Models of Concurrency

Frank D. Valencia

Uppsala University, Sweden
August 2003
Concurrency systems can combine: E.g., The Internet (a complex one)

- Timed systems,
- Secure systems,
- Mobile systems,
- Reactive systems,
  - Synchronous & Asynchronous systems,
  - Message-Passing & Shared-Memory systems,

Examples:

each other.

Concurrency systems: Multiple agents (processes) that interact among

...Concurrency is Everywhere

Model of Concurrency

Aug., 2003
Concurrent computation is usually:

Non-deterministic (undepictable).

Reactive (or Interactive)

Non-Terminating

Models for sequential computation (functions $f$: Inputs $\rightarrow$ Outputs) don't apply.

Need for formal models to describe and analyze concurrent systems.

...Concurrency: A Serious Challenge.
Probably because concurrency is a very broad (young) area.

- There is no yet a "canonical (all embracing) model" for concurrency.
- New models typically arise as extensions of well-established ones.
- There are several (too many?) models focused in specific phenomena.

In concurrency theory:

\[ \text{calculus} \]

Formal models must be simple, expressive, formal and provide techniques (e.g., A)

Need for Formal Models to describe and analyze concurrent systems.

Concurrency: A Serious Challenge.
Some Well-Established Models:

- Petri Nets: First well-established concurrency theory—extension of automata
- Sarpasa’s CCS (Shared Memory Communication)
- Milner’s π-calculus (CCS Extension to Mobility)
- Milner’s CCS & Hoare’s CSP (Synchronous Communication)
- Process Algebras (Process Calculi):

Need Formal Models to describe and analyze concurrent systems.

Concurrency: a Serious Challenge.
Agenda

• Basic concepts from Automata Theory. (1 Class)

• CCS (3 Classes)
  ▶ Basic Theory (2 Classes)
  ▶ Applications: Concurrency Work Bench Tool (1 Class)

• $\pi$-calculus (3 Classes)
  ▶ Basic Theory (2 Classes)
  ▶ Applications (1 Class).

• CCP (1 Class)
  ▶ Basic Theory
  ▶ Timed Extension: Ntcc and its Applications.

• Petri Nets (1/2 Class) (Optional)
  ▶ Basic Theory

• Current Hot Topics in Concurrency (1/2 Class)
  ▶ Security
  ▶ Verification/Bioinformatics.
Usually \( (q, a, b) \in \mathcal{L} \) is written as \( b \in \mathcal{L}_a(q) \). We omit \( \mathcal{L}_a \) when it is understood.

\[ \mathcal{O} \times A \times \mathcal{O} \subseteq \mathcal{L} = (\forall)_L \]

is the set of transitions.

\[ \mathcal{O} \subseteq f\mathcal{O} = (\forall)_f \mathcal{S} \]

is the set of accepting (or final) states.

\[ \mathcal{O} \subseteq 0\mathcal{O} = (\forall)_0 \mathcal{S} \]

is the set of initial states.

\[ \{ \cdots, b, 0b, 00b \} = \mathcal{O} = (\forall) \mathcal{S} \]

is the set of states.

Where

An automata over an alphabet \( \mathcal{A} \) is a tuple \( (\mathcal{O}, \mathcal{O}_0, \mathcal{F}, \mathcal{L}) \).
Automaton Example

Above \( A \) over \( \{a, b\} \) with \( S(A) = \{q_0, q_A, q_B, q_f\} \), \( S_0(A) = \{q_0\} \), \( S_f(A) = \{q_f\} \), \( T(A) = \{q_0 \xrightarrow{a} q_A, \ldots\} \).
Regular Expression (e.g., $a(qc + e)^*$ are equally expressible to $\text{FSA}$.

is regular iff $S = \mathcal{L}(A)$ for some $\text{FSA}$ $A$.

Regular sets are those recognizable by finite-state automata ($\text{FSA}$) i.e., $S$ accepted by $A$.

The language of (or recognized by) $A$, $\mathcal{L}(A)$, is the set of sequences and $(\forall)S \in \mathcal{L}(A)$, $\exists u \in \text{Act}^*$ \& $\exists v \in \text{Act}^*$ such that $uv$ is accepted by $A$ over $\text{Act}$.

Definition. (Acceptance, Regularity)

Regular Sets
forall s' ∈ T(B)

(1) Prove 2.b and 2.c.

Exercises:

(a) prove that language equivalence of two given FSA is decidable.
(b) Construct a FSA A such that

\( s \in L(A) \) if and only if every suffix of s, s' \( s' \in L(B) \).

