CHR (Constraint Handling Rules, Thom Frühwirth) is...
- very high level ➔ focus on what, not how (e.g. implicit fixpoint computation)
- multi-headed rule based ➔ very expressive; declarative reading
- a language extension (embedded in a host language) ➔ no interfacing problems
- used to write (application tailored) constraint solvers ➔ also a general-purpose language

→ Compact programs; easy to understand, modify & experiment with

Optimizing compilation: Guard Reasoning

Example CHR program: (simple integer interval solver)

```prolog
:- constraints in(?int,+interval).
:- chr_type interval --> int::int.

X in A:B <=/=> A>B | fail.
X in A:B <=/=> A>=:=B | X is A.

pragma Passive(C2).
```

Example query:

```prolog
?- V in 2:5, V in 3:7, V in 1:3

rule 3
V in 3:5 ,  V in 1:3

rule 3
V in 3:3

rule 2

V = 3
Yes
```

Compiled code:

```prolog
X in I :-
  `in/2::0'(X,I,).
  `in/2::0'(__,A:B,T) :-
    A>B, !,
    remove_code
    fail.
  `in/2::0'(X,A:B,T) :-
    A>=:=B, !,
    remove_code
    X is A.
  `in/2::0'(X,A:B,T) :-
    A<=:<B,
    find_partner(X in C:D)
    C<=:<D, !,
    remove_code
    X in max(A,C):min(B,D).
  `in/2::0'(X,C:D,T) :-
    C<=:<D,
    find_partner(X in A:B)
    A<=:<B, !
    remove_code
    X in max(A,C):min(B,D).
  `in/2::0'(X,I,T) :-
    insert_code.
```

Acknowledgements
This research was funded in part by the FWO project G.0144.03:
“Program Analysis for better memory management and higher performance of
HAL, a constraint logic programming language.”

Further information
Jon Sneyers
Division of Informatics (DTAI)
Celestijnenlaan 200A
3001 Leuven (Heverlee), Belgium
Jon.Sneyers@cs.kuleuven.ac.be

Supervisor: Bart Demoen
Tel.: +32 16 32 75 44
Fax: +32 16 32 79 96
www.cs.kuleuven.ac.be