Calculi of Concurrency: CCS and π-calulus

Jan Małuszyński
Ulf Nilsson

http://www.ida.liu.se/ ulfni/conc/

Motivation

- Growing number of concurrent systems
- Growing complexity of the systems

We want:

- to understand them,
- to reason about them (correctness).

The problems:

1. what are the primitive concepts of concurrency?
2. how to achieve correctness?

The objective of the course:
to present a widely accepted approach to (1) and (2), and a prototype supporting tool.

Designing a System

1. (Re-)formulate requirements
2. (Re-)design a model
3. Analysis of the model w.r.t. requirements
4. Repeat from 1 or 2 if necessary

Formal methods:

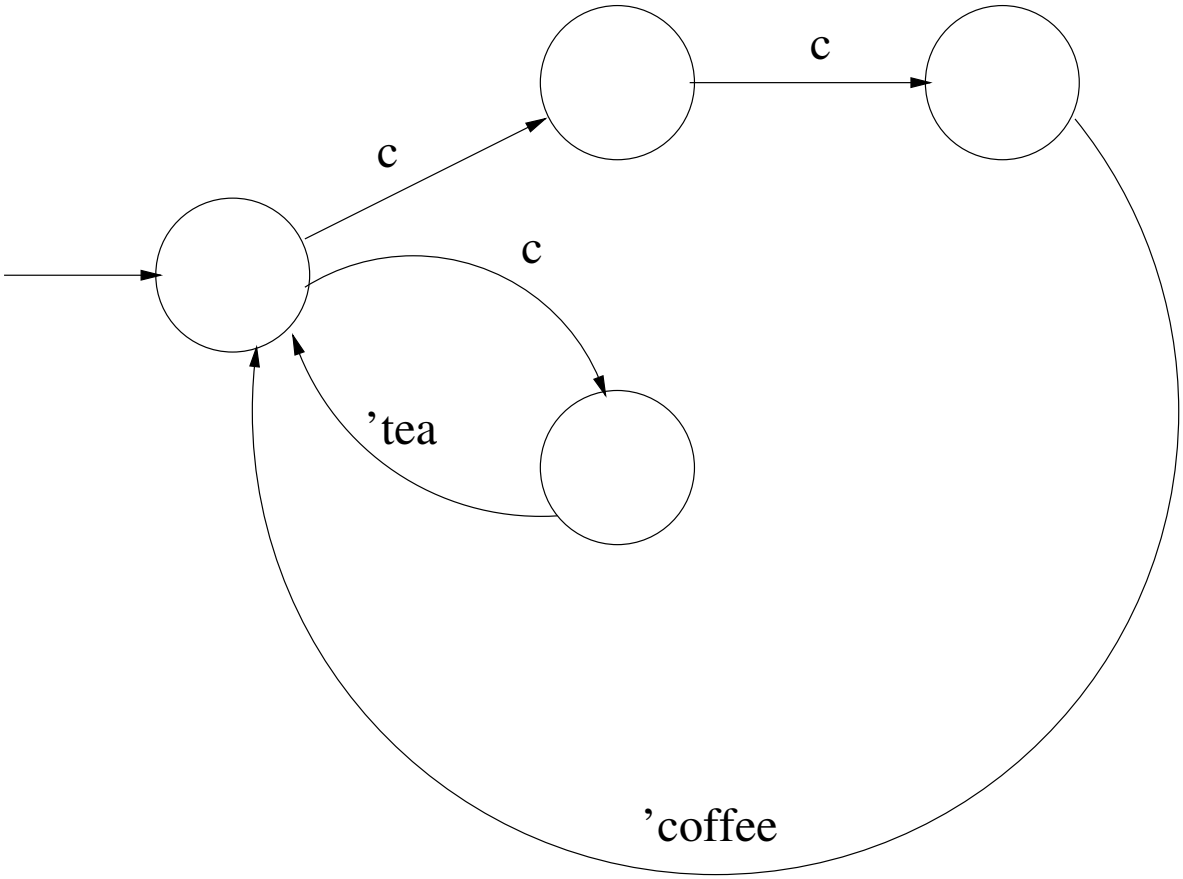
- Objective: automation of (3)
- Require: Formal description languages for
  - models
  - expected properties of the system

