Capirsi per collaborare: i tipi come metafora della comunicazione

Tre mattine all'Università

a cura della Commissione Orientamento di Informatica

Viviana Bono

“l’area grigia dell’informatica”

Cosa serve per creare le cose belle che avete visto?

... e i linguaggi di programmazione!

Perché scrivere un programma?

Per risolvere un problema!

1° passo: analizzare il problema e identificare una procedura di soluzione = ***algoritmo***

2° passo: algoritmo → ***programma*** (programmazione in-the-small).

Problema complesso (es: gestione automatizzata di una segreteria studenti) → ***sistema software*** = insieme di componenti organizzate in un architettura, programmazione in-the large)

linguaggio

- *alfabeto*: insieme di simboli con cui si possono costruire i termini del linguaggio (*lessico*)
- *sintassi*: definita da una *grammatica* che fornisce le regole di composizione dei termini in *frasi ben formate* del linguaggio
- *semantica*: definisce il significato delle frasi ben formate del linguaggio

Analizzatore sintattico (*parser*):

analizza frasi e decide se sono frasi ben formate del linguaggio o no

Una grammatica (o sintassi)

Regole

Frase → ***ParteNominale*** ***ParteVerbale*** .
ParteNominale → ***Nome*** ***Relativa***$_{opt}$
ParteVerbale → ***VerboIntransitivo*** |
VerboTransitivo ***ParteNominale***
Nome → ***NomeProprio*** | ***Articolo*** ***NomeComune***
Relativa → **che** ***ParteVerbale***
NomeProprio → **Mario** | **Lucia**
NomeComune → **cane** | **gatto**
Articolo → **il** | **un**
VerboIntransitivo → **corre** | **scappa**
VerboTransitivo → **insegue** | **raggiunge**

Nota: opt = opzionale (cioè che può esserci oppure no)

Esempi di frasi corrette secondo la sintassi precedente

Mario corre.

Il cane scappa.

Il cane insegue il gatto.

Mario insegue Mario.

Lucia insegue Mario che scappa.

Lucia che insegue Mario raggiunge il cane.

Mario insegue il cane che insegue un gatto che insegue Lucia.

Il gatto raggiunge il gatto che raggiunge il gatto che raggiunge il gatto che corre.

Un cane raggiunge un gatto che insegue il cane.

Mario che insegue Mario raggiunge Lucia che insegue Lucia.

...

Esempi di frasi scorrette secondo la stessa sintassi

Il Mario corre.
Mario corre *(manca il punto finale!)*
Cane insegue cane.
Il gatto insegue.
Un cane raggiunge un Mario.
Il gatto insegue il topo.
Corre il cane.
Il cane che scappa.
Che cane.
...

linguaggio di programmazione

- *frasi ben formate* = programmi
- programma = insieme di “istruzioni”
- *semantica* = esecuzione del programma

linguaggio di programmazione

- Per esempio Java, C, C++, BASIC, Pascal, JavaScript, Ruby, PHP, …
- Proprietà:
  - Non ambiguo
  - Preciso
  - Conciso
  - Espressivo
  - Alto livello (molte astrazioni)

Un programma

```
public class HelloJava {
   public static void main(String[] args) {
      System.out.println(“Hello, Java!");
   }
}
```

Interpretazione e compilazione

$\Rightarrow$ occorre tradurre il programma (scritto in un linguaggio ad alto livello) in istruzioni in linguaggio macchina

Due tecniche per effettuare questa traduzione:

- Compilazione (“traduco tutto poi eseguo”)
- Interpretazione (“traduco ed eseguo istruzione per istruzione”)

