Contents

BASIC CONCEPTS IN CONCURRENCY **5**

- Definitions 5
  - A sequence of actions 5
  - process 5
  - Threads, 5
    - in specific languages 5
    - Some authors 5
- Execution Order 5
- nondeterminism. 5
  - A program is deterministic if, 5
  - a program is nondeterministic if 5
  - makes it difficult 5
- mechanisms 6
  - communication between processes, 6
  - coordination between processes, 6
  - atomicity, 6
- message-passing mechanisms: 6
  - Buffering. 6
    - unbuffered communication 6
  - Synchronicity. 6
    - With asynchronous communication, 6
  - Message Order. 6
    - If a mechanism preserves transmission order, 6
- An action is atomic 6
  - A nonatomic action 7
  - any concurrent programming language 7
- Mutual Exclusion 7
  - for certain operations, 7
  - critical section. 7
  - mechanisms. 7
    - Mutual exclusion: 7
    - Progress: 7
    - Bounded waiting: 7
  - A general approach 7
  - Using a loop to wait for some condition is called 8
- Deadlock 8
  - technique that prevents deadlock 8
  - two-phase locking, 8
- Semaphores 8
  - A standard semaphore 8
    - is represented 8
    - Initially, 8
    - The maximum 8
    - If a program contains several shared resources, 8

When a process waits on a semaphore, ... 8
If the value is greater than 0, ... 8
If a wait is executed on a semaphore whose integer value is 0, ... 9
Suspending a process 9
When a process leaves a critical section, ... 9
If the queue is not empty, ... 9
If no process is waiting, ... 9
Monitors 9
Monitors, ... 9
A monitor consists of 9
local data 9
In traditional terminology, ... 9
In modern terminology, ... 9

THE ACTOR MODEL **10**
Each actor 10
three basic actions that an actor may perform: 10
Actor computation 10
an actor program 10
An actor is dormant until 10
When an actor receives a message, ... 10
After executing its script, ... 10
The replacement behavior 10
In any computation, ... 10
messages 11
the mail system, ... 11
Every message must be sent to 11
When an actor A specifies a replacement behavior, ... 11
A message from one actor to another 11
A task has three parts: 11
a behavior may be defined as a function of parameters, which are called 11
actor-specific concept 11
forwarder. 11
General aspects of actors 11
An actor changes state 11
The motivation for using asynchronous communication 11
problems 12

CONCURRENT ML **12**
Threads and Channels 12
A CML process is called 12
When a CML program begins, ... 12
When spawn f is evaluated, ... 12
The return value of the function 12
The thread that evaluates spawn f is called 12

the thread running f() is called 12
the parent–child relation 12
Another CML primitive, ... 12
is forever. 12
channel. 13
operations on channels 13
CML message passing 13
Channels are created by 13
Functions Using Threads and Channels 13
A guarded command, ... 13
The intent 13
Events 13
we can decompose send into two parts: 13
Selective Communication 13
The select operator 13
The function choose 14

Paradigma de scripting **14**
En principio, cualquier lenguaje 14
los lenguajes específicamente de scripting están pensados para 14
típicamente un "script"se ejecuta de 14
Tipos de lenguajes de scripting 14
propósito general, ... 14
específicos de dominio 14
Los lenguajes "pegamento" (glue languages) 14
Algunos ejemplos 14
son especialmente útiles para 15
Los lenguajes de macros que 15
Otras herramientas como 15
Con el advenimiento de interfaces gráficas de usuario, ... 15
surgió un tipo especializado de lenguaje de scripting. 15
A estos lenguajes también se les llama 15
Muchos programas de aplicación grandes incluyen 15
muchos juegos de computadora utilizan 15

SECURITY **16**
One way that attackers may try to gain access to a system 16
Mobile code, ... 16
The two main mechanisms for managing risks raised by mobile code are called 16
The idea behind sandboxing, ... 16
Code signing 16
Buffer Overflow Attack 16
if the number of bytes in the buffer are not checked, then 16
By writing over the return address of a function, the attacker . . 16
Java code is not 16

