Concurrency Theory

Xiaoju Dong(董笑菊)

BASICS

Department of Computer Science

Shanghai Jiao Tong University

xjdong@sjtu.edu.cn

问题1

公安机关正在调查一宗盗窃案，现获得事实如下：

- A或B盗窃了文物
- 若A盗窃了文物，则作案时间不可能在午夜前
- 若B证词正确，则在午夜前屋里灯光未灭
- 若B证词不正确，则作案时间发生在午夜前
- 午夜时屋里灯光灭了

试问谁是盗窃犯？

问题1

公安机关正在调查一宗盗窃案，现获得事实如下：

- A或B盗窃了文物
- 若A盗窃了文物，则作案时间不可能在午夜前
- 若B证词正确，则在午夜前屋里灯光未灭
- 若B证词不正确，则作案时间发生在午夜前
- 午夜时屋里灯光灭了

试问谁是盗窃犯？

命题逻辑

问题2

电灯开关

两个开关A、B同时控制一盏灯C,

(1) 只要有一个开关处于开启状态灯就会亮

(2) 只有两个开关之一处于开启状态灯才亮

请具体列出灯C在开关A和B处于什么情况下会亮

问题2

电灯开关

两个开关A、B同时控制一盏灯C,

(1) 只要有一个开关处于开启状态灯就会亮

(2) 只有两个开关之一处于开启状态灯才亮

请具体列出灯C在开关A和B处于什么情况下会亮

数字逻辑

Formal Methods

- In computer science, specifically software engineering and hardware engineering, formal methods are a particular kind of mathematically based techniques for the specification, development and verification of software and hardware systems
- The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design.

Formal Methods

- Formal methods are best described as the application of a fairly broad variety of theoretical computer science fundamentals, in particular logic calculi, formal languages, automata theory, discrete event dynamic system and program semantics, but also type systems and algebraic data types to problems in software and hardware specification and verification

Concurrency

- In computer science, **concurrency** is a property of systems
  - several computations are executing simultaneously
  - and potentially interacting with each other

Concurrency

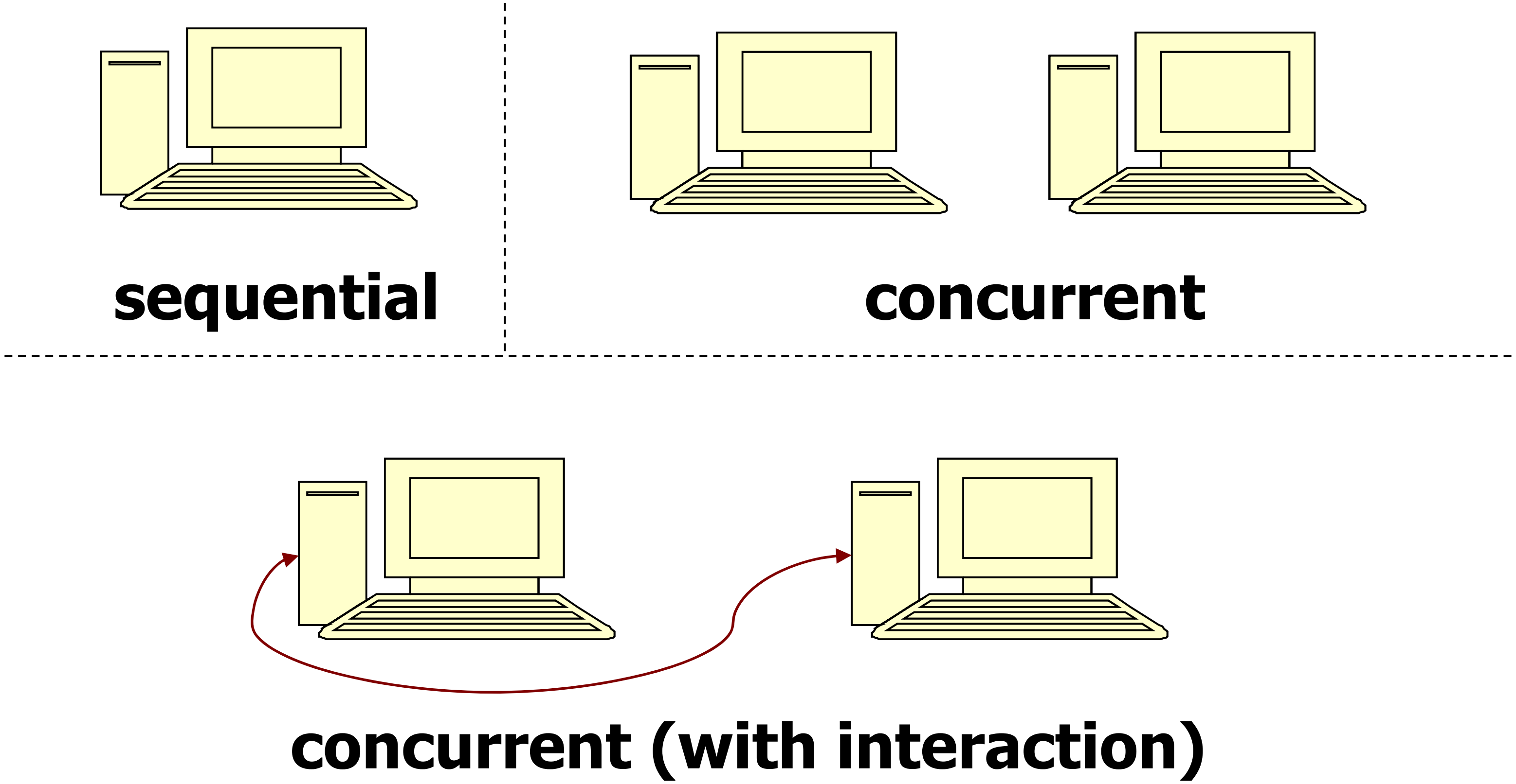
- **Computation**
  - any type of calculation
  - use of computer technology in Information processing (e.g. query in a database)
  - (electronic)computers, quantum computers, DNA computers, molecular computers, etc.
  - a process following a well-defined model expressed in an algorithm, protocol, etc.

