Choreographic Compilation of Decentralized Comprehension Patterns

Iliano Cervesato
iliano@cmu.edu

Edmund S.L. Lam
sllam@qatar.cmu.edu

Ali Elgazar
aee@cmu.edu

Carnegie Mellon University

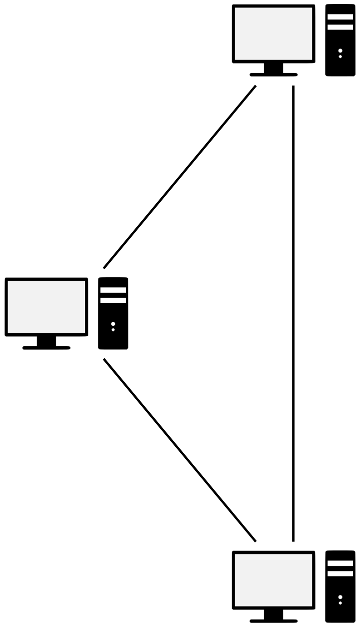
Supported by QNRF grants NPRP 4-341-1-059, NPRP 4-1593-1-290 and JSREP 4-003-2-001

RuleML 2016

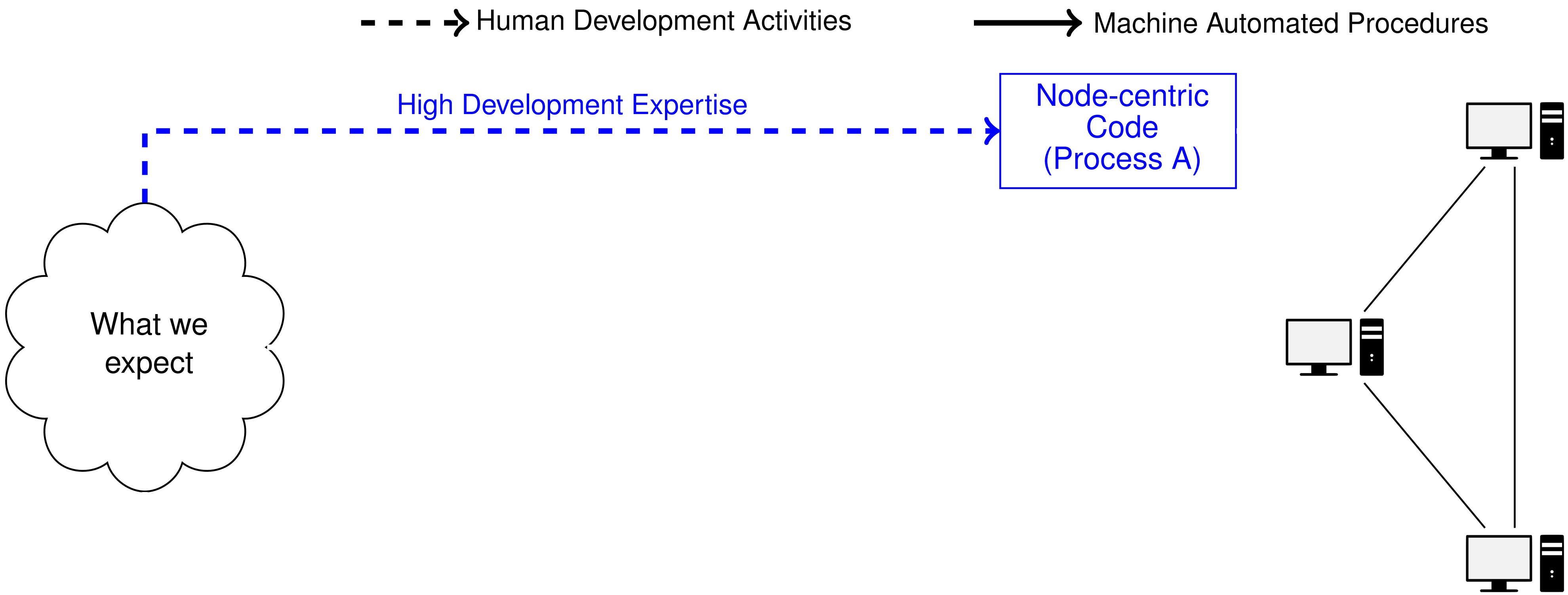
Outline

Traditional Concurrent/Distributed Programming