The Java Sandbox 17
    The JVM executes Java bytecode in 17
    The word sandbox is used to indicate that, ... 17
    mechanisms: 17
Class Loader 17
    contributes to the Java sandbox in three ways: 17
        separates trusted class libraries from 17
        provides separate name spaces for 17
        places code into 17
The Bytecode Verifier and Virtual Machine Run-Time Tests 17
    provide the following guarantees: 17
        No Stack Overflow or Underflow: 17
        All Methods are Called with Parameters of the Correct Type: 18
        No Illegal Data Conversions (Casts) Occur: 18
        Private, Public, Protected, and Default Accesses must be Legal: 18
The Security Manager 18
    The security manager is 18
    The job of the security manager is 18
    The security manager does its job by 18
    Each protection domain has two attributes 18
    The signer is 18
    The location is 18
    A standard security manager will 18
    A running virtual machine can have only one 19
    once a security manager has been 19
    how the standard Java library uses the security manager 19
        the program makes a call to a potentially dangerous operation 19
        If the operation is permitted, ... 19
        If the operation is not permitted, ... 19
        One limitation of the security manager using exceptions is that 19
Security and Type Safety 19
    all of the sandbox mechanisms rely on 19
    ways that type errors can allow code to perform arbitrary actions. 19
        One way for a C/C++ program to call an arbitrary function is 19
        type confusion between object types would 19
        a buggy class loader and linker. 20

BASIC CONCEPTS IN CONCURRENCY

Definitions

A sequence of actions

may be called a process, thread, or task.

process

is often used to refer to an operating system process, which generally runs in its own address space.

Threads,

in specific languages
like Concurrent ML and Java, may run under control of the language run-time system, sharing the same operating system address space.

Some authors
define thread to mean **“lightweight process,”** which means a process that does not run in a separate operating system address space.

Execution Order

Concurrent programs may have many possible execution orders.

nondeterminism.

A program is deterministic if,

for each sequence of program inputs, there one sequence of program actions and resulting outputs.

a program is nondeterministic if

there is more than one possible sequence of actions corresponding to a single input sequence.

makes it difficult

to design and debug programs.

It is difficult to think carefully about millions of possible execution orders

mechanisms

communication between processes,

achieved by mechanisms such as buffered or synchronous communication channels, broadcast, or shared variables or objects,

coordination between processes,

which may explicitly or implicitly cause one process to wait for another process before continuing,

atomicity,

which affects both interaction between processes and the handling of error conditions.

message-passing mechanisms:

Buffering.

If communication is buffered, then every data item that is sent remains availableuntil it is received.In

unbuffered communication
a data item sent before the receiver is ready to accept that it may be lost.

Synchronicity.

In synchronous communication, the sender cannot transmit a data item unless the receiver is ready to receive it.

With asynchronous communication,

the sending process may transmit a data item and continue executing even if the receiver is not ready to receive the data.

Message Order.

A communication mechanism may preserve transmission order or it may not.

If a mechanism preserves transmission order,
then a sequence of messages will be received in the order that they are sent.

An action is atomic

if every execution will either complete successfully or terminate in a state that is indistinguishable from the state in which the action was initiated.

A nonatomic action

may involve intermediate states that are observable by other processes. A nonatomic action may also halt in error before the entire action is complete, leaving some trace of its partial progress.

any concurrent programming language

must provide some atomic actions, because, without some guarantee of atomicity, it is extremely difficult to predict the behavior of any program.

Mutual Exclusion

for certain operations,

it is important to restrict concurrency so that only one process proceeds at a time.

critical section.

A critical section is a section of a program that accesses shared resources. Because a process in a critical section may disrupt other processes, it may be important to allow only one process into a critical section at a time.

mechanisms.

Mutual exclusion:
Only one process at a time may be in its critical section.