Interpretazione vs compilazione

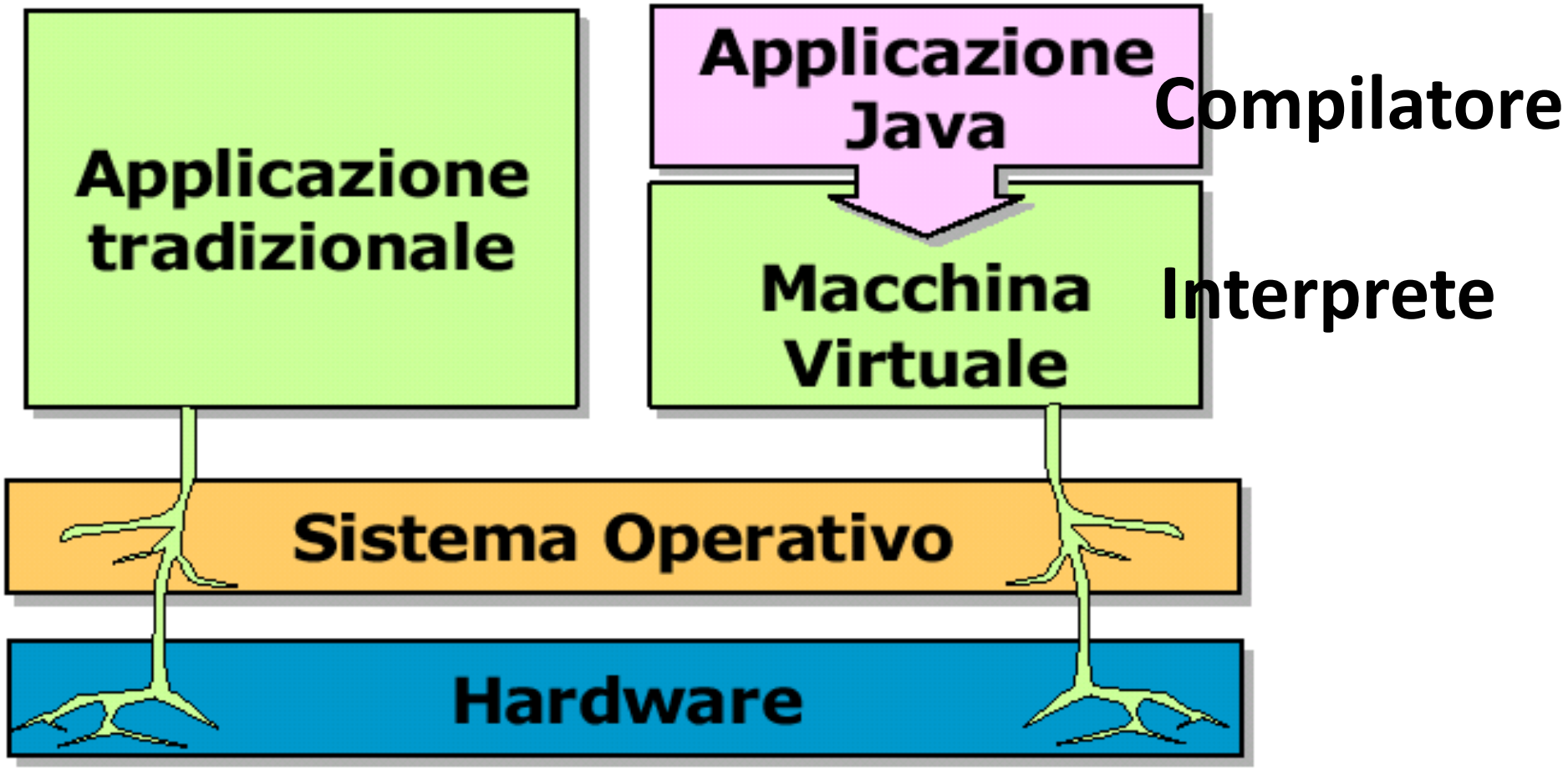
Intepretazione

- Inefficiente
- “Portabile”

Compilazione

- Efficiente
- “Poco portabile”

linguaggi semi-compilati (semi-interpretati): Java (e C#)

