Concurrent programs

Concurrent program syntax

Language

add a parallel composition statement: *stat* $||$ *stat*

semantics: $s_1 \mid\mid s_2$

- execute s_1 and s_2 in parallel
- allowing an arbitrary interleaving of atomic statements
  (expression evaluation or assignments)
- terminates when both s_1 and s_2 terminate

Hoare logic:

extended by Owicki and Gries [Owicki76]

first idea:

$$\frac{\{P_1\}\ s_1\ \{Q_1\} \qquad \{P_2\}\ s_2\ \{Q_2\}}{\{P_1 \wedge P_2\}\ s_1 \mid\mid s_2\ \{Q_1 \wedge Q_2\}}$$

but this is unsound

Concurrent programs: rule soundness

Issue:

$$\frac{\{P_1\}\ s_1\ \{Q_1\} \qquad \{P_2\}\ s_2\ \{Q_2\}}{\{P_1 \land P_2\}\ s_1 \,||\, s_2\ \{Q_1 \land Q_2\}}$$ is not always sound

example:

given $s_1 \stackrel{\text{def}}{=} X \leftarrow 1$ and $s_2 \stackrel{\text{def}}{=}$ **if** $X = 0$ **then** $Y \leftarrow 1$, we derive:

$$\frac{\overline{\{X = Y = 0\}\ s_1\ \{X = 1 \land Y = 0\}} \qquad \overline{\{X = Y = 0\}\ s_2\ \{X = 0 \land Y = 1\}}}{\{X = Y = 0\}\ s_1 \,||\, s_2\ \{\text{false}\}}$$

Solution:

the proofs of $\{P_1\}\ s_1\ \{Q_1\}$ and $\{P_2\}\ s_2\ \{Q_2\}$ must not interfere

Concurrent programs: rule soundness

interference freedom

given proofs Δ_1 and Δ_2 of $\{P_1\}\ s_1\ \{Q_1\}$ and $\{P_2\}\ s_2\ \{Q_2\}$

Δ_1 does not interfere with Δ_2 if:

for any Φ appearing before a statement in Δ_1
for any $\{P_2'\}\ s_2'\ \{Q_2'\}$ appearing in Δ_2
$\{\Phi \wedge P_2'\}\ s_2'\ \{\Phi\}$ holds
and moreover $\{Q_1 \wedge P_2'\}\ s_2'\ \{Q_1\}$

i.e.: the assertions used to prove $\{P_1\}\ s_1\ \{Q_1\}$ are stable by s_2

example:

given $s_1 \stackrel{\text{def}}{=} X \leftarrow 1$ and $s_2 \stackrel{\text{def}}{=}$ **if** $X = 0$ **then** $Y \leftarrow 1$, we derive:

$$\frac{\overline{\{X = 0 \wedge Y \in [0,1]\}\ s_1\ \{X = 1 \wedge Y \in [0,1]\}} \quad \overline{\{X \in [0,1] \wedge Y = 0\}\ s_2\ \{X \in [0,1] \wedge Y \in [0,1]\}}}{\{X = Y = 0\}\ s_1 \,||\, s_2\ \{X = 1 \wedge Y \in [0,1]\}}$$

Concurrent programs: rule completeness

Issue: incompleteness

$\{X = 0\}\ X \leftarrow X + 1 \mid\mid X \leftarrow X + 1\ \{X = 2\}$ is valid

but no proof of it can be derived

Solution: auxiliary variables

introduce explicitly program points and program counters

example:

$^{\ell 1}\, X \leftarrow X + 1\, ^{\ell 2} \mid\mid {}^{\ell 3}\, X \leftarrow X + 1\, ^{\ell 4}$

with auxiliary variables $pc_1 \in \{1, 2\}$, $pc_2 \in \{3, 4\}$

we can now express that a process is at a given control point
and distinguish assertions based on the location of other processes

$s_1 \stackrel{\text{def}}{=} {}^{\ell 1}\, X \leftarrow X + 1\, ^{\ell 2}$, $s_2 \stackrel{\text{def}}{=} {}^{\ell 3}\, X \leftarrow X + 1\, ^{\ell 4}$

$\{(pc_2 = 3 \wedge X = 0) \vee (pc_2 = 4 \wedge X = 1)\}\ s_1\ \{(pc_2 = 3 \wedge X = 1) \vee (pc_2 = 4 \wedge X = 2)\}$

$\{(pc_1 = 1 \wedge X = 0) \vee (pc_1 = 2 \wedge X = 1)\}\ s_2\ \{(pc_1 = 1 \wedge X = 1) \vee (pc_1 = 2 \wedge X = 2)\}$

$\Longrightarrow \{pc_1 = 1 \wedge pc_2 = 3 \wedge X = 0\}\ s_1 \mid\mid s_2\ \{pc_1 = 2 \wedge pc_2 = 4 \wedge X = 1\}$

in fact, auxiliary variables make the proof method complete

Conclusion

Conclusion

- logic allows us to reason about program correctness
- verification can be reduced to proofs of simple logic statements

Issue: automation

- annotations are required (loop invariants, contracts)
- verification conditions must be proven

to scale up to realistic programs, we need to automate as much as possible

Some solutions:

- automatic logic solvers to discharge proof obligations
  SAT / SMT solvers
- abstract interpretation to approximate the semantics
  - fully automatic
  - able to infer invariants

Bibliography

[Apt81] **K. Apt**. *Ten Years of Hoare's logic: A survey* In ACM TOPLAS, 3(4):431–483, 1981.

[Cousot02] **P. Cousot**. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation.* In TCS, 277(1–2):47–103, 2002.

[Dijkstra76] **E.W. Dijkstra**. *Guarded commands, nondeterminacy and formal derivation of program* In Comm. ACM, 18(8):453–457, 1975.

[Floyd67] **R. Floyd**. *Assigning meanings to programs* In In Proc. Sympos. Appl. Math., Vol. XIX, pages 19–32, 1967.

[Hoare69] **C.A.R. Hoare**. *An axiomatic basis for computer programming* In Commun. ACM 12(10), 1969.

[King69] **J.C. King**. *A program verifier* In PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, 1969.

[Owicki76] **S. Owicki & D. Gries**. *An axiomatic proof technique for parallel programs I* In Acta Informatica, 6(4):319–340, 1976.