_i \rangle : \{\varphi_i\} P_i \{\psi_i\}$, then the above rule would have led to circular reasoning, since then $A_1 \rightarrow C_1 \rightarrow A_2 \rightarrow C_2 \rightarrow A_1$ might have been implied. Because of this reason the rule would have become unsound. To see this, choose $A \equiv \mathit{true}$ and $A_1 \equiv A_2 \equiv C_1 \equiv C_2 \equiv \mathit{false}$. Then in this changed interpretation, the above rule would have implied $\langle \mathit{true}, \mathit{false} \rangle \{\varphi\}\ P\ \{\mathit{false}\}$, which contradicts the intuitive meaning of A-C formulae given earlier.

To avoid such problems, in defining the validity of $\langle A_i, C_i \rangle : \{\varphi_i\}\ P_i\ \{\psi_i\}$, we have required that if φ_i holds in the initial state then:

(i) C_i holds initially, and

(ii) C_i holds after every communication provided A_i holds after all **preceding** communications.

Hence, false cannot be used as the commitment in a valid A-C formula with assumption true, i.e., the definition of the validity of A-C correctness formulae incorporates an induction step. The associated inductive argument is part of the soundness proof of the parallel composition rule, and no longer needs to be given, when applying this rule.

Consequently, the above rule plays the same rôle for the parallel composition of synchronously communicating processes as Hoare's **loop** rule (Rule 9.9 in Chapter 9) plays for iterative constructs.

We deduced: $\vdash \underbrace{\langle (\#A>0 \to odd(last(A))) \wedge (\#B>0 \to odd(last(B))) \rangle}_{Ass_1}\ \underbrace{\{\#D=\#A=\#B=0\}\ P\ \{false\}}_{C_1}$, $\langle \#D=\#A=\#B \geq 1 \to even(last(D)) \rangle$?

Next we construct an env. Q for P s.t. $\vdash \langle true, Ass_1 \rangle : \{\#B=\#A=0\}\ Q\ \{false\}$, i.e., whose C_Q implies Ass_1

Example 6.1 We continue the example from Figure 6.1 and propose an implementation of P using distributed communication in Figure 6.3.

