Decentralized Execution of Constraint Handling Rules for Ensembles

Edmund S. L. Lam  Iliano Cervesato
sllam@qatar.cmu.edu  iliano@cmu.edu

Carnegie Mellon University, Qatar
Department of Computer Science

Funded by the Qatar National Research Fund as project NPRP 09-667-1-100
(Effective Programming for Large Distributed Ensembles)

16th Sept 2013, PPDP’13
1. Introduction
2. The $CHR^e$ Language
3. Operational Semantics
4. Conclusion
Main-stream Distributed Programming

- Distributed programming is fast becoming *inevitable*!
  - Multicore architectures
  - Cloud computing and big data
  - Mobile smart phones and tablets
- Main-stream demand for software that:
  - Execute in a decentralized manner.
  - Communicate and coordinate as a collective ensemble.
- We explore adapting Constraint Handling Rules (CHR) for this purpose!
  - Its declarative and concurrent.
  - Online (Computation can start on partial inputs)
  - Anytime (Computation can be interrupted for approximate)
Constraint Handling Rules (CHR), Traditionally

- General form of a CHR rule:

\[
\begin{align*}
\text{Rule Name} & \quad r \quad \odot \quad P \\
\text{Propagated Heads} & \quad \setminus \quad S \quad \iff \quad G \quad \mid \quad B \\
\text{Simplified Heads} & \quad \iff \quad G \\
\text{Body} & \quad \mid \quad B
\end{align*}
\]

- CHR rules are applied to a multiset of constraints, known as the *constraint store* \( St \).

  Informally: If we have \( P \cup S \) in \( St \), such that \( G \) is true, then replace \( S \) in \( St \) with \( B \).

- Short forms:
  - If \( P \) is empty, we write \( r \odot S \iff G \mid B \).
  - If \( S \) is empty, we write \( r \odot P \Rightarrow G \mid B \).
Example of CHR Program: Find all shortest path

\[\text{base} @ \text{edge}(X, Y, D) \implies \text{path}(X, Y, D)\]

\[\text{elim} @ \text{path}(X, Y, D1) \setminus \text{path}(X, Y, D2) \iff D1 \leq D2 | \text{true}\]

\[\text{trans} @ \text{edge}(X, Y, D1), \text{path}(Y, Z, D2) \implies X \neq Z | \text{path}(X, Z, D1 + D2)\]

Example of CHR derivations (e for edge, p for path):

\[\begin{array}{l}
\text{base} \quad \{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8)\}
\\
\text{base} \quad \{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5)\}
\\
\text{trans} \quad \{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8)\}
\\
\text{trans} \quad \{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8), p(l_0, l_2, 7)\}
\\
\text{trans} \quad \{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8), p(l_0, l_2, 7), p(l_0, l_2, 9)\}
\\
\text{elim} \quad \{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8), p(l_0, l_2, 7)\}
\end{array}\]
Why Decentralized and Distribute Solving?

- Exploiting distributed computing resources.
  (e.g., shortest path, page rank, minimal spanning tree, distributed sorting)

- Not feasible for centralized storage.
  (e.g., P2P mobile applications, embedded device programming)
Outline

1. Introduction
2. The CHRe Language
3. Operational Semantics
4. Conclusion
Recalling “Find Shortest Path” in CHR

\[
\text{base } \odot \text{ edge}(X, Y, D) \implies \text{path}(X, Y, D)
\]

\[
\text{elim } \odot \text{ path}(X, Y, D1) \setminus \text{path}(X, Y, D2) \iff D1 \leq D2 \mid \text{true}
\]

\[
\text{trans } \odot \text{ edge}(X, Y, D1), \text{path}(Y, Z, D2) \implies X \neq Z \mid \text{path}(X, Z, D1 + D2)
\]

\[
\begin{align*}
&\gtrsim e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3)\]
&\Gcirc e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3)
&\qquad , p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_0, 4), p(l_1, l_2, 2), p(l_2, l_0, 11), p(l_3, l_2, 8), p(l_3, l_1, 3)\]
&\Gcirc e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3)
&\qquad , p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_0, 4), p(l_1, l_2, 2), p(l_2, l_0, 11), p(l_3, l_2, 8), p(l_3, l_1, 3)
&\qquad , p(l_0, l_2, 9), p(l_0, l_2, 7), p(l_1, l_0, 13), p(l_1, l_3, 5), p(l_2, l_1, 16), p(l_2, l_3, 12), p(l_3, l_0, 19)
&\qquad , p(l_3, l_0, 7), p(l_3, l_2, 5)\]
&\ldots
&\Gcirc e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3)
&\qquad , p(l_0, l_1, 4), p(l_0, l_2, 6), p(l_0, l_3, 1), p(l_1, l_0, 4), p(l_1, l_2, 2), p(l_1, l_3, 5), p(l_2, l_0, 11)
&\qquad , p(l_2, l_1, 15), p(l_2, l_3, 12), p(l_3, l_0, 19), p(l_3, l_1, 3), p(l_3, l_2, 5)\]
\]

