

Introduction to Formal Methods

Academic year 2025/2026

Emilio Tuosto

emilio.tuosto@gssi.it

<https://cs.gssi.it/emilio.tuosto>

A couple of reasons to be rigorous

Exercise -1: Describe the behaviour of the program below

```
print("Start")
a = 0.0
while (a != 100.0):
    a += 0.1
    print("moving on")
# ...
```

A couple of reasons to be rigorous

[<https://www.omg.org/spec/BPMN/2.0/>]

A converging **Inclusive Gateway** is used to merge a combination of alternative and parallel paths. A control flow *token* arriving at an **Inclusive Gateway** MAY be synchronized with some other *tokens* that arrive later at this **Gateway**. The precise synchronization behavior of the **Inclusive Gateway** can be found on page 292.

292

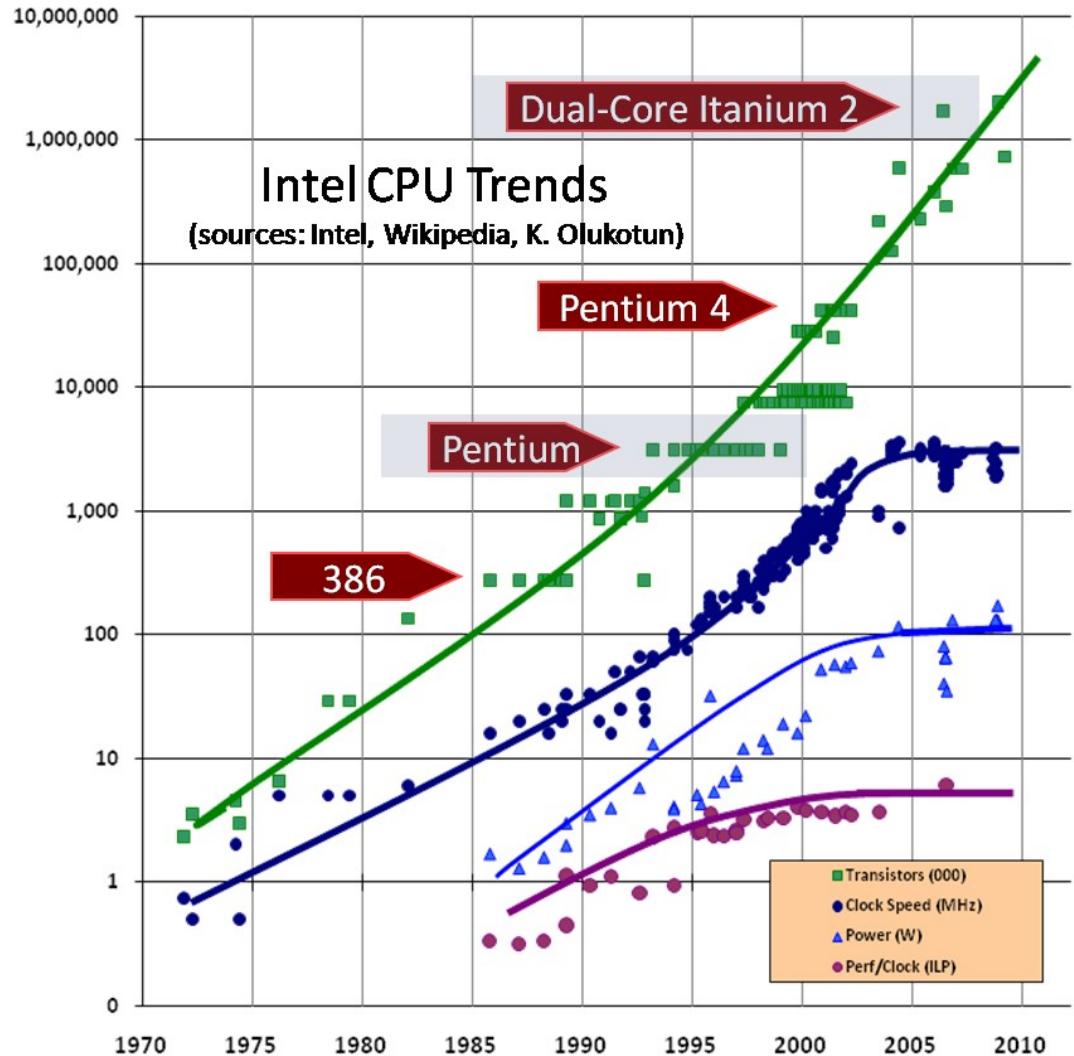
Business Process Model and Notation, v2.0

[<https://stackoverflow.com/questions/1812990/incrementing-in-c-when-to-use-x-or-x>]