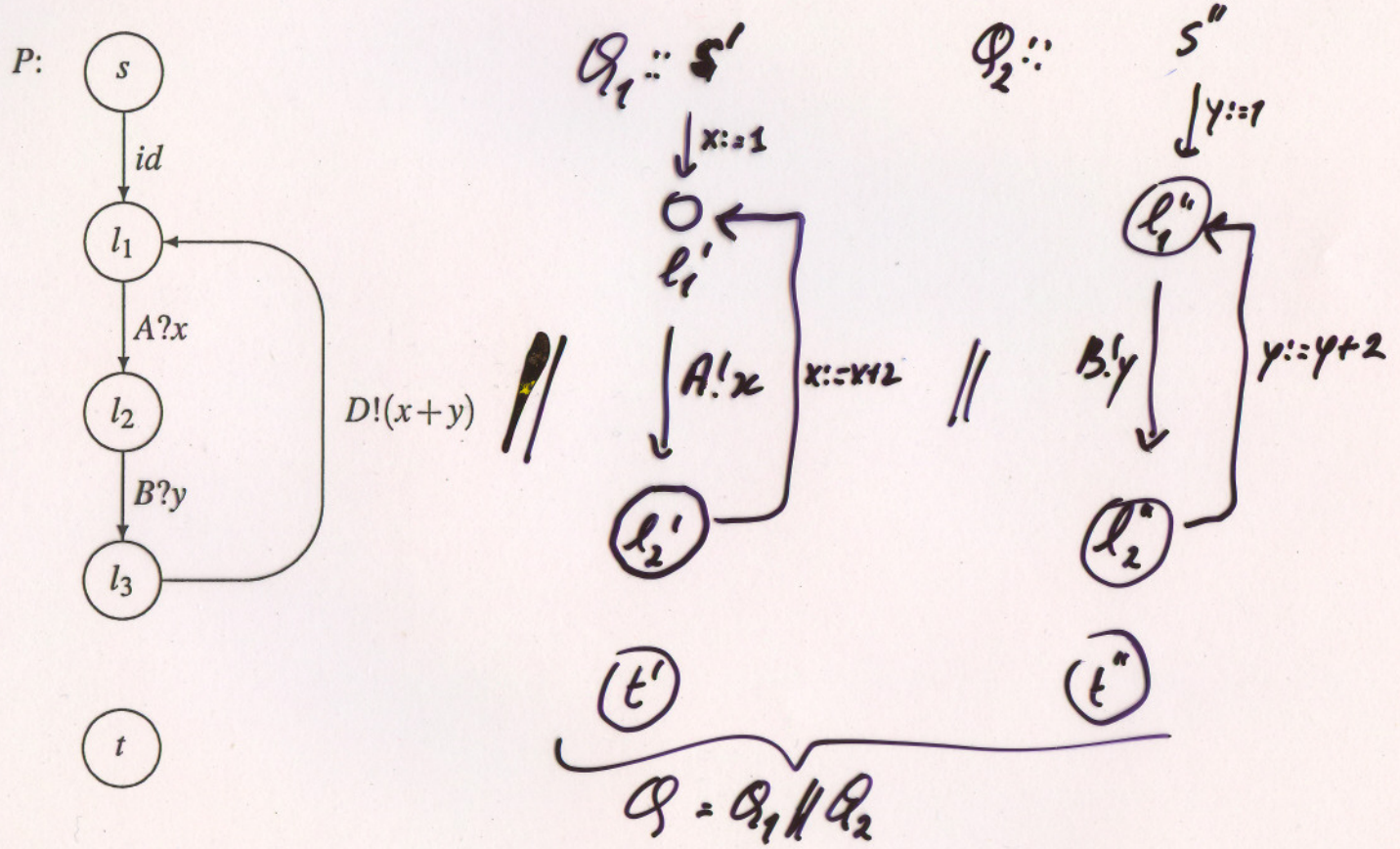


Fig. 6.3. Structure of adder P.

Combining $P \| Q_1 \| Q_2$ leads to a new. with $\langle A, C \rangle \stackrel{def}{=} \langle true, \#D=\#A=\#B \geq 1 \to even(last(D)) \rangle$

$pre \stackrel{def}{=} \#A=\#B=\#D=0$

$post \stackrel{def}{=} false$

We deduced : $\vdash \langle \underbrace{(\#A>0 \to odd(last(A))) \wedge (\#B>0 \to odd(last(B)))}_{Ass_1}, \underbrace{\{\#D=\#A=\#B=0\}\ P\ \{false\}}_{C_1} \rangle$, $\underbrace{(\#D=\#A=\#B \geq 1 \to even(last(D)))}$:

Next we construct an env. Q for P s.t. $\vdash \langle true, Ass_1 \rangle : \{\#B=\#A=0\}\ Q\ \{false\}$, i.e., whose C_Q implies Ass_1

E.g., consider Q_1 satisfying

$\vdash \langle true, \#A \geq 1 \to odd(last(A)) \rangle : \{\#A=0\}\ Q_1\ \{false\}$

