Verifikation nebenläufiger Programme
Wintersemester 2004/05

Ulrich Hannemann Jan Bredereke

17 Compositionality

17.1 Compositionality: the Concept

The technical property which is required of program correctness methods for supporting the verify-while-develop paradigm is called *"compositionality"*, whose definition is recalled below:

> *"That a program meets its specification should be verified on the basis of specifications of its constituent components only, without additional need for information about the interior construction of those components."*

To make this verification strategy possible, programs and their components are specified using predicates *over their observable behaviour, only.* Such specifications are called *assertional.* Consequently, assertional specifications of the constituent components of a program never depend on any additional knowledge about the underlying execution mechanism of those components. Consequently, *compositional verification should be based on an assertional specification language.*

What does a compositional proof method consist of?

It consists of:

1. *Basic techniques* for proving that a program P, which is not decomposed any further, satisfies φ.
2. *Compositional proof techniques* to handle the case that P is composed of parts $P_1, \ldots, P_n$, i.e., $P = op^{lang}(P_1, \ldots, P_n)$, $n \geq 1$, with op^{lang} some operator of the programming language.

For the latter kind of program operators we develop *compositional proof rules*, i.e., logical inference rules of the form:

> "From P_1 *satisfies* φ_1 and $\ldots P_n$ *satisfies* φ_n infer P *satisfies* φ."
> [Zwi89]

In order to do so we need to be more specific about assertional specifications through predicates. They should not refer to the interior construction of components. Therefore, the available information for specifying a component consists only of:

- a description of the desired observable behaviour of the component P_i (as given by the assertional specification φ_i), and
- a description of the interface $obs(P_i)$ of that component P_i.

This leads to the introduction of restrictions upon the specifications φ_i of P_i, whose purpose it is to guarantee that φ_i is invariant under independent execution of P_j, $j \neq i$. These restrictions concern the sets of observables allowed to occur within φ_i, and impose that those observables, which are involved in independent executions of P_j, $j \neq i$, are disjoint from the observables of φ_i.

These considerations lead to the following definition of a *compositional proof rule for a program operator* $op^{lang}(P_1, \ldots, P_n)$:

$$\frac{\begin{array}{c}\textit{for } i = 1, \ldots, n,\ P_i \textit{ satisfies } \varphi_i \textit{ and } obs(P_i) \subseteq O_i,\\ op^{spec}(\varphi_1, \ldots, \varphi_n, \varphi; obs(P_1), \ldots, obs(P_n))\end{array}}{op^{lang}(P_1, \ldots, P_n) \textit{ satisfies } \varphi,}$$

where $obs(P_i)$ refers to the interface of P_i, i.e., P_i's set of observables, $O_1, \ldots, O_n$ denotes given sets of observables, $\varphi_1, \ldots, \varphi_n$, and φ expresses assertional specifications, and $op^{spec}(\varphi_1, \ldots, \varphi_n, \varphi; obs(P_1), \ldots, obs(P_n))$ expresses points (i) and (ii) above[1].

Various proof methods are considered, each giving separate interpretations to φ, *"P satisfies φ"*, $obs(P_i) \subseteq O_i$, op^{lang}, and op^{spec}. For each of these methods it is essential that the *basic technique* to deal with programs which are not decomposed further (although adaptation rules may be applied) are formulated and, in particular, that appropriate (meta-)predicates $op^{spec}(\varphi_1, \ldots, \varphi_n, \varphi; obs(P_1), \ldots, obs(P_n))$ for the compositional proof rules contained in the respective proof method are identified.

17.2 A Compositional Proof Method

As an illustration of this strategy, we introduce compositionally-inductive assertion networks for reasoning about the sequential components P_i of a parallel system $P \equiv P_1 \| \ldots \| P_n$. Then we introduce compositional proof rules for deducing properties of the whole system. The basic idea of a compositionally-inductive assertion network is the definition of the verification conditions of P_i in terms of a single *logical* history variable h which records the sequence of communications generated by that component.

Here one distinguished *history variable* h is assumed to exist, $h \in VAR$. A state $\sigma \in \Sigma$ thus assigns as the meaning to h a sequence of communications, i.e., $\sigma(h) \in (CHAN \times VAL)^*$ denotes a sequence of communication records, which are pairs consisting of a channel name and a value. The idea behind this is that history variable h represents the sequence of communications of the given concurrent system. For every basic synchronous transition diagram P we require for every state transformation f and boolean condition b of P that $VAR(f) \subseteq (VAR \backslash \{h\})$ and $VAR(b) \subseteq (VAR \backslash \{h\})$. This requirement formalises the condition that the history variable h does not occur in any program.