- - - → Human Development Activities ——→ Machine Automated Procedures

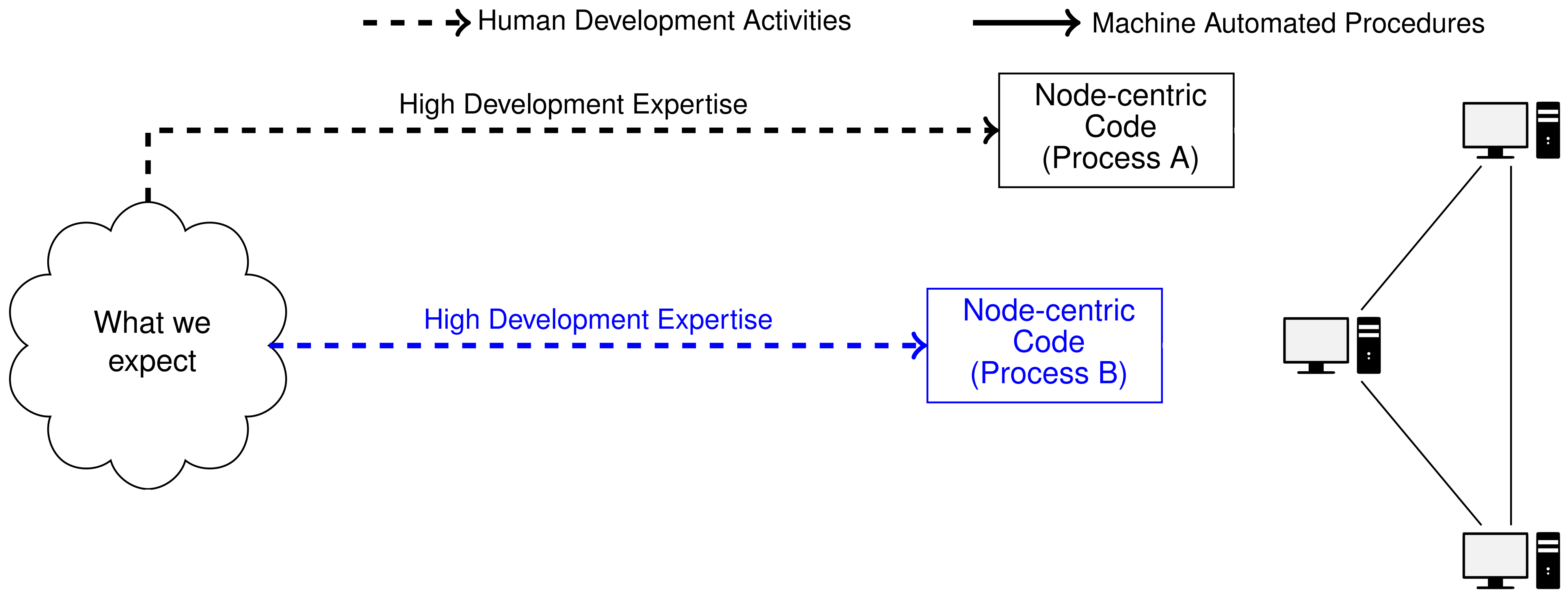


Traditional Concurrent/Distributed Programming



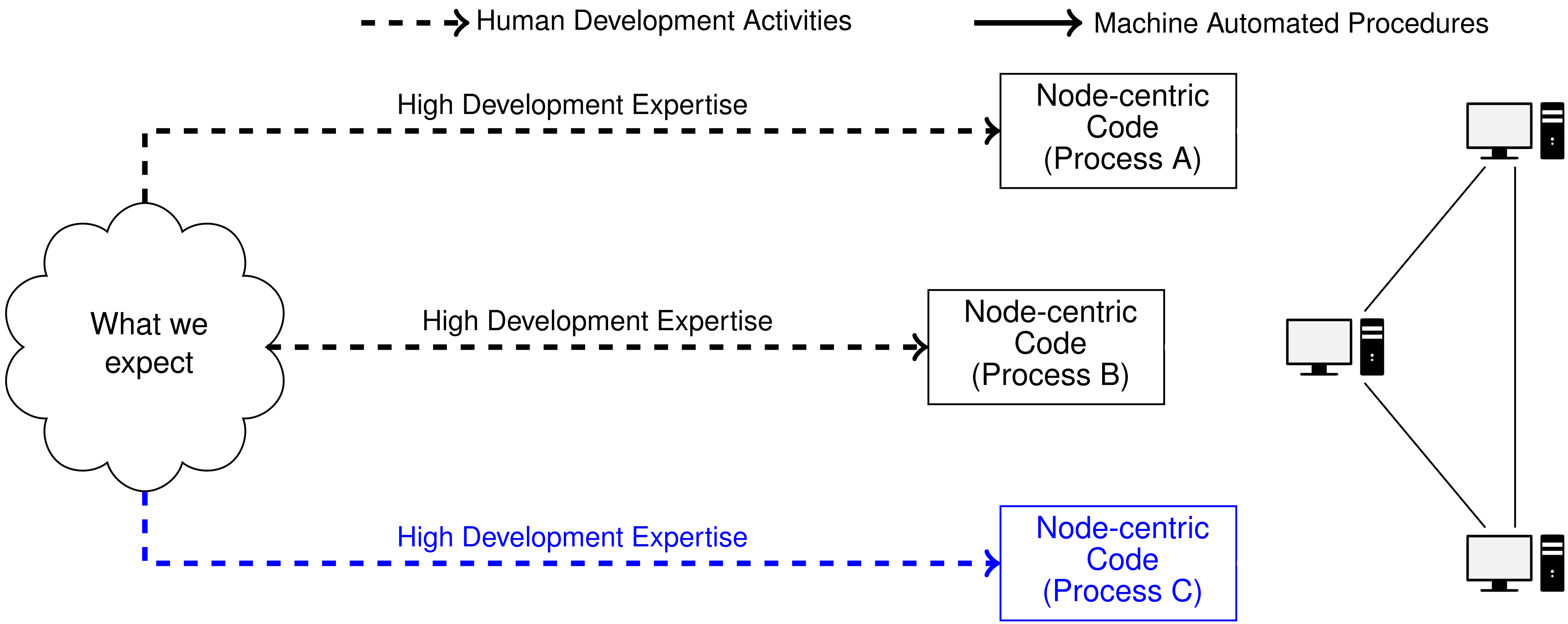
Start by writing code for a process A (e.g., a consumer)

Traditional Concurrent/Distributed Programming



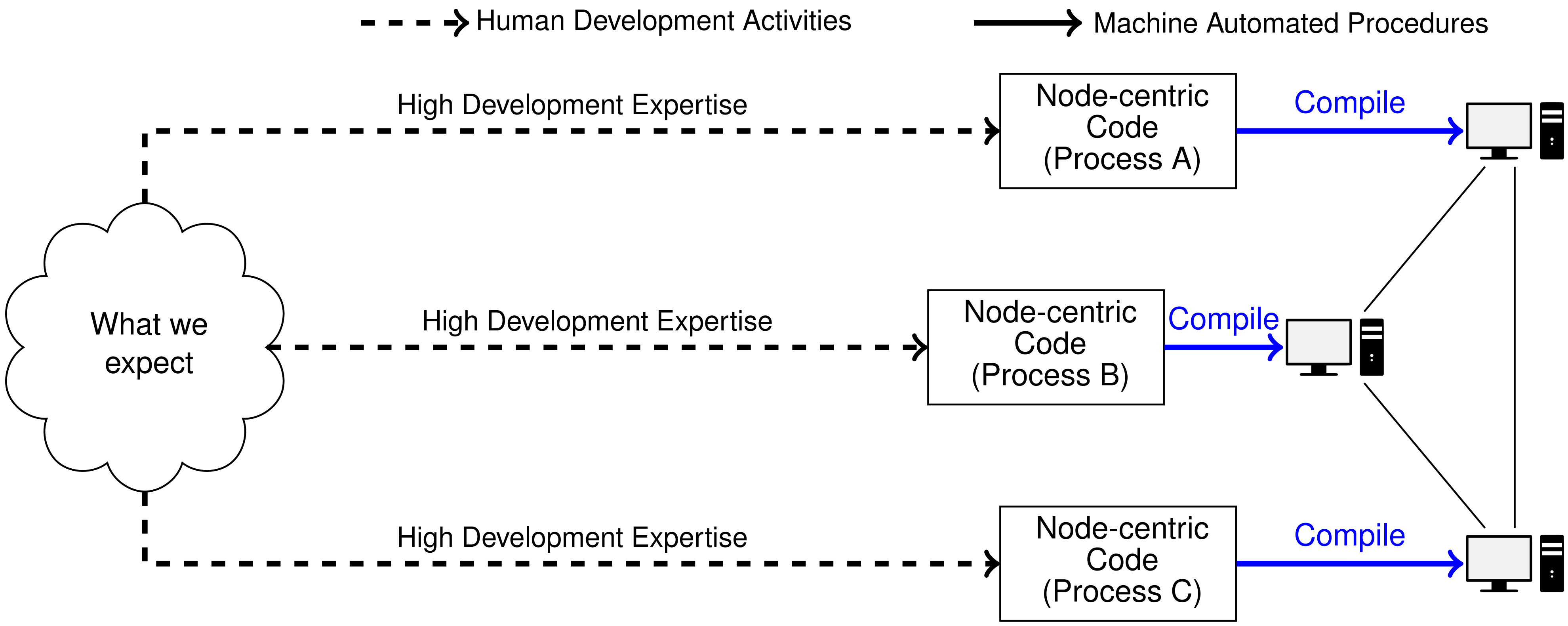
Proceed to writing code for A's dual B (e.g., a producer)