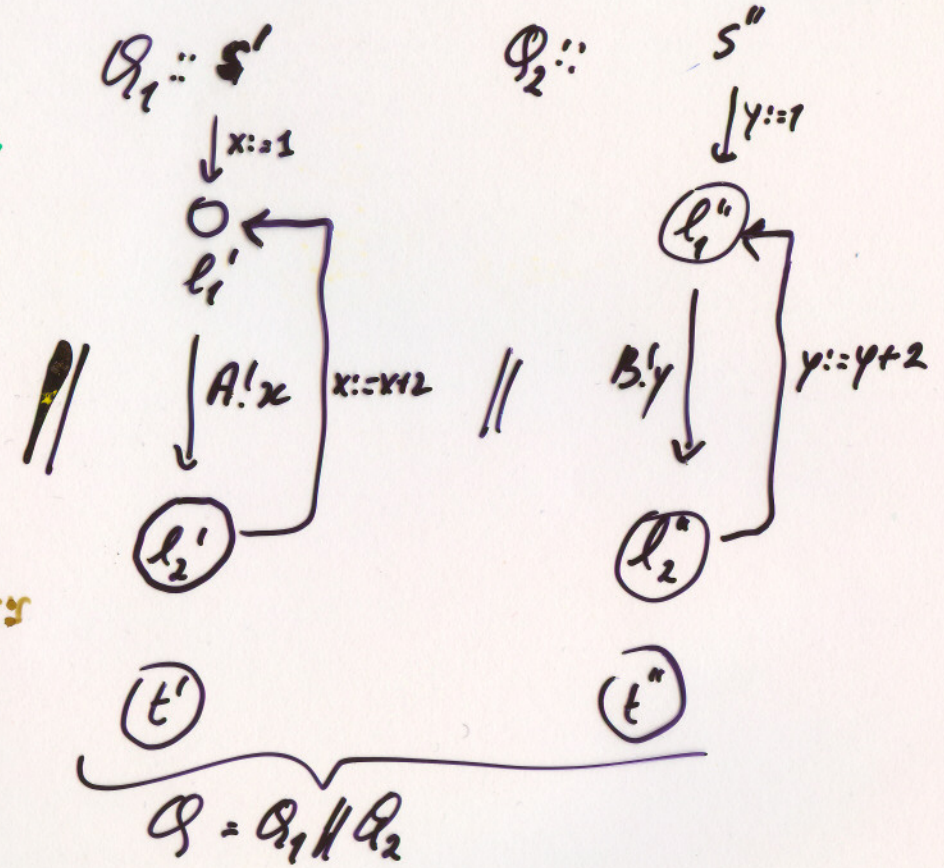
with Q_2 satisf.:

$\vdash \langle true, \#B \geq 1 \to odd(last(B)) \rangle : \{\#B=0\}\ Q_2\ \{false\}$

By appl. the par. comp. rule applied to $Q = Q_1 // Q_2$
we obtain the spec. for Q above.

Since $\vdash true \wedge \underbrace{(\#A \geq 1 \to odd(last(A)))}_{A_{Q_1}} \to \underbrace{true}_{A_{Q_2}}$ } the add. VCs for appl. the par comp rule.

and similarly $\vdash A_{Q_2} \wedge C_{Q_2} \to A_{Q_1}$

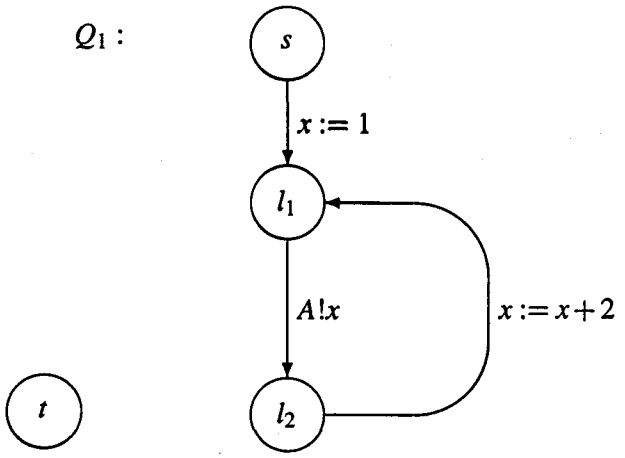


Combining $P // Q_1 // Q_2$ leads to a netw. with $\langle A, C \rangle \overset{def}{=} \langle true, \#D=\#A=\#B \geq 1 \to even(last(D)) \rangle$