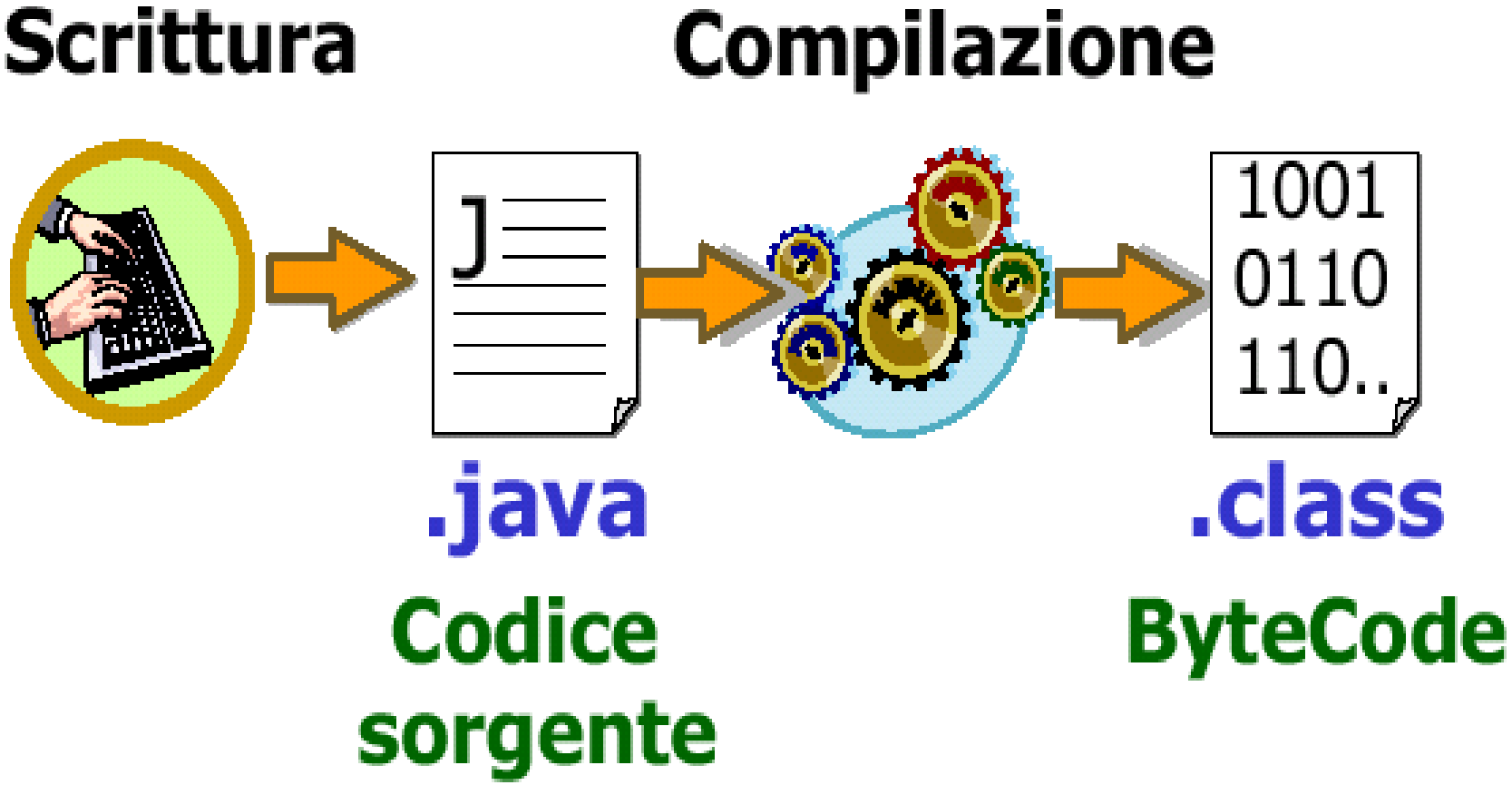


Il programma di prima...

