Concurrent program logics seminar

slides for 2011-05-09

(with tiny revisions)

Georg Neis

Last week: Early Papers

C.A.R. Hoare: An axiomatic basis for computer programming. CACM 1969.

Susan Owicki, David Gries: An axiomatic proof technique for parallel programs I. Acta Informatica 6, 319-340 (1976)

Review: Hoare Logic (1969)

- Very simple imperative language
- Axiomatic semantics
- Specification: **P {C} Q**
- Partial vs. total correctness
- Example:

  $$\text{If} \quad \vdash P \wedge B\{S\}P \ \text{then} \quad \vdash P\{\textbf{while}\ B\ \textbf{do}\ S\}\neg B \wedge P$$

- (Completeness)

Review: Owicki/Gries (1975)

- Extension of Hoare Logic to a concurrent language (shared memory)
- New constructs:

$$\frac{\{P \wedge B\}\ S\ \{Q\}}{\{P\}\ \textbf{await}\ B\ \textbf{then}\ S\ \{Q\}}$$

$$\frac{\{P1\}\ S1\ \{Q1\}, \ldots, \{Pn\}\ Sn\ \{Qn\}\ \text{are interference-free}}{\{P1 \wedge \ldots \wedge Pn\}\ \textbf{cobegin}\ S1\ //\ \ldots\ //\ Sn\ \textbf{coend}\ \{Q1 \wedge \ldots \wedge Qn\}}$$

- Key notion: interference freedom
- Relative completeness

Today: Rely-Guarantee

Viktor Vafeiadis: Modular fine-grained concurrency verification. PhD thesis. Sections 2.2 and 2.3 only. (entire thesis)

Joey W. Coleman, Cliff B. Jones: A structural proof of the soundness of rely/guarantee rules. Journal of Logic and Computation 17: 807-841 (2007)

Leonor Prensa Nieto: The rely-guarantee method in Isabelle/HOL. ESOP 2003.

Owicki/Gries not compositional

- "**In a compositional proof system, the specification of a parallel program can be derived from the specification of the components without knowing the implementation of these components.** This is important for the correct development of complex programs, where one would like to verify the design at stages where implementation details are still unknown." *(Nieto)*
- Owicki/Gries: need proofs of components to check non-interference
- Jones: describe interference in the specification

Operational semantics

"In order to show that the inference rules used for (concurrent) program constructs are sound, an independent semantics is needed." *(Coleman/Jones)*

$$\frac{}{(\mathbf{skip}; C_2), \sigma \to C_2, \sigma} \quad (\textsc{Seq1})$$

$$\frac{C_1, \sigma \to C_1', \sigma'}{(C_1; C_2), \sigma \to (C_1'; C_2), \sigma'} \quad (\textsc{Seq2})$$

$$\frac{}{(C_1 + C_2), \sigma \to C_1, \sigma} \quad (\textsc{Cho1})$$

$$\frac{}{(C_1 + C_2), \sigma \to C_2, \sigma} \quad (\textsc{Cho2})$$

$$\frac{B(\sigma)}{\mathsf{assume}(B), \sigma \to \mathbf{skip}, \sigma} \quad (\textsc{Assume})$$

$$\frac{}{C^*, \sigma \to (\mathbf{skip} + (C; C^*)), \sigma} \quad (\textsc{Loop})$$

$$\frac{C, \sigma \to^* \mathbf{skip}, \sigma'}{\langle C \rangle, \sigma \to \mathbf{skip}, \sigma'} \quad (\textsc{Atom})$$

$$\frac{C_1, \sigma \to C_1', \sigma'}{(C_1 \| C_2), \sigma \to (C_1' \| C_2), \sigma'} \quad (\textsc{Par1})$$

$$\frac{C_2, \sigma \to C_2', \sigma'}{(C_1 \| C_2), \sigma \to (C_1 \| C_2'), \sigma'} \quad (\textsc{Par2})$$