The course presents:

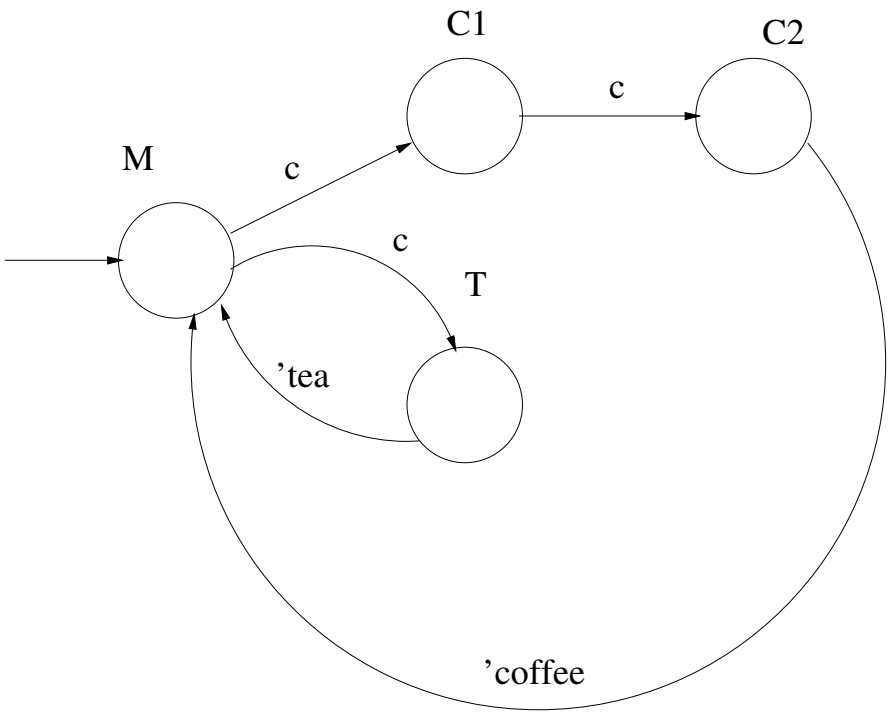
- examples of such languages;
- a tool for automatic model verification

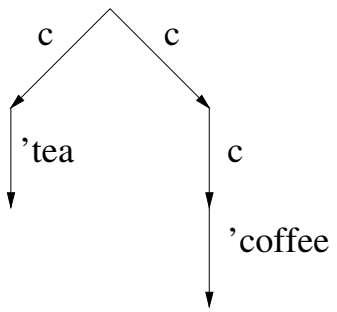
Sequential processes

Finite state systems



Behavior





Concurrent processes

Concurrent Process:
A set of (sequential) Processes

Sequential Process:

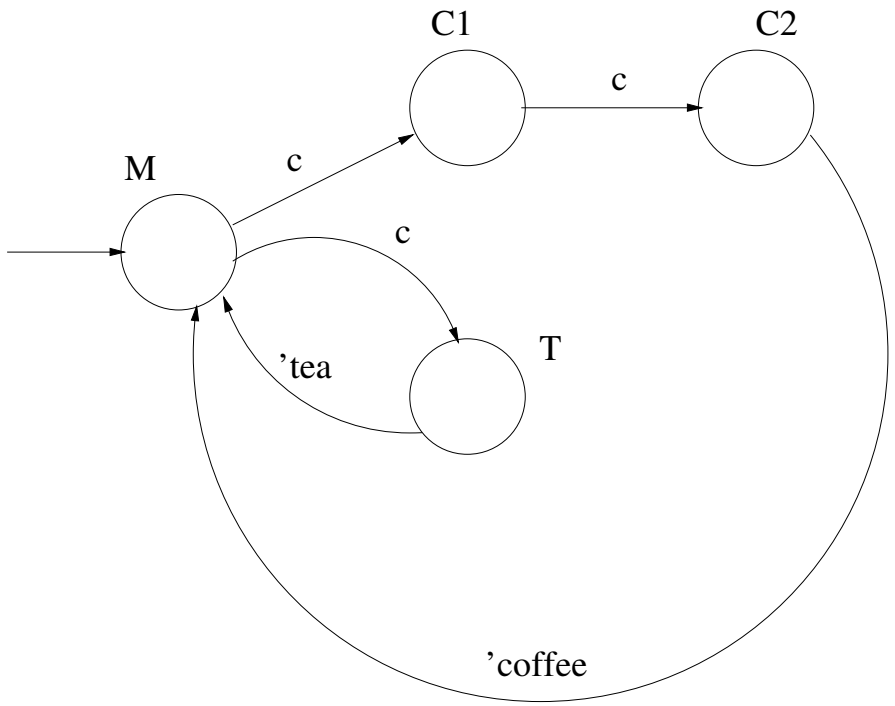
- Finite State Transition System (LTS)
- Sequential Process Expression