Concurrency

- **Interaction** (or communication)
  - Hand-shaking (synchronization)
  - Value-passing
  - Name-passing
  - Process-passing
  - ……

Concurrency

- An example – sorting



Concurrency

- An example – sorting
- More examples ?

Concurrency

- An example – sorting
- More examples ?
  - reading & eating an apple
  - chatting & downloading
  - CS
  - web service
  - cells in our body
  - ……

Concurrency theory

- An active field of research in theoretical computer science
- Formalisms for modeling and reasoning about concurrency
- One of the first proposals:
  - Carl Adam Petri, Petri Nets, in the early 1960s

Complex concurrent systems

- Shared resources

(1) x := 1 (2) x := 0

x := x+1

Sequential: determined

Concurrency: non-determined

Content

- Process calculi
  - Calculus of Communicating Systems (CCS)
  - Name Passing Calculus (Pi calculus)
- Petri nets

References

- CCS
  - *Communication and Concurrency.*
    Robin Milner. Prentice Hall, 1989.
- Pi calculus
  - *Communicating and Mobile Systems: The π-calculus.*
    Robin Milner. Cambridge University Press, 1999.
  - The π-calculus: A Theory of Mobile Processes.
    Davide Sangiorgi. Cambridge University Press, 2001.
- Petri nets
  - *Petri nets – an introduction* ,
    Wolfgang Reisig, Springer-Verlag, 1982

Review

Sequential Computation

- The Concept of Computation
- Sequential Computation
  - a linearly ordered sequence of atomic actions

