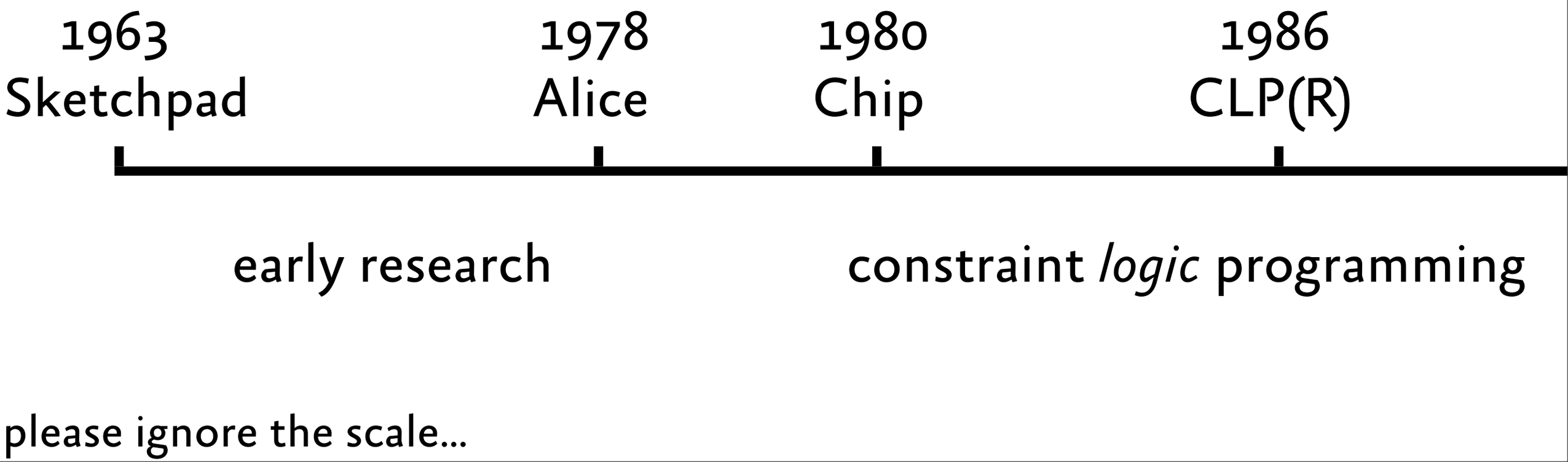
Constraint Programming

Marco Kuhlmann & Guido Tack

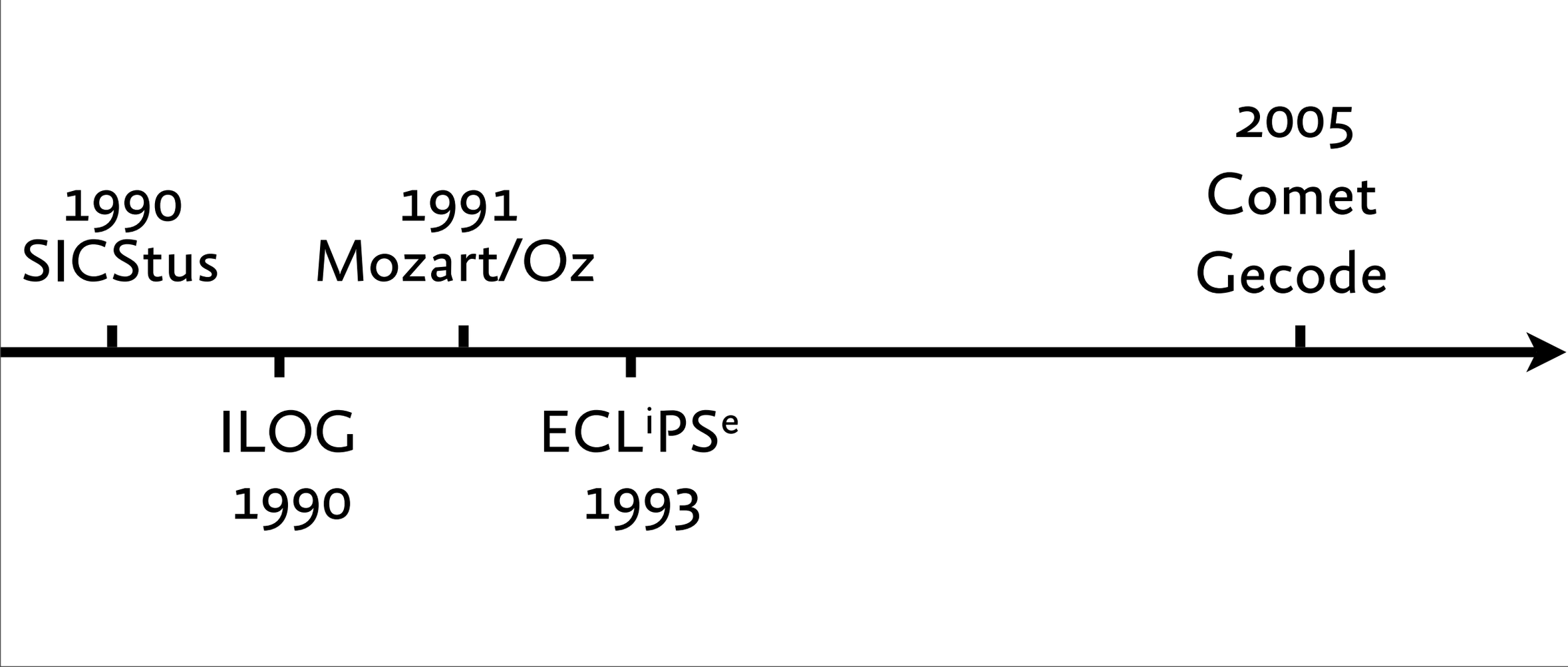
Lecture 2

Historical notes

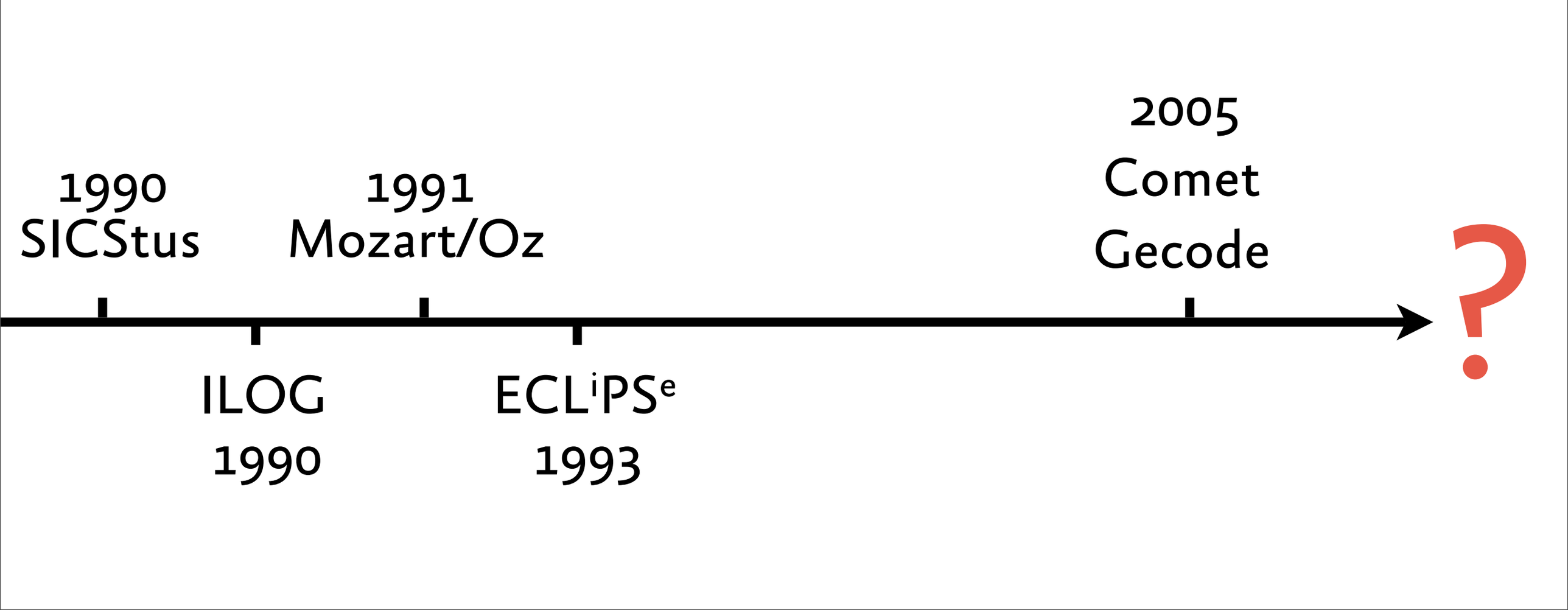


please ignore the scale...

Historical notes



Historical notes



Logic Programming

```
foo(a).
foo(b).
g(X) :- X=[Y, Z], foo(Y), foo(Z).
```

Logic Programming

```
foo(a).
foo(b).
g(X) :- X=[Y, Z], foo(Y), foo(Z).


| ?- g(X).
X = [a,a] ? ;
X = [a,b] ? ;
X = [b,a] ? ;
X = [b,b]
```

Constraint Logic Programming

```
foo(X) :- fd_domain(X, 1, 3).
g(Y,Z) :- foo(Y), foo(Z), Y #< Z,