Communication:
"handshaking" through complementary labels

Concurrent Process Expression:

- describes the structure of the system
- the global behavior can be described by an LTS

Process expressions (Examples)



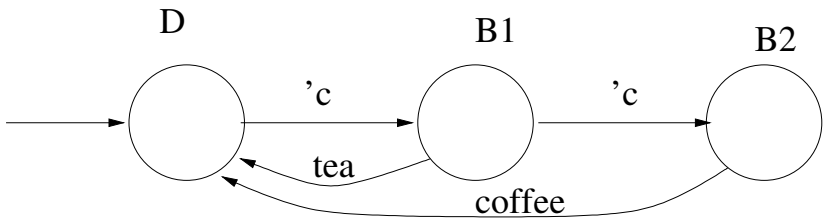
M = c.C1 + c.T

C1 = 'tea.M

C1 = c.C2

C2 = 'coffee.M

S = (D | M)



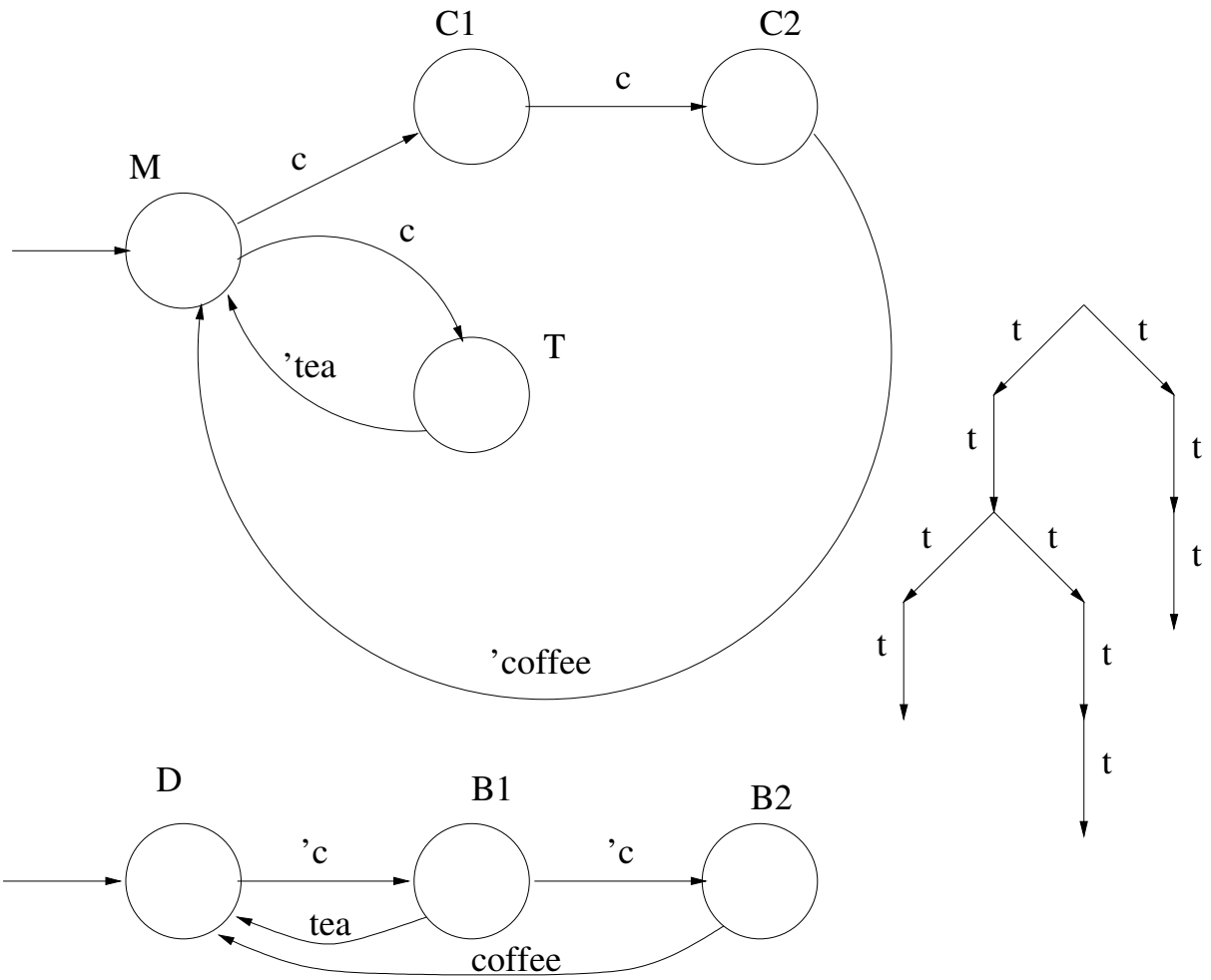