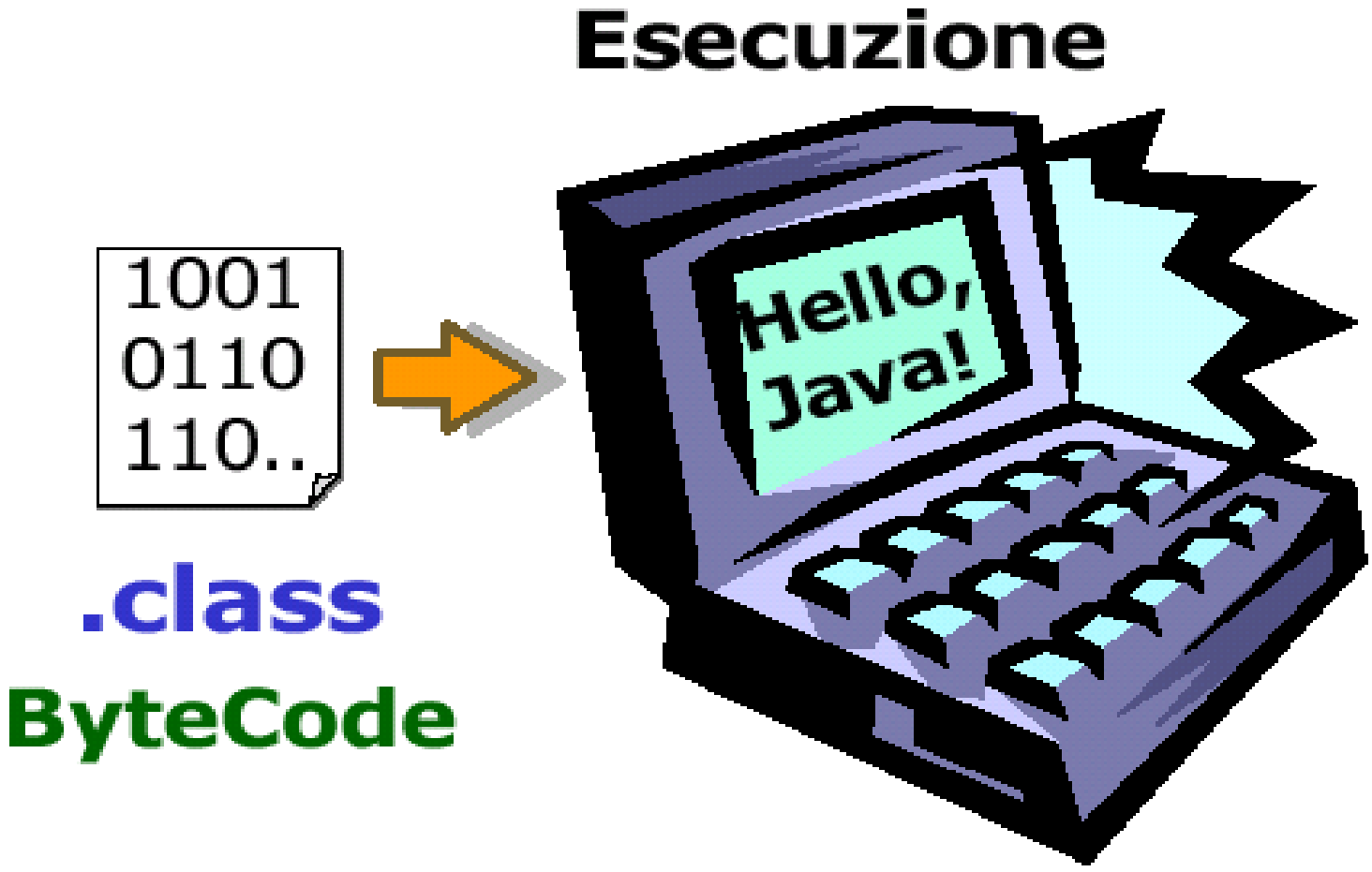
```
public class HelloJava {
   public static void main(String[] args) {
      System.out.println(“Hello, Java!");
   }
}
```