Progress:

If no processes are in their critical section and some process wants to enter a critical section, it becomes possible for one waiting process to enter its critical section.

Bounded waiting:

no waiting process should have to wait indefinitely.

if one process halts in a critical section, other processes will eventually be able to enter the critical section.

A general approach

The main idea is that each process executes some kind of **wait** action before and executes some kind of signal action afterward.

Using a loop to wait for some condition is called

busy waiting.

Deadlock

occurs if a process can never continue because the state of another process. processes are waiting for the lock held by the other process. In this situation, neither process can proceed and deadlock has occurred.

technique that prevents deadlock

two-phase locking.

two-phase locking,

a process is viewed as a sequence of independent tasks. For each task, the process must first acquire all locks that are needed to perform the task. Before proceeding from one task to the next, a process must release all locks.

Semaphores

A standard semaphore

is represented

by an integer variable, an integer maximum, and a queue of waiting processes.

Initially,
the integer variable is set to the maximum.

The maximum
indicates the number of processes that may enter a critical section at the same time; in many situations the maximum is one.

If a program contains several shared resources,
the program may use a separate semaphore for each resource.

When a process waits on a semaphore,

the wait operation checks the integer value of the semaphore.

If the value is greater than 0,
this indicates that a process may proceed. The value is decremented before proceeding to limit the number of processes that are allowed to proceed.

If a wait is executed on a semaphore whose integer value is 0,

then the process is suspended and placed on a queue of waiting processes.

Suspending a process

is an operating system operation that keeps the process from continuing until the process is resumed.

When a process leaves a critical section,

The signal operation checks the semaphore queue to see if any process is suspended.

If the queue is not empty,
one of the suspended waiting processes is allowed to run.

If no process is waiting,
then the integer value of the semaphore is incremented.

Monitors

Monitors,

place the responsibility for synchronization on the operations that access data. Monitors are similar to abstract data types, with all synchronization placed in the operations of the data type. This makes it easier to write correct client code.

A monitor consists of

one or more procedures, an initialization sequence, and local data.

local data
are accessible only by the monitor procedures, which are responsible for all synchronization associated with concurrent access to the data.

In traditional terminology,

a process **enters** the monitor by invoking one of its procedures.

In modern terminology,

a monitor might be called a **synchronized object.**

THE ACTOR MODEL

Each actor

is a form of reactive object, executing some computation in response to a message and sending out a reply when the computation is done. Actors do not have any **shared state,** but use buffered asynchronous message passing for all communication.

three basic actions that an actor may perform:

It may send communication to itself or other actors,

It may create actors,

It may specify a replacement behavior,

which is essentially another actor that takes the place of the actor that creates it for the purpose of responding to later communication.

Actor computation

is reactive, which means that computation is performed only in response to communication.

an actor program

creates some number of actors and sends them messages. All of these actors can react to messages concurrently, but there is no explicit concept of thread.

An actor is dormant until

it receives communication.

When an actor receives a message,

the script of the actor may specify subsequent communication and a set of new actors to create.

After executing its script,

the actor returns to its dormant state.

The replacement behavior

specifies how the actor will behave when it receives another message.

In any computation,

each actor receives a linearly ordered sequence of messages.

messages

are not guaranteed to arrive in the order in which they are sent.

the mail system,

routes and buffers messages between actors.

Every message must be sent to

a mail address; one actor may communicate with another if it knows its mail address.

When an actor A specifies a replacement behavior,

the replacement behavior will be the script for a new actor that receives all messages addressed to the mail address of A.

A message from one actor to another

is called a task.

A task has three parts:

A unique tag, distinguishing it from other tasks in the system,

A target, which is the mail address of the intended receiver,

A communication, which is the data contained in the message.

a behavior may be defined as a function of parameters, which are called

acquaintances:

actor-specific concept

forwarder.

A forwarder actor does nothing itself, except forward all tasks to another actor.

General aspects of actors

An actor changes state

only by becoming another actor after it completes processing input.