$pre \overset{def}{=} \#A=\#B=\#D=0$

$post \overset{def}{=} false$

Example 7.37 Since we want to use P as an even number generator, we have to provide a program Q_1 that sends odd values via channels A and B as required in the assumption Ass_1 in Example 7.36. This program sends odd numbers via



channel A and can serve as part of the environment of P. We can give a local proof of

$$\vdash \langle true, \#A \geq 1 \rightarrow odd(last(A)) \rangle : \{\#A = 0\}\ Q_1\ \{false\}.$$

If we modify the program Q_1 of Figure 7.6 such that the output statement $A!x$ is replaced by $B!y$, and call the resulting process Q_2, then $Q_1 || Q_2$ constitutes an environment in which P will generate only even numbers.

Because

$$\vdash \langle true, \#B \geq 1 \rightarrow odd(last(B)) \rangle : \{\#B = 0\}\ Q_2\ \{false\},$$

by an application of the parallel composition rule we deduce that

$$\vdash \langle true, ((\#A \geq 1 \rightarrow odd(last(A))) \wedge (\#B \geq 1 \rightarrow odd(last(B)))) \rangle : \{\#A = \#B = 0\}\ Q_1 || Q_2\ \{false\},$$

since $\models true \wedge (\#A \geq 1 \rightarrow odd(last(A))) \rightarrow true$ and $\models true \wedge (\#B \geq 1 \rightarrow odd(last(B))) \rightarrow true$.

We see that the commitment of $Q_1 || Q_2$ is exactly the assumption Ass_1 of Example 7.36, and since

$$\models true \wedge ((\#A \geq 1 \rightarrow odd(last(A))) \wedge (\#B \geq 1 \rightarrow odd(last(B)))) \rightarrow Ass_1$$

and

$$\models true \wedge \#D = \#A = \#B \geq 1 \rightarrow even(last(D)) \rightarrow true$$

hold, we can apply the parallel composition rule again to obtain (after some simplifications using the consequence rule):

$$\vdash \langle true, \#D = \#A = \#B \geq 1 \rightarrow even(last(D)) \rangle : \{\#A = \#B = \#D = 0\}\ Q_1 || Q_2 || P\ \{false\}.$$

That is, with Q_1 and Q_2 as the input generating environment, P acts as an even number generator, as desired. □

EX 7.38

The par. composition rule is COMPOSITIONAL, i.e., it only operates on the SPECIFICATIONS of its components, and then deduces a SPEC. of ~~its~~ their parallel comp.

Of P we deduced: A_P

$\langle (\#A>0 \to odd(last(A))) \wedge (\#B>0 \to odd(last(B))),\ \underbrace{(\#D=\#A=\#B \geq 1 \to even(last(D)))}_{C_P} \rangle :: P \ldots$

Put in an env. Q satisfying

$\langle true, A_P \rangle \ldots Q \ldots$

We composed P with Q ... $\Rightarrow$ P||Q satisfying $\langle true, \#D=\#A=\#B \geq 1 \to even\,last(D) \rangle :: C_P \ldots$ P||Q...

Now we offer a different env. to P:

