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 Taks 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

$$ScSpec(i,X) \equiv \sum \{ b_k . ScSpec(i,X-\{k\}) \mid k \in X\} \quad \text{if } i \in X$$

$$ScSpec(i,X) \equiv a_i.ScSpec(i+1,X \cup \{i\}) + \sum \{ b_k . ScSpec(i,X-\{k\}) \mid k \in X\} \quad \text{if } i \notin X$$

Example: A distributed scheduler

- **Implementation:** We build the scheduler, Sched, as a ring of n cells each linked to one task
- Cell:

  $A \equiv a.C \qquad C \equiv c.E \qquad E \equiv b.D + d.B$

  $B \equiv b.A \qquad D \equiv d.A$

  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 $A(a_k,b_k,c_k,\underline{c}_{k-1})$, B_k for $B(a_k,b_k,c_k,\underline{c}_{k-1})$, etc.

- Definition $\quad Sched \equiv (\nu\ c_1)\ldots(\nu\ c_n)\ (\ A_1 \mid \prod \{\ D_k \mid k \neq 1\})$

- Theorem 1 (Correctness of the implementation wrt the specification):

  $$Sched = ScSpec(1,\varnothing)$$

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:

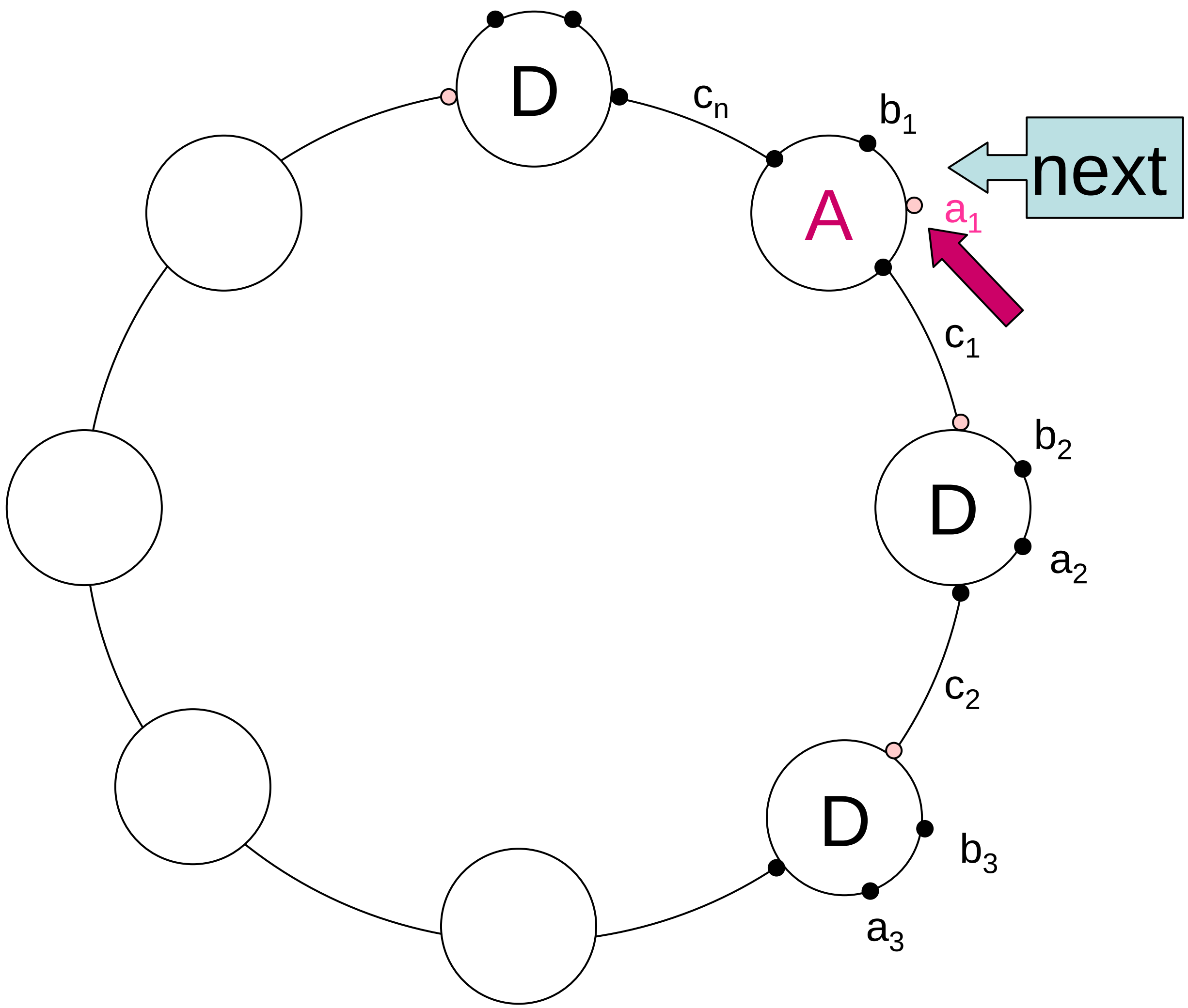
$$\mathrm{Sched}(i,X) \equiv (\nu\, \mathbf{c})\ (B_i \mid \prod \{D_k \mid k \notin X\} \mid \prod \{E_m \mid m \in X-\{i\}\}) \qquad \text{if } i \in X$$