The motivation for using asynchronous communication

is that it is easily implemented on a wide-area network.

problems

message order, message delivery, and coordination between sequences of concurrent actions

CONCURRENT ML

Threads and Channels

A CML process is called

a thread.

When a CML program begins,

it consists of a single thread. This thread may create additional threads by using the **spawn** primitive:

When spawn f is evaluated,

the function call f() is executed as a separate thread.

The return value of the function

is discarded, but the function may communicate with other threads by using channels.

The thread that evaluates spawn f is called

the parent thread

the thread running f() is called

the child thread.

the parent–child relation

does not have any impact on thread execution either thread may terminate without affecting the other.

Another CML primitive,

is forever.

This function takes an initial value and a function that can be repeatedly applied to this value:

channel.

For each CML type 'a, there is a type 'a chan of channels that communicate values of type 'a.

operations on channels

recv : 'a chan → 'a → send : ('a chan * 'a) → unit

CML message passing

is synchronous, meaning that communication occurs only when both a sender and a receiver are ready to communicate.

Channels are created by

the channel constructor,

channel : unit → 'a chan

Functions Using Threads and Channels

Because channels are "just another type" in CML, we can define functions that take channel arguments and return channel results.

A guarded command,

has historically been written in the form

Condition ⇒ Command

consists of a predicate and an action.

The intent

is that, if the Condition is true, then the Command may be executed.

Events

we can decompose send into two parts:

the code that will cause a **send** to occur is the **send event,** and doing the send is called **synchronizing on this event.**

Selective Communication

The select operator
for selective communication can be defined from a primitive function on events that chooses an event out of a list.

The function choose

choose : 'a event list → 'a event

takes a list of events and returns one event from the list. choose must return an event that can be synchronized on if there is one in the list.

Paradigma de scripting

En principio, cualquier lenguaje

puede ser utilizado como un lenguaje de scripting, siempre que cuente con las librerías o bindings para un entorno específico.

los lenguajes específicamente de scripting están pensados para

ser muy rápidos de aprender y escribir, ya sea como archivos ejecutables o de forma interactiva en un (REPL).

típicamente un "script"se ejecuta de

principio a fin, como un "guión", sin un punto de entrada explícito.

Tipos de lenguajes de scripting

propósito general,

fueron dise~nados originalmente para "pegar"las salidas y entradas de otros programas, para construir sistemas más grandes.

específicos de dominio

están pensados para extender las capacidades de una aplicación o entorno, ya que permiten al usuario personalizar o extender las funcionalidades mediante la automatización de secuencias de comandos.

Los lenguajes "pegamento" (glue languages)

son los que se especializan para conectar componentes de software.

Algunos ejemplos

los pipes y las secuencias de comandos del shell. conectar un servidor web con diferentes servicios web, como una base de datos.

son especialmente útiles para

comandos personalizados para una consola de comandos;

programas más peque~nos que los que están mejor implementados en un lenguaje compilado;

programas de ¸contenedor”para archivos ejecutables, como manipular archivos y hacer otras cosas con el sistema operativo antes o después de ejecutar una aplicación

secuencias de comandos que pueden cambiar;

prototipos rápidos de una solución.

Los lenguajes de macros que

manipulan componentes de un sistema o aplicación específicos pueden funcionar también como lenguajes de pegamento.

Otras herramientas como

AWK también pueden ser considerados lenguajes pegamento, al igual que cualquier lenguaje implementado por un motor de Windows Script Host

Con el advenimiento de interfaces gráficas de usuario,

surgió un tipo especializado de lenguaje de scripting.

Estos lenguajes interactúan con los gráficos de ventanas, menús, botones, etc., estos lenguajes se usan normalmente para automatizar las acciones del usuario.

A estos lenguajes también se les llama
”macros”si el control es a través de la simulación de las acciones de presionar teclas o clics del ratón,

Muchos programas de aplicación grandes incluyen

un peque~no lenguaje de programación adaptado a las necesidades del usuario de la aplicación.

muchos juegos de computadora utilizan

un lenguaje de programación específico para expresar las acciones de personajes no jugadores y del entorno del juego.