Sta in un file: `HelloJava.java`

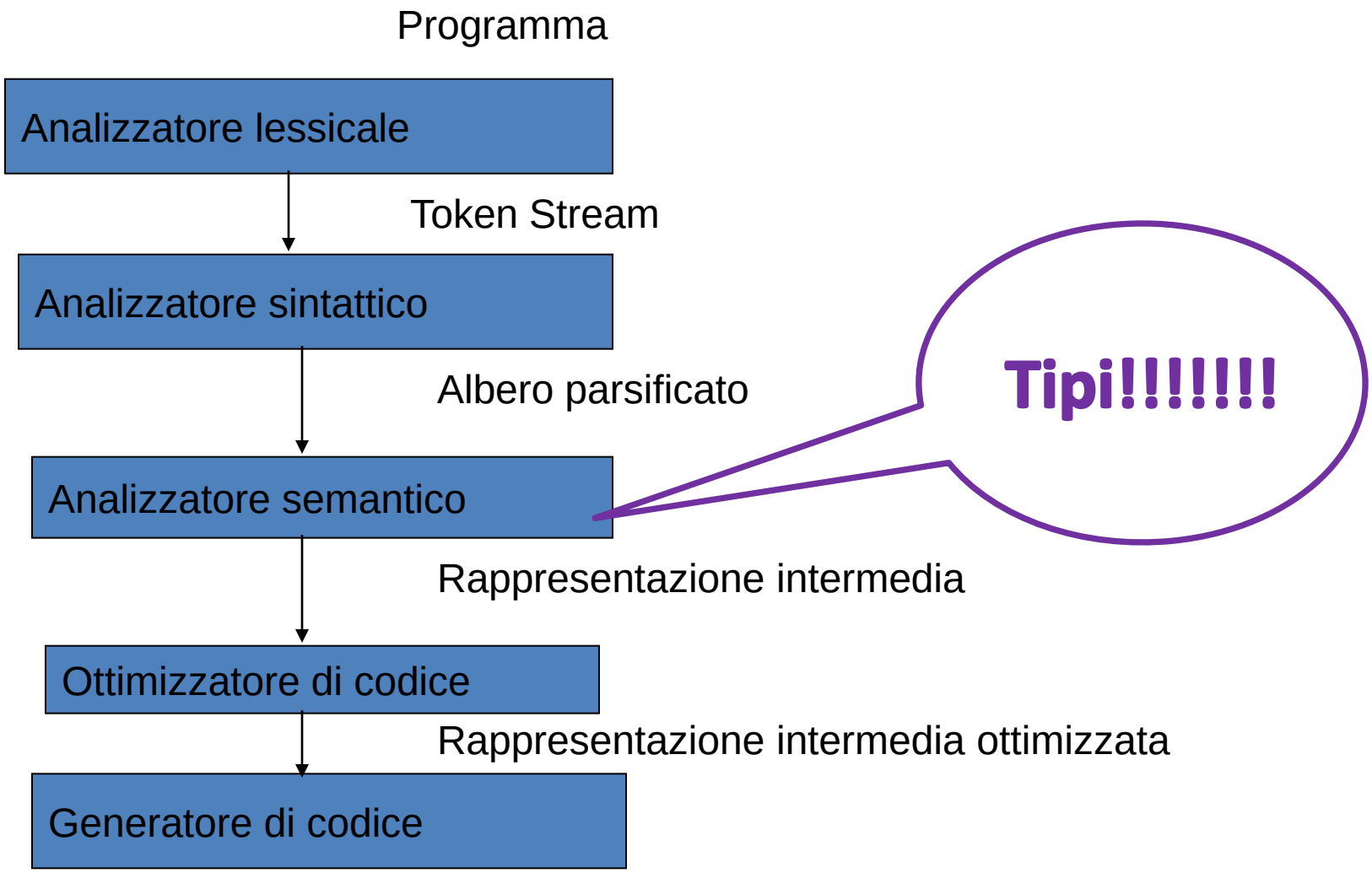
Compilo...


Scrittura
Compilazione
J
1001
0110
110..
.java
Codice
sorgente
.class
ByteCode

Eseguo...



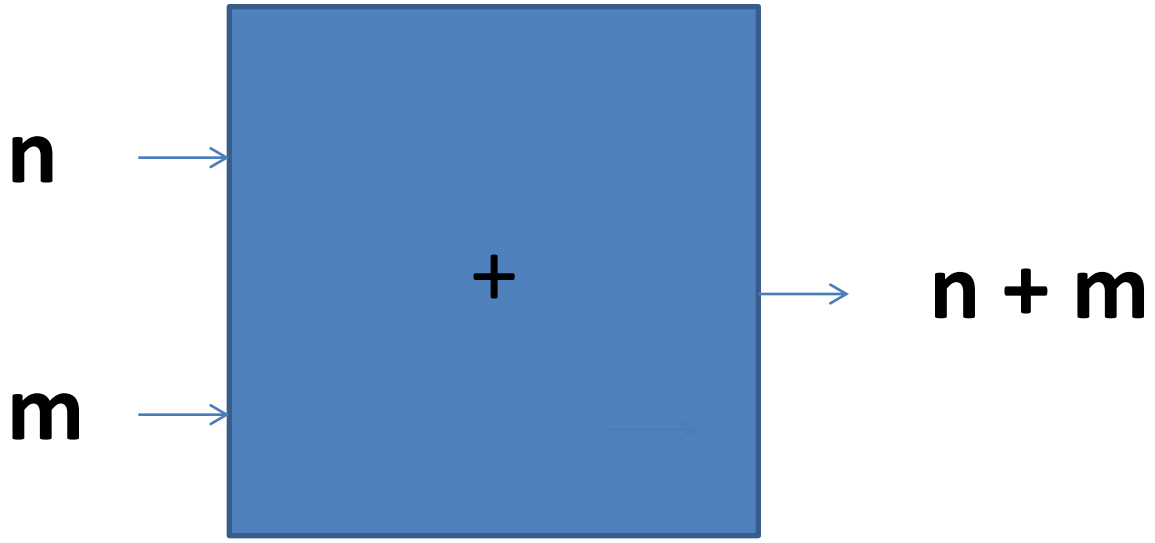
Anatomia di un compilatore



Codice macchina (**non** per Java: bytecode)