How can we distribute the constraints, but still collectively solve?
Explicit Location Annotations

\[ \text{base} @ \text{[X]edge}(Y, D) \implies \text{[X]path}(Y, D) \]

\[ \text{elim} @ \text{[X]path}(Y, D1) \setminus \text{[X]path}(Y, D2) \iff D1 \leq D2 | \text{true} \]

\[ \text{trans} @ \text{[X]edge}(Y, D1), \text{[Y]path}(Z, D2) \implies X \neq Z | \text{[X]path}(Z, D1 + D2) \]

<table>
<thead>
<tr>
<th>Location ( l_0 )</th>
<th>Location ( l_1 )</th>
<th>Location ( l_2 )</th>
<th>Location ( l_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( { e(l_1, 5), e(l_3, 1) } )</td>
<td>( { e(l_0, 4), e(l_2, 2) } )</td>
<td>( { e(l_0, 11) } )</td>
<td>( { e(l_2, 8), e(l_1, 3) } )</td>
</tr>
<tr>
<td>( \implies * )</td>
<td>( \implies * )</td>
<td>( \implies * )</td>
<td>( \implies * )</td>
</tr>
<tr>
<td>( { e(l_1, 5), e(l_3, 1) ) , ( p(l_1, 5), p(l_3, 1) } )</td>
<td>( { e(l_0, 4), e(l_2, 2) ) , ( p(l_0, 4), p(l_2, 2) } )</td>
<td>( { e(l_0, 11) ) , ( p(l_0, 11) } )</td>
<td>( { e(l_2, 8), e(l_1, 3) ) , ( p(l_2, 8), p(l_1, 3) } )</td>
</tr>
<tr>
<td>( \implies * )</td>
<td>( \implies * )</td>
<td>( \implies * )</td>
<td>( \implies * )</td>
</tr>
<tr>
<td>( { e(l_1, 5), e(l_3, 1) ) , ( p(l_1, 5), p(l_3, 1) ) , ( p(l_2, 9), p(l_2, 7) } )</td>
<td>( { e(l_0, 4), e(l_2, 2) ) , ( p(l_0, 4), p(l_2, 2) ) , ( p(l_0, 13), p(l_3, 5) } )</td>
<td>( { e(l_0, 11) ) , ( p(l_0, 11) ) , ( p(l_1, 15) ) , ( p(l_3, 12) } )</td>
<td>( { e(l_2, 8), e(l_1, 3) ) , ( p(l_0, 19), p(l_1, 3) ) , ( p(l_2, 5) } )</td>
</tr>
</tbody>
</table>

How to execute in decentralized manner, yet preserve soundness (w.r.t abstract CHR semantics)?
Multiset Matching in $CHR^e$ Rules

\[
\text{base} \circ [X]\text{edge}(Y, D) \Longrightarrow [X]\text{path}(Y, D) \\
\text{elim} \circ [X]\text{path}(Y, D1) \setminus [X]\text{path}(Y, D2) \Longleftrightarrow D1 \leq D2 \mid \text{true} \\
\text{trans} \circ [X]\text{edge}(Y, D1), [Y]\text{path}(Z, D2) \Longrightarrow X \neq Z \mid [X]\text{path}(Z, D1 + D2)
\]

Location $l_0$

\[
\begin{align*}
\langle & e(l_1, 5), e(l_3, 1) \rangle \\
\Rightarrow^* & \langle e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1) \rangle \\
\Rightarrow^* & \langle e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1), p(l_2, 9), p(l_2, 7) \rangle \\
\Rightarrow^* & \langle e(l_1, 5), e(l_3, 1), p(l_1, 4), p(l_2, 6), p(l_3, 1) \rangle \\
\end{align*}
\]

Location $l_1$

\[
\begin{align*}
\langle & e(l_0, 4), e(l_2, 2) \rangle \\
\Rightarrow^* & \langle e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2) \rangle \\
\Rightarrow^* & \langle e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_0, 13), p(l_3, 5) \rangle \\
\Rightarrow^* & \langle e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_3, 5) \rangle \\
\end{align*}
\]