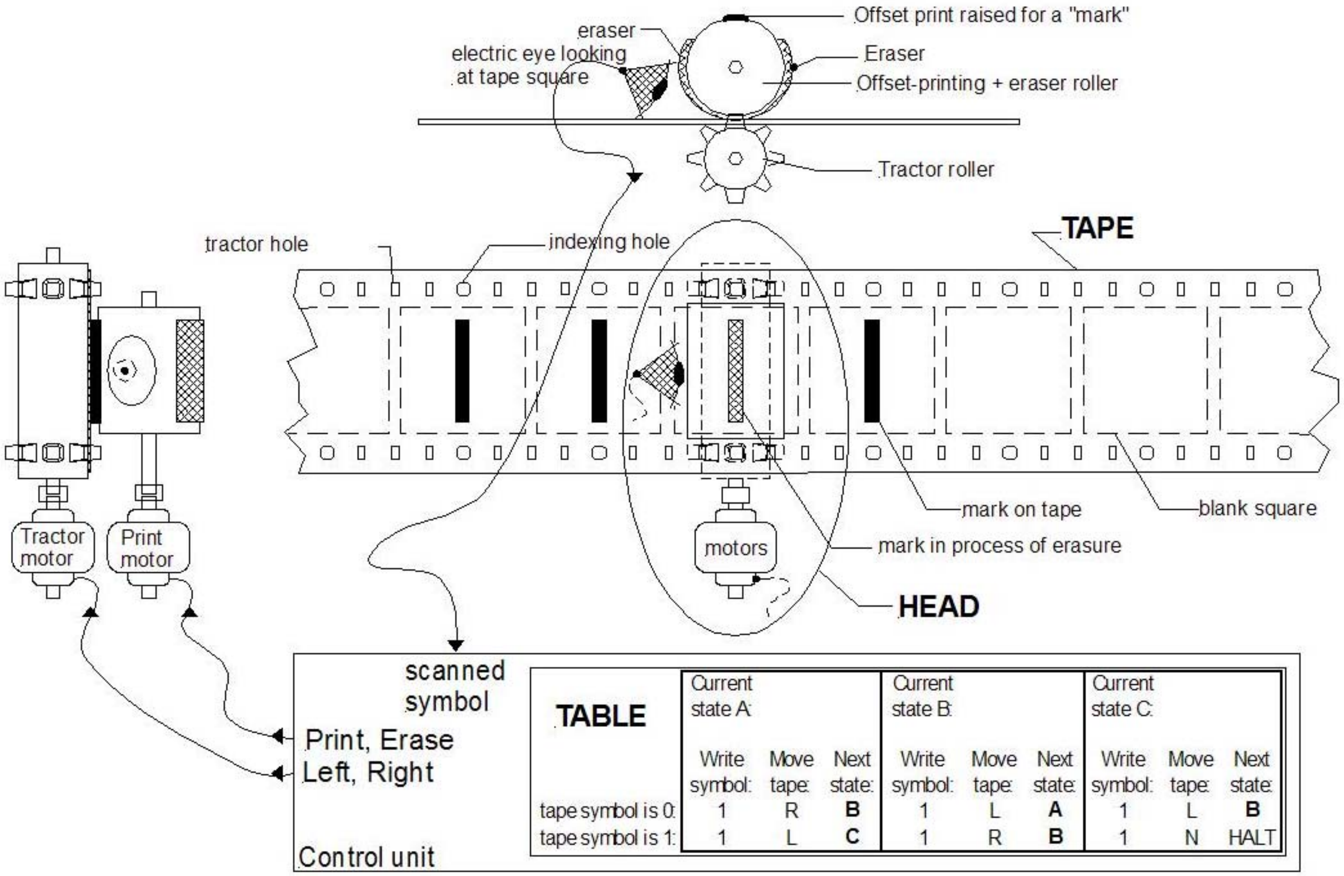
Some Models

- Theory of Sequential Computation
  - Turing machine (Turing)
    - Computation as mechanical operation
  - λ-Calculus (Church)
    - Computation as β-reduction
  - Recursion theory (Kleene etc.)
    - Computation as function composition
    - Computation denotes function

Turing Machine

- A **Turing machine** is a theoretical device that manipulates symbols on a strip of tape according to a table of rules
  - tape
  - head
  - transition function (a table of instructions, an action table)
  - state register
- Reference
  - *Elements of the theory of computation*, Harry Lewis
  - *Introduction to Theory of Computation*, Michael Sipser

Turing Machine



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

λ-Calculus

- Computation as Term Rewriting
- Term

$$t := x \quad \text{variable}$$
$$tt' \quad \text{application}$$
$$\lambda x.t \quad \text{abstraction}$$

- Reduction

$$(\lambda x.t)s \to t\{s/x\}$$

Examples:

$$(\lambda x.xx)\ (\lambda x.xx) \to (\lambda x.xx)\ (\lambda x.xx)$$
$$u((\lambda x.t)s) \to u(t\{s/x\})$$
$$(\lambda x.f(xx))\ (\lambda x.f(xx)) \to f((\lambda x.f(xx))\ (\lambda x.f(xx)))$$

Fixed Point Theorem

- For all lambda expressions F there exists X such that FX=X.

Proof. Suppose $W \equiv \lambda x.F(xx)$ and $X \equiv WW$.

$X \equiv WW \equiv (\lambda x.F(xx))W = F(WW) \equiv FX.$

Recursive Function

- Computable Functions as Recursive Functions
- Basic Idea
  - Some initial functions
  - Some rules to compose new functions
    - Composition
    - Recursion
    - Minimalization

Recursive Function

- The Zero function

  $f(x) = 0$
- The Successor function

  $f(x) = x+1$
- The Projection function

  $f(x_1, \ldots, x_n, i) = x_i$
- Substitution/composition

  $f(y_1, \ldots, y_n)=f(g(x_1), \ldots, g(x_n))$, where $y_i=g(x_i)$

Recursive Function

- Recursion

  h(x, 0) = f(x); h(x, y+1) = g(x, y, h(x, y))

  e.g. x+y: x+0=x; x+(y+1)=(x+y)+1

- Minimization

$$\mu y(f(\mathbf{x}, y) = 0) \simeq \begin{cases} \text{the least } y \text{ such that} \\ \quad f(\mathbf{x}, y) \text{ is defined for all } z \leq y, \text{ and} \\ \quad f(\mathbf{x}, y) = 0, \\ \text{undefined if otherwise.} \end{cases}$$

Ackermann Function

The Ackermann function is defined as follows:

$$
\begin{aligned}
\psi(0, y) &\simeq y + 1, \\
\psi(x+1, 0) &\simeq \psi(x, 1), \\
\psi(x+1, y+1) &\simeq \psi(x, \psi(x+1, y)).
\end{aligned}
$$

The Ackermann function is not primitive recursive.

