The K.U.Leuven CHR System: Implementation and Application

Tom Schrijvers, Bart Demoen
{tom.schrijvers,bart.demoen}@cs.kuleuven.ac.be.

Katholieke Universiteit Leuven, Belgium
Overview of the Talk

1. Implementation
   ♦ 1.1 Setup and Evolution
   ♦ 1.2 Optimizations
   ♦ 1.3 Evaluation

2. Application
   ♦ 2.1 Well-Founded Semantics
   ♦ 2.2 Java Memory Model

3. Conclusion
   ♦ 3.1 Future Work
   ♦ 3.2 Discussion Topics
1. Implementation
1.1 Setup and Evolution

K.U.Leuven CHR system consists of two parts:

- the **runtime**
  - strongly based on Christian Holzbaur’s
  - finetuned for performance

- the preprocessing **compiler**
  - CHR $\rightarrow$ Prolog
1.1 Setup and Evolution

- **hProlog** (Bart Demoen)
  - evaluate new attributed variables
  - optimized compilation of CHR

- **XSB** (David S. Warren)
  - along with hProlog’s attributed variables
  - integration of CHR with tabling (ICLP’04)

- **SWI-Prolog** (Jan Wielemaker)
  - along with hProlog’s attributed variables
  - first CLP capabilities in SWI-Prolog
1.2 Optimizations

Based on other systems and own ideas:

- heuristic reordering of constraints in multi-headed rules
- early scheduling of guards
- detection of never attached constraints
  \[\rightarrow\] other constraints in same rule passive
  \[\rightarrow\] avoid space and time overhead
1.2 Optimizations

- functional dependencies inference
  $\implies$ once transformation

a(X,_) \ a(X,_) $\iff$ true.
b(X), ..., a(X,Y), ... $\implies$ ...

procedure_b(X) :-
  ...
  % retrieve other constraints
  once (retrieve_a(X,Constraint),
  ...
  % retrieve other constraints
  !,
  ...
  ...
1.2 Optimizations

- inference of unmatched arguments
  $$\implies$$ no attaching

entry(Key, Value) \ lookup(Key, Query)\n<=> Query = Value.

?- entry(123, [Var1, Var2, ..., Var10]),
  lookup(123, Query).
entry/2 is redundantly (de/at)tached
1.2 Optimizations

- inference of unmatched arguments conditions for argument in all rules:
  - ♦ argument is a variable
  - ♦ variable only appears once in heads and guard
  - ♦ variable may appear in (anti-monotonic) var test:

\[
\begin{align*}
\text{fib}(N,M1) \setminus \text{fib}(N,M2) & \iff \text{var}(M2) \mid M1 = M2. \\
\text{fib}(N,M) & \implies N \leq 1 \mid M = 1. \\
\text{fib}(N,M) & \implies N > 1 \mid N1 \text{ is } N-1, \text{fib}(N1,M1), \\
& \quad \quad N2 \text{ is } N-2, \text{fib}(N2,M2), M \text{ is } M1 + M2.
\end{align*}
\]
1.3 Evaluation

Factors:

- **Plain Prolog performance**

<table>
<thead>
<tr>
<th>SICStus</th>
<th>Yap</th>
<th>hProlog</th>
<th>XSB</th>
<th>SWI-Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>100.0%</td>
<td>78.0%</td>
<td>76.2%</td>
<td>146.8%</td>
</tr>
</tbody>
</table>

- **CHR optimizations + instantiation errors (disabled)**

- **Attributed variables implementation**
  - global store is hotspot
  - short reference chains
## 1.3 Evaluation

