Verifikation nebenläufiger Programme
Wintersemester 2004/05

Ulrich Hannemann   Jan Bredereke

12 Towards Total Correctness

12.1 Proving Convergence for Shared-Variable Concurrency

As mentioned before, diagram $P_1 \parallel \ldots \parallel P_n$ is φ-convergent if it does not admit infinite φ-computations. In this section, we extend the method for proving convergence of sequential transition diagrams from the last chapter to the parallel composition of such diagrams operating upon shared variables. We shall do this along the same lines as we have done in generalising Floyd's inductive assertion method for proving partial correctness of sequential diagrams to the Owicki & Gries method for proving partial correctness of concurrent shared-variable diagrams, as worked out in the previous sessions.

12.1.1 A Global Method

Since the semantics of a concurrent transition diagram is given by constructing an equivalent sequential transition diagram (up to observations based on interleaving), one immediately gets a global method for proving convergence by adapting Floyd's wellfoundedness method (Definition 5.1) to the resulting diagram.

Definition 12.1 (Global method for proving convergence) Let L be the set of *global* locations $l \in L$ of $P \equiv P_1 \parallel \ldots \parallel P_n$. To prove that P is φ-convergent:

1. Find an assertion network $\mathcal{Q}$ for the *global* locations of P, and prove that it is inductive, and that $\models \varphi \rightarrow \mathcal{Q}_s$ holds, where s denotes the initial location of P.

2. Choose a wellfounded set $(W, \prec)$ and a network of partially-defined ranking functions ρ for the global locations of P such that

$$\rho_l : \Sigma \rightarrow W \text{ for every } l \in L.$$

3. Prove *local consistency,* of Q_l and ρ_l, i.e., that, for every $l \in L$, $\mathcal{Q}_l$ implies that ρ_l is defined, i.e., for every $\sigma \in \Sigma$,

$$\sigma \models \mathcal{Q}_l \text{ implies } \rho_l(\sigma) \in W$$

holds.

4. Prove that for $l, l' \in L$, every transition $l \xrightarrow{a} l'$ of P, with $a = b \rightarrow f$, decreases the ranking function, i.e.,

$$\models \mathcal{Q}_l \wedge b \rightarrow \rho_l \succ (\rho_{l'} \circ f). \qquad \Box$$

Soundness and semantic completeness of this global method for proving convergence follow from the corresponding proofs for the sequential method and from the definition of the semantics of the parallel composition of diagrams.

Therefore, we wish to improve on that by examining a more localised approach. Our task is again to find a way to formulate the verification conditions for the whole program in terms of verification conditions for its composing processes.

12.1.2 A Local Method

Looking at the problem from another angle, we assume that for each process P_i there exists a local assertion network and a network of local ranking functions to a wellfounded set $(W_i, \prec_i)$. Our task is to find verification conditions for (global) convergence in terms of these local quantities. Since we have the global method already at our disposal, we look at whether we can derive a global proof for convergence from these local conditions. Now there are two networks to be constructed: one consisting of assertions, and one of ranking functions.

The global assertion network can be derived in the same way as we have done for proving partial correctness: for a global location $l = \langle l_1, \dots l_n \rangle$, define

$$\mathcal{Q}_l \stackrel{\text{def}}{=} \mathcal{Q}_{l_1} \wedge \dots \wedge \mathcal{Q}_{l_n}.$$

The following two conditions, namely, local correctness and the interference freedom test:

$$\models \mathcal{Q}_l \wedge b \rightarrow \mathcal{Q}_{l'} \circ f, \tag{1}$$

$$\models \mathcal{Q}_l \wedge \mathcal{Q}_{l''} \wedge b \rightarrow \mathcal{Q}_{l''} \circ f, \tag{2}$$

for any transition $l \xrightarrow{a} l'$ of P_i, with $a = b \rightarrow f$ and location l'' of P_j, for $j \neq i$, ensure that the network is inductive by Theorem 10.3.

Definition 12.2 (Componentwise order) Let $W \stackrel{\text{def}}{=} W_1 \times \dots \times W_n$ and $(W_i, \prec_i)$ be wellfounded sets, $i = 1, \dots n$. For
$(w_1, \dots, w_n)$, $(w'_1, \dots, w'_n) \in W$, define the componentwise order as follows:

$$(w_1, \dots, w_n) \prec (w'_1, \dots, w'_n) \text{ iff } \exists i. w_i \prec_i w'_i \text{ and } \forall j \neq i. w_j \preceq_j w'_j. \qquad \square$$

We leave it as an exercise to prove that $(W, \prec)$ is also a wellfounded set.