The screenshot shows a Stack Overflow question page. The left sidebar has navigation links: Home, PUBLIC, Questions (which is selected), Tags, Users, Companies, and COLLECTIVES. The main content area displays a question titled "Incrementing in C++ - When to use x++ or ++x?". Below the title, it says "Asked 12 years, 11 months ago" and "Viewed 251k times". The question text is: "I'm currently learning C++ and I've learned about the incrementation a while ago. I know that you can use "+x" to make the incrementation before and "x++" to do it after." The question has 118 upvotes. A comment below the question asks, "Still, I really don't know when to use either of the two... I've never really used "+x" and things always worked fine so far - so, when should I use it?" To the right of the main content, there's a sidebar with "The Overflow" section containing links to "Making new da sponso" and "Stop re test: Mi".

A reason to go concurrent

[<https://i.extremetech.com/imagery/content-types/03zc6ghfKswe41smvPXi8Zh/images-6.jpg>]



[<https://www.comsol.com/blogs/havent-cpu-clock-speeds-increased-last-years>]

The screenshot shows a blog post on the COMSOL website. The header features a blue banner with icons for products, videos, events, blog, and learning center. The main content area has a dark blue background with white and orange icons representing people, microphones, and arrows.

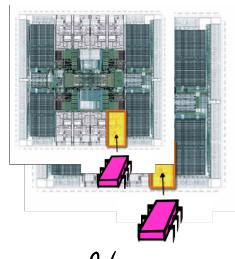
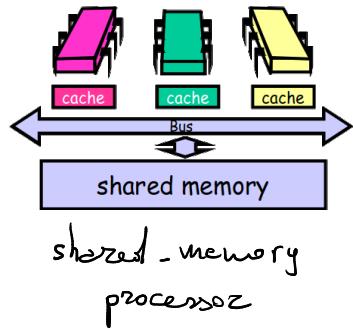
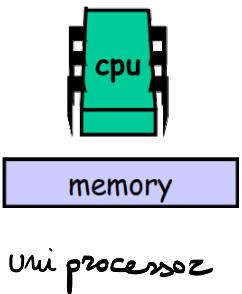
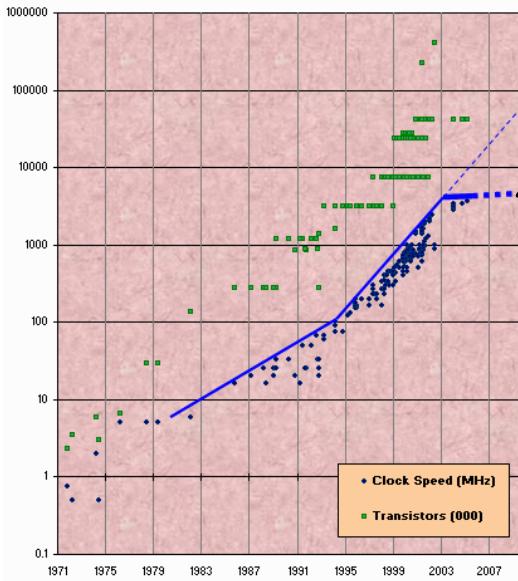
COMSOL Blog
Why Haven't CPU Clock Speeds Increased in the Last Few Years?

by Pär Person Mattsson | November 13, 2014

The first computer I used was a real performance beast. Equipped with Intel's 486 clocking in at

Categories: COMSOL Now
Get New Posts by Email
Leave a Comment

the art - multi-processor programming



Hw → Efficiency is no longer in "hw thing" → Sw

▷ programming constructs everywhere

- "new" languages
 - Clojure
 - Scala
 - Erlang / Elixir
 - Ballerina
 - Concurrents

▷ libraries (Akka)

▷ Modelling languages

- BPEL
- BPMN

Job interviews and prime numbers

print

"On the first day of your new job, your boss asks you to ~~find~~ all primes between 1 and 10^{10} (never mind why), using a parallel machine that supports ten concurrent threads. This machine is rented by the minute, so the longer your program takes, the more it costs. You want to make a good impression. What do you do?"

[Herlihy, Shavit: The Art of Multiprocessor Programming. Elsevier, 2012.]