The parallel combination of these compositionally-inductive assertion networks is defined in terms of a simple semantic characterisation of the variables and channels involved in a predicate. The notion of channels involved in a predicate is defined in terms of a natural generalisation of the projection operation on histories to predicates.

[1]The clause $P_i \subseteq O_i$ is added here to cover restrictions on the components, e.g., the case in shared-variable concurrency where a process is required to have exclusive write-access to a variable. Its general purpose is to be able to impose further restrictions on $P_1, \ldots, P_n$, although these are not considered here.

In order to model an input statement $C?x$ which involves the assignment of an *arbitrary* value to x, we need the introduction of quantifiers.

For a predicate φ we define $\sigma \models \forall x.\varphi$ iff for every semantic expression $e : \Sigma \to \mathit{VAL}$, we have $(\sigma : x \mapsto e(\sigma)) \models \varphi$, and similarly $\sigma \models \exists x.\varphi$ by $\sigma \models \neg\forall x.\neg\varphi$.

Definition 17.1 (Compositionally-inductive assertion network) The local assertion network $\mathcal{Q}$ for a *sequential* synchronous transition diagram $P \equiv (L, T, s, t)$ is called *compositionally inductive* if:

- For $l \xrightarrow{a} l'$ a local transition of P, i.e., $a \equiv b \to f$ for some boolean b and state transformation f, one has

$$\models \mathcal{Q}_l \wedge b \to \mathcal{Q}_{l'} \circ f.$$

- For $l \xrightarrow{a} l'$ an output transition of P, i.e., $a \equiv b; C!e \to f$, for some boolean b, channel C and state transformation f, one has

$$\models \mathcal{Q}_l \wedge b \to \mathcal{Q}_{l'} \circ (f \circ g),$$

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

- For $l \xrightarrow{a} l'$ an input transition of P, i.e., $a \equiv b; C?x \to f$, for some boolean b, channel C and state transformation f, one has

$$\models \mathcal{Q}_l \wedge b \to \forall x.\mathcal{Q}_{l'} \circ (f \circ g),$$

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

We denote by $P \vdash \mathcal{Q}$ that $\mathcal{Q}$ is a compositionally-inductive assertion network for P. □

Definition 17.2 A *partial correctness statement* is of the form $\{\varphi\}\ P\ \{\psi\}$, where φ and ψ are predicates, also called the precondition and postcondition, which involve, amongst others, logical variable h, and P denotes either a synchronous transition diagram or a parallel system. □

Formally, validity of a partial correctness statement $\{\varphi\}\ P\ \{\psi\}$, notation: $\models \{\varphi\}\ P\ \{\psi\}$, is defined now with respect to the semantics $\mathcal{O}$.

Definition 17.3 (Validity of partial correctness formulae) We define $\models \{\varphi\}\ P\ \{\psi\}$, with $P = (L, T, s, t)$ a sequential synchronous transition diagram, by:

for every $(\sigma, \sigma', \theta) \in \mathcal{O}_t(P)$ such that $(\sigma : h \mapsto \langle\rangle) \models \varphi$, we have

$$(\sigma' : h \mapsto \theta) \models \psi.$$

Similarly, we define $\models \{\varphi\}\ P\ \{\psi\}$, for P a parallel system, in terms of $\mathcal{O}(P)$ by:

for every $(\sigma, \sigma', \theta) \in \mathcal{O}(P)$ such that $(\sigma : h \mapsto \langle\rangle) \models \varphi$, we have

$$(\sigma' : h \mapsto \theta) \models \psi.$$

□

In this way the validity of a partial correctness statement $\{\varphi\}\ P\ \{\psi\}$ is defined with respect to computations of P which start in a state in which the history variable h is initialised to the empty sequence, reflecting the fact that we only consider top-level synchronous networks. We can impose this restriction because we do not consider the sequential composition of parallel systems (that is, we consider only top-level parallelism).

Rule 17.1 (Basic diagram rule) For $P \equiv (L, T, s, t)$ a sequential synchronous transition diagram, we have the following basic diagram rule:

$$\frac{P \vdash \mathcal{Q}}{\{\mathcal{Q}_s\}\ P\ \{\mathcal{Q}_t\}}$$

Moreover, for synchronous transition diagrams we have the following initialisation rule.

Rule 17.2 (Initialisation rule)

$$\frac{\{\varphi \wedge (h = \langle\rangle)\}P\{\psi\}}{\{\varphi\}P\{\psi\}}$$

where $\sigma \models h = \langle\rangle \Leftrightarrow \sigma(h) = \langle\rangle$.