Traditional Concurrent/Distributed Programming



In general, it's not always just producer and consumer

Traditional Concurrent/Distributed Programming

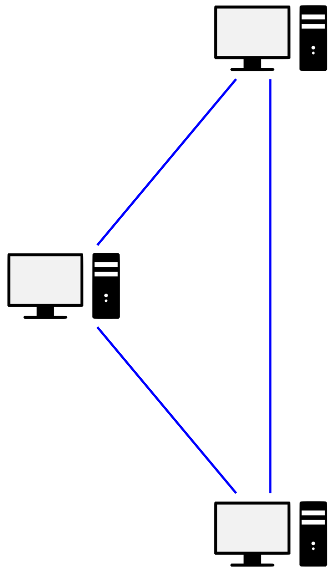


Current practice, but it's costly, tedious and error-prone

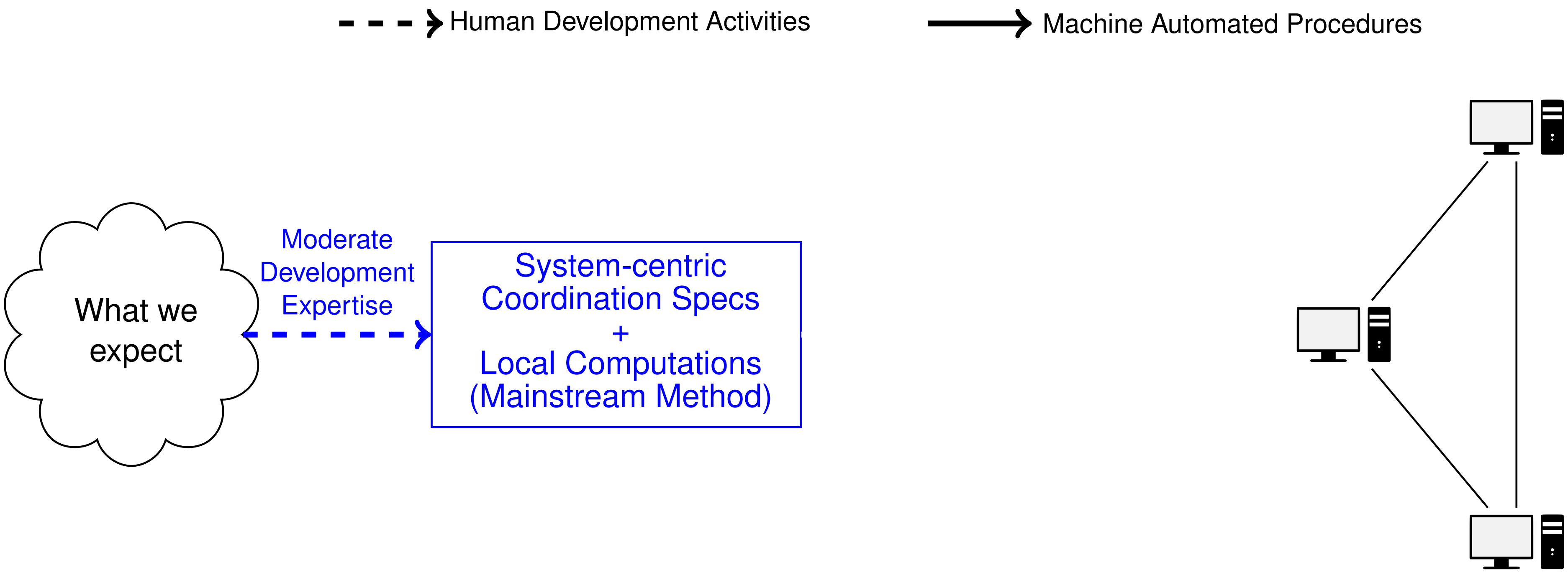
System-Centric Programming

Human Development Activities

Machine Automated Procedures

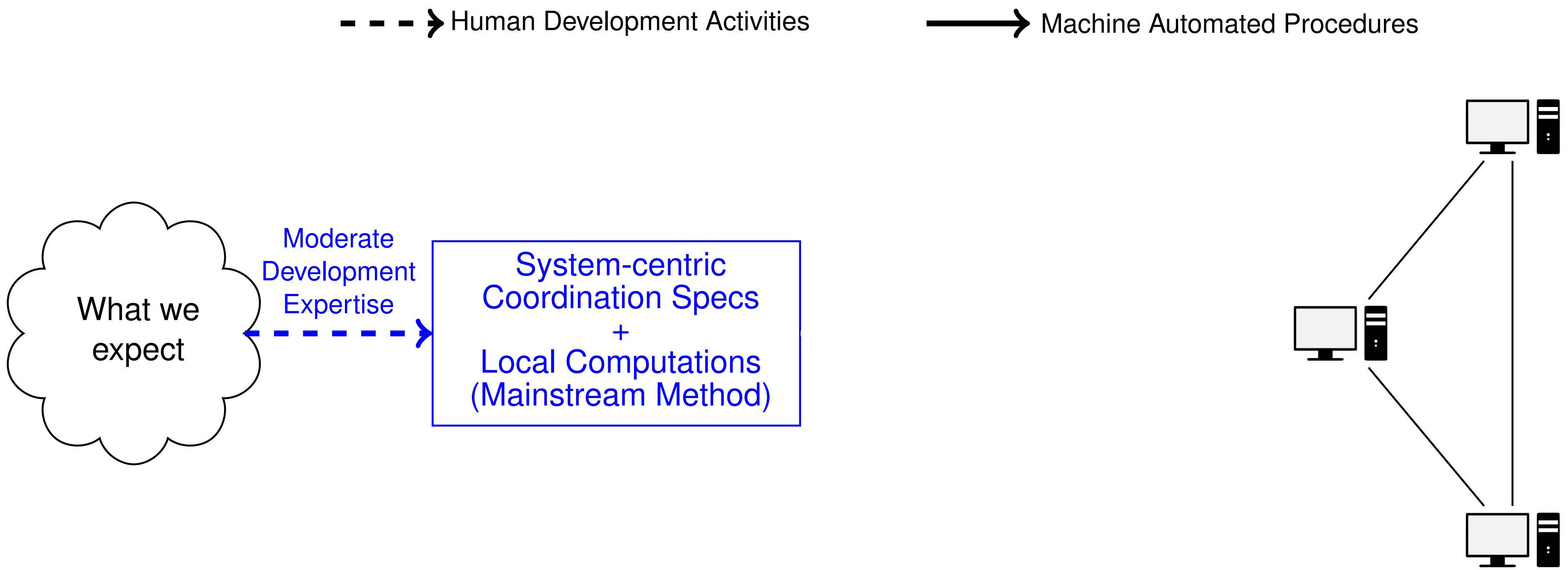


