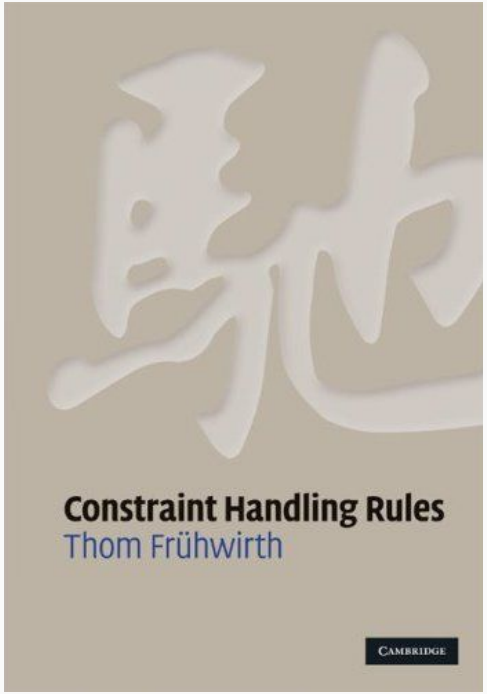
# From Constraint Handling Rules to Prolog

Alexander Grooff & Arnold Overwater

# Constraint Handling Rules

- Designed by Thom Frühwirth
- Created for constraint solving



# Constraint Handling Rules - Syntax

```
Rule = Rulename? {Constraint “,”}+ RuleType Guard? {Constraint “,”}+
```

```
X < Y → Y < Z | X < Z
```

# Constraint Handling Rules - Rules

Two kinds of rules:

- Simplification: $X \leq Y \land Y \leq X \Leftrightarrow X = Y$
- Propagation: $X \leq Y \land Y \leq Z \Rightarrow X \leq Z$

# Constraint Handling Rules - Example

```
upto(N),fib(A,AV),fib(B,BV) ⇒
    B === A+1,B<N | fib(B+1,AV+BV)
```

Constraint Store

| ID | Constraint |
|---|---|
| 1 | `fib(1,1)` |
| 2 | `fib(2,1)` |
| 3 | `upto(10)` |
| 4 | `fib(3,2)` |
| 5 | `fib(4,3)` |
| 6 | `fib(5,5)` |
| 7 | `fib(6,8)` |
| 8 | `fib(7,13)` |
| 9 | `fib(8,21)` |
| 10 | `fib(9,34)` |
| 11 | `fib(10,55)` |

# Switch to Prolog

- CHR based on Prolog
- Advised to switch to Prolog

# Prolog - Syntax

- Literals, Atoms, Variables and CompoundTerms
  - Atoms start with lowercase
  - Variables start with uppercase
- Based on rules, facts and queries
  - Rule: a(X) :- b(X).
  - Fact: a(X). This is equivalent to a(X) :- true.
  - Query: ?- a(1).

# Prolog - Example

```
reverse(List, Reversed) :-
        reverse(List, [], Reversed).

reverse([], Reversed, Reversed).
reverse([Head|Tail], SoFar, Reversed) :-
        reverse(Tail, [Head|SoFar], Reversed).

?- reverse([a, b, c, d], X).
```

# Prolog - Solving process

- Evaluate
  - Attempts to evaluate query
  - Tries to find a matching rule/fact
- Unification
  - Compares terms in head of rule vs query
- Substitution
  - Replace unbound variables with new value

# Prolog - Query variables

```
a(1).

?- a(X).
```

- Start as unbound variables
- If a match is found, return binding

# Prolog - Demo

# Questions