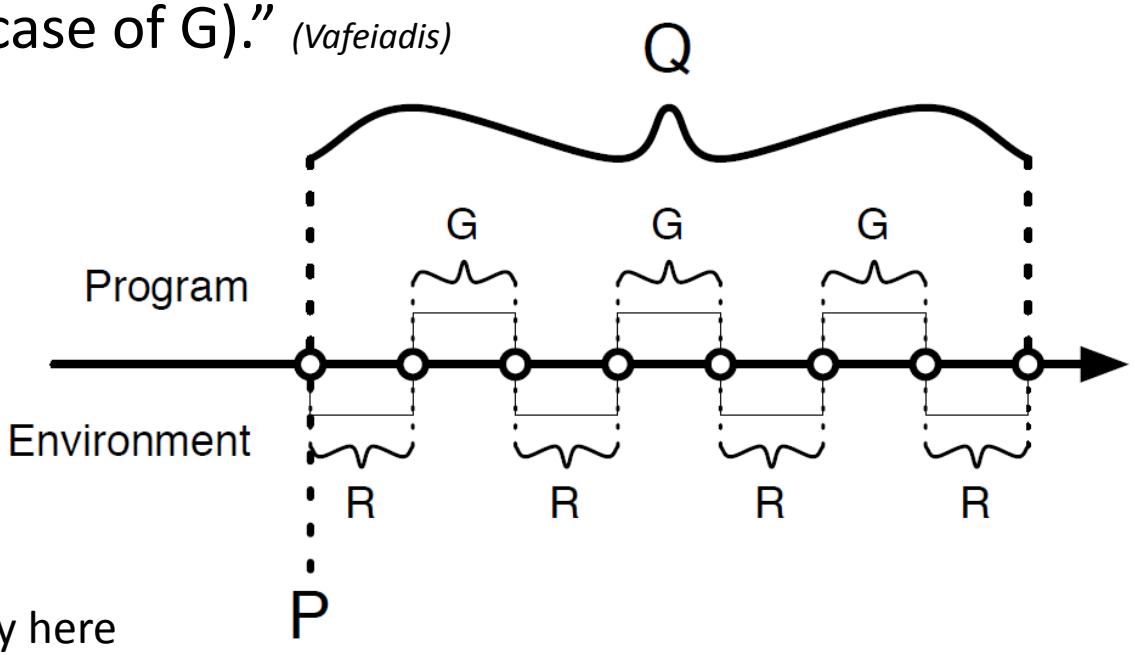
$$\frac{}{(\mathbf{skip} \| \mathbf{skip}), \sigma \to \mathbf{skip}, \sigma} \quad (\textsc{Par3})$$

$$\mathbf{if}(B)\ C_1\ \mathbf{else}\ C_2 \stackrel{\text{def}}{=} (\mathsf{assume}(B); C_1) + (\mathsf{assume}(\neg B); C_2)$$

$$\mathbf{while}(B)\ C \stackrel{\text{def}}{=} (\mathsf{assume}(B); C)^*; \mathsf{assume}(\neg B)$$

Rely-guarantee conditions

- Interference becomes part of specification, in the form of two binary relations: $\{P, R\} \vdash S \textbf{ sat } (G, Q)$
- "R and G summarise the properties of the individual atomic actions invoked by the environment (in the case of R) and the thread itself (in the case of G)." *(Vafeiadis)*



Note: post-conditions are binary here

Partial vs. total correctness

$\{P, R\}\ C \Vdash \text{sat}\ (G, Q)$ means: If C is started in a state that satisfies P and in an environment where each transition satisfies R, then:

- Coleman/Jones (total correctness)
  1. Any of C's transitions satisfies G, and
  2. C terminates in a state that satisfies Q.
- Nieto, Vafeiadis (partial correctness)
  1. Any of C's transitions satisfies G, and
  2. if C terminates, then in a state that satisfies Q.
- Noone (partial correctness, version 2)
  1. If C terminates, then its transitions satisfy G and
  2. its final state satisfies Q.

Rules for partial correctness (1)

$$\frac{}{\mathbf{skip}\ \mathsf{sat}_{\mathsf{RG}}\ (\mathit{true}, R, G, \mathsf{ID})}$$

the post-condition should say R*, and G must be reflexive

(RG-SKIP)

$$\frac{C_1\ \mathsf{sat}_{\mathsf{RG}}\ (P_1, R, G, (Q_1 \wedge P_2)) \qquad C_2\ \mathsf{sat}_{\mathsf{RG}}\ (P_2, R, G, Q_2)}{(C_1; C_2)\ \mathsf{sat}_{\mathsf{RG}}\ (P_1, R, G, (Q_1; Q_2))}$$

(RG-SEQ)

Rules for partial correctness (2)

$$\frac{(P;R) \Rightarrow P \qquad (R;Q) \Rightarrow Q \qquad (Q;R) \Rightarrow Q \qquad C \ \mathsf{sat}_{\mathsf{RG}}\ (P, \mathrm{ID}, \mathsf{True}, (Q \wedge G))}{\langle C \rangle \ \mathsf{sat}_{\mathsf{RG}}\ (P, R, G, Q)} \quad (\text{RG-ATOMIC})$$

$$\frac{C_1 \ \mathsf{sat}_{\mathsf{RG}}\ (P, (R \vee G_2), G_1, Q_1) \qquad C_2 \ \mathsf{sat}_{\mathsf{RG}}\ (P, (R \vee G_1), G_2, Q_2)}{(C_1 \| C_2) \ \mathsf{sat}_{\mathsf{RG}}\ (P, R, (G_1 \vee G_2), (Q_1 \wedge Q_2))} \quad (\text{RG-PAR})$$

