Programming Paradigms

Type Systems (Part 5)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Overview

- Introduction
- Types in Programming Languages
- Polymorphism
- Type Equivalence
- Type Compatibility 
- Formally Defined Type Systems

Type Compatibility

- **Check whether combining two values is valid according to their types**
- **"Combining" may mean**
  - Assignment: Are left-hand side and right-hand side compatible?
  - Operators: Are operands compatible with the operator and with each other?
  - Function calls: Are actual arguments and formal parameters compatible?

Compatible ≠ Equal

Most PLs: Types may be compatible

even when not the same

Example (C):

```
double d = 2.3;
float f = d * 2;
int i = f;
printf("%d\n", i);
```

Compatible ≠ Equal (2)

- **Rules of PL define which types are compatible**
- **Examples of rules**
  - Can assign subtype to supertype:

    ```
    lhs = rhs;
    ```

  - Different number types are compatible with each other
  - Collections of same type are compatible, even if length differs

Type Conversions

When types aren't equal, they must be converted

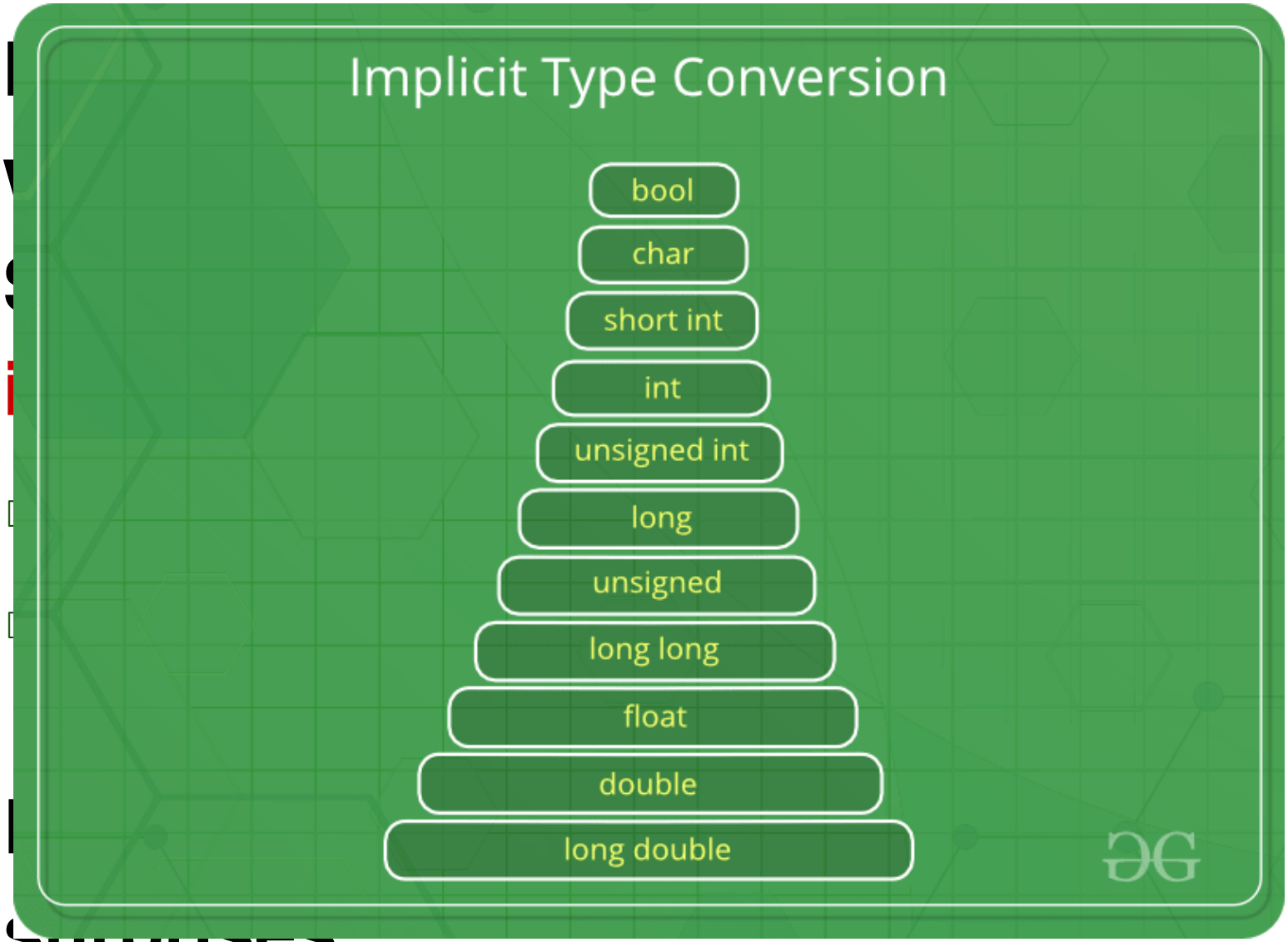
- Option 1: Cast = explicit type conversion
  - Programmer changes value's type from T1 to T2
- Option 2: Coercion = implicit type conversion
  - PL allows values of type T1 in situation where type T2 expected
- Both options: Actual conversion happens at runtime

Coercions in C

- **Most primitive types are coerced whenever needed**
- **Some coercions may loose information**
  - `float` to `int`: Loose fraction
  - `int` to `char`: Causes `char` to overflow (and will give unexpected result)
- **Enable compiler warnings to avoid surprises**

Coercions in C

- I
- S
- I



Surprises

Source: *geeksforgeeks.org*

Coercions in C: Demo

Demo: coercions.c

compile with gcc -Wconversion

Coercions in JavaScript

- **Almost all types are coerced when needed**
  - Rationale: Websites shouldn't crash
- **Some coercions make sense:**
  - `"number:" + 3` yields `"number:3"`
- **Many others are far from intuitive:**
  - `[1, 2] << "2"` yields `0`

More details and examples:
***The Good, the Bad, and the Ugly: An Empirical Study of Implicit Type Conversions in JavaScript*. Pradel and Sen. ECOOP 2015**