Plan of the talk

- Nondeterminism in Concurrency vs “Classical” Nondeterminism in Computation Theory
- An example of a concurrent formalism in which nondeterminism is essential to achieve Turing-completeness
- Control of Nondeterminism: The expressive power of mixed choice
Nondeterminism in Concurrency

vs

Nondeterminism in Computation Theory

and

Classical nondeterministic sequential formalisms
Nondeterminism in Computation Theory and in classical sequential formalisms
Nondeterminism in Computation theory and classical sequential formalisms

- Nondeterminism in Computation Theory:
  - Nondeterministic Turing Machines

- Examples of nondeterministic sequential formalisms
  - Logic Programming
  - Nondeterministic Pascal (Pascal + “choose” construct)

- Nondeterminism is used to provide “natural” solutions to problems which are intrinsically nondeterministic
Examples of problems which are intrinsically nondeterministic

- Parsing problem: Determine whether a string is generated by a certain grammar
  - String abbc  
    Grammar:  \[ A = aA | aB \quad B = bB | bc \]
- Find all the paths that connect two nodes in a graph
Examples of problems which are intrinsically nondeterministic

- Parsing problem: Determine whether a string is generated by a certain grammar
  - String abbc
  - Grammar: $A = aA \mid aB$, $B = bB \mid bc$

- Find all the paths that connect two nodes in a graph
Features of nondeterminism in Computation Th / Sequential Formalisms

- In general the computation is regarded the whole tree
- There can be successful, failed, and infinite branches
- Only the successful branches count: failure is universal
Features of nondeterminism in Computation Th / Sequential Formalisms

- **Nondeterminism in general adds expressive power to formalisms which are not Turing-complete, but:**

- **Nondeterminism does not add expressive power to TM:** in general, the nondeterministic computation can be simulated by a deterministic one. For instance, by breadth-first exploration, or (if there are no infinite paths) by backtracking

- **Nondeterministic TM are not more expressive than TM**
Nondeterminism in Concurrency
Nondeterminism in Concurrency

- Nondeterminism usually arises from the fact that there are several independent components / processes.
- The various components / processes usually evolve at different speeds, and they may interact in unpredictable ways.
Example: a distributed system for reservations of tickets for a performance
Example: a distributed system for reservations of tickets for a performance

- The assignment of the seats to the customers depends on many factors: the order by which the customers contact the agencies, the resolution of the conflicting requests between the agencies, etc.

- If coordination is not carefully organized, then “bad things” can happen, for instance the same seat could be sold to two different customers.

- Backtracking is usually too complicated / too expensive, or even impossible. And in some cases, even if we backtrack, there is no guarantee to avoid ending up in the same problem.

  - Example: the dining philosophers
Features of nondeterminism in concurrency

- A computation is just one of the possible branches

- “bad branches” count (failures, including deadlocks, and infinite branches, including livelocks)

- \textit{failure is existential}
Features of nondeterminism in concurrency

- Nondeterminism cannot be avoided
- We need mechanisms to control nondeterminism in order to avoid “bad paths”
  - The control of nondeterminism may provide new criteria (other than Turing-completeness) for discriminating among the expressive power of different formalisms.
An example of a concurrent formalism in which nondeterminism is essential to achieve Turing-completeness
Control of Nondeterminism: The expressive power of mixed choice