D = 'c.B1

B1 = 'c.B2

B1 = tea.D

B2 = coffee.D

Joint Behavior



Correctness

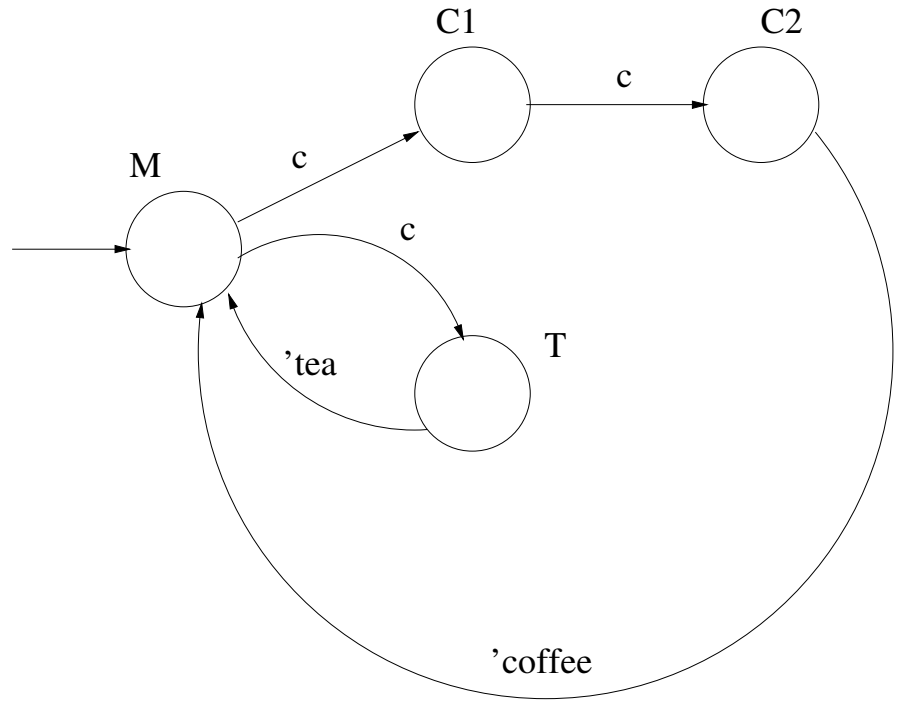
Compare the model with a specification

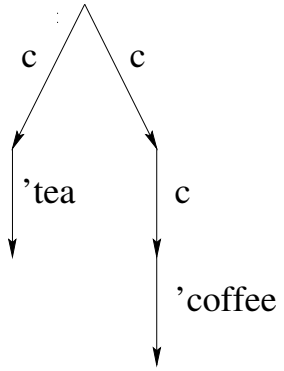
- **Model:** a process expression
- **Specification:**
  - another process expression; equivalence/entailment check
  - another specification language; satisfaction check