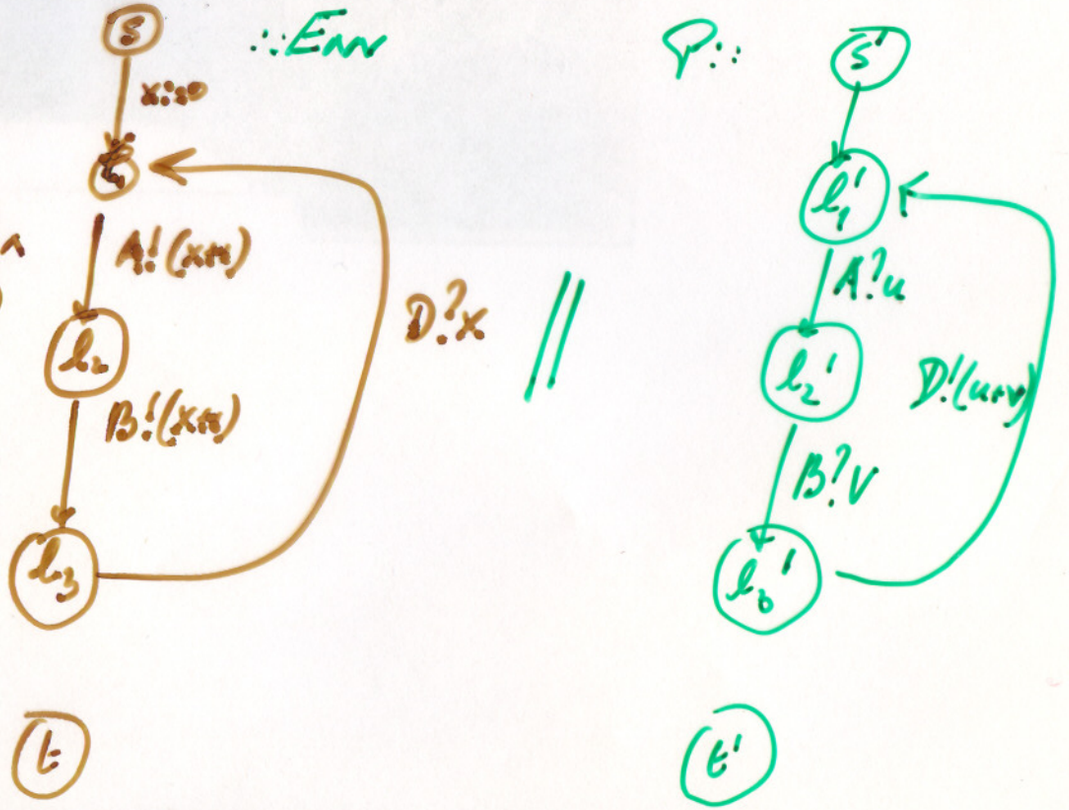
satisfying as commitment

$C_2 \overset{def}{=} (\#A \geq 1 \to (odd(last(A)) \wedge (A,1) \preceq h \wedge \{A\}^{\downarrow})) \wedge$
$(\#B \geq 1 \to (odd(last(B)) \wedge (B,1) \preceq h \wedge \{B\}^{\downarrow}))$

with C_P as assumption, i.e., along D even values are received.

The composition Env || P again satisf

$\langle true, C_P \rangle \ldots Env||P$



7.23

Rule 7.12 (Consequence rule)

$$\frac{\begin{array}{c}\langle A,C\rangle : \{\varphi\}\ P\ \{\psi\} \\ A' \to A,\ \varphi' \to \varphi, \\ C \to C',\ \psi \to \psi'\end{array}}{\langle A',C'\rangle : \{\varphi'\}\ P\ \{\psi'\}}$$

Rule 7.13 (Prefix invariance) Let $cset \subseteq CHAN$ be a set of channels, and $t \in Lvar$.

$$\frac{\langle A,C\rangle : \{\varphi\}\ P\ \{\psi\}}{\langle A, C \wedge t \preceq h{\downarrow}cset\rangle : \{\varphi \wedge t = h{\downarrow}cset\}\ P\ \{\psi \wedge t \preceq h{\downarrow}cset\}}.$$

Rule 7.14 (Assumption closure) Let $cset$ satisfy $Chan(A) \subseteq cset \subseteq CHAN$ and $t \in Lvar$.

$$\frac{\langle A,C\rangle : \{\varphi\}\ P\ \{\psi\}}{\begin{array}{c}\langle A, C \wedge \forall t.(t_0 \preceq t \prec h{\downarrow}cset \to A\{t/h\})\rangle : \\ \{\varphi \wedge t_0 = h{\downarrow}cset\}\ P\ \{\psi \wedge \forall t.(t_0 \preceq t \preceq h{\downarrow}cset \to A\{t/h\})\}\end{array}}$$