For global location $l = \langle l_1, \dots, l_n \rangle$, we define

$$\rho_l \stackrel{\text{def}}{=} (\rho_{l_1}, \dots, \rho_{l_n}).$$

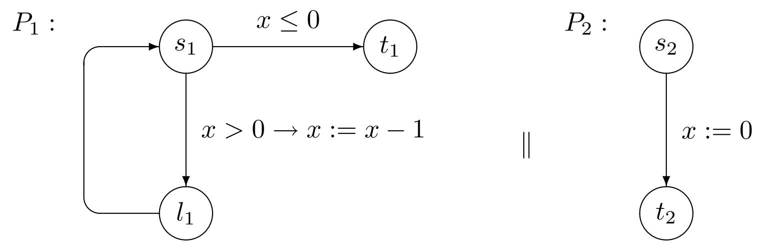
To prove that this function indeed decreases along the transitions of P, the following two additional conditions on local correctness and the interference-freedom test for the ranking functions suffice:

$$\models \mathcal{Q}_l \wedge b \rightarrow \rho_l \succ_i (\rho_{l'} \circ f), \tag{3}$$

$$\models \mathcal{Q}_l \wedge \mathcal{Q}_{l''} \wedge b \rightarrow \rho_{l''} \succeq_j (\rho_{l''} \circ f), \tag{4}$$

for any transition $l \xrightarrow{a} l'$ of P_i, with $a = b \rightarrow f$, and location l'' of P_j, for $j \neq i$.

Example 12.3 As a first step, we show that the following simple program $P_1 \| P_2$ is convergent.



Let the value of x be a natural number. Define all the assertions to be simply *true* and the ranking-function network as follows:

$$\begin{array}{ll} \rho_{s_1}(x) \overset{\text{def}}{=} (x,1) & \rho_{s_2}(x) \overset{\text{def}}{=} 1 \\ \rho_{l_1}(x) \overset{\text{def}}{=} (x,2) & \rho_{t_2}(x) \overset{\text{def}}{=} 0 \\ \rho_{t_1}(x) \overset{\text{def}}{=} (0,0), & \end{array}$$

where the wellfounded set for the first ranking-function network is the lexicographical ordering of $I\!N \times I\!N$ over $(I\!N, <)$, and for the second ranking-function network is $(I\!N, <)$ itself. Local correctness and interference freedom are both trivially satisfied. □

Definition 12.4 (A local method for proving convergence) Consider $P \equiv P_1\| \ldots \|P_n$. To prove that P converges (i.e., terminates or deadlocks) w.r.t. precondition φ, we define the following local method for proving convergence of the parallel composition of transition diagrams P_i:

1. Augment P by introducing auxiliary variables; every action $b \rightarrow f$ can be extended as follows: $b \rightarrow f \circ g$, where $g(\bar{z})$ is a state transformation involving only the auxiliary variables $\bar{z}$ not occurring in P or φ. This leads to an augmented transition system $P' \equiv P'_1 \parallel \ldots \parallel P'_n$.

2. Associate a predicate $\mathcal{Q}_l$ with every location l of P'_i. Choose a wellfounded set $(W_i, \prec_i)$ and a network of partial ranking functions $\rho \overset{\text{def}}{=} \{\rho_l \mid l \in P'_i\}$. Prove that the assertion and ranking functions are *locally* consistent, namely, predicate $\mathcal{Q}_l$ implies that ranking function ρ_l is defined:
$$\models \mathcal{Q}_l(\sigma) \Rightarrow \rho_l(\sigma) \in W_i, \text{ for } \sigma \in \Sigma.$$

3. Prove *local correctness* of every P'_i. This involves proving two properties: that the local assertion network of P'_i is inductive, and that the ranking functions decrease along every local transition. That is, for every transition $l \xrightarrow{a} l'$ of P'_i, with $a = b \rightarrow f$, we have to prove:
$$\left.\begin{array}{ll} \bullet & \models \mathcal{Q}_l \wedge b \rightarrow \mathcal{Q}_{l'} \circ f \\ \bullet & \models \mathcal{Q}_l \wedge b \rightarrow \rho_l \succ_i (\rho_{l'} \circ f) \end{array}\right\} \text{(LC).}$$

4. Prove *interference freedom* of both the local assertion networks and the local networks of ranking functions, where the latter amounts to showing that the ranking functions in one process do not increase due to transitions in other processes. That is, for every transition $l \xrightarrow{a} l'$ of P'_i, with $a = b \rightarrow f$, and location l'' of P'_j, for $j \neq i$, prove that:
$$\left.\begin{array}{ll} \bullet & \models \mathcal{Q}_l \wedge \mathcal{Q}_{l''} \wedge b \rightarrow \mathcal{Q}_{l''} \circ f \\ \bullet & \models \mathcal{Q}_l \wedge \mathcal{Q}_{l''} \wedge b \rightarrow \rho_{l''} \succeq_j (\rho_{l''} \circ f) \end{array}\right\} \text{(IFT).}$$

5. Prove that:

    - $\models \varphi \rightarrow (\bigwedge_i \mathcal{Q}_{s_i}) \circ f$ for some state transformation f whose write variables belong to the set of auxiliary variables $\bar{z}$. Here s_i denotes the initial location of P'_i. $\Box$

Soundness of the local method for proving convergence follows in a straightforward manner from the soundness of the global method.

