Ptolemy II Type System

Yuhong Xiong
Edward A. Lee

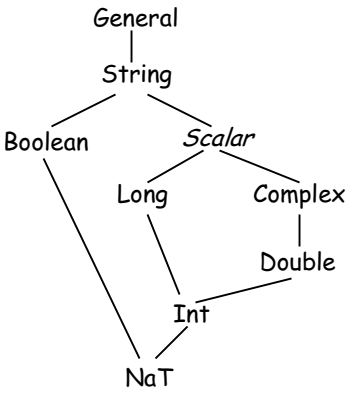
UC Berkeley

Objective

- Support polymorphic actors
- Increase safety
- Prevent information loss due to data type conversion during data transport

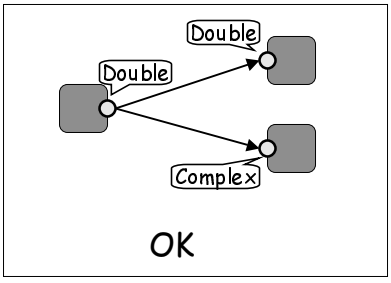
The Type Lattice

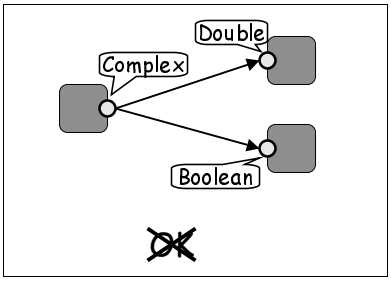
- Specifies the lossless type conversion relation
- A type can be **losslessly** converted to any type greater than it



Type Compatibility Rule

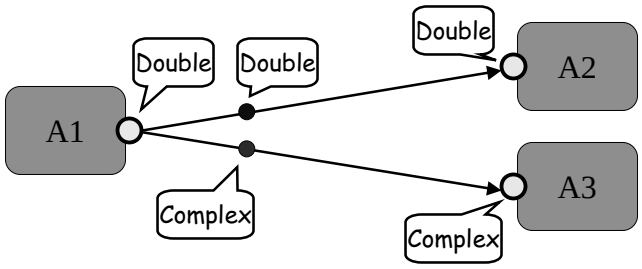
- The type of an output port must be less than or equal to the types of connected inputs.





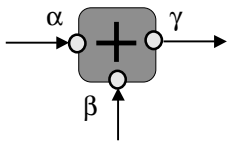
Run-time Type Conversion

- If a token sent to an input port is not an instance of the type of that port, but can be converted to that type, Ptolemy II does the conversion automatically.



Polymorphic Actors

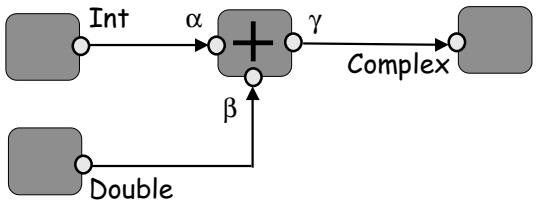
- Types are represented by type variables.
- Type constraints are described by inequalities over the type lattice



$\alpha \leq \gamma$
$\beta \leq \gamma$
$\gamma \leq$ Complex

Type Constraints

- Topology also enforces type constraints



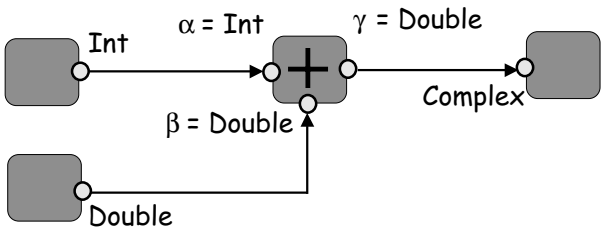
Int $\leq \alpha$
Double $\leq \beta$
$\gamma \leq$ Complex

$\alpha \leq \gamma$
$\beta \leq \gamma$
$\gamma \leq$ Complex

Type Resolution Algorithm

- General algorithm to solve constraints over lattice given by Rehof and Mogensen
  - Initialize all the type variables to NaT
  - Repeatedly update the variables to a greater type, until
  - All the constraints are satisfied, or determines that constraints not satisfiable
- Equivalent to searching for the least fixed point of a monotonic function
- Runs in linear time

Example

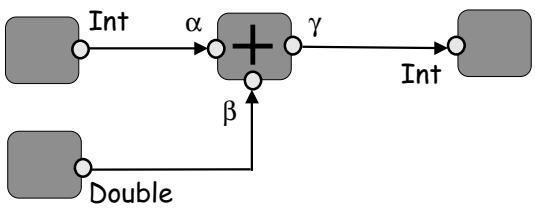


Int ≤ α
Double ≤ β
γ ≤ Complex

α ≤ γ
β ≤ γ
γ ≤ Complex

Type Conflict

- The set of type constraints are not satisfiable
- Some type variables are resolved to NaT
- Some type variables are resolved to an abstract type



Conclusion

- Static typing combined with run-time type checking prevent information loss during data transfer
- Run-time type conversion helps simplify actor development
- Type constraints and type resolution algorithm support polymorphic actors