

Translating Constraint Handling Rules into Action Rules

Tom Schrijvers, Neng-Fa Zhou, Bart Demeo
K.U.Leuven, Belgium CUNY, USA





- 1 Action Rules
- 2 CHR2AR
 - Two-Headed Rules
 - Single-Headed Rules
 - Multi-Headed Rules
- 3 Optimization
- 4 Implementation & Evaluation
- 5 Conclusion



- 1 Action Rules
- 2 CHR2AR
 - Two-Headed Rules
 - Single-Headed Rules
 - Multi-Headed Rules
- 3 Optimization
- 4 Implementation & Evaluation
- 5 Conclusion

Action Rules

4/29

- ▶ rule-based
- ▶ event-driven
- ▶ applications:
 - ▶ constraint propagators (FD, finite set)
 - ▶ GUIs
- ▶ embedded in B-Prolog (N. Zhou)

Action Rule Syntax

$$\text{Agent, Conditions, \{Events\} \Rightarrow Body.}$$

Semantics

For *Agent*, whenever one of *Events* happens and if *Conditions* are satisfied, call *Body*.

Example

$$p(X), \text{var}(X), \{\text{event}(X, \text{Msg})\} \Rightarrow \text{writeln}(\text{Msg}).$$

Conditions:

- ▶ optional
- ▶ fixed set of builtins: `var/1`, `nonvar/1`, ...

Events:

- ▶ generated: when agent is created
- ▶ `ins(X)`: when `X` is instantiated
- ▶ `event(Channel,Message)`: custom event
 - ▶ `Channel`: variable to post event on
 - ▶ with `post_event(Channel,Message)`

The Language

7/29

Commitment Rule

Syntax

$$\text{Agent}, \text{Conditions} \Rightarrow \text{Body}.$$

Semantics

When an event wakes *Agent*, but no AR applies, and if *Conditions* are satisfied, call *Body* and kill *Agent*.

Example

$$p(X), \text{nonvar}(X) \Rightarrow \text{true}.$$

Example

8/29

Program:

```
p(X), var(X), {event(X,Msg), ins(X)} => writeln(Msg).  
p(X), nonvar(X) => true.
```

Query:

```
?- p(X),  
   post_event(X,hello),  
   post_event(X,world),  
   X = 1.
```

hello

world

Comparison of notable features

	AR	CHR
#heads	one	many
fire	many times	once
conditional	conditions, events	guards
keep head	action rule	propagation
remove head	(commitment rule)	simplification
impl. semantics	<i>breadth first</i>	refined (ω_r)

CHR2AR: Compile CHR to AR

Advantages

- ▶ also rule-based
- ▶ higher-level than attributed variables (?)
- ▶ benefits from AR efficiency
- ▶ new view on CHR compilation

Disadvantages

- ▶ single-headed rules
- ▶ different semantics



- 1 Action Rules
- 2 CHR2AR**
 - Two-Headed Rules
 - Single-Headed Rules
 - Multi-Headed Rules
- 3 Optimization
- 4 Implementation & Evaluation
- 5 Conclusion

Constraint Representation

12/29

$$\text{constr}(Cno, Alive, History, X)$$

- ▶ *Cno*: constraint identifier
- ▶ *Alive*: variable when alive, 0 when dead
- ▶ *History*: part of propagation history
- ▶ *X*: constraint arguments

Compilation Example

13/29

Very simple example to illustrate the principle:

Example CHR Code

```
a(X), b(X) ==> writeln(hello).
```

Constraint Setup

14/29

```
a(X) :-  
    gen_constr_id(Cno),  
    Constr = constr(Cno,Alive,History,X),  
  
    % “store” new constraint  
    get_channel(a_1_1,Channel),  
    agent_a_1_1(Channel,Cno,Alive,History,X),  
  
    % “activate” new constraint  
    get_channel(b_1_1,PChannel),  
    post_a_1(PChannel,Alive,Constr,X).
```

Occurrence Code

15/29

passive partner = agent
active constraint = event
 index = occurrence channel

```

a_1_1(Channel,Cno,Alive,History,X), var(Alive),
  {event(Channel,Partner), ins(Alive)} =>
    Partner = constr(PCno,PAlive,PHistory,PX),
    ( Cno == PCno, var(PAlive), X == PX,
      check_history(History,r1,PCno),
      check_history(PHistory,r1,Cno) ->
        update_history(History,r1,PCno),
        writeln(hello)
    );
    true
  ).
a_1_1(_____) => true.
  
```

Constraint (Re)Activation

16/29

synchronization of occurrences!

activate : generated

reactive : ins(X)

```
post_a_1(PChannel, Alive, Constr, X), var(Alive),  
  {generated, ins(Alive), ins(X)} =>  
    post_event(PChannel, Constr).  
post_a_1(_, _, _, _) => true.
```

Single-Headed Rules

17/29

Naive Approach

Turn them into two-headed rules.