Stark difference between base and elim rules, and the trans rule.
Localized Multiset Matching

base @ \([X]edge(Y, D) \rightarrow [X]path(Y, D)\)

elim @ \([X]path(Y, D1) \setminus [X]path(Y, D2) \iff D1 \leq D2 \mid true\)

trans @ \([X]edge(Y, D1), [Y]path(Z, D2) \rightarrow X \neq Z \mid [X]path(Z, D1 + D2)\)

Location \(l_0\)

\[\{e(l_1, 5), e(l_3, 1)\}\]

\[\rightarrow^* \{e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1)\}\]

\[\rightarrow^* \{e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1), p(l_2, 9), p(l_2, 7)\}\]

...\]

\[\rightarrow^* \{e(l_1, 5), e(l_3, 1), p(l_1, 4), p(l_2, 6), p(l_3, 1)\}\]

Location \(l_1\)

\[\{e(l_0, 4), e(l_2, 2)\}\]

\[\rightarrow^* \{e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2)\}\]

\[\rightarrow^* \{e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_0, 13), p(l_3, 5)\}\]

...\]

\[\rightarrow^* \{e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_3, 5)\}\]

Rule-heads of elim rule are 'localized'. Same for base rule.
Distributed Multiset Matching

base \( @ [X]edge(Y, D) \implies [X]path(Y, D) \)

elim \( @ [X]path(Y, D1) \setminus [X]path(Y, D2) \iff D1 \leq D2 | true \)

trans \( @ [X]edge(Y, D1), [Y]path(Z, D2) \implies X \neq Z | [X]path(Z, D1 + D2) \)

Rule-heads of trans rule are partially found in \( l_0 \) and \( l_1 \)!
We need manner of classification of $CHR^e$ rules.

Impose some conditions on the rule heads...

such that we ensure topological information of the problem is *sufficiently encoded* in rule heads.
n-Neighbor Restriction: Counting Neighbors

- **base**: \([X]\) edge\((Y, D) \implies [X] path(Y, D)\)

- **elim**: \([X]\) path\((Y, D1) \setminus [X] path(Y, D2) \iff D1 \leq D2 \mid \text{true}\)

- **trans**: \([X]\) edge\((Y, D1), [Y]\) path\((Z, D2) \implies X \neq Z \mid [X] path(Z, D1 + D2)\)

- We call \(X\) and \(Y\) matching locations.
- 0-neighbor restricted rules requires localized multiset matching.
- \(m\)-neighbor restricted rules (for \(m \geq 1\)) requires distributed multiset matching.
- But what’s the relation between \(X\) and \(Y\)?
**n-Neighbor Restriction: Imposing Connectivity**

- **$X$ and $Y$ must be directly connected.**
- Connection represented in constraints: Location $Y$ is referenced in location $X$’s constraint arguments.
- Forces *topology* (of $X$ and $Y$) to be specified as part of the constraint problem.
- 1-neighbor restriction is similar to link-restriction in [?], but without specialized 'link' constraints.
n-Neighbor Restriction, in General

- A n-Neighbor Restricted Rule in general:
  - Has $n + 1$ distinct matching obligations (i.e., $X$ and $Y_i$’s).
  - Exists one matching location $X$, that is directly connected to all others. (i.e., $\forall i \in I_n. \ Y_i \in FV(S_x \cup P_x)$)
- We call $X$ the primary location.
- “Star” Topology:
Outline

1. Introduction
2. The CHR\textsuperscript{e} Language
3. Operational Semantics
4. Conclusion
Executing 0-Neighbor Restricted Rules

0-neighbor restriction: **Local matching, send to and create neighbors.**

\[ r @ [X]P_x \setminus [X]S_x \iff G | [X]B_x, [Z_i]B_{z_i}, \exists D . [D_j]B_{d_j} \]

where \( Z_i \in \text{FV}(P_x \cup S_x) \) and \( D_j \in D \)

What happens when we apply \( r \)?
Execution 0-Neighbor Restricted Rules

0-neighbor restriction: Local matching, send to and create neighbors.

\[ r \odot [X]P_x \setminus [X]S_x \iff G \mid [X]B_x, [Z_i]B_{z_i}, \exists D : [D_j]B_{d_j} \]

where \( Z_i \in FV(P_x \cup S_x) \) and \( D_j \in D \)
Decentralized and asynchronous local CHR solvers, specifically $\omega_r$ [?].

