Verifikation nebenläufiger Programme
Wintersemester 2004/05

Ulrich Hannemann Jan Bredereke

19 Proof Method and Application

19.1 A-C-inductive assertion networks

We adapt Floyd's method to the additional requirements of A-C formulae and define for $B \equiv (L, T, s, t)$ an A-C-inductive assertion network $\mathcal{Q}(A, C) : L \to \mathcal{P}(\Sigma)$, i.e., we associate with each location l of B a predicate $\mathcal{Q}_l$ as follows:

Definition 19.1 (A-C-inductive assertion networks) $\mathcal{Q}$ is an A-C-inductive assertion network w.r.t. A and C for $B \equiv (L, T, s, t)$ if:

- $\models \mathcal{Q}_s \to C$.
- In case of an internal transition $l \xrightarrow{a} l' \in T$, $a \equiv b \to f$, we require
$$\models \mathcal{Q}_l \wedge b \wedge A \to \mathcal{Q}_{l'} \circ f.$$
- In case of an output transition $l \xrightarrow{a} l' \in T$, $a \equiv b; D!e \to f$, we require, for $v = e(\sigma)$,
$$\models \mathcal{Q}_l \wedge A \wedge b \to ((A \to \mathcal{Q}_{l'}) \wedge C) \circ (f \circ g),$$
where $g(\sigma) \overset{\text{def}}{=} (\sigma : h \mapsto \sigma(h) \cdot (D, v))$.
- 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 $v \in VAL$,
$$\models \mathcal{Q}_l \wedge A \wedge b \to ((A \to \mathcal{Q}_{l'}) \wedge C) \circ (f \circ g),$$
where $g(\sigma) \overset{\text{def}}{=} (\sigma : x, h \mapsto v, \sigma(h) \cdot (D, v))$. □

Remark 19.2 In the last two clauses of this definition we require only $(A \to \mathcal{Q}_{l'})$ to hold after the transition, since the idea behind A-C reasoning is to use the assumption A whenever possible.

We abbreviate the existence of an A-C-inductive assertion network $\mathcal{Q}$ w.r.t. A and C for B by $\mathcal{Q}(A, C) \vdash B$. We have the following rule for deriving A-C specifications of basic transition diagrams.

Rule 19.1 (Basic diagram rule) For $B \equiv \langle L, T, s, t \rangle$:
$$\frac{\mathcal{Q}(A, C) \vdash B}{\langle A, C \rangle : \{\mathcal{Q}_s\}\ B\ \{\mathcal{Q}_t\}}.$$

Example 19.3 (Even number generator) We demonstrate the application of the method by verifying the first specification of Example 18.1, i.e., for the program P of Figure 1 of Session 18 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:

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

We have to check five implications to prove that the assertion network above is an A-C-inductive assertion network w.r.t. assumption $Ass \equiv true$ and commitment $C \equiv (\#D = \#A = \#B \geq 1 \rightarrow last(D) = last(A) + last(B))$.

- $\models \mathcal{Q}_s \rightarrow C$ and $\models \mathcal{Q}_s \rightarrow \mathcal{Q}_{l_1}$ follow from the definition above.
- $\models \mathcal{Q}_{l_1} \rightarrow (\mathcal{Q}_{l_2} \wedge C) \circ g$, where $g(\sigma) \stackrel{\text{def}}{=} (\sigma : x, h \mapsto v, \sigma(h) \cdot (A, v))$ for arbitrary v, holds since only a communication via A took place and, consequently, the value received is the last value received via A. Since $\#A \neq \#B = \#D$ holds, we have also satisfied C. (Note that, since the assumption is identically true, it suffices to check that $\mathcal{Q}_{l_2} \circ g$ holds.)
- $\models \mathcal{Q}_{l_2} \rightarrow (\mathcal{Q}_{l_3} \wedge C) \circ g$, where $g(\sigma) \stackrel{\text{def}}{=} (\sigma : x, h \mapsto v, \sigma(h) \cdot (B, v))$ for arbitrary v, holds analogously.
- The most interesting verification condition is related to the output transition of P:
  $$\models \mathcal{Q}_{l_3} \rightarrow (\mathcal{Q}_{l_1} \wedge C) \circ g,$$
  where $g(\sigma) \stackrel{\text{def}}{=} (\sigma : h \mapsto \sigma(h) \cdot (D, \sigma(x) + \sigma(y)))$.
  We have one communication via channel D and, hence, $\mathcal{Q}_{l_1}$ is satisfied. Furthermore, $last(D) = x + y = last(A) + last(B)$, and, hence, C holds after the transition.