- ▶ add artificial partner *dummy* to every rule

$C \Leftrightarrow B$ becomes $\text{dummy} \setminus C \Leftrightarrow B$

$C \Rightarrow B$ becomes $\text{dummy}, C \Rightarrow B$

- ▶ add *dummy* to the initial constraint store

$?- \text{dummy}, \text{Query}.$

Single-Headed Rules

18/29

Improved Approach

- ▶ do not include partner
 - ▶ no propagation history and related overhead
 - ▶ no initial dummy constraint
- ▶ use *private* channel for synchronization (ω_r)

Multi-Headed Rules

19/29

Turn them into two-headed rules (source to source):

$$C_1, C_2, \dots, C_n \implies B$$

becomes

$$\left\{ \begin{array}{l} C_1, C_2 \implies P_2 \\ P_2, C_3 \implies P_3 \\ \dots \\ P_{n-1}, C_n \implies B \end{array} \right.$$

where P_i are auxiliary constraints (manifest partial joins)

- ▶ **RETE** (eager and manifest joins) vs **LEAPS** (on demand)
 - ▶ LEAPS asymptotically better
 - ▶ used by most CHR implementations (e.g. K.U.Leuven CHR)
- ▶ source transformation inaccurate:
 - ▶ refined semantics **not** preserved
may fire occurrences out of order
fix: synchronization channel
 - ▶ simplification rules
fix: carry **Alive** flags in P_i
 - ▶ *dead* partial joins
fix: link **Alive** flags of heads with those of P_i



- 1 Action Rules
- 2 CHR2AR
 - Two-Headed Rules
 - Single-Headed Rules
 - Multi-Headed Rules
- 3 Optimization**
- 4 Implementation & Evaluation
- 5 Conclusion

Many known CHR optimizations can be reused or adapted easily:

- ▶ never-stored constraints
- ▶ ground constraints
- ▶ late storage
- ▶ ...

A new Action Rule feature for better CHR indexing:

$$\text{post_event}(C_1 \wedge \dots \wedge C_n, \text{Event})$$

Event is only sent to agents listening on all of C_1, \dots, C_n .

e.g. $\text{post_event}(X \wedge Y, \text{Event})$ reaches

$p(X, Y)$

$p(X, A)$

$p(B, Y)$



- 1 Action Rules
- 2 CHR2AR
 - Two-Headed Rules
 - Single-Headed Rules
 - Multi-Headed Rules
- 3 Optimization
- 4 Implementation & Evaluation**
- 5 Conclusion

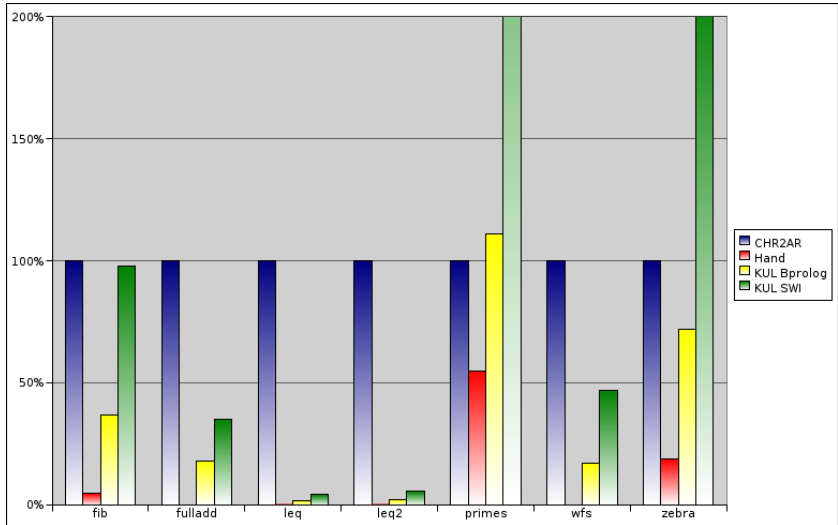
CHR2AR Compiler

- ▶ automatic (894 loc): basic schema with minor optimizations
- ▶ by hand: advanced optimizations

Comparison

K.U.Leuven CHR (7645 loc)

- ▶ in SWI-Prolog
- ▶ **new** port to B-Prolog





- 1 Action Rules
- 2 CHR2AR
 - Two-Headed Rules
 - Single-Headed Rules
 - Multi-Headed Rules
- 3 Optimization
- 4 Implementation & Evaluation
- 5 Conclusion

Results

- ▶ CHR2AR compilation schema
- ▶ RETE-like join algorithm: bad match for ω_r
- ▶ performance looks promising
- ▶ K.U.Leuven CHR port

Future Work

- ▶ automate optimizations
- ▶ trade-offs RETE vs. LEAPS
- ▶ alternate refined semantics?

Thank you!

29/29

Want to know more?

- ▶ Download the two systems
<http://www.probp.com/chr/>
- ▶ Technical Report CW 448, Dept. of Computer Science,
K.U.Leuven