(c) Regular sets are closed under (a) union, (b) complement, (c) intersection.

1. Deterministic and NonDeterministic FSA are equally expressive.

Proposition.

Automata: Some Nice Properties and Exercises
For interactive behavior.

If $T(A_1) = T(A_2)$ then language equivalence (trace equivalence) is too weak.

Example: Two Vending-Machines

In computer science.

Classic Automata Theory is solid and foundational, and it has several applications.

Automata Theory: What is it good for.

Mode of Concurrency

Aug. 2003
We need a stronger equivalence that does not validate the above.

\[ a \cdot c + q \cdot c = q \cdot c + q \]

That is, the states are equivalent.

\[ \text{(or automata)} \]

The theory allows to deduce that \[ a \cdot c + q \cdot c = q \cdot c + q \].

The problem with Classic Automata Theory.
A relation $R$ is a bisimulation if $R$ and its converse $R^{-1}$ are both simu-

• If $p \rightarrow q$ then there exists $q'$ such that $q \rightarrow p'$ and $(p', q') \in R$. 

A relation $R$ is a bisimulation if for every $(a, b) \in R$: 

A relation $R \subseteq S(L) \times S(L)$ is a bisimulation if $L$ be transition system.

Definition.

Transition systems are just automata in which final and initial states are irrelevant.

Simulation is Bisimulation Relations.
Question: If $d$ simulates $b$ and $b$ simulates $d$, are $d$ and $b$ similar?

Examples: In the previous example $p^0$ simulates $q^0$ but $q^0$ cannot simulate $p^0$, so $p^0$ and $q^0$ are not similar. See Example 3.7 and 3.11 in Milner’s Notes. Also, $p$ and $q$ are similar. We say that $p$ simulates $q$ if there exists a simulation $R$ such that $(d, p) \in R$. Thus, $(b, q) \in R$. 

Definition: We say that $p$ simulates $q$ if there exists a simulation $R$ such that $(d, p) \in R$. 

Bisimilarity.
Lence 8iven by Biisimilarity. Processes represented as transition systems and their behavioural equivalence.

Next: Process Calculi, in particular CCS.

Transition Systems, Bisimilarity.
Basic Classical Automata theory, and

Road Map

Mode of Concurrency

Aug '03
Expressiveness: How expressive are the constructs?

E.g.: Logical for expressing process specifications (Hennessy-Milner logic)

(Specificity) How to specify and prove process properties

(undecidability:

E.g.: Observable behavior, process equivalences, congruences and their

(Equivalences) How to compare processes

(Semantics) How to give meaning to the constructs

E.g.: Operational, denotational, or algebraic semantics

(Syntax) Constructs that fit the intended phenomena

Process Calculi: Key Issues
Actions $a$ and $\bar{a}$ are viewed as being "complementary." I.e., $a = \bar{a}$.

- Action $\top$ is called the silent or unobservable action.

- A set $\mathcal{A}$ of actions (ranged over by $a, b, \ldots$).

- $\mathcal{A} \cup \mathcal{J} = \mathcal{A}$ (A set of labels (ranged over by $1, 2, \ldots$).

- $\mathcal{N} \cup \mathcal{N} = \mathcal{J}$ (A set of names and names of co-names).

CCS: A calculus for synchronous communication.
The set of all processes is denoted by $p$.

\[
\{ q^n, \ldots, q^1 \} \models (d)^n f \quad \text{with} \quad d = (q^n \ldots q^1)\]

For each (call) $A(q^n, \ldots, q^1)$ there is a unique process definition.

Free names of $p^n q^n (p, q)$: Those with a not bound occurrence in $p$.

Bond names of $p^n f^n (p, q)$: Those with a bound occurrence in $p$.

$\langle q^n, \ldots, q^1 \rangle \models d \mid \emptyset + d \mid \emptyset \parallel d \mid d. \emptyset \mid 0 =: \ldots \emptyset. \emptyset \mid d$
CCS: Operational Semantics.
Example Write a CCS expression for the vending machine (in parallel with some

the states corresponding to \( p \) and are bisimilar.