Here $A\{t/h\}$ denotes substitution, i.e., $A\{t/h\} \stackrel{\text{def}}{=} A \circ f$, with $f : \Sigma \mapsto \Sigma$, $f(\sigma)(x) \stackrel{\text{def}}{=} \sigma(x)$ for $x \neq h$, and $f(\sigma)(h) \stackrel{\text{def}}{=} \sigma(t)$. Observe that this rule is consistent with Definition 7.32 in that also $A\{t_0/h\}$ is required to hold in the C and ψ parts of its conclusion.

Rule 7.15 (Initialisation)

$$\frac{\langle A,C\rangle : \{\varphi\}\ P\ \{\psi\}}{\langle A,C\rangle : \{\varphi \circ f\}\ P\ \{\psi\}}$$

where f is a function such that its write variables constitute a set of (logical) variables that do not occur in P, A, C, or ψ.

Rule 7.16 (Sequential composition)

$$\frac{\langle A,C\rangle : \{\varphi\}\ P_1\ \{\xi\},\ \langle A,C\rangle : \{\xi\}\ P_2\ \{\psi\}}{\langle A,C\rangle : \{\varphi\}\ P_1;P_2\ \{\psi\}}.$$

Soundness of the basic diagram rule

Theorem 7.41
The basic diagram rule is sound.

Lemma 7.42 Given the above, we have for $i = 0, \ldots, n$ that:

(i) $(\forall \theta' \preceq \theta_i.(\sigma : h \mapsto \sigma(h) \cdot \theta') \models A) \Rightarrow (\sigma_i : h \mapsto \sigma(h) \cdot \theta_i) \models Q_{l_i}$.
(ii) $(\forall \theta' \prec \theta_i.(\sigma : h \mapsto \sigma(h) \cdot \theta') \models A) \Rightarrow (\sigma_i : h \mapsto \sigma(h) \cdot \theta_i) \models C$, where $Q_{l_0} = Q_s$, and $Q_{l_n} = Q_t$.

Soundness of the sequential composition rule

Theorem 7.43
The sequential composition rule 7.16 is sound.

Soundness of the parallel composition rule

Theorem 7.44
The parallel composition rule 7.17 is sound.

Theorem 7.45

$$\models \langle A,C\rangle : \{\varphi\}\ P\ \{\psi\} \Rightarrow \vdash \langle A,C\rangle : \{\varphi\}\ P\ \{\psi\}.$$

The proof of this theorem proceeds by induction on the structure of program P. First we discuss the structure of this proof.

Given program P and restrictions on the environment consisting of an assumption A and a precondition φ, we will construct the *strongest postcondition* w.r.t. φ, A and P, and the *strongest commitment* w.r.t. φ, A and P, in order to characterise precisely all reachable states of P which are consistent with the environment specified by A and φ, and all possible traces of P which are consistent with that environment.

These two kinds of predicates will be used for constructing A-C-inductive assertion networks for basic transition diagrams. For given precondition φ, assumption A and basic synchronous transition diagram B, they allow us to generate the A-C-inductive assertion network $Q(A,SC(\varphi,A,B))$ w.r.t. assumption A and strongest commitment $SC(\varphi,A,B)$, by associating with each node l of B the strongest postcondition $SP_l(\varphi,A,B)$. That this yields an A-C-inductive assertion network will be established in Lemma 7.49 below, and allows one to

apply the basic diagram rule 7.11, after which some applications of the consequence rule yield the desired result.

To establish completeness for composite systems P one proceeds inductively. In this part we shall apply the usual notion of strongest postcondition $SP(\varphi,A,P)$ generalised to our setting.

The simple case is that of sequential composition: $P \equiv P_1;P_2$.