It grows faster than all the primitive recursive functions.

Church Turing Thesis

- Fact
  - Equivalence in terms of expressive power
- Church Turing Thesis
  - The computable functions are precisely the recursive functions
- Logical foundation of computation
  - Computation as logical object

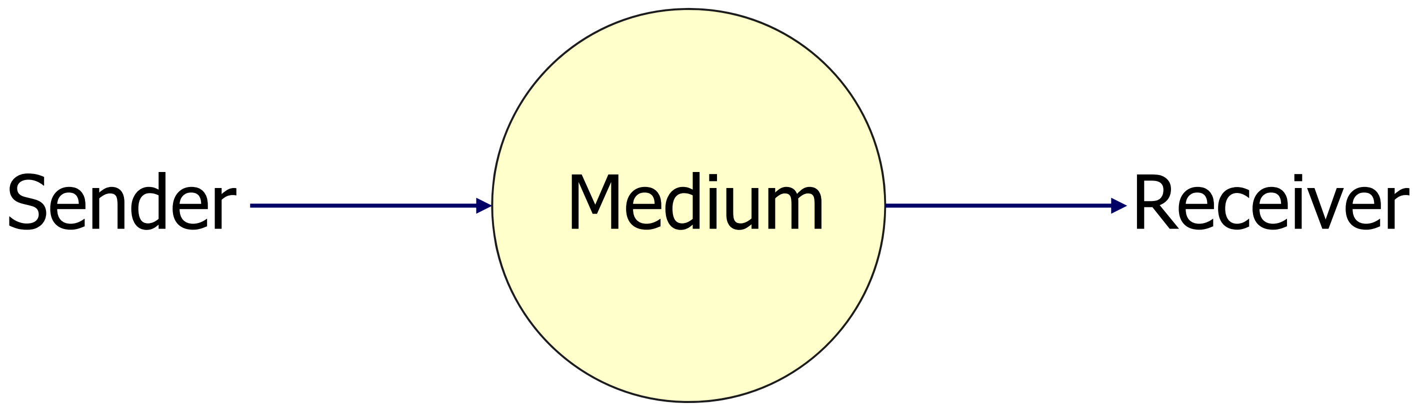
Concurrent System

- All Systems are Concurrent Systems
- Concurrency is an Intrinsic Property
- Concurrency is a Property Rather Than a Definition

EXAMPLES

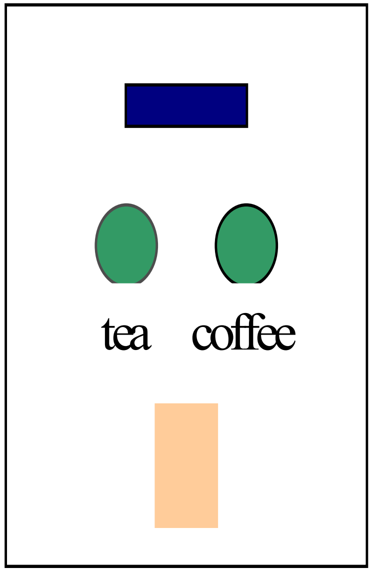
Example I: Ether Net

- Is the Capacity of Medium Unbounded?
- Is the Order of the Message Respected?



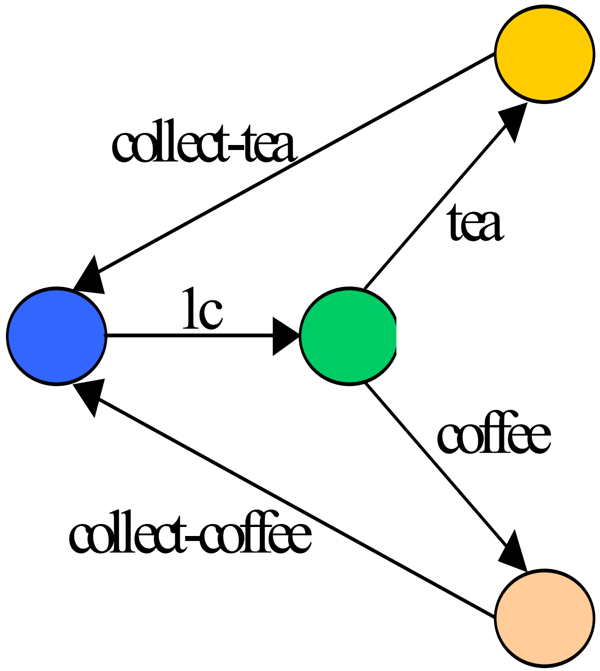
Example II: Vending Machine

- Vending Machine
  - The slot can accept 1c coin
  - The buttons can be pressed for a cup of tea or coffee
  - Tea or coffee can be collected from the tray
- Interaction between Machine and Buyer