Hence, define \( \mathcal{O} \sim p \) iff \( \mathcal{O} \sim \mathcal{O} \parallel p \). The labelled transition system of CCS has \( \mathcal{O} \) as its states and its transitions

CCS: Bisimilarity
between some $p_i$ and $p_j$. So, every move in $(\forall a)(\forall p)(p_i)\text{true}$(or a communication

\[
\left\{ p_i \leftarrow p_i \mid u\ p_i \mid \cdots \mid p_i \leftarrow p_i \mid u\ p_i \mid \cdots \right\} \cup \\
\left\{ p \not\in a, a \leftarrow a \mid u\ a \mid \cdots \mid a \leftarrow a \mid u\ p_i \mid \cdots \right\} \cup \\
\sim (u\ p_i \mid \cdots \mid \text{true}) (\forall a)
\]

summatation form: l.e.

More generally, we have the expansion law which allows to express systems in

Notice that $q.a.0 \parallel q.a.0 + q.a.0$.

CCS: The expansion law.
In fact, is a congruence. How can we prove this? The notion of congruence allows us to replace “equals with equals” where $C_p$ is a process context. i.e., we want $\sim$ to be a congruence.

$$[\emptyset] C_0 \sim [P] C_0$$

More generally we would like

$$R \parallel \emptyset \sim R \parallel P \parallel \emptyset \sim P$$

Suppose that $P \sim \emptyset$. We would like $P \sim R$.

Congruence Issues.
actions (i.e., actions \( \in J \)).

So, we look for another notion of equivalence focused in terms of \textit{observable} actions:

\[ p \not\sim p' \]

Notice that \( p \) is an \textit{unobservable} action. So \( \sim \) could be too.

In principle, \( p \) and \( p' \) should be \textit{equivalent} if another process (the environment, an observer) cannot observe any difference in their behavior.

\textbf{Other Equivalences: Observable Behavior}
The notion of experiment:

\[ \mathcal{J} \subseteq \mathcal{A} = \{ a_1, a_2, \ldots, a_n \} \]

where \( \mathcal{J} \) denotes a sequence of observable actions. Notice that for \( e = a_1, a_2, \ldots, a_n \),

\[ \mathcal{J} = \left( a_1 \right) \left( a_2 \right) \cdots \left( a_n \right) \]

A \( \mathcal{J} \) is valid if it is a sequence of observable actions. Think of an experiment as a sequence of observable actions. Think of any \( \mathcal{J} \) as an observation. An observation \( \mathcal{J} \) be observed by an action \( a \) by some \( P_{\mathcal{J}} \)'s environment.

Observations
\[(0 \parallel 0 \parallel 0 \parallel 0) \stackrel{t}{\sim} 0.0 + 0.0 \]

(not sensitive to deadlocks)

\[(0.0 + 0.0) \stackrel{t}{\sim} 0.0 \]

(not that nice:

\[(0.0 + 0.0) \stackrel{t}{\sim} 0.0 \]

\[\Downarrow \]

Examples:

\[\Leftrightarrow \emptyset \not\equiv! \Leftrightarrow \emptyset \]
Failures Equivalence

Definition. (*Failures Equivalence*). A pair \((e, L)\), where \(e \in \mathcal{L}^*\) (i.e. a trace) and \(L \subseteq \mathcal{L}\), is a failure for \(P\) iff

\[ (1) P \xrightarrow{e} P' \quad (2) P' \xrightarrow{l} \quad \text{for all } l \in L, \quad (3) P' \xrightarrow{\top} \]

\(P\) and \(Q\) are failures equivalent, written \(P \sim_f Q\), iff they have the same failures.

**Fact** \(\sim_f \subseteq \sim_t\).

**Examples.**

- \(\tau.P \sim_t P\)

- \(a.b.0 + a.c.0 \not\sim_f a.(b.0 + c.0)\) (*Exercise*: why?)

- \(a.b.0 + a.0 \not\sim_f a.b.0\).

- \(a.0 + b.0 \not\sim_f (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)\) (*Exercise*)
• $a.(b.c.0 + b.d.0) \sim_f a.b.c.0 + a.b.d.0$.

• Let $D = \tau.D$. We have $\tau.0 \not\sim_f D$. 
Weak Bisimilarity (Observational Equivalence)

Definition. A symmetric binary relation \( R \) on processes is a weak bisimulation iff for every \((P, Q) \in R:\)

. If \( P \xrightarrow{e} P' \) and \( e \in \mathcal{L}^* \) then there exists \( Q' \) such that \( Q \xrightarrow{e} Q' \) and \((P', Q') \in R.\)

\( P \) and \( Q \) are weakly bisimilar, written \( P \approx Q \) iff there exists a weak bisimulation containing the pair \((P, Q)\).

Examples. .

