Concepts of Concurrent Computation

Bertrand Meyer
Sebastian Nanz

Lecture 12: CCS Advanced Concepts
CCS: Weak Bisimulations
Refinement

- Further use of bisimulations: refinement of systems
- We would like to state that two processes $\text{Spec}$ and $\text{Imp}$ behave the same, where $\text{Imp}$ specifies the computation in greater detail
- This is not possible with strong bisimulations, as every action need be matched in equivalent processes
- Key to a weaker notion of equivalence: abstract from internal actions
- **Idea**: an external observer who focuses on visible actions but ignores all internal behavior
Weak bisimulation (1)

• We write $p \xrightarrow{\alpha} q$ if $p$ can reach $q$ via an $\alpha$-transition, preceeded and followed by zero or more $\tau$-transitions.

$\quad p \xrightarrow{\tau^*} p' \xrightarrow{\alpha} q' \xrightarrow{\tau^*} q$

Furthermore, $p \xrightarrow{\tau} q$ holds if $p = q$.

• This definition allows us to "erase" sequences of $\tau$-transitions in a new definition of behavioral equivalence: weak bisimulation.
Weak bisimulation (2)

• Definition (Weak bisimulation)

Let \((Q, A, \rightarrow)\) be a labeled transition system. \(R \subseteq Q \times Q\) is a weak bisimulation if \((p, q) \in R\) implies for all \(a \in A\)

• if \(p \xrightarrow{a} p'\) then \(q \xrightarrow{a} q'\) such that \((p', q') \in R\)

• if \(q \xrightarrow{a} q'\) then \(p \xrightarrow{a} p'\) such that \((p', q') \in R\)

Two states \(p\) and \(q\) are weakly bisimilar, \(p \approx q\), if there is a weak bisimulation \(R\) such that \((p, q) \in R\)
Example: Weak bisimulation

Consider the following CCS processes:

\[ P_0 = a.P_0 + b.P_1 + \tau.P_1 \]
\[ P_1 = a.P_1 + \tau.P_2 \]
\[ P_2 = b.P_0 \]
\[ Q_1 = a.Q_1 + \tau.Q_2 \]
\[ Q_2 = b.Q_1 \]

Is \( P_0 \approx Q_1 \)?

Yes, since \( \{(P_0, Q_1), (P_1, Q_1), (P_2, Q_2)\} \) is a weak bisimulation.
Conclusion

• Many "fundamental" models of concurrency: CCS, CSP, \(\pi\)-calculus
• The reason for this is that there are many forms of concurrency one might like to describe
• The \(\pi\)-calculus takes mobility into account, which is not the case for CCS and CSP
• Process calculi provide models of concurrency, not a programming languages - for "everyday use" too many details are abstracted away
• However, the formal techniques studied in process calculi can help to design better concurrent programming languages as well