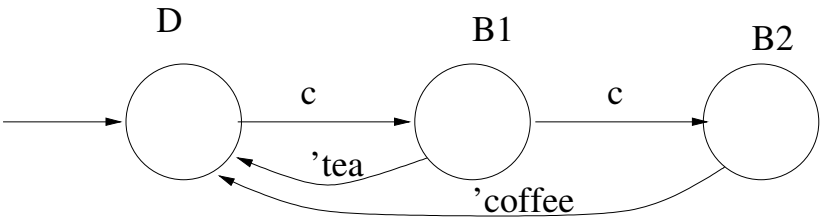
Define:

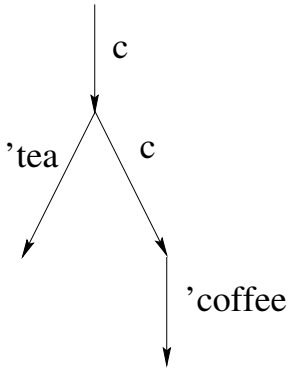
- suitable concept(s) of equivalence,
- suitable specification language(s) and concepts of satisfaction

Equivalences









Equivalences

The tea/coffeee processes

- are language equivalent
- are not observation equivalent

Objective: study suitable concepts of **equivalence**

Properties

Describe (expected) behavior of the process, e.g.

- an action may happen
  now:
  `prop teanow = <tea>tt`
  or some time in the future:
  `prop teaf =`
  `min X = <tea>tt` $\vee$ `[-]X`
- the process may reach a deadlock state
  `prop candeadlock =`
  `min X = [-]ff` $\vee$ `<->X`
- ...

The specification language of CWB-NC:
μ - calculus.

The Concurrency Workbench

Installed at IDA, makes possible:

- simulation of processes described in
  - CCS
  - CSP
  - Lotos
  - ...
- checking equivalences
- model checking vs. a specification in μ-calculus (or in CTL)

CWB - an Example

Process descriptions in CCS

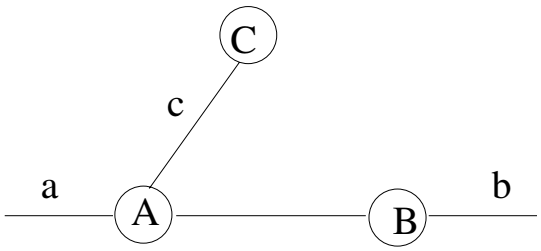
```
proc M = c.T + c.C1
proc T = tea.M
proc C1 = c.C2
proc C2 = coffee.M

proc M1 = c.T1
proc T1 = tea.M1 + c.C
proc C  = coffee.M1
```

Checking equivalence of M and M1

```
 eq -S obseq "M" "M1"
Building automaton...
States: 7
Transitions: 9
Done building automaton.
Transforming automaton...
Done transforming automaton.
FALSE...
M satisfies:
  <<c>>[[c]]ff
M1 does not.
```

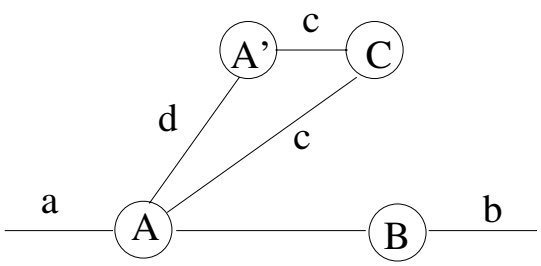
(Im)Mobility

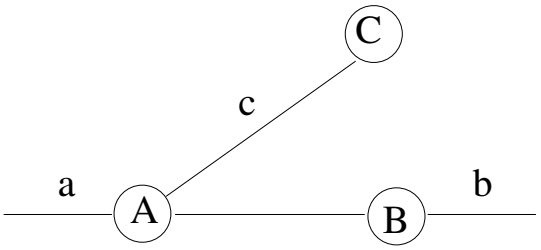


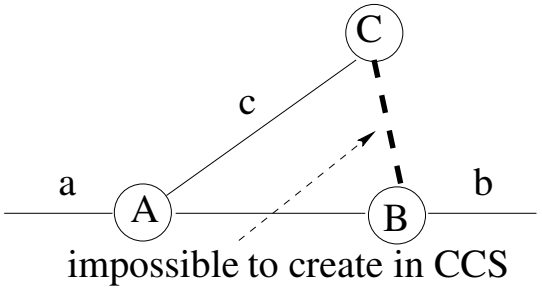
S = new c(A | C) | B

A = a.new d(A | A') + c.A''

A' = c.0







impossible to create in CCS

π-Calculus for Mobility

The principle:
Mobility = Creating **links** in a virtual space

The solution: Extending CCS Expressions:
links passed as action parameters

The Schedule

April 4, (JM) Introduction

April 11, (JM) Reactive sequential processes. Bisimulation. Chapter 2,3.

