# On the Expressive Power of Priorities in CHR

Maurizio Gabbrielli [1] Jacopo Mauro [1]
Maria Chiara Meo [2]

[1]Dipartimento di Scienze dell'Informazione, University of Bologna

[2]Dipartimento di Scienze, University of Chieti-Pescara

CHR Working Week, Ulm 2009

# Outline

# Motivations

## Claim

In [De Koninck et al. - 2007] it is claimed that “priorities do improve the expressivity of CHR”

## Our Contribution

- formal ground for this informal claim using a notion of expressivity coming from the field of concurrency theory
- dynamic priorities do not augment the expressivity

# CHR

## CHR

Constraint Handling Rules is a high-level programming language based on multi-headed, committed-choice, guarded multiset rewrite rules.

*Thom Frühwirth*

## CHR$^{rp}$

CHR$^{rp}$ extends CHR with user-defined priorities.

# CHR - syntax

- two types of constraints
  - *CHR constraints* or User defined constraints
  - *Built-in constraints* (we assume a given constraint theory which describes their meaning)
- three types of rules

$$\begin{array}{cc} \textit{propagation} & r@H \Rightarrow C \mid B \\ \textit{simplification} & r@H' \Leftrightarrow C \mid B \\ \textit{simpagation} & r@H \setminus H' \Leftrightarrow C \mid B \end{array}$$

- a program: sequence of rules
- a goal: multiset or sequence of constraints

# CHR$^{rp}$ - syntax

- priorities ($p$) are arithmetic expressions
- the rules are extended with priorities in the following way

$$\begin{array}{cc} \textit{propagation} & p :: r@H \Rightarrow C \mid B \\ \textit{simplification} & p :: r@H' \Leftrightarrow C \mid B \\ \textit{simpagation} & p :: r@H \setminus H' \Leftrightarrow C \mid B \end{array}$$

- if a priority has a variable then it is dynamic, static otherwise

# Operational semantics - 1

three different operational semantics considered:

- $\omega_t$ - the traditional semantics for CHR
  the rule

$$r \; @ \; H \setminus H' \Leftrightarrow C \mid B$$

  - can fire if $H \cup H'$ are in the store and $C$ is satisfied
  - when fired $H'$ deleted and $B$ added
  - propagation rule fires only once

# Operational semantics - 2

- $\omega_r$ - the refined semantics for CHR
  - introduced to model the execution mechanism of the current implementations
  - based on active constraints
  - order of the rules and constraints matters

- $\omega_p$ - the traditional semantics for CHR$^{rp}$
  - only rules with highest priority can fire

# CHR by example

## Less than or equal program in CHR

$$\text{reflexivity} \ @ \ \text{leq}(X, Y) \iff X = Y \mid true$$
$$\text{antisymmetry} \ @ \ \text{leq}(X, Y), \text{leq}(Y, X) \iff X = Y$$
$$\text{transitivity} \ @ \ \text{leq}(X, Y), \text{leq}(Y, Z) \implies \text{leq}(X, Z)$$

## Shortest path program in CHR$^{rp}$

$$1 :: \text{source}(V) \implies \text{dist}(V, 0)$$
$$1 :: \text{dist}(V, D_1) \backslash \text{dist}(V, D_2) \iff D_1 \leq D_2 | true$$
$$D + 2 :: \text{dist}(V, D), \text{edge}(V, C, U) \implies \text{dist}(U, D + C)$$

# $\omega_t$ semantics

**Solve** $\langle \{c\} \uplus G, S, B, T\rangle_n \stackrel{\omega_t}{\rightarrow}_P \langle G, S, c \wedge B, T\rangle_n$ where $c$ is a built-in constraint

**Introduce** $\langle \{c\} \uplus G, S, B, T\rangle_n \stackrel{\omega_t}{\rightarrow}_P \langle G, \{c\#n\} \cup S, B, T\rangle_{n+1}$ where $c$ is a CHR constraint

**Apply** $\langle G, H_1 \cup H_2 \cup S, B, T\rangle_n \stackrel{\omega_t}{\rightarrow}_P \langle C \uplus G, H_1 \cup S, \theta \wedge B, T \cup \{t\}\rangle_n$ where $P$ contains a (renamed apart) rule

$$r\ @H_1' \backslash H_2' \Longleftrightarrow g \mid C$$

and there exists a matching substitution $\theta$ s.t. $\mathtt{chr}(H_1) = \theta H_1'$, $\mathtt{chr}(H_2) = \theta H_2'$, $\mathcal{CT} \models B \rightarrow \exists_{-Fv(B)}(\theta \wedge g)$ and $t = \mathtt{id}(H_1) +\!\!+\ \mathtt{id}(H_2) +\!\!+\ [r] \notin T$

