Concurrency 6

Specification and Verification in CCS

Catuscia Palamidessi
catuscia@lix.polytechnique.fr
Example: A distributed scheduler

• 1,...,n are tasks identifiers. Tasks have to be executed repeatedly, in a cyclic order. There can be more than one task executed at the same time, but the next instance of Task i cannot start before previous instance has finished.

• **Specification:** We use:
  – $a_k$ as the signal **start** to Task k and
  – $b_k$ as the signal that Task k has **terminated**

**Assume:**
– $X \subseteq \{1,...,n\}$ are the tasks in progress
– Task i is next

$$
\text{ScSpec}(i,X) = \sum \{ b_k \cdot \text{ScSpec}(i-X\{k\}) \mid k \in X \} \text{ if } i \in X
$$

$$
\text{ScSpec}(i,X) = a_i \cdot \text{ScSpec}(i+1,XU\{i\})
+ \sum \{ b_k \cdot \text{ScSpec}(i-X\{k\}) \mid k \in X \} \text{ if } i \not\in X
$$
Example: A distributed scheduler

- **Implementation:** We build the scheduler, Sched, as a ring of n cells each linked to one task
- **Cell:**
  \[
  A \equiv a.C \\
  B \equiv b.A \\
  C \equiv c.E \\
  D \equiv d.A \\
  E \equiv b.D + d.B
  \]

  Note: A stands for \(A(a,b,c,d)\), B stands for \(B(a,b,c,d)\), etc. We will also use \(A_k\) for \(A(a_k,b_k,c_k,c_{k-1})\), \(B_k\) for \(B(a_k,b_k,c_k,c_{k-1})\), etc.

- **Definition**  \(\text{Sched} \equiv (\forall c_1) \ldots (\forall c_n) \left( A_1 | \prod \{ D_k | k \neq 1 \} \right)\)

- **Theorem 1** (Correctness of the implementation wrt the specification): \(\text{Sched} = \text{ScSpec}(1, \emptyset)\)
Scheduler: Proof of correctness

• The meaning of the various cells:
  – \( A_i \): Task \( i \) is next, and it is ready to initiate
  – \( B_i \): Task \( i \) is next, but it is not ready to initiate
  – \( D_i \): Task \( i \) is not next, but it is ready to initiate
  – \( E_i \): Task \( i \) is not next, and it is not ready to initiate

• Definition:
  \[
  \text{Sched}(i,X) \equiv (\nu c) (B_i \mid \prod \{ D_k \mid k \notin X \} \mid \prod \{ E_m \mid m \in X-{i} \} ) \quad \text{if} \quad i \in X
  \]
  \[
  \text{Sched}(i,X) \equiv (\nu c) (A_i \mid \prod \{ D_k \mid k \notin X \cup \{i\} \} \mid \prod \{ E_m \mid m \in X \} ) \quad \text{if} \quad i \notin X
  \]

• Proposition 2: \( \text{Sched}(i,X) = \text{ScSpec}(i,X) \)

• Theorem 1 is a particular case of Proposition 2
Implementation of the scheduler: how it works
Implementation of the scheduler: how it works
Implementation of the scheduler: how it works

6 Novembre 2003  Concurrency 6
Implementation of the scheduler: a possible future configuration
Scheduler: Proof of Correctness

Proposition 2: \[ \text{Sched}(i,X) = \text{ScSpec}(i,X) \]

Proof

- Lemma 3
  - (1) \( (\nu c_i)(C_i | D_{i+1}) = \tau.(\nu c_i)(E_i | A_{i+1}) \)
  - (2) \( (\nu c_i)(C_i | E_{i+1}) = \tau.(\nu c_i)(E_i | B_{i+1}) \)

  Proof: By expansion law

- Lemma 4
  - \( \text{Sched}(i,X) = \sum \{ b_k \cdot \text{Sched}(i,X\{k\}) \mid k \in X \} \) if \( i \in X \)
  - \( \text{Sched}(i,X) = a_i \cdot \text{Sched}(i+1,X\{i\}) \)
    \[ + \sum \{ b_k \cdot \text{Sched}(i,X\{k\}) \mid k \in X \} \] if \( i \notin X \)

  Proof: By Expansion law and Lemma 3

From Lemma 4 and the Definition law we obtain that \( \text{Sched}(i,X) = \text{ScSpec}(i,X) \) \( \square \)
Example: Counter

- It is possible in CCS to create structures which grow and shrink dynamically. Examples include unbounded queues and stacks, and counters.

- Specification of a Counter
  A counter is an object that can be
  - tested for zero  \textit{zero}
  - incremented  \textit{inc}
  - decremented  \textit{dec}

\[
\text{Count}_0 \equiv \text{inc. Count}_1 + \text{zero. Count}_0
\]

\[
\text{Count}_n \equiv \text{inc. Count}_{n+1} + \text{dec. Count}_{n-1} \quad n > 0
\]
Example: Counter

- Implementation: A structure obtained by linking together a process $B$ and $n$ copies of a process $C$ specified as follows:

  $B \equiv \text{inc.}(B^C) + \text{zero}.B$
  $C \equiv \text{inc.}(C^C) + \text{dec}.D$
  $D \equiv d.C + z.B$

  $$P^Q \equiv (\nu i')(\nu z')(\nu d')(P(z,d,i',z',d') \mid Q(z',d', \text{inc,zero,dec}))$$

  **Note:** $B$, $C$ and $D$ stand for $B(z,d,\text{inc,zero,dec})$, $C(z,d,\text{inc,zero,dec})$, and $D(z,d,\text{inc,zero,dec})$ respectively.

  $(P^Q)$ stands for $(P^Q)(z,d,\text{inc,zero,dec})$.

  **Proposition:** $^\wedge$ is associative, i.e. $P^{(Q^R)} = (P^Q)^R$
Example: Counter

- **Implementation:**
  Definition: \( C^{(n)} \equiv B^{C^{C^{\ldots^{C}}}} \) (n times)

- **Theorem (Correctness):** \( C^{(n)} = \text{Count}_n \)
  **Proof**
  
  **Lemma:**
  
  (1) \( C^D \approx D^C \)
  
  (2) \( B^D \approx B^B \)
  
  (3) \( B^B = B \)

  We can now prove that
  
  - \( C^{(0)} = \text{inc. } C^{(1)} + \text{zero. } C^{(0)} \text{ and} \)
  
  - \( C^{(n)} = C^{(n-1)}^C \text{ for } n > 0 \text{ by definition} \)
    \[ = \text{inc. (} C^{(n-1)}^C C^{)} + \text{dec. (} C^{(n-1)}^D \) \text{ by expansion law} \]
    \[ = \text{inc. } C^{(n+1)} + \text{dec. } C^{(n-1)} \text{ by the lemma above} \]

  Hence \( C^{(n)} \) satisfies the same equations as \( \text{Count}_n \). By the unique solution law we can conclude \( C^{(n)} = \text{Count}_n \). □