Theorem 12.5 (Soundness)
The local method for proving convergence given in Definition 12.4 is sound.

Semantic completeness of the method for proving convergence is shown in [dRdBH+01]. Assume $P \equiv P_1 \parallel \ldots \parallel P_n$ is φ-convergent, then our task is to find a proof using the proposed local verification method. However, the (compositional) completeness proof of the Owicki & Gries method for proving partial correctness, as described in Session 11, *cannot* be extended with appropriate ranking functions to a completeness proof of the local method for proving convergence. This is because the (local) predicates used in the compositional completeness proof describe the behaviour of a process with respect to *any* environment. However, convergence of a process clearly depends on a *particular* environment. Therefore one has to construct an alternative completeness proof for proving partial correctness which is based on so-called *reachability* predicates, which do take the particular given environment of a process into account.

This proof is omitted here.

12.2 Proving Deadlock Freedom

A transition is said to be *enabled* if its boolean condition is satisfied. A process is said to be *blocked* at a certain location if it has not terminated and none of its transitions are enabled there. When blocked, a process waits until some other process causes one of its transitions (i.e., of the former process) to become enabled. For concurrent programs, it is not harmful to allow some of its processes to be blocked from time to time, and actually such blocking is essential to the correct functioning of some algorithms – this is called synchronisation. However, if adequate care is not exercised, it might happen that at a certain point all the processes of a system become blocked. Then no process can make a move anymore, and the system is said to be *deadlocked*. When we looked at proving partial correctness and convergence, we considered deadlocked behaviour as acceptable. However, in general, deadlock is clearly undesirable, for almost all the concurrent programs that we are aware of are either supposed to terminate normally or run forever, that is, they should never end up in deadlock. In this section a method for verifying deadlock freedom is investigated, that is, for proving that deadlock does not occur. Sometimes deadlock freedom is also called absence of blocking, or success.

Suppose the program under consideration is $P_1 \parallel \cdots \parallel P_n$, its precondition is φ, and for each process P_i there exists a local assertion network $\mathcal{Q}_i$ which satisfies steps 1 through 4 of the Owicki & Gries method plus the first clause of step 5. Process P_i can only be blocked in state σ at local location l from which there are m transitions with boolean conditions $c_1, \cdots, c_m$ respectively, if

$$b_l \stackrel{\text{def}}{=} \mathcal{Q}_l \wedge \neg(c_1 \vee \cdots \vee c_m)$$

holds in σ. The complete program $P_1 \parallel \cdots \parallel P_n$ is blocked, if some of its processes have terminated, while the remaining processes (at least one) are blocked. Let L_i be the set of locations of P_i and t_i be its exit location. Introducing the predicate

$$B_i \overset{\text{def}}{=} \bigvee_{l \in L_i \setminus \{t_i\}} b_l,$$

deadlock can only occur in a state σ if

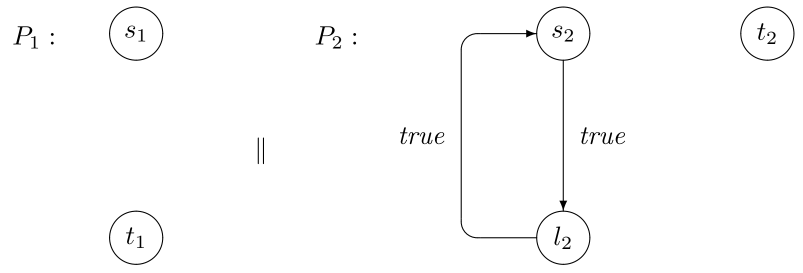
$$\models (\bigwedge_{i=1}^{n} (\mathcal{Q}_{t_i} \vee B_i) \ \wedge \ (\bigvee_{i=1}^{n} B_i))(\sigma)$$

holds. This is quite obvious, since for any process P_i, either it has terminated or is blocked when deadlock occurs, so $\mathcal{Q}_{t_i}(\sigma)$ or $B_i(\sigma)$ holds, respectively. Moreover, at least one of the processes P_j is blocked, therefore $B_j(\sigma)$ holds. To this end, it is easy to see that the following deadlock-freedom condition (DFC) due to Owicki & Gries [OG76]:

$$\models (\bigwedge_{i=1}^{n} (\mathcal{Q}_{t_i} \vee B_i) \ \wedge \ (\bigvee_{i=1}^{n} B_i)) = \mathit{false} \tag{5}$$

ensures that deadlock will not occur.