An example of shared memory concurrency

Printing all the prime numbers below 10^{10} ... sequentially

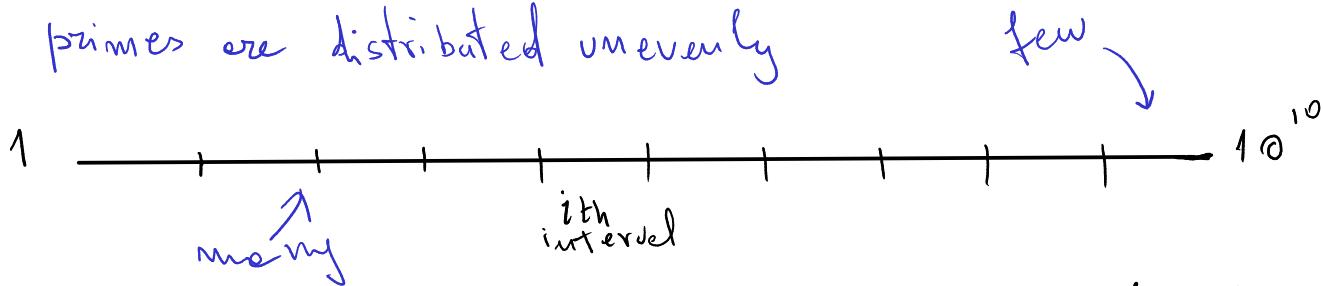
```
1 void primeSeq {  
2     for (j = 1, j<10^10; j++) {  
3         if (isPrime(j))  
4             print(j);  
5     }  
6 }
```

Now let's try concurrently



Split the interval & launch a thread on each position

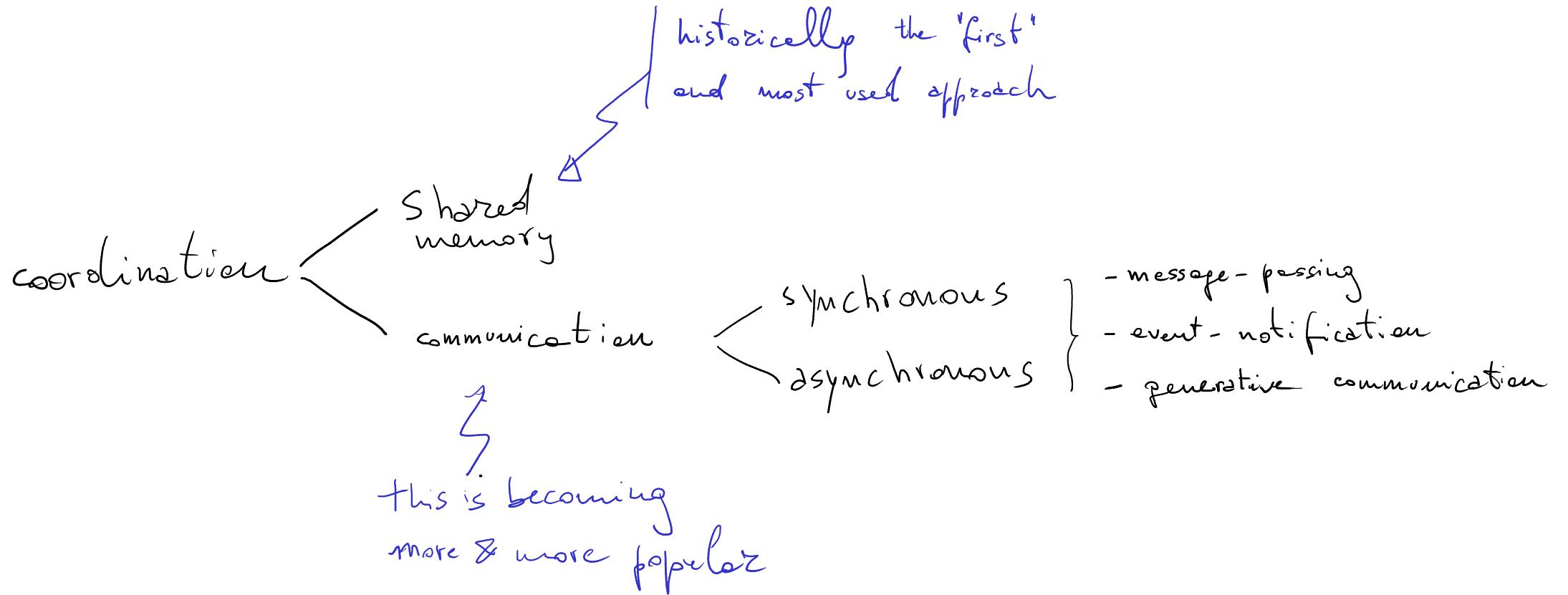
primes are distributed unevenly



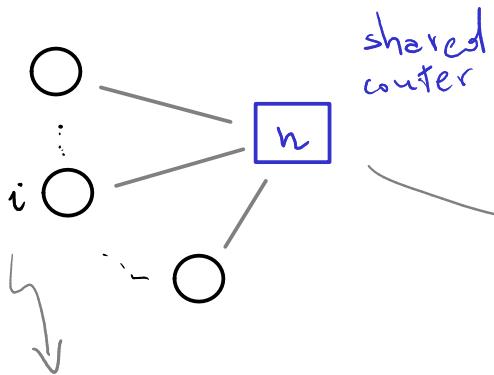
```
void primePrint(int i){ // i non-negative  
    for (j = i*10^9+1, j<(i+1)*10^9; j++) {  
        if (isPrime(j))  
            print(j);  
    }  
}
```

How good is this idea?