The choice of a single history variable h enforces restrictions on the parallel composition rule which serve the same purpose as the condition on disjointness of the (program) variables. In order to define a rule for parallel composition we introduce the following operation on predicates, known as *chaotic closure* [Zwi89].

We formulate these conditions in terms of a predicate also *involving*, apart from program variables, *channel names*. We introduce the chaotic closure $\varphi \uparrow \bar{C}$ of φ w.r.t. $\bar{C}$, where $\bar{C}$ expresses a set of channel names, to indicate that the value of φ, as far as h is concerned, only depends on the *projection* of the global history h on the channels $\bar{C}$. Formally, we have the following definition.

Definition 17.4 (Semantic characterisation of channel dependency) Let φ be a predicate and $\bar{C}$ a set of channels. We denote by the *chaotic closure* $\varphi \uparrow \bar{C}$ the predicate

$$\begin{aligned}\sigma \models \varphi \uparrow \bar{C} \Leftrightarrow \quad & \text{there exists } \sigma' \models \varphi \text{ s.t. } \sigma(x) = \sigma'(x), \text{for } x \in \mathit{VAR} \setminus \{h\}, \\ & \text{and } \sigma(h){\downarrow}\bar{C} = \sigma'(h){\downarrow}\bar{C}.\end{aligned}$$

That is, if $\sigma \models \varphi$ then $\sigma' \models \varphi \uparrow \bar{C}$ holds for all σ' which are obtained from σ by interleaving $\sigma(h)$ with *arbitrary* communication records (D, ν) with $D \notin \bar{C}$, thus justifying the name *chaotic* closure.

Note that $\varphi \uparrow \bar{C} = \varphi$ indicates that, as far as the dependency of the value of φ upon the value of h is concerned, the value of φ only depends on the *projection* of the global history h on the channels $\bar{C}$. More formally, $\varphi \uparrow \bar{C} = \varphi$ indicates that for every σ and σ', such that σ and σ' are the same, except that $\sigma(h){\downarrow}\bar{C} = \sigma'(h){\downarrow}\bar{C}$ holds for the values which they assign to the history variable h, we have

$$\sigma \models \varphi \text{ if and only if } \sigma' \models \varphi.$$

If $\varphi \uparrow \bar{C} = \varphi$ then we also say that 'φ *only involves the channels of* $\bar{C}$'. $\mathit{Chan}(\varphi)$ is defined as the minimal set of channels $\bar{C}$ such that $\varphi \uparrow \bar{C} = \varphi$. $\square$

Example 17.5 In this example we explain the definition of the chaotic closure operator, and the notion that 'φ *only involves the channels of* $\bar{C}$'.

1. Which channels are involved in $(h\downarrow\{C\}) = \langle C, 0\rangle$? In $((h \downarrow \{C\}) = \langle C, 0\rangle) \uparrow \{C\}$ we interleave the value of h with arbitrary communication records over channels different from C. However, the projection operator $\downarrow\{C\}$ applied to the resulting communication sequence projects all communications not involving channel C. That is, the values of $(h\downarrow\{C\} = \langle C, 0\rangle) \uparrow \{C\}$ and $(h\downarrow\{C\} = \langle C, 0\rangle)$ are the same. This implies that $((h\downarrow\{C\}) = \langle C, 0\rangle)$ only involves channel C.

2. Which channels are involved in $(h = \langle C, 0\rangle)$? In $(h = \langle C, 0\rangle) \uparrow \{C\}$ we interleave the value of h, as above, with arbitrary communication records over channels different from C. This influences the value of $(h = \langle C, 0\rangle)$, because these arbitrary interleavings are visible through the occurrence of h in $(h = \langle C, 0\rangle)$. Hence, $(h = \langle C, 0\rangle) \uparrow \{C\}$ and $(h = \langle C, 0\rangle)$ are different. Therefore, $(h = \langle C, 0\rangle)$ does not involve $\{C\}$.

3. Considering $h = \langle C, 0\rangle$, again, we ask ourselves which channels are involved in $h = \langle C, 0\rangle$. In $(h = \langle C, 0\rangle) \uparrow (\text{CHAN} \setminus \{C\})$ we interleave h with arbitrary communication records involving C, only. This leads to values, which are different from those of $h = \langle C, 0\rangle$ without those interleavings. That is, only by considering the chaotic closure of $h = \langle C, 0\rangle$ with respect to the full set of available channels CHAN, i.e., only by *not interleaving h with any communication records at all*, do we obtain $(h = \langle C, 0\rangle) \uparrow \text{CHAN} = (h = \langle C, 0\rangle)$. This implies that $h = \langle C, 0\rangle$ involves (the whole of) CHAN, which is a consequence of the unprojected occurrence of h in $h = \langle C, 0\rangle$. $\Box$

Similarly, as in Session 2, we restrict ourselves in the applications of the proof methods discussed in the remainder of this chapter to predicates φ for which there exist finite sets of channels which they involve. For such predicates φ one can prove that $Chan(\varphi)$ exists and is finite.