All 'send' and 'receive' are asynchronous, thanks to monotonicity of CHR! (i.e. if $St \mapsto^* St'$ then $(St \cup St'') \mapsto^* (St' \cup St'')$)

Please see paper for technical details and results.
Executing 1-Neighbor Restricted Rules

\[
\text{swap} : [X] \text{neighbor}(Y) \setminus [X] \text{color}(C_1),
\]
\[
[Y] \text{color}(C_2) \iff [Y] \text{color}(C_1), [X] \text{color}(C_2)
\]

\[
\begin{align*}
\text{St}_x \cup \{n(y), c(pink)\}^y \\
\text{St}_y \cup \{n(x), c(brown)\}^x \\
\end{align*}
\]

\[
\begin{align*}
\text{match} \quad \leftrightarrow \\
\text{swap} \\
\text{add body constraints}
\end{align*}
\]

We need to deal with competing (over-lapping) rule applications!
Executing 1-Neighbor Restricted Rules

- Executing 1-Neighbor Restricted rules requires some form of consensus protocol between primary and neighbor matching location.
- Consensus between primary and neighbor location to commit to a specific 1-neighbor restricted rule instance.
- For now, we assume a reliable network (i.e., fault-free)
Consensus in 1-Neighbor Restriction

\[ \text{swap} \circ @ [X] \text{neighbor}(Y) \setminus [X] \text{color}(C1), [Y] \text{color}(C2) \iff [Y] \text{color}(C1), [X] \text{color}(C2) \]

\( \text{St}_x \sqcup \downarrow n(y), c(pink) \downarrow \)

\( \text{St}_x \sqcup \downarrow n(y), c(pink) \downarrow \)

\( \text{St}_x \sqcup \downarrow n(y) \downarrow \)

\( \text{St}_x \sqcup \downarrow n(y), c(brown) \downarrow \)

Location \( x \)

Do you have \( \downarrow c(C2) \) ?

Yes, I do! \( C2 = \text{brown} \)

Commit, with \( c(pink) \)!

Yes, committed!

Location \( y \)

\( \text{St}_y \sqcup \downarrow n(x), c(brown) \downarrow \)

\( \text{St}_y \sqcup \downarrow n(x), c(brown) \downarrow \)

\( \text{St}_y \sqcup \downarrow n(x), c(pink) \downarrow \)
Implementing Consensus in $\omega_0^e$

- We implement this consensus in $\omega_0^e$, with source-to-source encoding $\rightsquigarrow_{1Nb}^\text{basic}$:

$$\text{swap} @ [X]\text{neighbor}(Y) \setminus [X]\text{color}(C_1), [Y]\text{color}(C_2) \iff [Y]\text{color}(C_1), [X]\text{color}(C_2)$$

$$\begin{align*}
\text{swap1} @ [X]\text{neighbor}(Y), [X]\text{color}(C) & \Rightarrow [Y]\text{req}(X, Y, C) \\
\text{swap2} @ [Y]\text{color}(C') \setminus [Y]\text{req}(X, Y, C) & \iff [X]\text{match}(X, Y, C, C') \\
\text{swap3} @ [X]\text{neighbor}(Y) \setminus [X]\text{color}(C), [X]\text{match}(X, Y, C, C') & \iff [Y]\text{commit}(X, Y, C, C') \\
\text{swap4a} @ [Y]\text{commit}(X, Y, C, C'), [Y]\text{color}(C') & \iff [X]\text{color}(C'), [Y]\text{color}(C) \\
\text{swap4b} @ [Y]\text{commit}(X, Y, C, C') & \iff [X]\text{color}(C)
\end{align*}$$

- **Synchronization constraints** to drive the consensus building:
  - req: Represents match of X’s obligations.
  - match: Represents match of Y’s obligations.
  - commit: Represents X’s commitment to the rule instance.
Implementing Consensus in $\omega_0^e$

\[
\text{St}_x \cup \downarrow n(y), c(pink) \uparrow \n
\]

$\text{St}_x \cup \downarrow n(y), c(pink) \uparrow$

$\rightarrow_{\text{swap1}} \text{St}_x \cup \downarrow n(y), c(pink) \uparrow$

$[X]n(Y), [X]c(C) \Rightarrow [Y]\text{req}(X, Y, C)$

Commit primary match obligations

\[
\text{St}_x \cup \downarrow n(y), c(pink), \\
m\text{match}(x, y, pink, brown) \uparrow \\
\rightarrow_{\text{swap3}} \text{St}_x \cup \downarrow n(y) \uparrow
\]