$$\mathrm{Sched}(i,X) \equiv (\nu\, \mathbf{c})\ (A_i \mid \prod \{D_k \mid k \notin X \cup \{i\}\} \mid \prod \{E_m \mid m \in X\}) \qquad \text{if } i \notin X$$

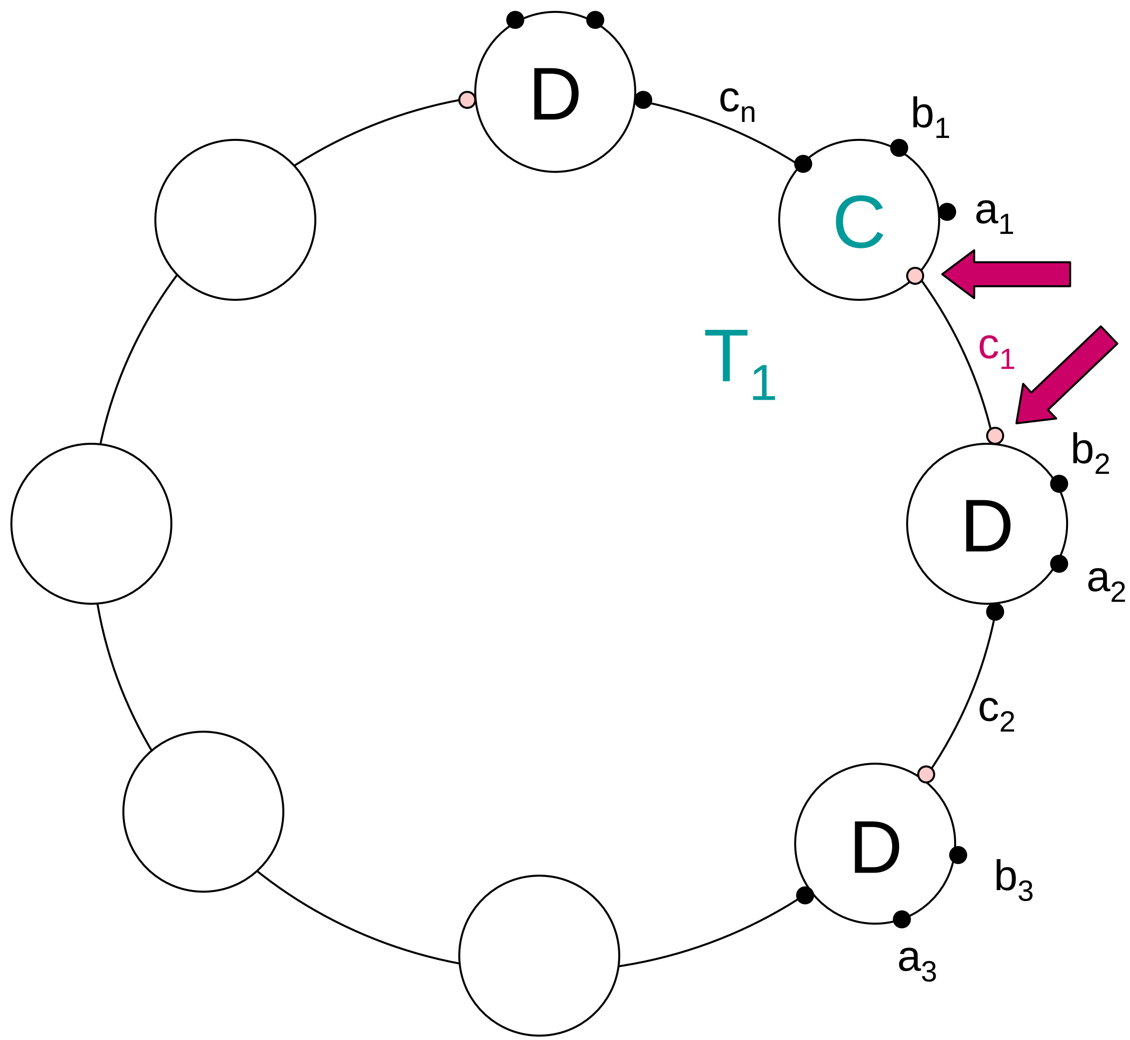
- Proposition 2: Sched(i,X) = ScSpec(i,X)