          fd_labeling([Y,Z]).
```

Constraint Logic Programming

```
foo(X) :- fd_domain(X, 1, 3).
g(Y,Z) :- foo(Y), foo(Z), Y #< Z,
          fd_labeling([Y,Z]).
| ?- g(X,Y).
X = 1
Y = 2 ? ;
X = 1
Y = 3 ? ;
X = 2
Y = 3
```

Constraint Logic Programming

```
foo(X) :- fd_domain(X, 1, 3).
g(Y,Z) :- foo(Y), foo(Z), Y #< Z,
          fd_labeling([Y,Z]).
| ?- g(X,Y).
X = 1
Y = 2 ? ;
X = 1
Y = 3 ? ;
X = 2
Y = 3
```

Model:

constraint program

=

logic program

=

logical formula

Constraint Logic Programming

```
foo(X) :- fd_domain(X, 1, 3).
g(Y,Z) :- foo(Y), foo(Z), Y #< Z,
          fd_labeling([Y,Z]).
| ?- g(X,Y).
X = 1
Y = 2 ? ;
X = 1
Y = 3 ? ;
X = 2
Y = 3
```

Languages/Systems:

GNU Prolog, BProlog, SICStus Prolog, ECL^{i}PS^{e}

Concurrent Constraint Programming

Concurrent Constraint Programming

- Von-Neumann architecture: store *values*
  - operations: `read` and `write`

Concurrent Constraint Programming

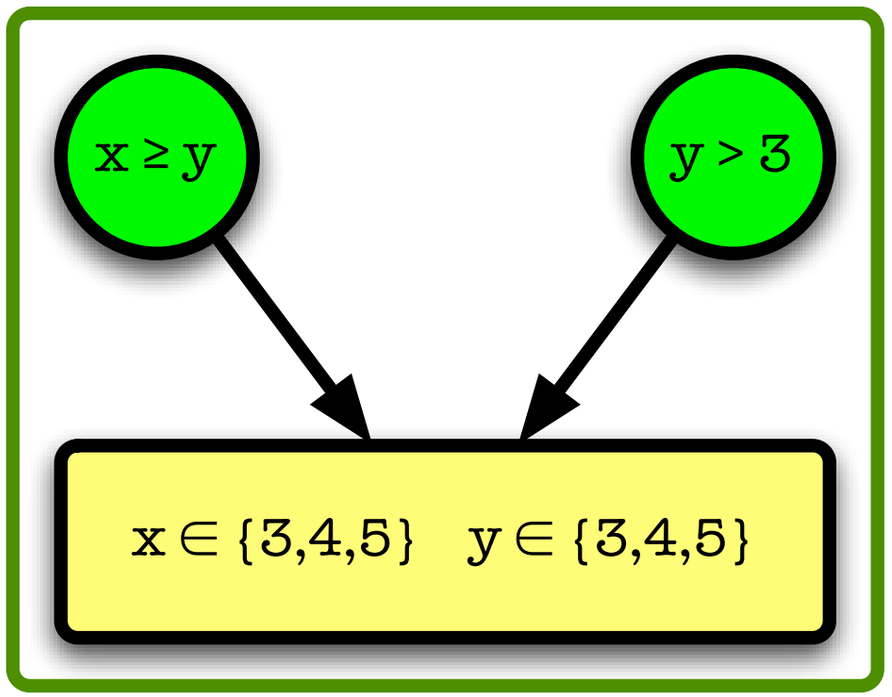
- Von-Neumann architecture: store *values*
  - operations: `read` and `write`
- cc architecture: store *constraints*
  - operations: `ask` and `tell`
  - communication through variables