Example 12.6 We prove that the following program is deadlock free.



Process P_1 can never execute any transition (not even a transition to terminate), so it is always blocked. Process P_2 executes its loop infinitely, therefore, the complete program is deadlock free. We choose the assertion networks as below:

$$\begin{array}{ll} \mathcal{Q}_{s_1} \overset{\text{def}}{=} \mathit{true}, & \mathcal{Q}_{s_2} \overset{\text{def}}{=} \mathit{true}, \\ \mathcal{Q}_{t_1} \overset{\text{def}}{=} \mathit{true}, & \mathcal{Q}_{l_2} \overset{\text{def}}{=} \mathit{true}, \\ & \mathcal{Q}_{t_2} \overset{\text{def}}{=} \mathit{false}. \end{array}$$

It is easy to see that they are inductive. The deadlock-freedom condition DFC in this case requires us to prove

$$\models (\mathcal{Q}_{t_1} \vee B_1) \wedge (\mathcal{Q}_{t_2} \vee B_2) \wedge (B_1 \vee B_2) = \mathit{false}$$

and this is trivial as the second conjunct evaluates to false. □

For establishing semantic completeness of this method, we again need to employ auxiliary variables in defining the necessary predicates. Auxiliary variables do not change the control flow of the program, and therefore do not alter the possibility regarding deadlock.

In summary, the method for proving deadlock freedom is very similar to the method for proving partial correctness (Definition 9.5); the only difference is that now instead of proving $\models \mathcal{Q}_{t_1} \wedge \ldots \wedge \mathcal{Q}_{t_n} \rightarrow \psi$, we prove the deadlock-freedom condition.

Definition 12.7 (Proof method of Owicki & Gries for deadlock freedom) In order to prove that a concurrent program is deadlock free relative to some precondition φ, prove steps 1 to 5 of the Owicki & Gries method given in Definition 9.5 except that the second clause of point 5 is replaced by the deadlock-freedom condition:

$$\models (\bigwedge_{i=1}^{n}(\mathcal{Q}_{t_i} \vee B_i) \ \wedge \ (\bigvee_{i=1}^{n} B_i)) = \mathit{false}. \tag{6}$$

□

Theorem 12.8 (Soundness and semantic completeness)
The method for proving deadlock freedom relative to a precondition φ as defined above is both sound and semantically complete.

Proof
Soundness has already been argued in this section, so we only present the completeness result in some detail.

As just pointed out, auxiliary variables do not change the possibility regarding deadlock. Hence, to prove that $P_1 \parallel \ldots \parallel P_n$ is deadlock free, we only have to prove the same result for the program $P'_1 \parallel \ldots \parallel P'_n$ augmented with history variables. We choose again the same assertion network as in the (compositional) completeness proof of partial correctness, i.e., with every location l of P_i we associate the predicate $SP'_l(\varphi, P_i)$. We prove the claim by showing that if the DFC does not hold, then $P'_1 \parallel \ldots \parallel P'_n$ is not deadlock free.

So, suppose there exist a state and history pair $\langle \sigma, \theta \rangle$ such that

$$\models (\bigwedge_{i=1}^{n}(SP'_{t_i}(\varphi, P_i) \vee B_i) \ \wedge \ (\bigvee_{i=1}^{n} B_i))(\langle \sigma, \theta \rangle)$$

holds, then for every $1 \leq i \leq n$, $(SP'_{t_i}(\varphi, P_i) \vee B_i)(\langle \sigma, \theta \rangle)$ holds. This means that there exists a location l of P' such that each process P_i is either terminated or deadlocked; moreover, from this configuration no transitions are enabled. By the fact that $\bigvee_{i=1}^{n} B_i(\langle \sigma, \theta \rangle)$ holds, at least one process has not terminated. Hence program $P'_1 \parallel \ldots \parallel P'_n$ is deadlocked at $\langle l; (\sigma, \theta) \rangle$.

■

References

[dRdBH+01] Willem-Paul de Roever, Frank S. de Boer, Ulrich Hannemann, Jozef Hooman, Yassine Lakhneche, Mannes Poel, and Job Zwiers. *Concurrency Verification.* Number 54 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, April 2001.

[OG76] S. Owicki and D. Gries. Verifying properties of parallel programs: An axiomatic approach. *Communications of the ACM*, 19(5):279–286, May 1976.