Here the induction hypothesis will be used to prove:

(i) completeness w.r.t. φ and A for P_1, and
(ii) completeness w.r.t. $SP(\varphi,A,P_1)$ and A for P_2,

after which the sequential composition rule will be applied to establish

$$\vdash \langle A,C\rangle : \{\varphi\}\ P_1;P_2\ \{SP(SP(\varphi,A,P_1),A,P_2)\}.$$

Using that $SP(SP(\varphi,A,P_1),A,P_2) = SP(\varphi,A,P_1;P_2)$ holds as usual, the result then follows from Lemma 7.53 (ii).

The most interesting case is $P \equiv P_1 || P_2$.
The problem here is that when

$$\models \langle A, C \rangle : \{\varphi\}\ P_1 || P_2\ \{\psi\}$$

holds, in general C cannot be equivalently expressed as $C_1 \wedge C_2$, and, similarly, ψ cannot be written as $\psi_1 \wedge \psi_2$, where $C_i, \psi_i, i = 1,2$, satisfy the restrictions imposed in the parallel composition rule.

As a result, the A-C formulae which can be proved directly using this rule are too weak. In fact, a second rule, the prefix-invariance rule, is needed to overcome some of these limitations. This can be seen as follows.

One restriction mentioned in the parallel composition rule is that

$$Chan(A_i, C_i, \psi_i) \cap Chan(P_j) \subseteq Chan(P_i),\ i \neq j,\ i, j = 1,2,$$

i.e., A_i, C_i and ψ_i do not involve channels of P_j which are not connected to P_i. In general, this rule only allows us to deduce commitments of the form $C_1 \wedge C_2$ and postconditions of the form $\psi_1 \wedge \psi_2$. However, when C_1 and ψ_1 refer to a channel D of P_1 which is not connected to P_2, and C_2 and ψ_2 to a channel E of P_2 which is not connected to P_1, the relative order to the communications over D and E prior to execution of $P_1 || P_2$ cannot be expressed this way (this order may have been referred to in φ). It is here that the application of the prefix-invariance axiom is crucial to establish completeness.

Another problem arises from the fact that, as remarked previously, validity of A-C formulae embodies an induction argument (this has been worked out in [ZdBdR84]). This, together with the fact that $\models A \wedge C_i \rightarrow A_j$, $i \neq j$, $i, j = 1,2$, occurs in the premise of the parallel composition rule, suggests a

mutually recursive relationship between C_i and A_j, whose explicit expression would considerably complicate proofs. To get around this problem a purely combinatorial trick is used, which is based on the assumption-closure rule, and on Lemma 7.55, which captures this recursive dependency between C_i and A_j as a property of the underlying data domain, which on the level of our completeness proof can be incorporated using the consequence rule.

As a consequence, the case $P \equiv P_1 || P_2$ of our completeness proof has a rather combinatorial flavour.

Definition 7.46 For $l \neq s$,

$$SP_l(\varphi,A,B) \stackrel{\text{def}}{=} \{\sigma \mid \exists \sigma_0,\sigma',\theta.(\sigma_0,\sigma',\theta) \in O_l(B) \wedge \sigma = (\sigma' : h \mapsto \sigma_0(h)\cdot\theta) \wedge \sigma_0 \models \varphi \wedge \forall \theta' \preceq \theta.(\sigma_0 : h \mapsto \sigma_0(h)\cdot\theta') \models A\}$$

$$SP_s(\varphi,A,B) \stackrel{\text{def}}{=} \{\sigma \mid \sigma \models \varphi\}.$$

□

Definition 7.47

$$SC(\varphi,A,B) \stackrel{\text{def}}{=} \{\sigma \mid \exists l,\sigma_0,\sigma',\theta.(\sigma_0,\sigma',\theta) \in O_l(B) \wedge \sigma(h) = \sigma_0(h)\cdot\theta \wedge \sigma_0 \models \varphi \wedge \forall \theta' \prec \theta.(\sigma_0 : h \mapsto \sigma_0(h)\cdot\theta') \models A\}.$$

□

Note that $SC(\varphi,A,B)$ is indeed a trace predicate.