Concurrent Constraint Programming

- Von-Neumann architecture: store *values*
  - operations: `read` and `write`
- cc architecture: store *constraints*
  - operations: `ask` and `tell`
  - communication through variables
- Languages/systems:
  - cc(FD), AKL, Mozart/Oz

Concurrent Constraint Programming

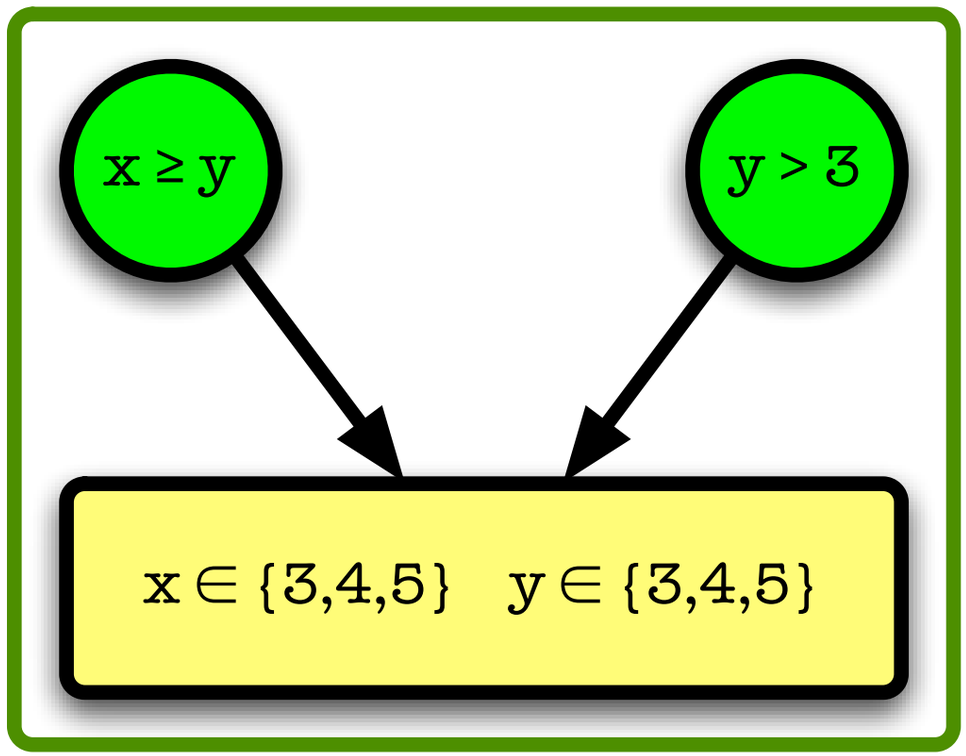
- Von-Neumann architecture: store *values*
  - operations: `read` and `write`
- cc architecture: store *constraints*
  - operations: `ask` and `tell`
  - communication through variables
- Languages/systems:
  - cc(FD), AKL, Mozart/Oz



Constraint Programming with Gecode/J

- **Quick reminder of last lecture**
- **Walk-through for *Send More Money***
- **Some modeling techniques**
- **Presentation of first graded lab**

Computation Space



constraint store with connected propagators