System-Centric Programming



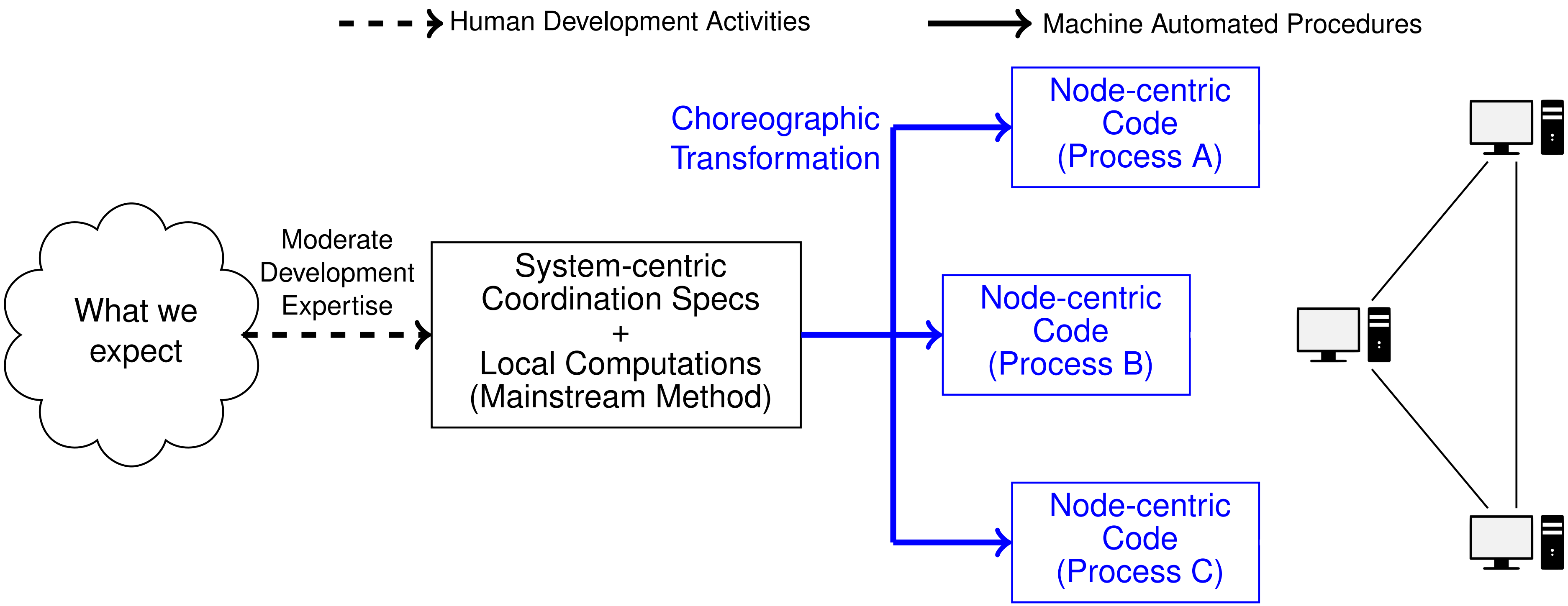
- Write coordination code *as a single entity*

System-Centric Programming



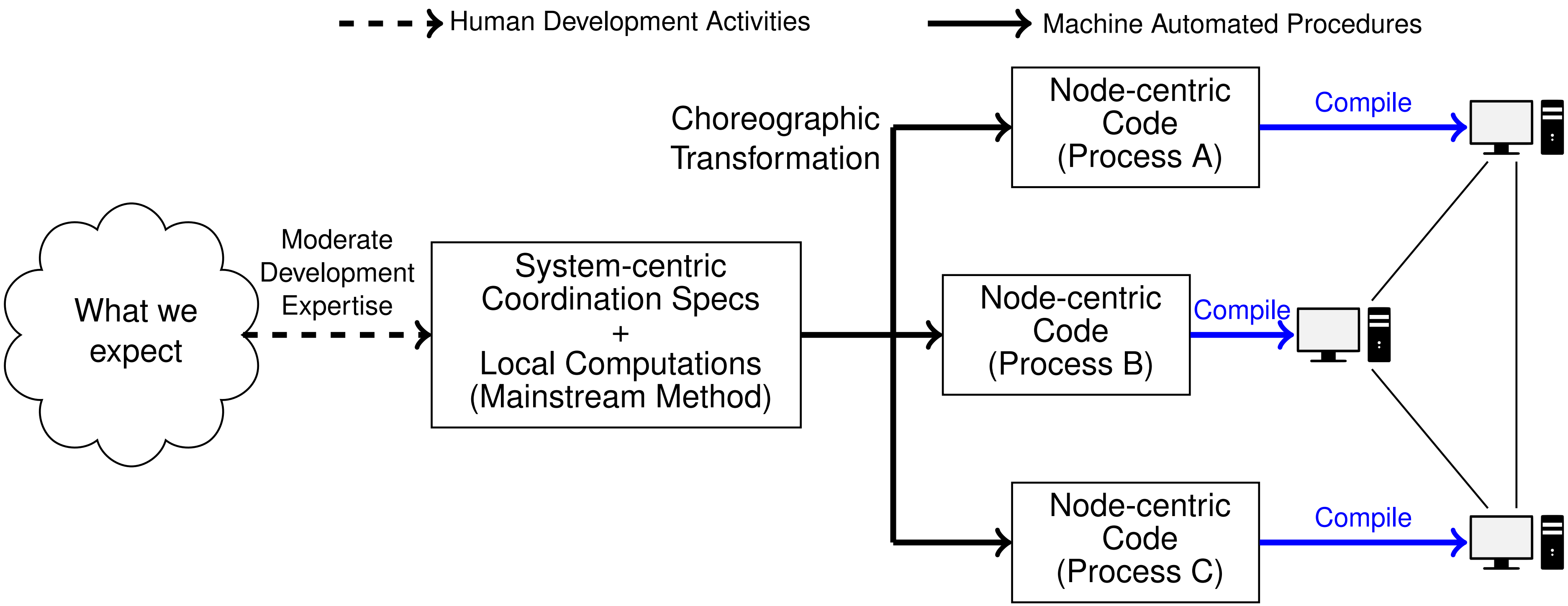
- Write coordination code *as a single entity*
- by combining the best of two worlds
  - Coordination: declarative + concise high-level specifications
  - Local Computation: familiar + rich mainstream programming methodologies

