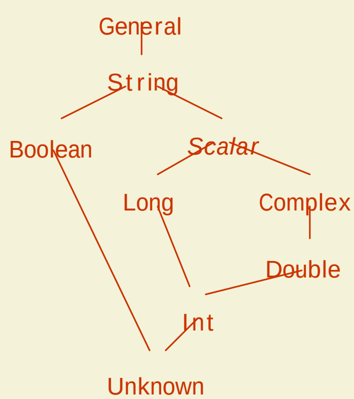
Extensions to the Ptolemy II Type System

Yuhong Xiong
Edward A. Lee

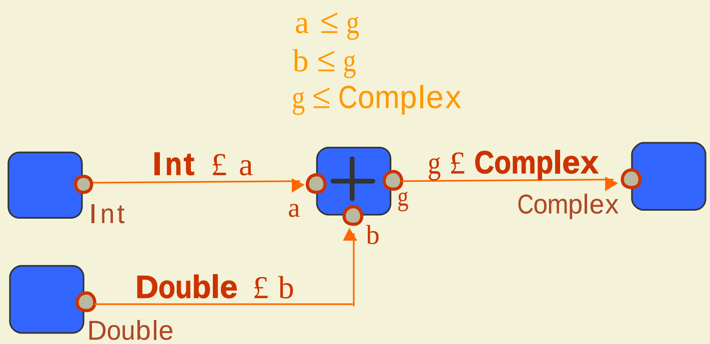
Ptolemy Miniconference
Berkeley, CA, March 22-23, 2001

Type Lattice



- Specifies the lossless type conversion or subtype relations

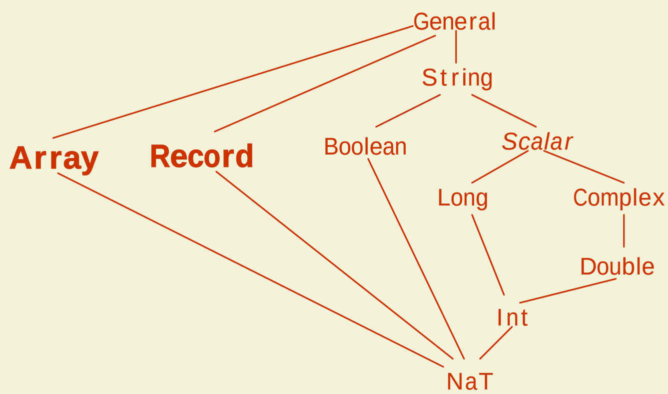
Type Constraints



Extensions

- Structured types
- Using monotonic functions in type constraints

Structured Types



Goals and Questions

- Arbitrary element types. E.g. (int)array, ((int)array)array, array of records, records containing arrays, …
- Type constraints between element types and the types of other objects in system
- Order relation among structured types?
- Structured types admitted by the inequality solving algorithm?
- Convergence on infinite lattice?

Order Relation

- Array: order by element type:
  - (Int)array £ (Double)array
- Record: subtype

{item: String, val: Double}

{item: String, val: Int}   {item: String, val: Double, id: Int}

Inequality Solving Algorithm

- Admits (Rehof and Mogensen '96):

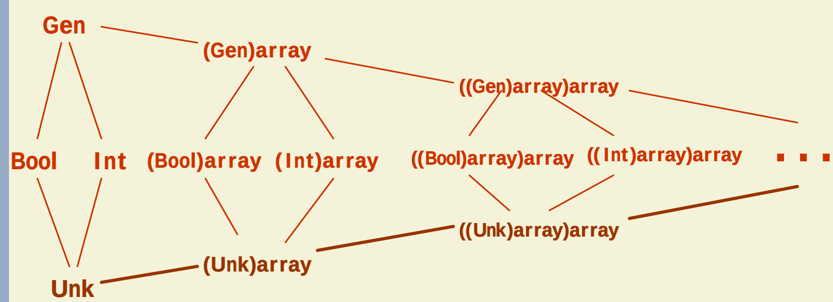
const
α   £   const
f(α)   α

- Extension:

(Int)array £ (α)array

  - know the "structure" of right-hand-side, so can update the variable to make right-hand-side the least upper bound of both side

Convergence on Infinite Lattice



"Bad" constraint:

$(\alpha)\mathrm{array} \le \alpha$

$(\mathrm{unk})\mathrm{array} \le \mathrm{unk}$

$((\mathrm{unk})\mathrm{array})\mathrm{array} \le (\mathrm{unk})\mathrm{array}$

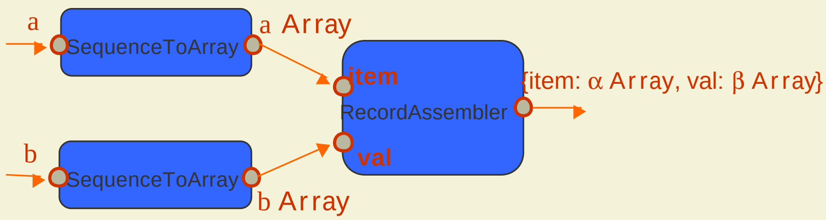
…

Convergence on Infinite Lattice (cont.)

- **All chains from any constant type to the top are finite**
- **Infinite updates with unknown types can be detected by setting a bound on the depth of structured types**

Actors Manipulating Structured Types

- SequenceToArray
- ArrayToSequence
- ArrayAppend
- ArrayElement
- ArrayExtract
- ArrayLength
- RecordAssembler
- RecordDisassembler
- RecordUpdater



Using Monotonic Functions in Type Constraints

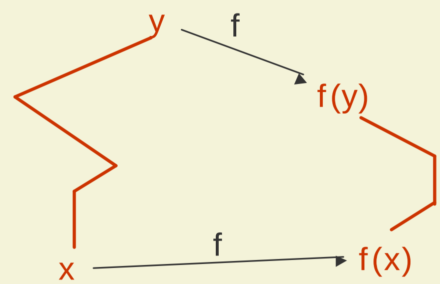
- Inequality solving algorithm admits:

const α f(α) £ const α

- Inequalities **f(**a**)** £ b can express complex type constraints

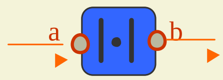
Monotonic Function

- A function $f: T \rightarrow T$ is monotonic if
  $x \le y$ implies $f(x) \le f(y)$



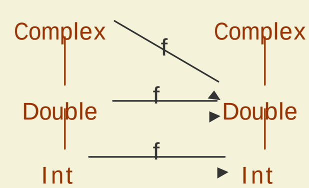
AbsoluteValue Actor

- **Requirements**
  - Works for Int, Long, Fix, Double, Complex
  - Output type is the same as the input, unless input is Complex
  - Output type is Double when input is Complex

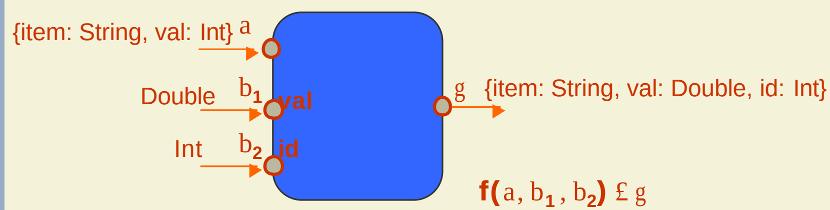


$f(a) \le b$, where

$f(a)$ = Double if a = Complex
= a otherwise



RecordUpdater Actor



a	b_1	b_2	$f(a, b_1, b_2)$
unknown	unknown	unknown	unknown
unknown	Double	Int	unknown
{item: String, val: Int}	Double	Int	{item: String, val: Double, id: Int}

Conclusion

- Structured types increases the expressiveness of the modeling environment
- Monotonic function can describe complex type constraints