- \( \tau.P \approx P \), \( a.\tau.P \approx a.P \)
- However, \( a.0 + b.0 \not\approx a.0 + \tau.b.0 \)
- \( a.(b.c.0 + b.d.0) \not\approx a.b.c.0 + a.b.d.0. \) (Exercise)
- Let \( D = \tau.D \). We have \( \tau.0 \approx D \).
- \( a.0 + b.0 \not\approx (\nu c)(c.0 || \bar{c}.a.0 || \bar{c}.b.0) \) (Exercise)
An alternative definition of weak bisimulation.

Proposition. \( R \) is a weak bisimulation iff

\[
\hat{R} \subseteq \hat{A} \quad \text{such that} \quad \forall a \in \hat{A} \exists p \in R \quad \text{and} \quad a = a \quad \text{and} \quad a \in R \quad \text{and} \quad a \in \forall \hat{R}.
\]

An alternative definition of weak bisimulation:

where \( a = a \) if \( a \in \mathcal{J} \), i.e., observable, otherwise \( a = e \).

Testing bisimulation using the previous definition could be hard, there are

\( e \) instead of \( e \). Fortunately, we have an alternative formulation.

\( e \) instead of \( e \). Fortunately, we have an alternative formulation.

an infinite many experiments \( e \). Fortunately, we have an alternative formulation.
Process Logic: Verification and Specification

A more traditional way in Computer Science is to use logics for specification and verification of properties. A logic whose formulae can express, e.g.,

\[ \text{forall } \text{predicates } \forall x. \text{pred} \text{; } \exists x. \text{pred} \text{.} \]

Processes can be used to specify and verify the behaviour of systems, e.g., vending machines. For instance, the behaviour of a vending machine can be specified by a formula that expresses that it will never accept a bad action or eventually execute a good action.

\[ \text{forall } \text{actions } \forall a. \text{action} \text{; } \exists a. \text{action} \text{.} \]
evolve into a which satisfies $P$.

$a \in K$ and then evolve into a that satisfy $P$.

$a \in K$ then it must be possible for $P$.

The boolean operators are interpreted as in propositional logic.

where $K$ is a set of actions

$H[K] \mid H \langle K \rangle \mid H \land I \mid H \lor I \mid \text{true} \mid \text{false}$

The syntax of the logic.

Hennessey-Milner Logic
Theorem. \( P \iff Q \leftrightarrow P \iff P \uparrow \uparrow \iff P \iff P \).

Notice that \( P \models P \uparrow \uparrow P \iff P \neq P \).

\( \langle \{c\} \rangle \lor \langle \{q\} \rangle \langle \{a\} \rangle = P \)

Example. Let \( P_1 = a(q0 + c0) \), \( P_2 = a \cdot q0 + a \cdot c0 \). Also let

\( P \models P_1 \iff \langle K \rangle \models P_2 \).

Semantics of Henney-Young Model of Concurrency
asserts (of a given trace $s$) that at every point in $s$, holds.

asserts (of a given trace $s$) that at some point in $s$, holds.

$\{\bot\} \cap T$ holds.

Boolean operators are interpreted as usual.

Formulate assert properties of traces.

where $T$ is a set of non-silent actions.

$\Box | \Diamond | \lnot \Diamond | \Diamond \lor \top | \Diamond \land \top | \bot | \text{false} | \text{true}$

The syntax of the formulae is given by

A Linear Temporal Logic.
$H = \models \langle \ell', s \rangle \text{ if there is a } \ell \leq \ell' \text{ s.t. } \varphi$ iff $H \models \langle \ell', s \rangle \text{ if for all } \ell \text{ s.t. } \ell < \ell' \varphi$,
$H = \models \langle \ell', s \rangle \text{ and } H = \models \langle \ell'', s \rangle$ iff $H = \models \langle \ell', s \rangle \lor H = \models \langle \ell'', s \rangle$,
$H = \models \langle \ell', s \rangle \text{ or } H = \models \langle \ell'', s \rangle$ iff $H = \models \langle \ell', s \rangle \land H = \models \langle \ell'', s \rangle$,
$T = \models \langle \ell', s \rangle$ iff $\exists T \in \ell \land T \in \ell''$, written $s = \models H$ iff $\langle \ell', s \rangle$ is a model of $H$. An infinite sequence of actions $s = a_1, a_2, \ldots \text{ satisfies } (\tau \text{ is a model of } H$.}

Semantics of the temporal logic.
Question. Does the other direction of the theorem hold?