The application of the basic diagram rule establishes the correctness formula above. □

Example 19.4 (Continuation of the previous example) The assumption that the environment always provides odd numbers via channels A and B is formalised by

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

Using this assumption and the following assertion network:

$$\begin{aligned} \mathcal{Q}_s &\stackrel{\text{def}}{=} \#D = \#A = \#B = 0,\\ \mathcal{Q}_{l_1} &\stackrel{\text{def}}{=} \#D = \#A = \#B,\\ \mathcal{Q}_{l_2} &\stackrel{\text{def}}{=} \#B = \#D = (\#A - 1) \wedge last(A) = x \wedge odd(last(A)),\\ \mathcal{Q}_{l_3} &\stackrel{\text{def}}{=} \#A = \#B = (\#D + 1) \wedge last(A) = x \wedge last(B) = y \wedge\\ &\qquad odd(last(A)) \wedge odd(last(B)),\ \text{and}\\ \mathcal{Q}_t &\stackrel{\text{def}}{=} false, \end{aligned}$$

one can prove similarly, as above,

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

where $C_1 \overset{\text{def}}{=} (\#D = \#A = \#B \geq 1 \to even(last(D)))$. 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

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

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

While we can conclude that $y = last(B)$ holds as before, the claim of $\mathcal{Q}_{l_3}$ 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. $\square$

19.2 Some general rules

For a valid formula $\langle A, C\rangle : \{\varphi\}\ P\ \{\psi\}$ we can always weaken the commitment or the postcondition without invalidating the formula. Symmetrically we can assure the same commitment/postcondition whenever we impose stricter restrictions on P's environment, i.e., the assumption and the precondition. This justifies the consequence rule below.

Rule 19.2 (Consequence rule)

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

Another observation on valid assumption-commitment formulae is that we cannot change the communication history that was generated *before* the process at hand was invoked.

Rule 19.3 (Prefix invariance) Let $cset \subseteq CHAN$ be a set of channels, and $t \in Lvar$.

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

An immediate consequence of the definition of the validity of A-C formulae is the fact that for any valid formulae not only the commitment holds after any communication action but also the assumption holds for all proper prefixes up to that point, i.e., the latter is an additional commitment.

Rule 19.4 (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 18.4 in that also $A\{t_0/h\}$ is required to hold in the C and ψ parts of its conclusion.

Additionally we have a conjunction rule and an initialisation rule.

Rule 19.5 (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.

19.3 Composition rules

The rule for sequential composition of processes is standard.

Rule 19.6 (Sequential composition)

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

Next we discuss the proof obligations for assumptions and commitments in the parallel composition rule.

Rule 19.7 (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 $P_1 \parallel P_2$.

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:

1. C_i holds initially, and
2. 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.

Derivability of an A-C formula $\langle A, C\rangle : \{\varphi\}\ P\ \{\psi\}$ in this A-C proof method is expressed by

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

19.4 Application of the above proof method

We illustrate the assumption-commitment paradigm by continuing Example 19.3, and construct the environment for process P as in Figure 1 of Session 18.

Example 19.5 Since we want to use P as an even number generator, we have to provide a program $\mathcal{Q}_1$ that sends odd values via channels A and B as required in the assumption Ass_1 in Example 19.4. 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

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

If we modify the program Q_1 of Figure 1 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

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

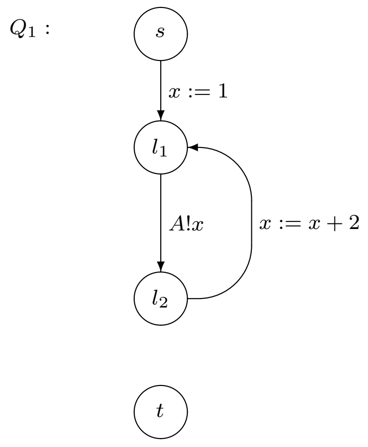

Figure 1: Odd numbers generator.

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 19.4, 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. $\square$

Next we illustrate the case of mutually dependent processes when the output of P is an input for Q and vice versa.

Example 19.6 We have a process Env as the environment for P, getting input via D and sending values via A and B. The output of process Env depends on its inputs via channel D, except for the first values sent via A and B. We now assume that all inputs via D are even values by defining assumption $Ass_2 \overset{\text{def}}{=} \#D \geq 1 \rightarrow even(last(D))$.

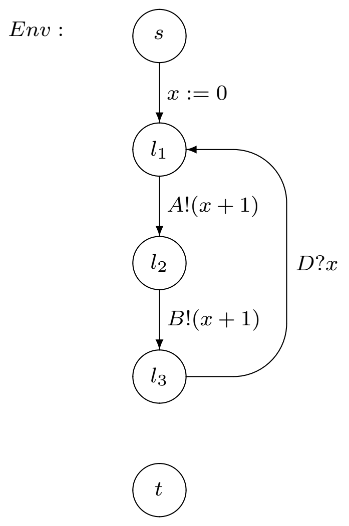

Figure 2: Another odd numbers generator.