We can now formulate the following rule for parallel composition.

Rule 17.3 Let $P \equiv P_1 \parallel P_2$ in the rule

$$\frac{\{\varphi_1\}\ P_1\ \{\psi_1\},\ \{\varphi_2\}\ P_2\ \{\psi_2\}}{\{\varphi_1 \wedge \varphi_2\}\ P\ \{\psi_1 \wedge \psi_2\}},$$

provided ψ_i does not involve the variables of P_j and $\psi_i \uparrow Chan(P_i) = \psi_i$, $i \neq j$. (Recall that $Chan(P_i)$ indicates the channels of P_i.)

The following example shows that the restriction on channels is necessary.

Example 17.6 Consider a network $C!0 \parallel D!0$ (abstracting from the locations of the components). Locally, we can prove

$$\{h = \langle\rangle\}\ C!0\ \{h = \langle(C, 0)\rangle\} \text{ and } \{h = \langle\rangle\}\ D!0\ \{h = \langle(D, 0)\rangle\}.$$

Applying the above rule (without checking whether its restrictions hold) leads to

$$\{h = \langle\rangle\}\ C!0 \parallel D!0\ \{\mathit{false}\},$$

which gives rise to incorrect results when we further compose the open system $C!0 \parallel D!0$. However, we observe that this specification of $C!0 \parallel D!0$ does hold (and can be derived by the AFR-method) when we view $C!0 \parallel D!0$ as a *closed* system (i.e., under the semantics $\mathcal{M}$). In the example above we derive that, e.g., postcondition $h = \langle (C, 0) \rangle$ involves *all* channels, and, hence, the conditions upon the postconditions ψ_i in the above rule for parallel composition are violated. $\square$

We conclude the exposition of the proof method with the following consequence and initialisation rules.

Rule 17.4 For P a sequential synchronous transition diagram or a parallel system we have the usual consequence rule:

$$\frac{\varphi \to \varphi',\ \{\varphi'\}\ P\ \{\psi'\},\ \psi' \to \psi}{\{\varphi\}\ P\ \{\psi\}}.$$

Rule 17.5 For P a parallel system we have the initialisation rule:

$$\frac{\{\varphi\}\ P\ \{\psi\}}{\{\varphi \circ f\}\ P\ \{\psi\}},$$

where f is a state function such that its write variables are not involved in P or ψ.

We denote derivability of a partial correctness statement $\{\varphi\}\ P\ \{\psi\}$, using the rules above, we denote by $\vdash \{\varphi\}\ P\ \{\psi\}$.

17.3 Application

Example 17.7 We consider the parallel system $P \equiv P_1 \| P_2 \| P_3$ and prove that

$$\vdash \{true\}\ P\ \{x = 1 \wedge y = 2 \wedge z = 0\}.$$

P_1:

$$\begin{tikzcd}
s_1 \arrow[r, "A!0"] & l_1^1 \arrow[r, "B!1"] & t_1
\end{tikzcd}$$

P_2:

$$\begin{tikzcd}
s_2 \arrow[r, "B?x"] & l_2^1 \arrow[r, "C!(x+1)"] & t_2
\end{tikzcd}$$

P_3:

$$\begin{tikzcd}
& l_3^1 \arrow[dr, "C?y"] & \\
s_3 \arrow[ur, "A?z"] \arrow[dr, "C?y"'] & & t_3 \\
& l_3^2 \arrow[ur, "A?z"'] &
\end{tikzcd}$$

As the assertion network we use for P_1 the following set of predicates which is compositionally inductive:

$$\begin{aligned}
\mathcal{Q}_{s_1} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{A,B\} = \langle\rangle \\
\mathcal{Q}_{l_1^1} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{A,B\} = \langle(A,0)\rangle \\
\mathcal{Q}_{t_1} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{A,B\} = \langle(A,0),(B,1)\rangle.
\end{aligned}$$

Similarly we have for P_2:

$$\begin{aligned}
\mathcal{Q}_{s_2} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{B,C\} = \langle\rangle \\
\mathcal{Q}_{l_2^1} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{B,C\} = \langle(B,x)\rangle \\
\mathcal{Q}_{t_2} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{B,C\} = \langle(B,x),(C,x+1)\rangle.
\end{aligned}$$