System-Centric Programming



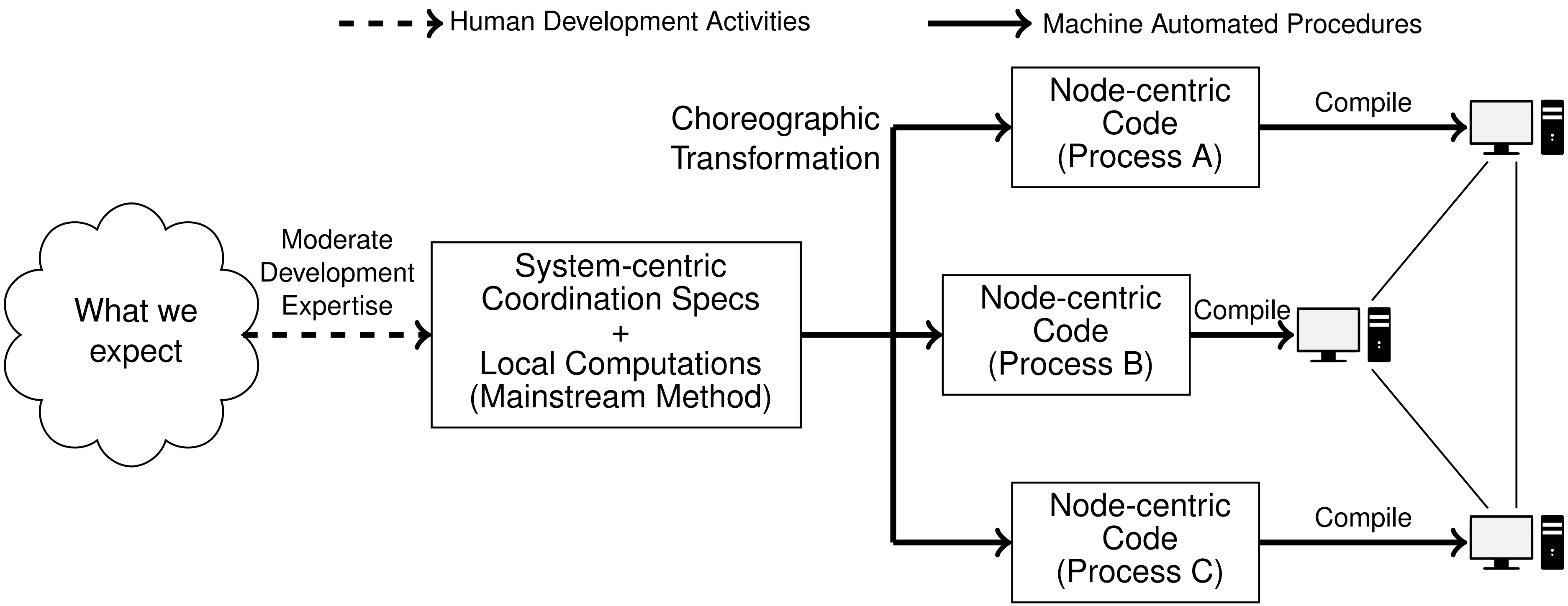
- Choreographic transformation: generate node-centric code . . .

System-Centric Programming



- Choreographic transformation: generate node-centric code . . .
- . . . that can be compiled into machine code

System-Centric Programming



- We focus on *CoMingle*:
  - A high-level rule-based declarative coordination language
  - Facilitates the coordination of mobile ensembles (Java + Android)
  - Great for distributed event-driven applications

Coordinating Ensembles in CoMingle


```
rule init  :: [I]initRace(Ps)
              --o {[A]next(B)|(A,B)<-Cs}, [E]last(), {[P]all(Ps),[P]at(I)|P<-Ps},
                  {[P]renderTrack(Ls),[I]has(P)|P<-Ps} where (Cs,E) = makeChain(I,Ps).

rule start :: [X]all(Ps) \ [X]stRace() --o {[P]release()|P<-Ps}.

rule tap   :: [X]at(Y) \ [X]sendTap() --o [Y]recvTap(X).

rule exit  :: [X]next(Z) \ [X]exit(Y), [Y]at(X) --o [Z]has(Y), [Y]at(Z).

rule win   :: [X]last() \ [X]all(Ps), [X]exit(Y) --o {[P]decWinner(Y)|P<-Ps}.
```

- An example: Orchestrating a race across Android devices

Coordinating Ensembles in CoMingle


```
rule init  :: [I]initRace(Ps)
              --o {[A]next(B)|(A,B)<-Cs}, [E]last(), {[P]all(Ps),[P]at(I)|P<-Ps},
                  {[P]renderTrack(Ls),[I]has(P)|P<-Ps} where (Cs,E) = makeChain(I,Ps).

rule start :: [X]all(Ps) \ [X]stRace() --o {[P]release()|P<-Ps}.

rule tap   :: [X]at(Y) \ [X]sendTap() --o [Y]recvTap(X).

rule exit  :: [X]next(Z) \ [X]exit(Y), [Y]at(X) --o [Z]has(Y), [Y]at(Z).

rule win   :: [X]last() \ [X]all(Ps), [X]exit(Y) --o {[P]decWinner(Y)|P<-Ps}.
```

- CoMingle handles *coordination*
  - Multiset rewriting → coordination
  - Rewrite rules are parametric on computing nodes

Coordinating Ensembles in CoMingle


```
rule init   :: [I]initRace(Ps)
                 --o {[A]next(B)|(A,B)<-Cs}, [E]last(), {[P]all(Ps),[P]at(I)|P<-Ps},
                     {[P]renderTrack(Ls),[I]has(P)|P<-Ps} where (Cs,E) = makeChain(I,Ps).

rule start :: [X]all(Ps) \ [X]stRace() --o {[P]release()|P<-Ps}.

rule tap    :: [X]at(Y) \ [X]sendTap() --o [Y]recvTap(X).

rule exit   :: [X]next(Z) \ [X]exit(Y), [Y]at(X) --o [Z]has(Y), [Y]at(Z).

rule win    :: [X]last() \ [X]all(Ps), [X]exit(Y) --o {[P]decWinner(Y)|P<-Ps}.
```

- System-centric style: program coordination *as a whole*. For instance:
  - rule `exit` (*fully*) implements the “crossing over” of racing sprites
  - coordinates three distinct roles (X, Y and Z) of the same distributed event

Outline

Choreographic Compilation of Multiset Rewrite Rules

In previous works . . .

- For each system-centric rule like:

```
rule exit ::  [X]next(Z) \[X]exit(Y), [Y]at(X) -o
              [Z]has(Y), [Y]at(Z).
```

Choreographic Compilation of Multiset Rewrite Rules

In previous works . . .