- Theorem 1 is a particular case of Proposition 2

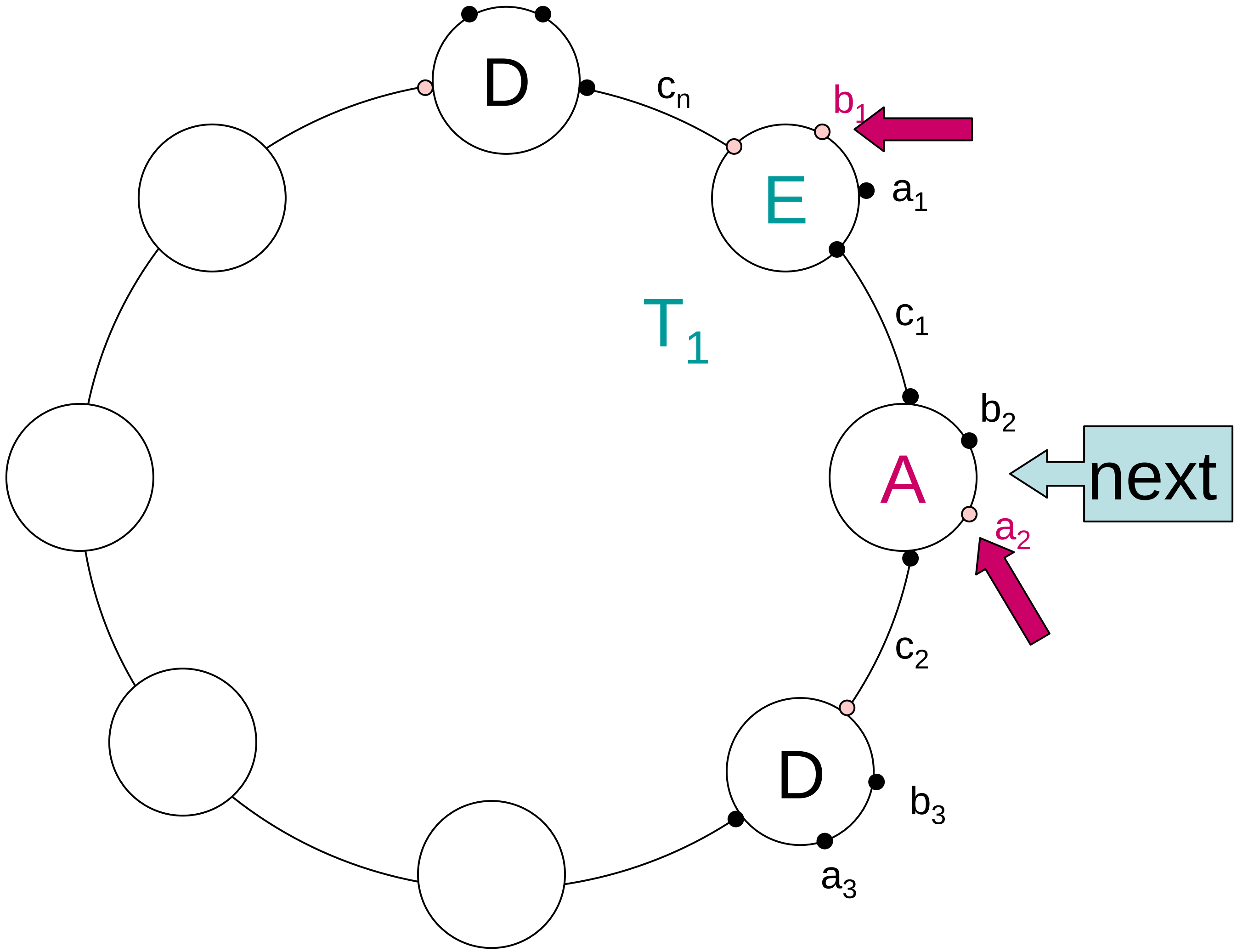
Implementation of the scheduler: how it works