SECURITY

One way that attackers may try to gain access to a system

is to send data over the network to a program that processes network input.

Mobile code,

code transmitted over the network before it is executed, provides another way for an attacker to try to gain access to a system

The two main mechanisms for managing risks raised by mobile code are called

sandboxing and code signing.

The idea behind sandboxing,

is to give a program a restricted execution environment

Code signing

is used to determine which person or company produced a class file.

With digital signatures, it is possible for the producer of code to sign it in a way that any recipient can verify the signature.

Buffer Overflow Attack

an attacker sends network messages that cause a program to read data into a buffer in memory.

if the number of bytes in the buffer are not checked, then

the attacker may be able to write past the allocated memory.

By writing over the return address of a function, the attacker

may cause the program to misbehave and "return" by jumping to instructions chosen by the attacker. This is a powerful form of attack.

Java code is not

vulnerable to buffer overflow attacks because the JVM checks array bounds at run time. This prevents a function from writing more data into an array than the array can hold.

The Java Sandbox

The JVM executes Java bytecode in

a restricted environment called the Java sandbox.

The word sandbox is used to indicate that,

when bytecode is executed, some operations that can be written in the Java language might not be allowed to proceed,

mechanisms:

the class loader, the Java bytecode verifier, run-time checks performed by the JVM, and the actions of the security manager.

Class Loader

contributes to the Java sandbox in three ways:

separates trusted class libraries from

untrusted packages by making it possible to load each with different class loaders.

provides separate name spaces for

classes loaded by different class loaders.

places code into
categories (called **protection domains**)

that let the security manager restrict the actions that specific code will be allowed to take.

The Bytecode Verifier and Virtual Machine Run-Time Tests

The Java bytecode verifier checks Java bytecode programs before they are executed, Complimentary run-time checks, are performed by the JVM.

provide the following guarantees:

No Stack Overflow or Underflow:

The verifier examines the way that bytecode manipulates data on the operand stack and guarantees that no method will overflow the operand stack allocated to it.

All Methods are Called with Parameters of the Correct Type:

This type-correctness guarantee prevents type-confusion attacks

No Illegal Data Conversions (Casts) Occur:

For example, treating an integer as a pointer is not allowed.

Private, Public, Protected, and Default Accesses must be Legal:

no improper access to restricted classes, interfaces, variables, and methods will be allowed.

The Security Manager

The security manager is

a single Java object that answers at run time.

The job of the security manager is

to keep track of which code is allowed to do which dangerous operations.

The security manager does its job by

examining the protection domain associated with a class.

Each protection domain has two attributes

a signer and a location.

The signer is

the person or organization that signed the code before it was loaded into the virtual machine. This will be null if the code is not signed by anyone.

The location is

the URL where the Java classes reside.

A standard security manager will

disallow most operations when they are requested by untrusted code and may allow trusted code to do whatever it wants.

A running virtual machine can have only one

security manager installed at a time.

once a security manager has been

installed, it cannot be uninstalled without restarting the virtual machine.

how the standard Java library uses the security manager

the program makes a call to a potentially dangerous operation

the associated Java API code asks the security manager whether the operation should be allowed.

If the operation is permitted,
the call to the security manager returns normally

and the Java API performs the requested operation.

If the operation is not permitted,

the security manager throws a SecurityException. This exception propagates back to the Java program.

One limitation of the security manager using exceptions is that
a hostile applet or other untrusted code may catch a security exception by using a try-finally block.

Security and Type Safety

all of the sandbox mechanisms rely on

type safety.

ways that type errors can allow code to perform arbitrary actions.

One way for a C/C++ program to call an arbitrary function is

through a function pointer.

type confusion between object types would

allow a program to perform dangerous actions.

a buggy class loader and linker.
made it possible to successfully load another class with a different interface and to use the first class name for objects of the second class.