- For each system-centric rule like:

```
rule exit ::  [X]next(Z) \[X]exit(Y), [Y]at(X) -o
              [Z]has(Y), [Y]at(Z).
```

- We derived a **node-centric operational interpretation**, by

Choreographic Compilation of Multiset Rewrite Rules

In previous works ...

- For each system-centric rule like:

```
rule exit ::  [X]next(Z) \[X]exit(Y), [Y]at(X) -o
              [Z]has(Y), [Y]at(Z).
```

- We derived a **node-centric operational interpretation**, by
  1. **choreographic compilation**: Transform system-centric rule into a set of local rules (PPDP'13)

Choreographic Compilation of Multiset Rewrite Rules

In previous works . . .

- For each system-centric rule like:

```
rule exit ::  [X]next(Z) \[X]exit(Y), [Y]at(X) -o
              [Z]has(Y), [Y]at(Z).
```

- We derived a **node-centric operational interpretation**, by
  1. **choreographic compilation**: Transform system-centric rule into a set of local rules (PPDP'13)
  2. **rule compilation**: Generate imperative code that implements local rewriting (CHR'14, APLAS'14)

Choreographic Compilation of Multiset Rewrite Rules

In previous works . . .

- For each system-centric rule like:

```
rule exit ::  [X]next(Z) \[X]exit(Y), [Y]at(X) -o
              [Z]has(Y), [Y]at(Z).
```

- We derived a **node-centric operational interpretation**, by
  1. **choreographic compilation**: Transform system-centric rule into a set of local rules (PPDP'13)
  2. **rule compilation**: Generate imperative code that implements local rewriting (CHR'14, APLAS'14)
- **Proof of Concept**: Integration with Android applications (Coordination'15, Wimob'15)

Choreographic Compilation of Multiset Rewrite Rules

In previous works . . .

- For each system-centric rule like:

```
rule exit ::  [X]next(Z) \[X]exit(Y), [Y]at(X) -o
              [Z]has(Y), [Y]at(Z).
```

- We derived a **node-centric operational interpretation**, by
  1. **choreographic compilation**: Transform system-centric rule into a set of local rules (PPDP'13)
  2. **rule compilation**: Generate imperative code that implements local rewriting (CHR'14, APLAS'14)
- **Proof of Concept**: Integration with Android applications (Coordination'15, Wimob'15)
- Choreographic compilation did not handle *multiset comprehensions*

In this Paper . . .

- We extend this choreographic compilation scheme
  - with system-centric comprehension patterns
  - with proofs of soundness and completeness w.r.t. abstract semantics (pure multiset rewriting)

In this Paper . . .

- We extend this choreographic compilation scheme
  - with system-centric comprehension patterns
  - with proofs of soundness and completeness w.r.t. abstract semantics (pure multiset rewriting)
- Comprehensions enable very concise implementations of *distributed aggregates*. E.g.:

$$\forall \left[\begin{array}{c} \wr [X] neighbor(N) \int_{N \rightarrow Ns}, \\ \wr [N] temp(T) \mid N \in Ns \int_{T \rightarrow Ts} \end{array} \right] \setminus [X] getAvg(Y) \multimap [Y] report(A)$$

where $A = sum(Ts)/size(Ts)$

In this Paper . . .

- We extend this choreographic compilation scheme
  - with system-centric comprehension patterns
  - with proofs of soundness and completeness w.r.t. abstract semantics (pure multiset rewriting)
- Comprehensions enable very concise implementations of *distributed aggregates*. E.g.:

$$\forall \left[\begin{array}{c} \wr [X]neighbor(N) \wr_{N \to Ns}, \\ \wr [N]temp(T) \mid N \in Ns \wr_{T \to Ts} \end{array} \right] \setminus [X]getAvg(Y) \multimap [Y]report(A)$$

where $A = sum(Ts)/size(Ts)$

  - Local comprehension: Get all $neighbor(N)$ of X

In this Paper ...

- We extend this choreographic compilation scheme
  - with system-centric comprehension patterns
  - with proofs of soundness and completeness w.r.t. abstract semantics (pure multiset rewriting)
- Comprehensions enable very concise implementations of *distributed aggregates*. E.g.:

$$\forall \left[\begin{array}{c} \wr [X]neighbor(N) \wr_{N \to Ns}, \\ \wr [N]temp(T) \mid N \in Ns \wr_{T \to Ts} \end{array} \right] \setminus [X]getAvg(Y) \multimap [Y]report(A)$$

where $A = sum(Ts)/size(Ts)$

  - Local comprehension: Get all $neighbor(N)$ of X
  - System-centric comprehension: Get all $temp(T)$ from each $N \in Ns$

Outline

Key Challenges: Deriving Node-centric Interpretations

- Deriving node-centric interpretations for system-centric rules:

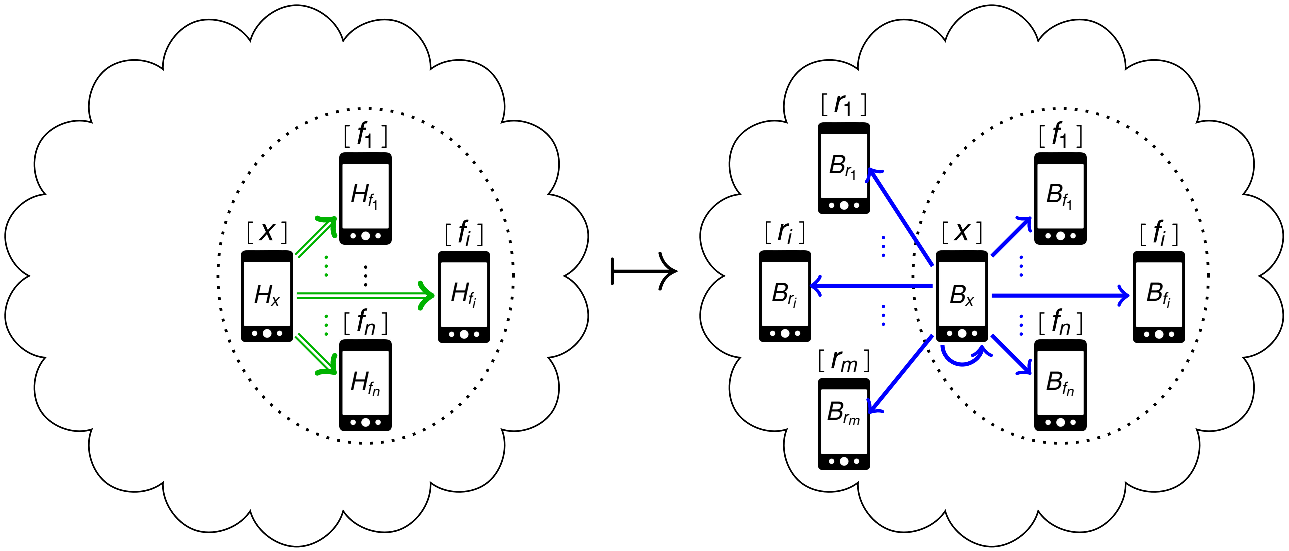
$$
\begin{aligned}
&[\,x\,]\,H_x,\ [\,f_1\,]\,H_{f_1},\ldots,\ [\,f_n\,]\,H_{f_n}\ \mid\ g \\
&\qquad \multimap\ [\,x\,]\,B_x,\ [\,f_1\,]\,B_{f_1},\ldots,\ [\,f_n\,]\,B_{f_n},\ [\,r_1\,]\,B_{r_1},\ldots,\ [\,r_m\,]\,B_{r_m}
\end{aligned}
$$