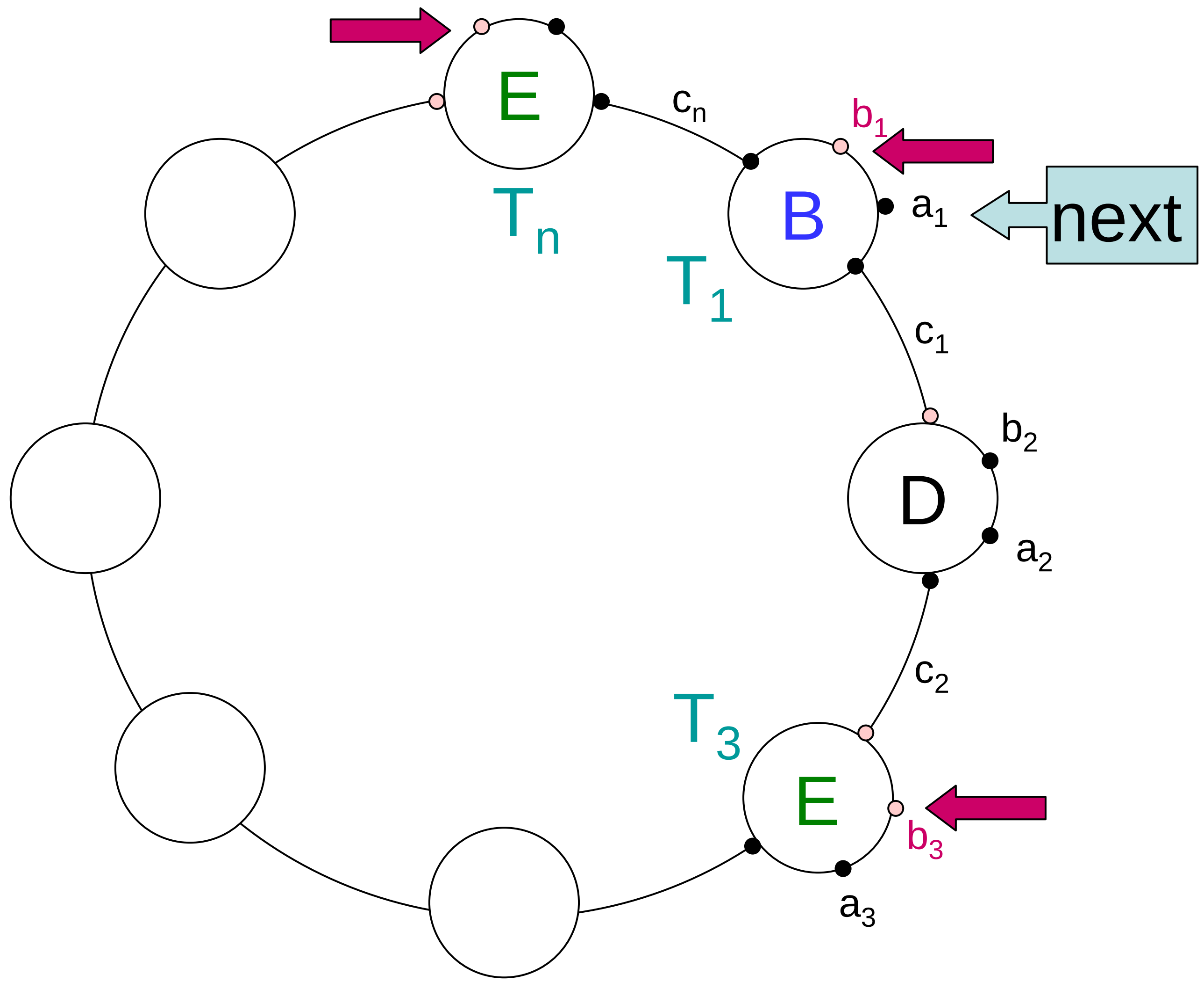
Implementation of the scheduler: how it works