Single-state vs. two-state *post*

$$\begin{aligned} &Await\text{: } [\![\ \forall\, V.\ \vdash P\ \textbf{sat}\ [pre \cap b \cap \{V\},\ \{(s,\ t).\ s = t\},\ UNIV,\\ &\qquad \{s.\ (V,\ s) \in guar\} \cap post];\ stable\ pre\ rely;\ stable\ post\ rely\ ]\!]\\ &\qquad \Longrightarrow \vdash Await\ b\ P\ \textbf{sat}\ [pre,\ rely,\ guar,\ post] \end{aligned}$$

"The rule for the await-statement is less obvious. By the semantics of the await-command, a positive evaluation of the condition and the execution of the body is done **atomically**. Thus, the state transition caused by the complete execution of *P* must satisfy the guarantee condition. This is reflected in the precondition and postcondition of *P* in the assumptions; **since these are sets of single states, the relation between the state before and after the transformation is established by fixing the values of the first via a universally quantified variable *V***." *(Nieto)*

Rules for partial correctness (2)

$$\frac{(P;R) \Rightarrow P \qquad (R;Q) \Rightarrow Q \qquad (Q;R) \Rightarrow Q \qquad C \ \mathsf{sat}_{\mathsf{RG}}\ (P, \mathrm{ID}, \mathsf{True}, (Q \wedge G))}{\langle C \rangle \ \mathsf{sat}_{\mathsf{RG}}\ (P, R, G, Q)} \quad (\text{RG-ATOMIC})$$

$$\frac{C_1 \ \mathsf{sat}_{\mathsf{RG}}\ (P, (R \vee G_2), G_1, Q_1) \qquad C_2 \ \mathsf{sat}_{\mathsf{RG}}\ (P, (R \vee G_1), G_2, Q_2)}{(C_1 \| C_2) \ \mathsf{sat}_{\mathsf{RG}}\ (P, R, (G_1 \vee G_2), (Q_1 \wedge Q_2))} \quad (\text{RG-PAR})$$

Rules for partial correctness (3)

$$\frac{C \ \mathsf{sat}_{\mathsf{RG}} \ (P, R, G, Q) \qquad P' \Rightarrow P \qquad R' \Rightarrow R \qquad G \Rightarrow G' \qquad Q \Rightarrow Q'}{C \ \mathsf{sat}_{\mathsf{RG}} \ (P', R', G', Q')} \ (\text{RG-WEAKEN})$$

$$\frac{C \ \mathsf{sat}_{\mathsf{RG}} \ (P, R, G, Q)}{C \ \mathsf{sat}_{\mathsf{RG}} \ (P, R, G, Q \wedge \overleftarrow{P} \wedge (G \vee R)^*)} \ (\text{RG-ADJUSTPOST})$$

Stability of P&Q under interference

Vafeiadis

- Stability checked only at rule for atomic blocks
- Pre-condition:
  $\overleftarrow{P} \wedge R \Rightarrow P$
- Post-condition:
  $Q \,|\, R \Rightarrow Q$
  $(R \,|\, Q) \Rightarrow Q$

Coleman/Jones

- Stability implicitly assumed everywhere?
- Pre-condition:
  $\overleftarrow{P} \wedge R \Rightarrow P$
- Post-condition:
  $Q \,|\, R \Rightarrow Q$
  $\overleftarrow{P} \wedge (R \,|\, Q) \Rightarrow Q$

Stability of B under interference

- “It is important to note that this property alone is not sufficient to establish that the test of a conditional construct such as If or While still holds during the execution of that construct’s body;” *(Coleman/Jones)*

- “Given the level of interference allowed in the language defined in Section 2, we can either add a proof requirement that evaluation of the b test is stable under R or we have to prove facts about the body of the while statement (respectively, the embedded statement of the if statement) without being able to take the b as an extra pre condition.” *(Coleman/Jones)*

- No rule for assume by Vafeiadis; issue not discussed by Nieto.

Rules for If and While

If-I

$$\frac{\begin{array}{l} b \textbf{ indep } R \\ \{P \wedge b, R\} \vdash body \textbf{ sat } (G, Q) \\ \overleftarrow{P} \wedge \overleftarrow{\neg b} \;\Rightarrow\; Q \end{array}}{\{P, R\} \vdash mk\text{-}If(b, body) \textbf{ sat } (G, Q)}$$

While-I

$$\frac{\begin{array}{l} bottoms(W, \neg b) \\ twf(W) \\ \{P, R\} \vdash body \textbf{ sat } (G, W \wedge P) \\ \overleftarrow{\neg b} \wedge R \;\Rightarrow\; \neg b \\ R \;\Rightarrow\; W^* \end{array}}{\{P, R\} \vdash mk\text{-}While(b, body) \textbf{ sat } (G, W^* \wedge P \wedge \neg b)}$$

Reflexivity and transitivity of R and G?