Under this assumption all outputs via A and B are odd numbers, as expressed by commitment

$$\begin{aligned} C_2 \overset{\text{def}}{=} & (\#A \geq 1 \rightarrow (odd(last(A)) \wedge (A,1) \preceq h{\downarrow}\{A\})) \wedge \\ & (\#B \geq 1 \rightarrow (odd(last(B)) \wedge (B,1) \preceq h{\downarrow}\{B\})). \end{aligned}$$

For process Env we prove

$$\vdash \langle Ass_2, C_2 \rangle : \{\#A = \#B = \#D = 0\}\ Env\ \{false\}.$$

We are still interested in the property that all values transmitted via channel D are even values. The program $Env \| P$ satisfies this property as can be seen as follows:

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 odd values back, P receives these odd values and sends their (even) sum back, etc.

This intuitive explanation why the program $Env \| P$ behaves correctly contains explicitly an induction argument. Our formal proof of the property mentioned does not require such an inductive argument explicitly. Our only proof obligations are

$$\models true \wedge C_2 \rightarrow Ass_1$$

and

$$\models true \wedge (\#D = \#A = \#B \geq 1 \rightarrow even(last(D))) \rightarrow Ass_2.$$

Applying the parallel composition rule (and some simplifications) then leads to

$$\begin{gathered} \vdash \langle true, (\#D = \#A = \#B \geq 1 \rightarrow even(last(D))) \rangle : \\ \{\#A = \#B = \#D = 0\}\ Env \| P\ \{false\}. \end{gathered} \tag{1}$$

The assumption-commitment paradigm allows for compositional reasoning. In particular, while verifying one component of a parallel program, some other components may not have been implemented yet – we only use their specifications. Note that the parallel composition rule only refers to the specification of P and Env and not to their implementation, and is therefore compositional.

Example 19.7 Consider the process R in Figure 3, which is just a slight modification of process Env above. Process R can replace process Env as environment

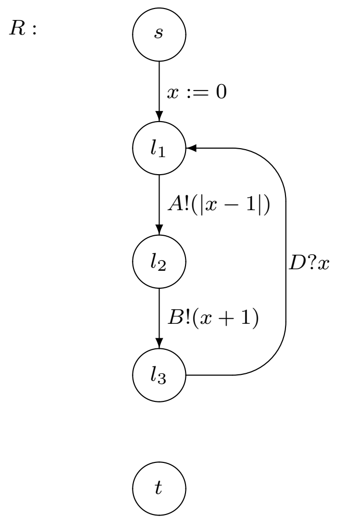

Figure 3: And yet another odd numbers generator.

for P since one can also prove that

$$\langle Ass_2, C_2\rangle : \{\#A = \#B = \#D = 0\}\ R\ \{false\}$$

is a valid correctness formula, and hence $P\|R$ will also satisfy

$$\begin{array}{c}\models \langle true, (\#D = \#A = \#B \geq 1 \rightarrow even(last(D)))\rangle : \\ \{\#A = \#B = \#C = 0\}\ P\|R\ \{false\}.\end{array} \tag{2}$$

Example 19.8 Whereas all previous formulae refer only to the last value sent via a channel, process R allows another use of program P: in environment R, P will send 2^i as the i-th value over D (here we abbreviate the i-th value sent along channel D by $D[i]$).

We get as the specification for the adder P

$$\langle Ass_3, C_3\ \rangle : \{\#A = \#B = \#D = 0\}\ P\ \{false\}$$

with $C_3 \overset{\text{def}}{=} \#D \geq 1 \rightarrow D[i] = 2^i$, and

$$Ass_3 \overset{\text{def}}{=} (\#A \geq 1 \rightarrow A[i] = 2^{(i-1)} - 1) \wedge (\#B \geq 1 \rightarrow B[i] = 2^{(i-1)} + 1).$$

Since P is still the same program, what are the restrictions on the environment to guarantee that output behaviour?

P just adds the values previously received via A and B. If we assume that we receive as $A[i]$ the value $2^{(i-1)} - 1$ and as $B[i]$ the value $2^{(i-1)} + 1$ then we have that

$$D[i] = A[i] + B[i] = (2^{(i-1)} - 1) + (2^{(i-1)} + 1) = 2 * 2^{(i-1)} = 2^i$$

as desired.

Now similarly as in the odd-even case, P and R mutually depend on each other's output. What we have to prove now for R is:

$$\vdash \langle C_3, Ass_3 \rangle : \{\#A = \#B = \#D = 0\}\ R\ \{false\},$$

which is done again by verifying an appropriate A-C-inductive assertion network. □