Lemma 7.48

(i) $\models \langle A, SC(\varphi,A,B)\rangle : \{\varphi\}\ B\ \{SP_t(\varphi,A,B)\}$.
(ii) $\models \langle A, C\rangle : \{\varphi\}\ B\ \{\psi\} \Rightarrow$
  (a) $\models SP_t(\varphi,A,B) \rightarrow \psi$.
  (b) $\models SC(\varphi,A,B) \rightarrow C$.

Lemma 7.49 $\mathcal{SP} : l \mapsto SP_l(\varphi,A,B)$ is an A-C-inductive assertion network w.r.t. A and $SC(\varphi,A,B)$ w.r.t. B.

After these preliminaries we start with our induction proof of Theorem 7.45 and consider the case that program B is a basic synchronous transition diagram (L,T,s,t) such that

$$\models \langle A,C\rangle : \{\varphi\}\ B\ \{\psi\}.$$

For predicates A and φ we construct SP_l predicates which form by Lemma 7.49 an A-C-inductive assertion network for B w.r.t. A and $SC(\varphi,A,B)$. Thus we can apply the basic diagram rule 7.11 and derive

$$\vdash \langle A,SC(\varphi,A,B)\rangle : \{SP_s(\varphi,A,B)\}\ B\ \{SP_t(\varphi,A,B)\}.$$

Since by Lemma 7.48 $\models SP_t(\varphi,A,B) \to \psi$ and $\models SC(\varphi,A,B) \to C$ hold, and $\models \varphi \to SP_s(\varphi,A,B)$ by definition of SP_s, we derive by an application of the consequence rule the desired result

$$\vdash \langle A,C\rangle : \{\varphi\}\ B\ \{\psi\}.$$

This establishes completeness in the sense of Theorem 7.45 for basic synchronous transition diagrams.

Strongest postcondition and strongest commitment for composed programs

We adapt Definitions 7.46 and 7.47 to the semantics of Definition 7.30.

Definition 7.50

$$\begin{aligned} SP(\varphi,A,P) \stackrel{\text{def}}{=} \{\sigma \mid \exists \sigma_0,\sigma',\theta.(\sigma_0,\sigma',\theta,\top) \in O[\![P]\!] \\ \wedge \sigma = (\sigma' : h \mapsto \sigma_0(h)\cdot\theta) \\ \wedge \sigma_0 \models \varphi \wedge \forall \theta' \preceq \theta.(\sigma_0 : h \mapsto \sigma_0(h)\cdot\theta') \models A \}. \end{aligned}$$ □

Remark 7.51 For a basic diagram B we have $\models SP_t(\varphi,A,B) \Leftrightarrow SP(\varphi,A,B)$.

Definition 7.52

$$\begin{aligned} SC(\varphi,A,P) \stackrel{\text{def}}{=} \{\sigma \mid \exists \sigma_0,\sigma',\theta,\tau.(\sigma_0,\sigma',\theta,\tau) \in O[\![P]\!] \\ \wedge \sigma(h) = \sigma_0(h)\cdot\theta \\ \wedge \sigma_0 \models \varphi \wedge \forall \theta' \prec \theta.(\sigma_0 : h \mapsto \sigma_0(h)\cdot\theta') \models A \}. \end{aligned}$$ □

The strongest postcondition SP and strongest commitment SC satisfy the following properties:

Lemma 7.53

(i) $\models \langle A, SC(\varphi,A,P)\rangle : \{\varphi\}\ P\ \{SP(\varphi,A,P)\}$.

(ii) $\models \langle A,C\rangle : \{\varphi\}\ P\ \{\psi\} \Rightarrow$

  (a) $\models SP(\varphi,A,P) \rightarrow \psi$.

  (b) $\models SC(\varphi,A,P) \rightarrow C$.

Proof Similar to that of Lemma 7.48. □