- o uneven load
- o should we look for an optimal split?



Exercise 0 Find a better multi-threaded program for printing the first 10^{10} primes



```
void primePrint( Counter counter ) {  
    long j = 0;  
    while (j < 10^10) {  
        j = counter.getAndIncrement();  
        if (isPrime(j))  
            print(j);  
    }  
}
```

```
public class Counter {  
    private long value;  
  
    public long getAndIncrement() {  
        return value++;  
    }  
}
```

THIS IS NOT
GOOD!

RACES

synchronised!

temp = value
value ++
return temp

```
public long getAndIncrement() {  
    synchronized {  
        temp = value;  
        value = temp + 1;  
    }  
    return temp;  
}
```

even better
WHY?

REFLECT about why this solution is better than splitting

Some terminology

DESIGN!

Concurrency

vs

Parallelism

compose "independent" stuff

deal with ~ a lot of stuff
AT ONCE

GOAL: "good" composition

break down problems
&

compose the solutions

run stuff simultaneously

do a ~ lot of stuff
AT ONCE

GOAL: "good" execution

PERFORMANCE

A refresher on minimal maths

$$A \subseteq B \quad \forall x \in A. \quad x \in B$$

$$A \subset B \quad A \subseteq B \wedge B \not\subseteq A$$

$$A = B \quad A \subseteq B \wedge B \subseteq A$$

$$A \cup B = \{x : x \in A \text{ or } x \in B\}$$

$$A \cap B = \{x : x \in A \text{ and } x \in B\}$$

$$A \setminus B = \{x : x \in A \text{ and } x \notin B\}$$

$$2^A = \{X \mid X \subseteq A\}$$

$$\bigcup_{i \in I} A_i$$

$$\bigcap_{i \in I} A_i$$

Relations

$$A \times B = \{(a,b) : a \in A \text{ and } b \in B\}$$

$$Id_A = \{(a,a) \mid a \in A\}$$

Let $R \subseteq A \times B$ (i.e., R is a relation with domain A and codomain B)

- inverse $R^{-1} = \{(b,a) \in B \times A \mid (a,b) \in R\}$

- composition $S \subseteq B \times C \Rightarrow R \circ S = \{(a,c) \in A \times C \mid \exists b \in B : (a,b) \in R \wedge (b,c) \in S\}$

If $A = B$

$$R^\circ = Id_A \quad R^{M+1} = R \circ R^M$$

$$a R b \Rightarrow R(a,b) \equiv (a,b) \in R$$

$$\prod_{i \in I} A_i$$

A refresher on induction

The induction principle is very useful, as you all probably know. Let's refresh it.

Proof method

To show that a property, say P , holds of every natural number n (i.e., to prove $P(n)$ for all n) it suffices to show that

- $P(0)$ is true &
- for all k , $P(k)$ implies $P(k+1)$

Example: for all n , $\text{sum}(n) = n(n+1)/2$ where $\text{sum}(k) = 1 + \dots + k$

- $\text{sum}(0) = 0 = 0(0+1)/2$
- for all k , if $\text{sum}(k) = k(k+1)/2$ then

$$\begin{aligned} \text{sum}(k+1) &= \text{sum}(k) + (k+1) && \text{by definition} \\ &= k(k+1)/2 + (k+1) && \text{by inductive hypothesis} \\ &= (k(k+1) + 2(k+1)) / 2 && \text{by arithmetic laws} \\ &= (k+1)(k+2)/2 && \text{by distributivity of multiplication over sum on natural numbers} \end{aligned}$$

Definitional mechanism

To define a set S inductively using a finite number of constructors f_1, \dots, f_n each with a finite arity on a set of 'basic elements'

- fix a set I of basic elements (you can think of the elements in I as constructors of arity 0) basis
- if e_1, \dots, e_k are in S and f is a constructor of arity k then $f(e_1, \dots, e_k)$ is an element of S induction
- nothing else can be an element of S closure

Example: $I = \{0\}$ and $s(_)$ is a constructor of arity 1, then the inductively defined set $S = \{0, s(0), s(s(0)), \dots\}$ is

isomorphic to natural numbers

(Indeed basis / induction / and closure boil down to the axioms of Peano).