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Christian Holzbaur</th>
<th>K.U. Leuven</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SICStus</td>
<td>Yap</td>
</tr>
<tr>
<td>bool</td>
<td>100.0%</td>
<td>74.1%</td>
</tr>
<tr>
<td>fib</td>
<td>100.0%</td>
<td>61.7%</td>
</tr>
<tr>
<td>fibonacci</td>
<td>100.0%</td>
<td>59.6%</td>
</tr>
<tr>
<td>leq</td>
<td>100.0%</td>
<td>96.0%</td>
</tr>
<tr>
<td>primes</td>
<td>100.0%</td>
<td>104.8%</td>
</tr>
<tr>
<td>ta</td>
<td>100.0%</td>
<td>82.6%</td>
</tr>
<tr>
<td>wfs</td>
<td>100.0%</td>
<td>63.6%</td>
</tr>
<tr>
<td>zebra</td>
<td>100.0%</td>
<td>52.6%</td>
</tr>
<tr>
<td>average</td>
<td>100.0%</td>
<td>74.4%</td>
</tr>
</tbody>
</table>
2. Application
2.1 Well-Founded Semantics

Algorithm to compute WFS of logic programs without arguments. Fixpoint of 2 steps:

- **step 1**
  - decide truth of atoms based on facts and already defined atoms

- **step 2**
  - only consider undefined atoms and positive body literals
  - apply same reasoning as in step 1
  - still undefined: false are considered false
  - now defined: remain undefined
2.1 Well-Founded Semantics

Prototype implementation in CHR

- sequencing of steps:
  flag constraints to enable/disble rules
- stop fixpoint:
  dirty flag constraint + order of rules
- different semantics step 1 vs. step 2:
  replace c1/n with c2/n
- available as wfs.chr

Refined operational semantics crucial!
2.1 Well-Founded Semantics

Some used CHR idioms/patterns/hacks:

\[ \begin{align*}
\text{witness2} & \ \backslash \ \text{witness2} \ \leftrightarrow \ \text{true}. \\
\text{phase2}, \ \text{nbucl}(At,\_\_) & \ \Rightarrow \ \text{witness2}, \ \text{undefined2}(At). \\
\text{phase2}, \ \text{pos}(At,Cl) & \ \Rightarrow \ \text{pos2}(At,Cl). \\
\text{phase2}, \ \text{aclause}(Cl,At) & \ \Rightarrow \ \text{aclause2}(Cl,At). \\
\text{phase2}, \ \text{nbplit}(Cl,N) & \ \Rightarrow \ \text{nbplit2}(Cl,N). \\
\text{phase2}, \ \text{witness2} \ # \ \text{ID} & \ \leftrightarrow \ \text{phase1} \ \text{pragma} \ \text{passive}(\text{ID}). \\
\text{phase2} \ \backslash \ \text{nbplit2}(\_\_,\_\_) \ # \ \text{ID} & \ \leftrightarrow \ \text{true} \ \text{pragma} \ \text{passive}(\text{ID}). \\
\text{phase2} \ \backslash \ \text{aclause2}(\_\_,\_\_) \ # \ \text{ID} & \ \leftrightarrow \ \text{true} \ \text{pragma} \ \text{passive}(\text{ID}). \\
\text{phase2} & \ \leftrightarrow \ \text{true}.
\end{align*} \]
2.2 Java Memory Model

Java Memory Model flawed

- memory model: interaction threads - main memory
- JSR-133: new memory model
- Concurrent Constraint-based Memory Machines (V. Saraswat):
  - framework for memory models
  - events + constraints
  - rules for ordering between events
  - rules for linking reads to writes
2.2 Java Memory Model

CHR implementation

- prove claim of CCMMs: generative models
- CHR used for
  - finite domain constraints
  - partial order
  - ordering and linking rules
- poster at ICLP’04
3. Conclusion
3. Conclusion

- fairly portable CHR system
  - hProlog
  - XSB
  - SWI-Prolog
- competitive performance
- tabling in XSB
3.1 Future Work

- Debugger (Jan Wielemaker)
- Optimizations
  - Instantiation and other declarations
  - Intelligent backtracking (+ backmarking)?
  - CHR without global store?
- Tabling specific features and applications in XSB
3.1 Discussion Topics

- One CHR Standard
  - Syntax
  - Semantics
- Portable Options and Pragmas
- Programming in the large?
- Benchmark Suite