Key Challenges: Deriving Node-centric Interpretations

- Deriving node-centric interpretations for system-centric rules:

$$[\,x\,]\,H_x,\ [\,f_1\,]\,H_{f_1},\ldots,\ [\,f_n\,]\,H_{f_n}\ \mid\ g$$
$$\multimap\ [\,x\,]\,B_x,\ [\,f_1\,]\,B_{f_1},\ldots,\ [\,f_n\,]\,B_{f_n},\ [\,r_1\,]\,B_{r_1},\ldots,\ [\,r_m\,]\,B_{r_m}$$

- Overview of system-centric rule application:



- System-centric matching $\Rightarrow$: obtaining consensus
- System-centric rewriting $\rightarrow$: delete + message passing

Key Challenges: Deriving Node-centric Interpretations

- Choreographic compilation to the rescue!

```
[X]swap(Y,P), [X]item(N), [Y]item(M) | N<=P, M>=P ⊸ [X]item(M), [Y]item(N)
```

Choreographic transformation (PPDP'13)

```
( [X]swap(Y,P),  )  | N<=P  ⊸  exists E. ( [X]swapLHSX(E,P,N),  )
( [X]item(N)     )                        ( [Y]swapReqY(E,X,P,N) )

( [Y]swapReqY(E,X,P,N), )  | M>=P  ⊸  [X]swapLHSY(E,Y,M)
( [Y]item(M)            )

[Y]swapReqY(E,X,P,N) ⊸ [X]swapAbort(E)

( [X]swapLHSX(E,P,N), ) ⊸ ( [X]item(M), )
( [X]swapLHSY(E,Y,M)  )   ( [Y]item(N)  )

( [X]swapLHSX(E,P,N), ) ⊸ ( [x]swap(Y,P), )
( [X]swapAbort(E)     )   ( [X]item(N)    )
```

Key Challenges: Deriving Node-centric Interpretations

- Choreographic compilation to the rescue!

$$\texttt{[X]swap(Y,P), [X]item(N), [Y]item(M) | N<=P, M>=P} \multimap \texttt{[X]item(M), [Y]item(N)}$$

Choreographic transformation (PPDP'13)

$$\begin{pmatrix} \texttt{[X]swap(Y,P),} \\ \texttt{[X]item(N)} \end{pmatrix} \texttt{ | N<=P} \multimap \texttt{exists E.} \begin{pmatrix} \texttt{[X]swapLHSX(E,P,N),} \\ \texttt{[Y]swapReqY(E,X,P,N)} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{[Y]swapReqY(E,X,P,N),} \\ \texttt{[Y]item(M)} \end{pmatrix} \texttt{ | M>=P} \multimap \texttt{[X]swapLHSY(E,Y,M)}$$

$$\texttt{[Y]swapReqY(E,X,P,N)} \multimap \texttt{[X]swapAbort(E)}$$

$$\begin{pmatrix} \texttt{[X]swapLHSX(E,P,N),} \\ \texttt{[X]swapLHSY(E,Y,M)} \end{pmatrix} \multimap \begin{pmatrix} \texttt{[X]item(M),} \\ \texttt{[Y]item(N)} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{[X]swapLHSX(E,P,N),} \\ \texttt{[X]swapAbort(E)} \end{pmatrix} \multimap \begin{pmatrix} \texttt{[x]swap(Y,P),} \\ \texttt{[X]item(N)} \end{pmatrix}$$

- In this paper, we extend this to handle *comprehension patterns*

Key Challenges: Deriving Node-centric Interpretations

- Choreographic compilation to the rescue!

```
[X]swap(Y,P), [X]item(N), [Y]item(M) | N<=P, M>=P ⊸ [X]item(M), [Y]item(N)
```

Choreographic transformation (PPDP'13)

```
( [X]swap(Y,P),  )  | N<=P  ⊸  exists E. ( [X]swapLHSX(E,P,N),  )
( [X]item(N)     )                       ( [Y]swapReqY(E,X,P,N) )

( [Y]swapReqY(E,X,P,N), )  | M>=P  ⊸  [X]swapLHSY(E,Y,M)
( [Y]item(M)            )

[Y]swapReqY(E,X,P,N) ⊸ [X]swapAbort(E)

( [X]swapLHSX(E,P,N), ) ⊸ ( [X]item(M), )
( [X]swapLHSY(E,Y,M)  )   ( [Y]item(N)  )

( [X]swapLHSX(E,P,N), ) ⊸ ( [x]swap(Y,P), )
( [X]swapAbort(E)     )   ( [X]item(N)    )
```

- In this paper, we extend this to handle *comprehension patterns*
- But multiset rewriting with comprehension patterns are *non-monotonic*!!

Key Challenges: Non-monotonicity + Comprehensions

- Some definitions:
  - Let $\mathcal{P}$ be a program and *St* be a rewrite state
  - $\mathcal{P} \rhd St \ \longmapsto \ St'$ denotes a single rewriting step
  - $\mathcal{P} \rhd St \ \longmapsto^{*} \ St'$ denotes zero or more steps

Key Challenges: Non-monotonicity + Comprehensions

- Some definitions:
  - Let $\mathcal{P}$ be a program and *St* be a rewrite state
  - $\mathcal{P} \rhd St \longmapsto St'$ denotes a single rewriting step
  - $\mathcal{P} \rhd St \longmapsto^* St'$ denotes zero or more steps
- Pure multiset rewriting is *monotonic*:
  - if $\mathcal{P} \rhd St \longmapsto St'$, then $\mathcal{P} \rhd St, St_2 \longmapsto St', St_2$

Key Challenges: Non-monotonicity + Comprehensions

- Some definitions:
  - Let $\mathcal{P}$ be a program and *St* be a rewrite state
  - $\mathcal{P} \rhd St \;\longmapsto\; St'$ denotes a single rewriting step
  - $\mathcal{P} \rhd St \;\longmapsto^*\; St'$ denotes zero or more steps
- Pure multiset rewriting is *monotonic*:
  - if $\mathcal{P} \rhd St \;\longmapsto\; St'$, then $\mathcal{P} \rhd St, St_2 \;\longmapsto\; St', St_2$
- Comprehension patterns introduce *non-monotonicity*:
  - Above does not always hold for comprehension patterns (CHR'14)