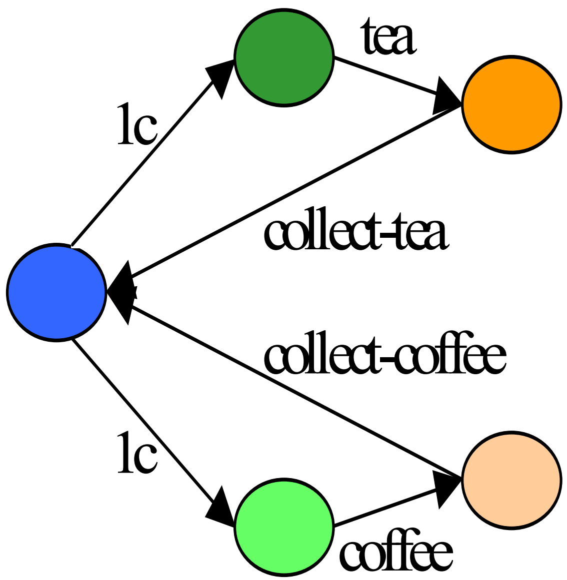
Transition

- The State Transition
  - Blue is the initial state
  - Green is the state after receiving 1c
  - One of the other two states is reached after the tea-button or the coffee-button is pressed



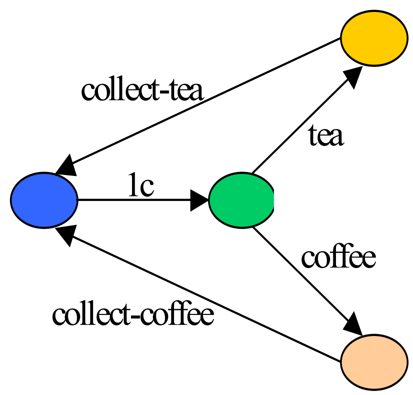
Nondeterminism

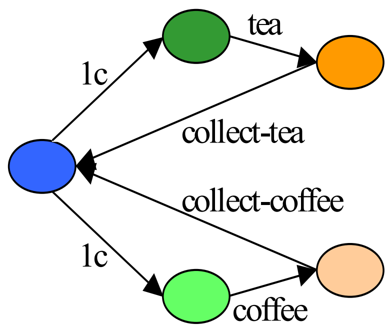
- The Vending Machine Out of Order
  - Having taking in 1c, the vending machine disable the tea-button or coffee-button nondeterministically



Equivalence

- Are The Two Systems Equivalent?
  - They are equivalent as automata
  - As concurrent systems they are not equivalent, because the ways buyers interact with them are different
- Observational Equivalence





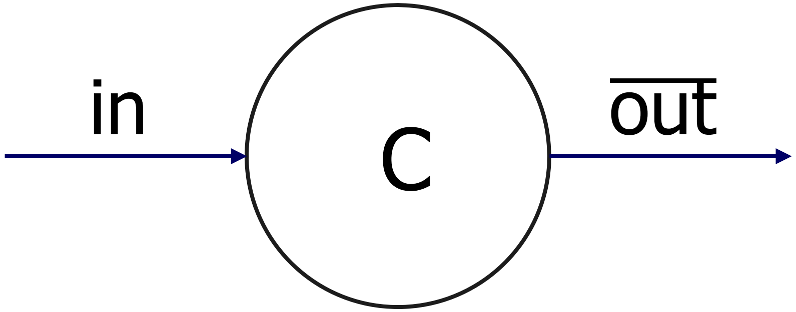
Example III: Concurrent Program

- Sequential Programs
  - X:=2
  - X:=1; X:=X+1
- Both Programs Assigns 2 to X
- Two Programs are Equivalent

- Concurrent Programs
  - X:=2 | X:=2
  - (X:=1; X:=X+1) | X:=2
- The Results are Nondeterministic
- Two Programs are not Equivalent
- Interleaving semantics