\[
\emptyset \models p \iff \emptyset \models p
\]

Then for every linear temporal formula \( \mathcal{F} \),

\[
\emptyset \models \mathcal{F} \iff \mathcal{F}
\]

Notice that the trace equivalent processes \( A(q, q', c) \) and \( B(q, q', c) \) satisfy

\[
\langle A(q, q', c) \rangle \triangleq a.q.B(q, q', c) \quad + \quad A.q.B(q, q', c)
\]

and

\[
(\langle A(q, q', c) \rangle A(q, q', c) \quad + \quad \langle A(q, q', c) \rangle A(q, q', c))
\]

Example. Consider the linear temporal logic
Road Map

✔ Basic classic automata theory and the limitation of language equivalence.

✔ Bisimilarity Equivalence for automata (transition systems).

✔ A basic process calculus for synchrony (CCS)
  ▶ CCS behaviour represented as transitions systems
  ▶ CCS behaviour compared by using bisimilarity, trace equivalence,
    failures equivalence, weak bisimilarity.
  ▶ Specification of CCS properties using HM and Temporal Logics.

• Mobility and the $\pi$ calculus.
Mobility

What kind of \textit{process mobility} are we talking about with?

- Processes move in the \textit{physical space}.

- Processes move in the \textit{virtual space} (of linked processes).

✓ \textit{Links} move, in the \textit{virtual space} (of linked processes).

The last one is the $\pi$-calculus’ choice; for economy, flexibility, and simplicity.

- In the $\pi$ calculus extends CCS with the ability of sending private and public \textit{links (names)}.
\[
\ldots \parallel d \parallel d \parallel d \ \\
\text{replicates} \ p, \ i.e., \ \exists d' \ \text{representing} \ d \ \\
\text{and} \ \nu a(x) \ \text{are the only binders.} \ p(\nu a)
\]

\[
\text{create a fresh name} \ a \ \text{private to} \ p
\]

\[
\text{receive a name on channel} \ a \ (\text{if any}), \ \text{and replace} \ x \ \text{within it in} \ p
\]

\[
\text{send} \ q \ \text{on channel} \ a \ \text{and then activate} \ p
\]

\[
\text{Names=Channels=Ports=Links.}
\]

\[
\text{where} \ \alpha =: \ \ldots \cdot \emptyset \ p
\]

\[
\begin{align*}
\text{if} \ & \text{then} \ q = a \ | \ d \ i \ | \ d(\nu a) \ | \ p \ . \ \alpha \ . \ \emptyset \ | \ \emptyset \ | \ d \\
\end{align*}
\]

\[
\text{Syntax: Calculus}
\]
The System Client || Server

\[
\text{The Client } \text{Client} = \text{Client} = s(e)(\text{Print})
\]

\[
\text{The printer-server Server} = (p(d)) s(u) (d) s(t) (p(d)) \text{Print}
\]

Client & Printer-Server

Mobile Example
Generates infinity many different names—and sends them all along channel $a$.

But $R$ and $P$, $Q$ cannot communicate.

$R$ contains three agents, $P$, $Q$ such that $P$, $Q$ can communicate with both and $P$.

Each process receives $Q$ and $R$ on channel $a$ so that only one (sequential) process receives $Q$ and $R$ on channel $a$.

Reads two names (ports) and sends the first along the second.

Reads something from port $a$ and sends it twice along port $b$.

Write an agent that

**Mobile Exercises**
\[ d_i \parallel d \equiv d_i \]

\[ \neg u \not\in \mathcal{D}(v, a) \parallel d \equiv (\mathcal{O} \parallel d)(v, a) \]

\[ p(v) \equiv p(q) \equiv p \quad 0 = 0(v) \]

\[ (\mathcal{H} \parallel \mathcal{O}) \parallel d = \mathcal{H} \parallel (\mathcal{O} \parallel d), \quad \mathcal{O} \equiv \mathcal{O} \parallel d \equiv 0 \parallel d \]

\[ \mathcal{O} \equiv d \quad \text{if and only if} \quad \mathcal{O} \equiv \mathcal{O} \parallel d \equiv 0 \parallel d \]

Process equivalence satisfies:

**Definition (Structural Congruence)** The relation \( \equiv \) given is the smallest

\[ \text{Intuitively, } \equiv \text{ describes irrelevant syntactic aspects of processes. Formally,} \]

\text{The reactive semantics of } \equiv \text{ consists of an structural congruence and the}

\text{Reactive Semantics of } \not\in \text{ Model of Concurrency