I tipi

I tipi sono un concetto semplice…



m e **n** possono essere numeri [2 + 3], variabili [x + y], espressioni [(x+2) + (y+4)]

Un programma

```
  public static void main(String[] args) {
    int x; int y;
    // int ~ naturali con segno

    x = 2; y = 4
    int z = x + y;
  }
}
```

dichiarazioni

Un altro programma

```
  public static void main(String[]
args) {
   String x; String y;
   // String ~ sequenze di caratteri

   x = "ciao"; y = "a tutti";
   String z = x + y;
  }
}
```

dichiarazioni

m + n (nel nostro caso: x + y)

x = 2, y = 4

→ 6

x = “ciao”, y = “a tutti”

→ “ciao a tutti”

m + n (nel nostro caso: x+y)

x = 2, y = “ciao”

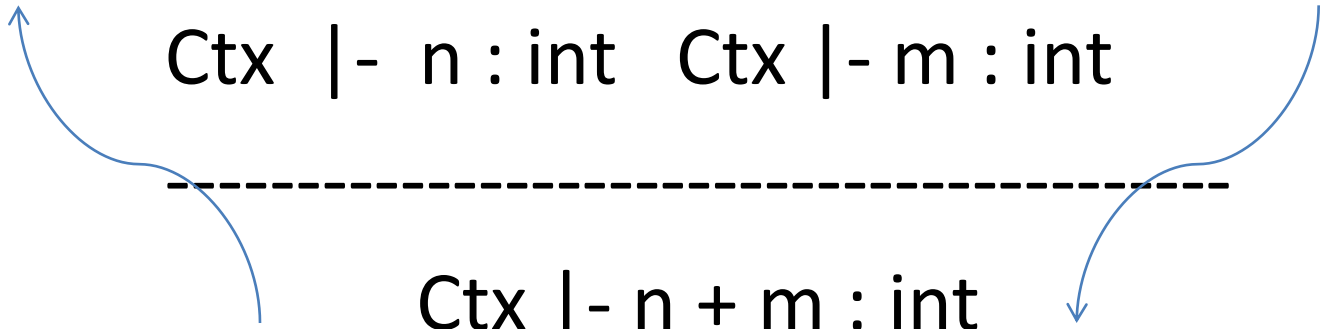
→

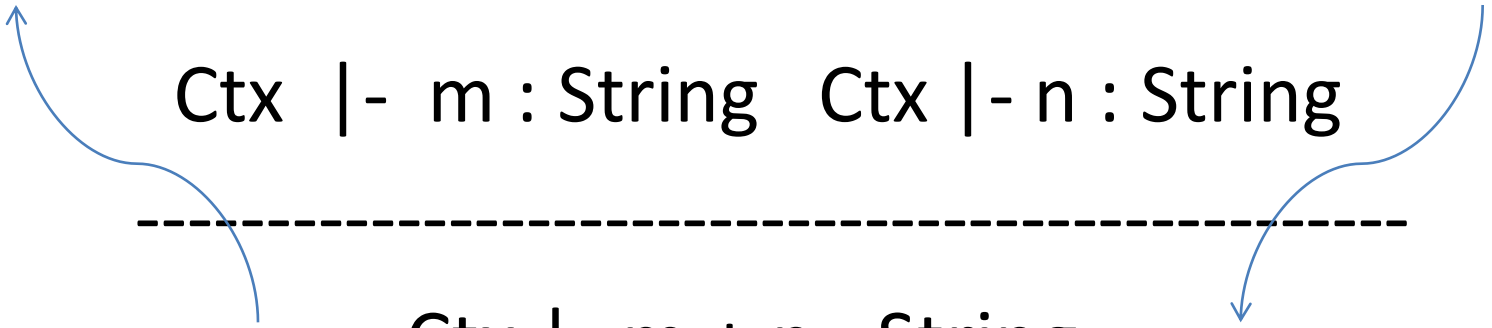
?