Example IV: A Unit Transmitter

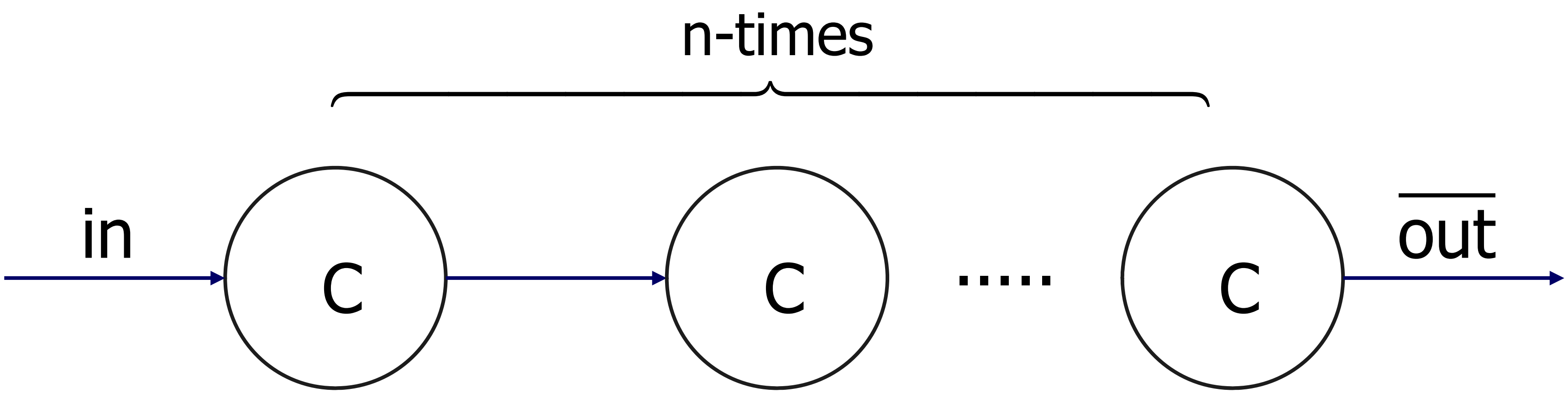
- The transmitter receives a message through in
- It sends out a message through out
- Repeat



$$C \stackrel{\text{def}}{=} in(x).C'(x)$$
$$C'(x) \stackrel{\text{def}}{=} \overline{out}(x).C$$

$$C \stackrel{\text{def}}{=} in(x).\overline{out}(x).C$$

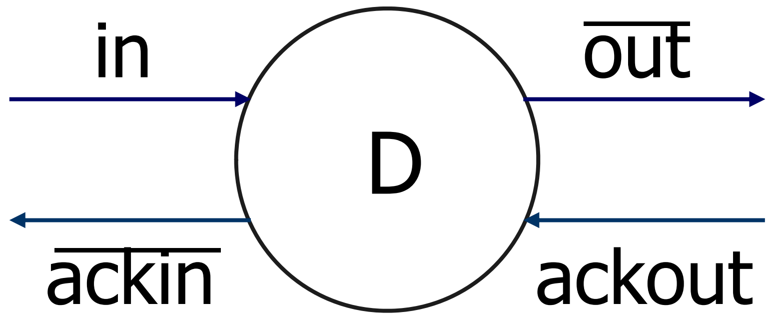
Connecting Unit Transmitter



How does it behave?

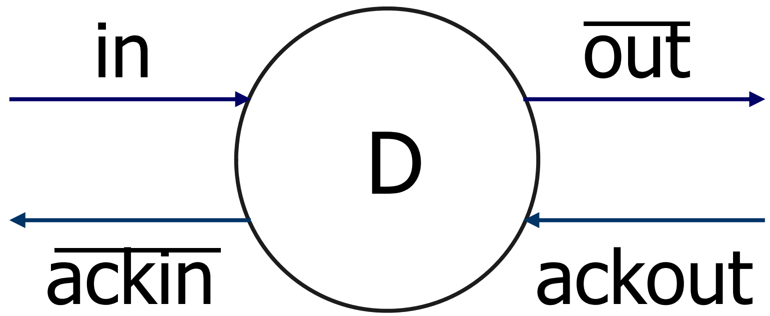
Example V: Transmitter with Acknowledgment

- The transmitter receives a message through in
- It sends out a message through out
- It receives acknowledgment through ackout
- It sends acknowledgment through ackin
- Repeat