Implementation of the scheduler: how it works



Implementation of the scheduler: a possible future configuration



Scheduler: Proof of Correctness

Proposition 2: Sched(i,X) = ScSpec(i,X)
Proof

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

  Proof: By exapansion law

- Lemma 4
  - $Sched(i,X) = \sum \{\, b_k.\, Sched(i,X-\{k\}) \mid k \in X \,\}$ if $i \in X$
  - $Sched(i,X) = a_i.Sched(i+1,X \cup \{i\})$
    $+$
    $\sum \{\, b_k.\, Sched(i,X-\{k\}) \mid k \in X \,\}$ if $i \notin X$

  Proof: By Expansion law and Lemma 3

From Lemma 4 and the Definition law we obtain that Sched(i,X) = ScSpec(i,X) □

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 $\underline{zero}$
  - incremented $\underline{inc}$
  - decremented $\underline{dec}$

$$Count_0 \equiv inc.Count_1 + zero.\ Count_0$$

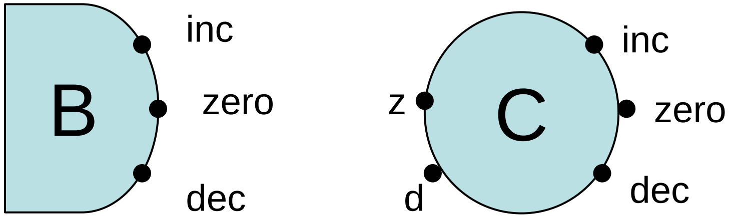
$$Count_n \equiv inc.Count_{n+1} + dec.\ Count_{n-1} \qquad n > 0$$

Example: Counter

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

$B \equiv inc.(B\hat{\ }C) + zero.B$

$C \equiv inc.(C\hat{\ }C) + dec.D$
$D \equiv \underline{d}.C + \underline{z}.B$



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

Note: B, C and D stand for B(z,d,inc,zero,dec), C(z,d,inc,zero,dec), and D(z,d,inc,zero,dec) respectively.

(P^Q) stands for (P^Q)(z,d,inc,zero,dec).

Proposition: ^ is associative, i.e. P^(Q^R) = (P^Q)^R

Example: Counter

- **Implementation:**

  Definition: $C^{(n)} \equiv B\hat{}C\hat{}C\hat{}\ldots\hat{}C$ (n times)

- **Theorem (Correctness):** $C^{(n)} = Count_n$

  Proof

  Lemma: (1) $C\hat{}D \approx D\hat{}C$

  (2) $B\hat{}D \approx B\hat{}B$

  (3) $B\hat{}B = B$

  We can now prove that

  - $C^{(0)} = inc.\ C^{(1)} + zero.\ C^{(0)}$ and

  - $C^{(n)} = C^{(n-1)}\ \hat{}C$ for $n > 0$ — by definition

    $= inc.\ (C^{(n-1)}\ \hat{}C\hat{}C) + dec.\ (C^{(n-1)}\ \hat{}D)$ — by expansion law

    $= inc.\ C^{(n+1)} + dec.\ C^{(n-1)}$ — by the lemma above

  Hence $C^{(n)}$ satisfies the same equations as $Count_n$. By the unique solution law we can conclude $C^{(n)} = Count_n$ □