And for P_3:

$$\begin{aligned}
\mathcal{Q}_{s_3} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{A,C\} = \langle\rangle \\
\mathcal{Q}_{l_3^1} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{A,C\} = \langle(A,z)\rangle \\
\mathcal{Q}_{l_3^2} &\stackrel{\text{def}}{=} \quad h{\downarrow}\{A,C\} = \langle(C,y)\rangle \\
\mathcal{Q}_{t_3} &\stackrel{\text{def}}{=} \quad (h{\downarrow}\{A,C\} = \langle(A,z),(C,y)\rangle \vee h{\downarrow}\{A,C\} = \langle(C,y),(A,z)\rangle).
\end{aligned}$$

Applying Rule 17.1 we derive

$$\begin{aligned}
&\vdash \{h{\downarrow}\{A,B\} = \langle\rangle\}\ P_1\ \{h{\downarrow}\{A,B\} = \langle(A,0),(B,1)\rangle\} \\
&\vdash \{h{\downarrow}\{B,C\} = \langle\rangle\}\ P_2\ \{h{\downarrow}\{B,C\} = \langle(B,x),(C,x+1)\rangle\} \\
&\vdash \{h{\downarrow}\{A,C\} = \langle\rangle\}\ P_3\ \{(h{\downarrow}\{A,C\} = \langle(A,z),(C,y)\rangle \\
&\qquad\qquad\qquad\qquad\qquad \vee h{\downarrow}\{A,C\} = \langle(C,y),(A,z)\rangle)\}.
\end{aligned}$$

Observe that the restrictions on the parallel composition rule 17.3 are satisfied, and we derive

$$\begin{gathered}
\vdash\ \{h{\downarrow}\{A,B\} = \langle\rangle \wedge h{\downarrow}\{B,C\} = \langle\rangle\} \\
P_1 \| P_2 \\
\{h{\downarrow}\{A,B\} = \langle(A,0),(B,1)\rangle \wedge h{\downarrow}\{B,C\} = \langle(B,x),(C,x+1)\rangle\}.
\end{gathered}$$

Since

$$\models h{\downarrow}\{A,B,C\} = \langle\rangle \rightarrow h{\downarrow}\{A,B\} = \langle\rangle \wedge h{\downarrow}\{B,C\} = \langle\rangle$$

and

$$\begin{aligned}
\models\ & (h{\downarrow}\{A,B\} = \langle(A,0),(B,1)\rangle \wedge h{\downarrow}\{B,C\} = \langle(B,x),(C,x+1)\rangle) \rightarrow \\
& h{\downarrow}\{A,C\} = \langle(A,0),(C,2)\rangle \wedge x = 1,
\end{aligned}$$

we obtain by the consequence rule

$$\vdash \{h{\downarrow}\{A,B,C\} = \langle\rangle\}\ P_1 \| P_2\ \{h{\downarrow}\{A,C\} = \langle(A,0),(C,2)\rangle \wedge x = 1\}.$$

Again we apply the parallel composition rule to $(P_1 \| P_2)$ and P_3 and derive

$$\begin{gathered}
\vdash \{h{\downarrow}\{A,B,C\} = \langle\rangle \wedge h{\downarrow}\{A,C\} = \langle\rangle\} \\
(P_1 \| P_2) \| P_3 \\
\{\ h{\downarrow}\{A,C\} = \langle(A,0),(C,2)\rangle \wedge x = 1 \wedge \\
(h{\downarrow}\{A,C\} = \langle(A,z),(C,y)\rangle \vee h{\downarrow}\{A,C\} = \langle(C,y),(A,z)\rangle)\}.
\end{gathered}$$

Now

$$\models (h{\downarrow}\{A,B,C\} = \langle\rangle) \rightarrow (h{\downarrow}\{A,B,C\} = \langle\rangle \wedge h{\downarrow}\{A,C\} = \langle\rangle),$$

and the postcondition above implies

$$x = 1 \wedge y = 2 \wedge z = 0.$$

By the consequence rule we derive, therefore,

$$\vdash \{h{\downarrow}\{A,B,C\} = \langle\rangle\}\ P_1 \| P_2 \| P_3\ \{x = 1 \wedge y = 2 \wedge z = 0\},$$

and the initialisation rule leads to

$$\vdash \{true\}\ P_1 \| P_2 \| P_3\ \{x = 1 \wedge y = 2 \wedge z = 0\}. \qquad \Box$$

References

[Zwi89] J. Zwiers. *Compositionality and Partial Correctness*, volume 321 of *LNCS*. Springer-Verlag, 1989.