$[X]n(Y) \setminus [X]c(C), [X]\text{match}(X, Y, C, C')$

$\Leftarrow [Y]\text{commit}(X, Y, C, C')$

Send $c(brown)$

$\text{St}_x \cup \downarrow n(y), c(brown) \uparrow$

Send $\text{req}(x, y, pink)$

Send $\text{match}(x, y, pink, brown)$

Commit neighbor match obligations

and complete rule instance application

Send $\text{commit}(x, y, pink, brown)$

$\text{St}_y \cup \downarrow n(x), c(brown) \uparrow$

$\rightarrow_{\text{swap2}} \text{St}_y \cup \downarrow n(x), c(brown) \uparrow$

$[Y]c(C') \setminus [Y]\text{req}(X, Y, C) \Leftarrow [X]\text{match}(X, Y, C, C')$

Send $c(brown)$

$\text{St}_y \cup \downarrow n(x), c(brown), \text{commit}(x, y, pink, brown) \uparrow$

$\rightarrow_{\text{swap4a}} \text{St}_y \cup \downarrow n(x), c(pink) \uparrow$

$[Y]\text{commit}(X, Y, C, C'), [Y]c(C') \Leftarrow [X]c(C'), [Y]c(C)$
Optimizations and Soundness

- $\leadsto_{\text{basic}}$ is the general encoding scheme.
- We can use optimized encoding schemes if primary or neighbor match obligation is known to be persistent. E.g.:

  \[
  \text{trans} \circ [X]\text{edge}(Y, D1), [Y]\text{path}(Z, D2) \Rightarrow X \neq Z \mid [X]\text{path}(Z, D1 + D2)
  \]

  $\leadsto_{\text{p-persist}}$

  \[
  \begin{align*}
  \text{trans1} \circ [X]\text{edge}(Y, D1) &\Rightarrow [Y]\text{req}(X, Y, D1) \\
  \text{trans2} \circ [Y]\text{req}(X, Y, D1), [Y]\text{path}(Z, D2) &\Rightarrow X \neq Z \mid [X]\text{path}(Z, D1 + D2)
  \end{align*}
  \]

  $\leadsto_{\text{p-persist}}$

- $\leadsto_{\text{p-persist}}$ is similar to link-restriction encoding in [?].
- We show soundness for composite encoding ($\leadsto_{\text{1Nb}}$):
  - Given $P_1 \leadsto_{\text{1Nb}} P_0$, derivations on $P_0$ has a correspondence to derivations on $P_1$.
  - See paper for details!
What about \textit{n}-Neighbor Restriction?

- \( r \otimes P_p, P_1, \ldots, P_i, \ldots, P_n \setminus S_p, S_1, \ldots, S_i, \ldots S_n \Leftrightarrow G \mid B \) where location \( p \) is the \textit{primary} and each \( i \) is the \( i^{th} \) \textit{neighbor}.

![Diagram]

- An instance of \( n + 1 \) consensus problem:
  - Location \( p \) as coordinator and a cohort.
  - Each location \( i \) as a cohort.
- Generalizes \( \sim_{1Nb} \) for a primary and \( n \)-neighbor locations.
- Formalized and proven in Technical Report [?].
Current Status

- Contributions, in this paper:
  - $n$-Neighbor Restriction
  - $\omega^e_0$ operational semantics for decentralized execution of $CHR^e$
  - Source-to-source encoding $\leadsto_{1Nb}$

- See the paper for technical details!

- See technical report [?] for proofs and more technical details (e.g., $\leadsto_{nNb}$)

- Prototype implementation in Python:
  - Available at GitHub, https://github.com/sllam/msre-py
  - Built on top of MPI libraries
  - Implements $\omega^e_0$ with $\leadsto_{1Nb}$.
  - Small set of examples (e.g., hyper-quicksort, GHS algorithm)
Related Works

- Meld [?]
- Distributed Datalog[?, ?]
- Compiling CHR to parallel hardware [?]
- Parallel CHR on Multicore [?]
- Multiple removals in parallel CHR rewriting [?]
Future Works

- High-Performance implementation:
  - Lower level runtime (e.g., C++)
  - With existing CHR optimizations (e.g., optimal join ordering, multiple removals, etc.)
  - Large scale benchmarking
  - Dynamic Load-balancing and fault tolerance.

- Other local CHR operational semantics (e.g., $CHR^{rp}$).

- Advance features, like aggregates, set comprehensions.

Thank You!