April 18, (JM) CCS, Chapter 4,5

May 2, (JM) Observational Equivalence - Examples, Chapter 6,7

May 9, (The students) Homework 1

May 16, (UN) Model checking with μ-calculus, The CWB manual +.

May 23 (JM) Introduction to π-calculus, Chapter 8-10.

May 30, (Invited talk?) A perspective of concurrency research
http://www.cs.sunysb.edu/ sas/sdcr/

June 6, (The students) Homework 2

"Chapters" refer to: **Robin Milner**, *Communicating and Mobile Systems: the π-calculus* CUP 1999)

CWB-NC

The Concurrency Workbench of
North Carolina or New Century

A system that provides

- notations for defining systems
  - ccs and variants (pccs, tccs)
  - csp
  - lotus
- notations for defining requirements (as above + HML, μ-calculus and CTL)
- Several types of equivalences
- Several preorder relations

Starting CWB-NC @ IDA

```
$ module add /home/tcslab/modules/prog/cwb

$ cwb-nc ccs

The Concurrency Workbench of North Carolina
(Version 1.11 --- May, 1998)

cwb-nc>
```

Example file: tmp.ccs

```
proc M = c.C1 + c.T
proc C1 = c.C2
proc C2 = 'coffee.M
proc T = 'tea.M

proc D = 'c.B1
proc B1 = 'c.B2 + tea.D
proc B2 = coffee.D
```

```
cwb-nc> load tmp.ccs

cwb-nc> cat M

===Agent===

c.C1 + c.T

cwb-nc> help ls

NAME

    ls

SYNOPSIS

    ls
...

cwb-nc> cd /home/ulfni
```

```
cwb-nc> compile M
Building automaton...
States: 4
Transitions: 5
Done building automaton.
0: c {1, 2}

1: c {3}

2: 'tea {0}

3: 'coffee {0}


Start States:  [0]
```

```
cwb-nc> ls

===Agent===

M
C1
T
C2
D
B1
B2


===Set===



===Formula===
```

```
cwb-nc> eq -S obseq M D
Building automaton...
States: 7
Transitions: 9
Done building automaton.
Transforming automaton...
Done transforming automaton.
FALSE...
M satisfies:
<<c>>tt
D does not.
```

Analyzes

Equivalence (e.g. observational, bisimulation and 3 more)

Preorder checking (2 types)

Reachability analysis

Model checking (HML/μ-calculus/CTL)

```
cwb-nc> min -S obseq M M-min
Building automaton...
States: 4
Transitions: 5
Done building automaton.
Transforming automaton...
Done transforming automaton.
Minimizing automaton...
Computing bisimulation...
Done computing bisimulation.
Done minimizing automaton.
cwb-nc>
```

Simulation

```
cwb-nc> sim M
M
1: -- c --> T
2: -- c --> C1
cwb-nc-sim> 1
T
1: -- 'tea --> M
cwb-nc-sim> 1
M
1: -- c --> T
2: -- c --> C1
cwb-nc-sim> back
Current state moved 1 step back.
cwb-nc-sim> current
T
1: -- 'tea --> M
cwb-nc-sim> 1
M
1: -- c --> T
2: -- c --> C1
```

```
cwb-nc-sim> history
1: M -- c -->
2: T -- 'tea -->
3: M
cwb-nc-sim> random 3
cwb-nc-sim> history
1: M -- c -->
2: T -- 'tea -->
3: M -- c -->
4: C1 -- c -->
5: C2 -- 'coffee -->
6: M
cwb-nc-sim> goto 3
Current state moved 3 steps back.
cwb-nc-sim> trace
 c 'tea
cwb-nc-sim> quit
```