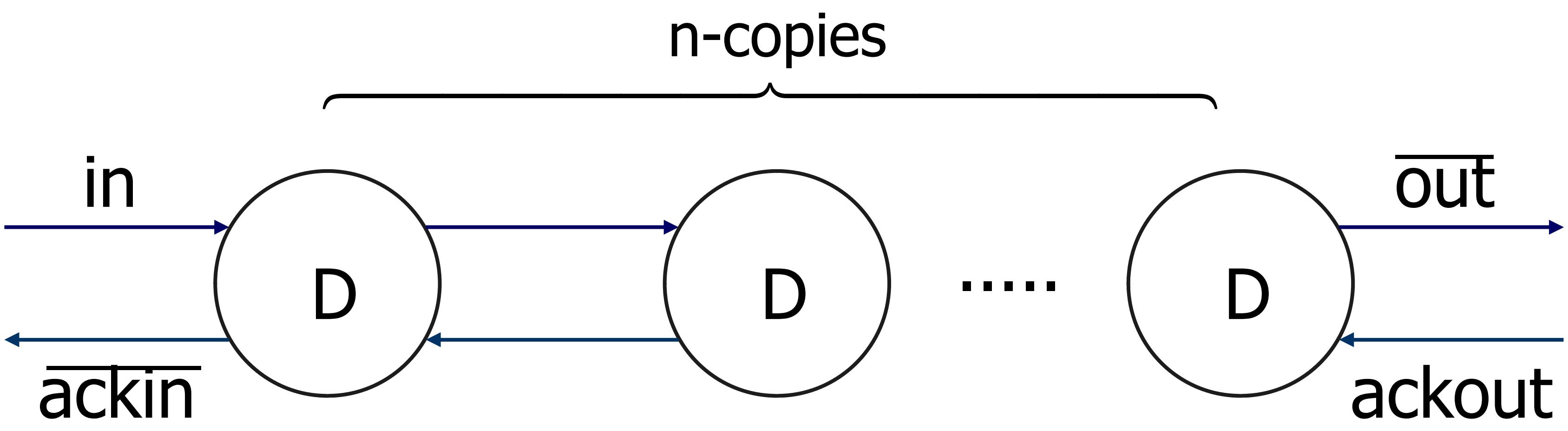


Example V: Transmitter with Acknowledgment

$$D \stackrel{\text{def}}{=} in(x).\overline{out}(x).ackout.\overline{ackin}.D$$

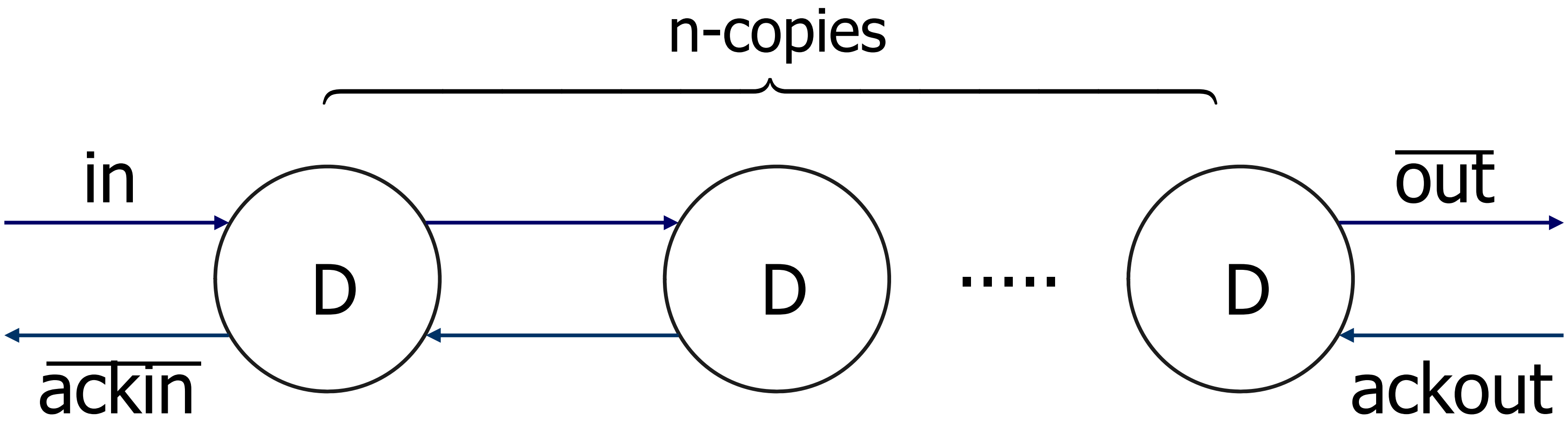


Chaining Acknowledgment



$$D \stackrel{\text{def}}{=} in(x).\overline{out}(x).ackout.\overline{ackin}.D$$

Chaining Acknowledgment



How does it behave?

Concurrent Computation

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\text{Sequential Computation} \arrow[r] \arrow[d] \arrow[dr, phantom, "\text{Fundamental Question}"] & \text{Concurrent Computation} \arrow[d] \\
\lambda\text{-Calculus} \arrow[r] & ?
\end{tikzcd}$$

Models of Concurrent Computing

- Petri Net (Petri;1962)
  - Information flow in concurrent systems
- Process Algebra (Bergstra;1970's)
  - Purely algebraic approach
- Process Calculus (Milner, Hoare;1970)
  - Operational approach
  - Subject to studies of combined approach