Regole di tipo (I)

dichiarazioni

Ctx, x : Tipo |- x :Tipo





Regole di tipo (II)

Ctx |- m : int Ctx|- n : String

|- m + n : ?????

Static typing vs. dynamic typing

Static*

- Più rigido
- Correttezza parziale
- Forma di docs automatica

→ più facile da mantenere

Dynamic**

- Più flessibile
- No garanzie di correttezza
- No docs diretta

→rapid prototyping

* Quello che abbiamo visto.
** es. JavaScript.

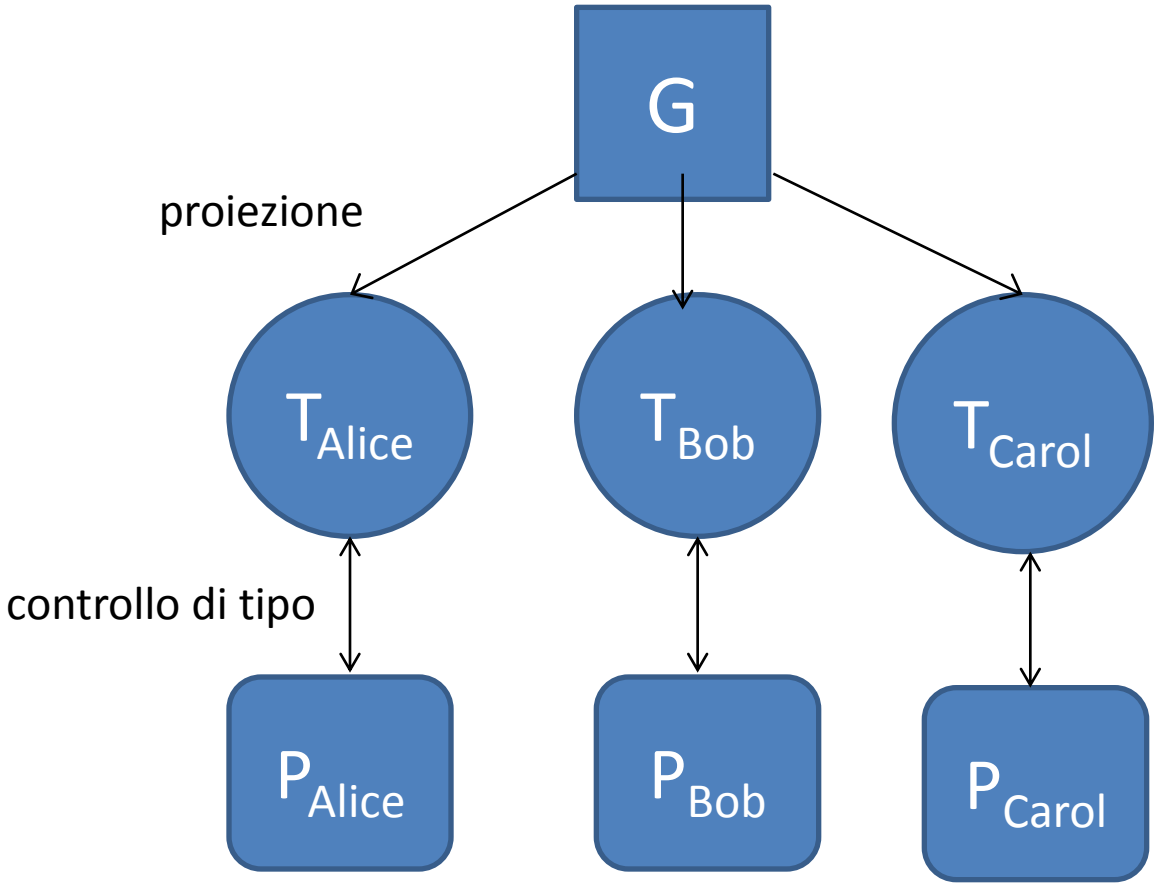
… ma Internet (& co) dov'è?

Programma in esecuzione = processo

Tanti processi in esecuzione che devono comunicare

Protocolli di comunicazione

Tipi comportamentali



Alice →(int) Bob
Bob→(int) Carol

TBob = Alice?int:
Carol!int:
end

PBob =
receive x from Alice;
send x+42 to Carol;
end