"Jones [5] first suggested that the rely and guarantee conditions be reflexive and transitive. However, **for the soundness proof only the reflexivity of the guarantee condition is necessary**. This is to ensure that transitions corresponding to the evaluation of boolean conditions (which do not affect the state) also satisfy the guarantee condition. If transitivity is also required another property, namely, observational equivalence, can be proven. […] in practice **finding guarantee conditions that are transitive is not easy**." *(Nieto)*

I spotted transitivity of rely conditions being used in the Coleman/Jones proofs.

Example

$ot \leftarrow \textbf{len}\, v + 1;\ et \leftarrow \textbf{len}\, v + 1;$
par
$\|\ (oc \leftarrow 1;$
  while $(oc < ot \wedge oc < et)$ **do**
    if $oc < ot \wedge pred(v(oc))$ **then** $ot \leftarrow oc$ **fi;**
    $oc \leftarrow oc + 2$
  od$)$
$\|\ (ec \leftarrow 2;$
  while $(ec < et \wedge ec < ot)$ **do**
    if $ec < et \wedge pred(v(ec))$ **then** $et \leftarrow ec$ **fi;**
    $ec \leftarrow ec + 2$
  od$)$
rap;
if $ot < et$ **then** $r \leftarrow ot$ **fi; if** $et < ot$ **then** $r \leftarrow et$ **fi**

pre $\forall i \in odds(\textbf{len}\, v) \cdot \delta(pred(v(i)))$
rely $v = \overleftarrow{v} \wedge et \leq \overleftarrow{et} \wedge ot \leq \overleftarrow{ot}$
guar $ot = \overleftarrow{ot} \vee ot < \overleftarrow{ot} \wedge pred(v(ot))$
post $\forall i \in odds(\textbf{len}\, v) \cdot i < ot \Rightarrow \neg\, pred(v(i))$

pre: {1..ot-1} instead of len v
rely: what about oc?
guar: what about oc?
different interpretation?

“**Our interest is development; we have argued in several publications that the steps of design provide the outline proof of correctness.** The formal rules offer the ability to generate verification conditions and to use a theorem prover if required. Proof rules for assignment statements are therefore of less interest than those for the combinators which let the designer decompose a large task into sub-programs. In fact, in the absence of complicated concepts like location sharing or interference, **assignments are unlikely to be wrong**.” *(Coleman/Jones)*

Nieto

- First formalization, done in Isabelle/HOL
- Number of parallel threads is parametric
- No nesting: each Pi in P1 || . . . || Pn must be a sequential command
- Expression evaluation assumed to be atomic
- Post-conditions are single-state predicates
- Partial correctness
- System is complete (proof uses extended(?) O/G as an intermediate stage)

Notes on completeness

"The rely/guarantee rules are intentionally incomplete: they model interference as a relation, **ignoring the environment's control flow.** Hence, they cannot directly prove properties that depend on the environment's control flow. Nevertheless, we can introduce auxiliary variables to encode the implicit control flow constraints, and use these auxiliary variables in the proof. Modulo introducing auxiliary variables, rely/guarantee is complete. The various completeness proofs [68] introduce an auxiliary variable that records the entire execution history. Of course, introducing such an auxiliary variable has a global effect on the program to be verified. Therefore, the completeness result **does not guarantee that a modular proof can be found** for every program." *(Vafeiadis)*

The view that rules like those in [Jon96] are proved as needed contrasts starkly with that of providing a (complete) axiomatic semantics for a language. The current authors note that the only non-trivial language for which this has been done is "Turing" [H+88]. **Our view absolves us from concerns about completeness because one can prove more rules as required.** This is fortunate because rely/guarantee rules have to fit many different styles of concurrent programming (depending on the intimacy of interference employed) and it is difficult to envisage a single canonical set. *(Coleman/Jones)*

...

"Further discussion of the trade-offs in designing rely/guarantee rules can be found in [CJ00] which includes the useful idea of a "dynamic invariant" that is not discussed further in the current paper." *(Coleman/Jones)*

"The pre-condition P, a single-state predicate, describes an assumption about the initial state that must hold for the program to make sense." *(Vafeiadis)*

"We do not enter here into a debate about the merits of operational versus denotational semantics (but see [Jon03b]); we do, of course, avoid the Baroque excesses caused by using what McCarthy called a "Grand State"." *(Coleman/Jones)*