1. Challenges of Parallel and Distributed Programming

- A notoriously laborious and difficult endeavor
- Wide range of technical difficulties (e.g., deadlock, atomicity, fault-tolerance).
- Traditional computational problems (e.g., correctness, completeness, termination).
- While ensuring scalability and performance effectiveness.

Open research problem:
- Distributed programming frameworks (e.g. Map reduce [DG08], Graph Lab [LKK+10], Pregel [MBAB+10], Mizan [KaKJ10]).
- Distributed programming languages (e.g. Erlang [AV90], X10 [SSvP07], NetLog [GW10], Meld [CAR0+12]).
- High-level programming abstractions (e.g. Join Patterns [TR11], Parallel CHR [LS11]).

We seek an approach that is declarative, based on logical foundations, expressive and concise.
Motivated by chemical reaction equations:

\[6\text{CO}_2 + 6\text{H}_2\text{O} \rightarrow 2\text{C}_2\text{H}_2\text{O}_6 + 6\text{O}_2\]

2. Introducing Rule-Based Multiset Rewriting

- Constraint Handling Rules (CHR) [Fr098]
- Rule-based constraint logic programming language.
- Based on multiset rewriting over first order predicate terms.
- Concurrent, committed choice and declarative.

CHR programs consist of a set of CHR rules of the following form:

| Informally means: If we have \( P \) and \( S \) such that \( G \) is satisfiable, replace \( S \) with \( B \).
| Example: Greatest common divisor (GCD)

\[
\begin{align*}
\text{base} & : \text{gcd}(0) \equiv \text{true} \\
\text{reduce} & : \text{gcd}(N \cdot M) \equiv 0 < N < M \lor \text{gcd}(M - N)
\end{align*}
\]

3. MSRE, Distributed Multiset Rewriting for Ensembles

- Elements are distributed across distinct locations \((k_1, k_2, \ldots, k_N)\), each possessing its own multiset of elements.

- Rewrite rules explicitly reference the relative location of constraints:

\[
\begin{align*}
\text{base rule:} & \quad \text{edge}_{k_1}(k_2, \ldots, k_N) \\
\text{elim rule:} & \quad \text{path}(k_1, k_2, \ldots, k_N)
\end{align*}
\]

4. Example: Parallel Mergesort

Parallel mergesort: Assumes tightly coupled ensembles (multicore, shared memory, etc.).

- New locations “dynamically” created to solve sub-problems.
- Completed sub-problems are transmitted to the “parent” location.

5. Example: Distributed Hyper-Quicksort

Distributed Hyper-Quicksort: Assumes loosely coupled ensembles (network, message passing interface, etc.)

- “Local” sorting algorithm Parallel merge sort rules

\[
\begin{align*}
\text{base rule:} & \quad \text{sorted}(X, X, X, \ldots, X) \equiv \text{true} \\
\text{merge rule:} & \quad \text{merge}(X, Y) \equiv \text{sorted}(X, Y, \ldots, X, Y, \ldots, X)
\end{align*}
\]

6. Multiset Comprehensions

- MSRE language includes multiset comprehension patterns
- Additional form of constraint patterns: \( p(i) \mid g(x \in t) \)
- \( t \) is the multiset of all bindings to \( x \)
- Example: pivoted swapping via comprehensions

\[
\begin{align*}
\text{swap}(X, Y, P) & \equiv \text{data}(X, D) \lor \text{data}(Y, D) \\
\text{pivot}(X, X, Y, P) & \equiv \text{data}(X, D) \lor \text{data}(Y, D)
\end{align*}
\]

7. Preliminary Experiment Results

- Execution times (ms) for various optimizations on programs with increasing input size.
- Preliminary, but promising: we get code reduction plus some performance improvement.

8. Research Outcome, So Far...

- Prototype Implementation:
  - Download: https://github.com/sllam/chrcp
  - Online Demo: http://rise4fun.com/msre

- Publications:
  - "Decentralized Execution of Constraint Handling Rules for Ensembles", PPDP’13
  - "Constraint Handling Rules with Multiset Comprehension Patterns", CHR’14
  - "Reasoning About Set Comprehensions", SMT’14
  - "Optimized Compilation of Multiset Rewriting with Comprehensions", APLAS’14

9. What’s Next?

- Practical applications for MSRE:
  - cloud computing applications
  - P2P apps on mobile devices, Android SDK (Funded by the QNRF, JSREP 4-003-2-001)
  - Logical interpretation of comprehensions: Logical proof system for MSRE
  - Linear logic + Fixed Points + Subexponentials

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