# $\omega_p$ semantics

**Solve** $\langle \{c\} \uplus G, S, B, T\rangle_n \overset{\omega_p}{\rightarrow}_P \langle G, S, c \wedge B, T\rangle_n$ where $c$ is a built-in constraint

**Introduce** $\langle \{c\} \uplus G, S, B, T\rangle_n \overset{\omega_p}{\rightarrow}_P \langle G, \{c\#n\} \cup S, B, T\rangle_{n+1}$ where $c$ is a CHR constraint

**Apply** $\langle \emptyset, H_1 \cup H_2 \cup S, B, T\rangle_n \overset{\omega_p}{\rightarrow}_P \langle C, H_1 \cup S, \theta \wedge B, T \cup \{t\}\rangle_n$ where $P$ contains a (renamed apart) rule

$$p :: r \,@ H_1' \backslash H_2' \Longleftrightarrow g \mid C$$

and there exists a matching substitution $\theta$ s.t. $\mathtt{chr}(H_1) = \theta H_1'$, $\mathtt{chr}(H_2) = \theta H_2'$, $\mathcal{CT} \models B \rightarrow \exists_{-Fv(B)}(\theta \wedge g)$ and $t = \mathtt{id}(H_1) +\!\!+\, \mathtt{id}(H_2) +\!\!+\, [r] \notin T$. Furthermore no rule of priority $p'$ and substitution $\theta'$ exists with $\theta' p' < \theta p$ for which the above conditions hold

# Observables

- initial configuration: the goal constraints are added into the store
- two final configuration:
  - failed (constraints in the store are unsatisfiable)
  - terminated (no rule can fire)
- observables are the data sufficient answers: terminated configurations that contain only built-in constraints

# Acceptable encoding

- language encoding with additional proprieties to fulfill
- motivation: discriminating differing (Turing powerful) languages
- in our work we require
  1. the observables remain the same
  2. compositionality of the goal encoding w.r.t. the conjunction of atoms

# CHR vs CHR$^{rp}$

**Theorem**

There exists no acceptable encoding of CHR$^{rp}$ in CHR

- idea of the proof:
  - considered the Last Man Standing Problem (LMS problem)
  - solved the problem in CHR$^{rp}$
  - shown that LMS can not be solved in CHR (under acceptability assumption)

**LMS problem solved in CHR$^{rp}$**

$$1 :: a(X), a(X) \Leftrightarrow X = no$$

$$2 :: a(X) \Leftrightarrow X = no | true$$

$$3 :: a(X) \Leftrightarrow X = yes$$

# $\omega_t$ VS $\omega_r$

**Theorem**

There exists no acceptable encoding of $CHR_{\omega_r}$ into $CHR_{\omega_t}$

- proof idea: using the LMS problem like in the previous case

**LMS Program in CHR with $\omega_r$ semantics**

$$a(X) \Leftrightarrow X = no | true$$

$$a(X) \Leftrightarrow X = yes | false$$

$$d(X), b(X), a(X) \Leftrightarrow X = no$$

$$a(X) \Leftrightarrow b(Y), b(X), c(X)$$

$$c(X), b(Y) \Leftrightarrow Y = yes, d(X)$$

$$d(X), b(Y) \Leftrightarrow X = yes | true$$

# Static vs dynamic priorities

**Theorem**

There is an acceptable encoding of $CHR^{rp}$ with dynamic priorities into $CHR^{rp}$ with static priorities

- encoding idea: instead of one rule execution
  1. detect which rules have the higher priority
  2. fire only one of these rules
- assumed that equalities and inequalities can be used as built in constraints

# CHR vs Prolog

- result: no acceptable encoding from CHR to Prolog (extension of a previous result [Di Giusto et al. 2009])
- Prolog program are considered w.r.t. the computed answer semantics
- assumed that no dynamic procedures are used
- an acceptable encoding from CHR to Prolog
  - preserves the compositionality of the goal
  - the Prolog program has no computed answers iff the CHR program has an empty data sufficient answer

# Conclusions

- we use the notion of acceptable encoding for studying the expressivity of CHR languages
- we proved that priorities improve the expressivity of CHR
- we proved that the refined semantics improve the expressivity of CHR considered with the traditional semantics
- we proved that dynamic priorities do not augment the expressivity of CHR with static priorities
- we extend a previous result showing that CHR can not be encoded in Prolog

# Future Work

We plan to

- investigate the relation between priorities and negation as absence
- consider the refined semantics for $CHR^{rp}$
- consider data qualified answers instead of data sufficient answers