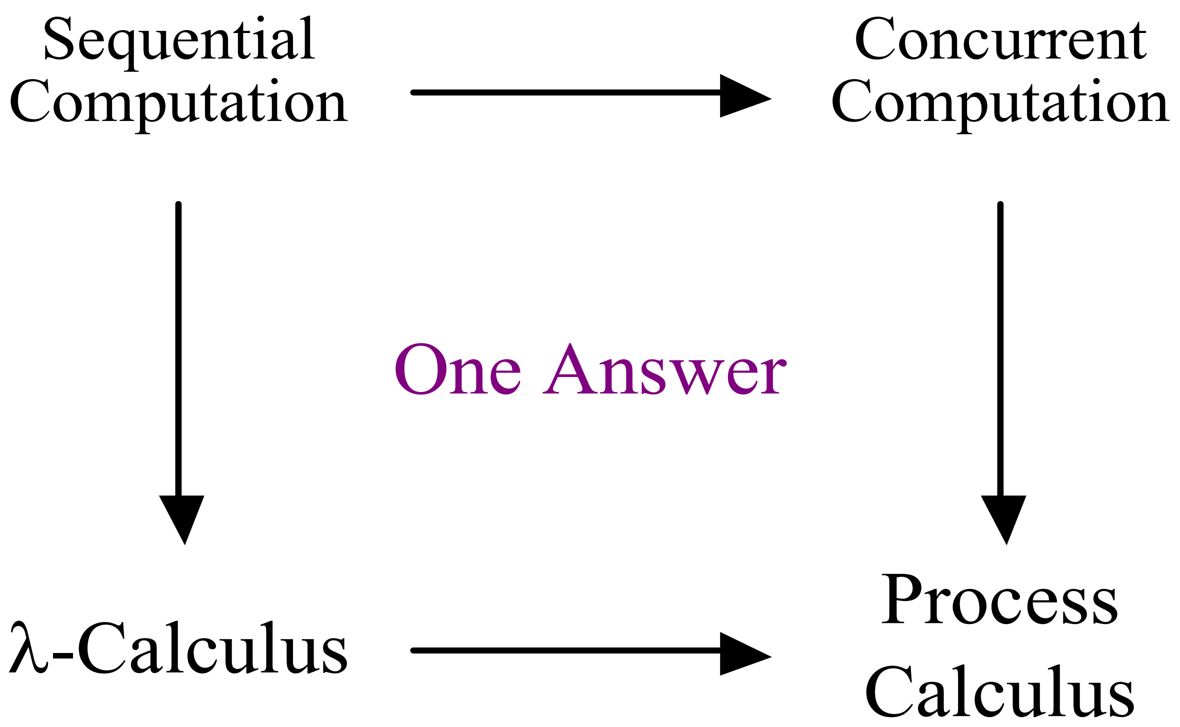
Concurrency Theory

- Studies on Semantic Models for
  - Distributed computing
  - Concurrent Computing
  - Mobile computing (mobile computation)
  - Grid Computing
  - Global Computing
  - Internet Computing
- Casts Light on Our Computing Practice

Concurrency Kaleidoscope

- Too Rich to be Understood at the Moment
- Too Rich to be Captured by one Formalism
- Many Complimentary Theories are Available

Process Calculus



Fundamental Questions

- Some Fundamental Questions:
  - What is a process?
  - How to describe a process?
  - What does a process do?
  - When are two processes equivalent?
- No Definite Answers, Of Course!
  - Research on the theory of process calculus proposes approximate answers

Analogy to λ-Calculus

- λ-Terms $\rightarrow$ Processes
  - What are processes?
- Application $\rightarrow$ Concurrent Composition
  - The fundamental operation for process calculi
- Reduction $\rightarrow$ Communication
  - Communications are atomic actions

The Mottos of Process Calculus

- Motto 1
  - All computing objects are processes;
    There are no other computing objects
- Motto 2
  - All computing actions are communications;
    There are no other kinds of actions
- Motto 3
  - A process lives to communicate

Syntactical Entities

- Communications Happen through Channels
- Two Classes of Syntactic Objects
  - Channels
  - Processes

Three Process Calculi

- CCS
  - Calculus of Communicating Systems
  - Milner;1969
- CSP
  - Communicating Sequential Processes
  - Hoare;1970
- π-Calculus
  - Milner,Parrow,Walker;1989

Robin Milner(1934-2010)

- 1991 Turing Award
- Three distinct and complete achievements:
  - LCF, the mechanization of Scott's Logic of Computable Functions, probably the first theoretically based yet practical tool for machine assisted proof construction
  - ML, the first language to include polymorphic type inference together with a type-safe exception-handling mechanism (http://smlnj.org/sml.html)
  - CCS, a general theory of concurrency. In addition, he formulated and strongly advanced full abstraction, the study of the relationship between operational and denotational semantics

Books

- Communication and Concurrency
  - Milner, Prentice Hall, 1989
- Communicating Sequential Processes
  - Hoare, OUP, 1980
- Communicating and Mobile Systems: the π-calculus
  - Milner, CUP, 1999
- The Pi-Calculus: a theory of mobile processes
  - Sangiorgi and Walker, CUP, 2001

Papers

- A Calculus of Mobile Processes
  - The Pioneering Paper
    - Milner, Parrow and Walker
    - Information and Computation, 1992
- An Introduction to the π- Calculus
  - J.Parrow. Chapter of Handbook of Processes Algebra. Elsevier. 2001
- Elements of interaction
  - R.Milner. Communication of ACM, (Jan):78-89, 1993

Websites

- Website for Mobile Process

  http://lampwww.epfl.ch/mobility/