Key Challenges: Non-monotonicity + Comprehensions

- Some definitions:
  - Let $\mathcal{P}$ be a program and St be a rewrite state
  - $\mathcal{P} \rhd St \longmapsto St'$ denotes a single rewriting step
  - $\mathcal{P} \rhd St \longmapsto^* St'$ denotes zero or more steps
- Pure multiset rewriting is *monotonic*:
  - if $\mathcal{P} \rhd St \longmapsto St'$, then $\mathcal{P} \rhd St, St_2 \longmapsto St', St_2$
- Comprehension patterns introduce *non-monotonicity*:
  - Above does not always hold for comprehension patterns (CHR'14)
  - A consequence of maximality of comprehensions

Outline

Choreographic Compilation of Comprehensions

- We extend our original choreographic compilation (PPDP’13) to handle comprehension patterns

Choreographic Compilation of Comprehensions

- We extend our original choreographic compilation (PPDP'13) to handle comprehension patterns
- Running example:

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \wr_{N \rightarrow Ns} \\ \wr [Y]data(M) \mid M \geq P \wr_{M \rightarrow Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \wr_{M \leftarrow Ms} \\ \wr [Y]data(N) \wr_{N \leftarrow Ns} \end{bmatrix}$$

- Swapping all *data*(N) at X for all *data*(M) at Y.

Choreographic Compilation of Comprehensions

- We extend our original choreographic compilation (PPDP'13) to handle comprehension patterns
- Running example:

$$\forall \left[\begin{array}{c} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns} \\ \wr [Y]data(M) \mid M \geq P \int_{M \to Ms} \end{array} \right] \multimap \left[\begin{array}{c} \wr [X]data(M) \int_{M \leftarrow Ms} \\ \wr [Y]data(N) \int_{N \leftarrow Ns} \end{array} \right]$$

- Swapping all $data(N)$ at X for all $data(M)$ at Y.
- Choreographic compilation must enforce *atomicity* of comprehension patterns

Choreographic Compilation of Comprehensions

$$\forall \left[\begin{array}{c} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns} \\ \wr [Y]data(M) \mid M \geq P \int_{M \to Ms} \end{array} \right] \multimap \left[\begin{array}{c} \wr [X]data(M) \int_{M \leftarrow Ms} \\ \wr [Y]data(N) \int_{N \leftarrow Ns} \end{array} \right]$$

Choreographic Compilation of Comprehensions

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \wr_{N \to Ns} \\ \wr [Y]data(M) \mid M \geq P \wr_{M \to Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \wr_{M \leftarrow Ms} \\ \wr [Y]data(N) \wr_{N \leftarrow Ns} \end{bmatrix}$$

$\downarrow$ Choreographic Compilation

$$\forall \begin{bmatrix} [X]swap(Y,P), \\ \wr [X]data(N) \mid N \leq P \wr_{N \to Ns}, \\ [X]Free^{data}, \ [X]Next(n) \end{bmatrix} \multimap \begin{bmatrix} [Y]Req^{swp}_{Y}(e,X,Ns,P), \\ [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Trans(e), \ [X]Next(n') \end{bmatrix}$$

where $e = H(X,n)$ and $n' = n+1$.

Local rewriting at X

$$\forall \begin{bmatrix} [Y]okSwap, \quad \wr [Y]data(M) \mid M \geq P \wr_{M \to Ms}, \\ [Y]Free^{data}, \ [Y]Req^{swp}_{Y}(e,X,Ns,P) \end{bmatrix} \multimap \begin{bmatrix} [Y]Trans(e), \\ [X]Ans^{swp}_{Y}(e,Y,Ms) \end{bmatrix}$$

Local rewriting at Y

$$\forall \begin{bmatrix} [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Ans^{swp}_{Y}(e,Y,Ms) \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \wr_{M \leftarrow Ms}, \quad \wr [Y]data(N) \wr_{N \leftarrow Ns}, \\ \wr [I]Free^{data} \wr_{I \leftarrow \wr X,Y \wr}, \ \wr [I]Done(e) \wr_{I \leftarrow \wr X,Y \wr} \end{bmatrix}$$

Commit rewriting

$$\forall \left(\wr [Z]Trans(e) \wr, \ [Z]Done(e) \multimap \emptyset \right)$$

Clean up

$$\forall \left(\wr [Y]Trans(e') \wr_{e' \to es} \backslash \ [Y]Req^{swp}_{Y}(e,X,Ns,P) \mid e \,\dot{\lll}\, es \multimap [X]Abort(e) \right)$$

where $e \,\dot{\lll}\, es$ iff $es = \emptyset$ or for some $e' \in es$ and $e < e'$

Roll back

Locking facts Transaction facts Staging facts

Choreographic Compilation of Comprehensions

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr[X]data(N) \mid N \leq P \int_{N \to Ns} \\ \wr[Y]data(M) \mid M \geq P \int_{M \to Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr[X]data(M) \int_{M \leftarrow Ms} \\ \wr[Y]data(N) \int_{N \leftarrow Ns} \end{bmatrix}$$

$\downarrow$ Choreographic Compilation

$$\forall \begin{bmatrix} [X]swap(Y,P), \\ \wr[X]data(N) \mid N \leq P \int_{N \to Ns}, \\ [X]Free^{data}, \ [X]Next(n) \end{bmatrix} \multimap \begin{bmatrix} [Y]Req^{swp}_{Y}(e, X, Ns, P), \\ [X]Wait^{swp}(e, Y, Ns, P), \\ [X]Trans(e), \ [X]Next(n') \end{bmatrix}$$

where $e = H(X, n)$ and $n' = n + 1$.

Local rewriting at X

$$\forall \begin{bmatrix} [Y]okSwap, \quad \wr[Y]data(M) \mid M \geq P \int_{M \to Ms}, \\ [Y]Free^{data}, \ [Y]Req^{swp}_{Y}(e, X, Ns, P) \end{bmatrix} \multimap \begin{bmatrix} [Y]Trans(e), \\ [X]Ans^{swp}_{Y}(e, Y, Ms) \end{bmatrix}$$

Local rewriting at Y

$$\forall \begin{bmatrix} [X]Wait^{swp}(e, Y, Ns, P), \\ [X]Ans^{swp}_{Y}(e, Y, Ms) \end{bmatrix} \multimap \begin{bmatrix} \wr[X]data(M) \int_{M \leftarrow Ms}, \quad \wr[Y]data(N) \int_{N \leftarrow Ns}, \\ \wr[I]Free^{data} \int_{I \leftarrow \wr X, Y \int}, \ \wr[I]Done(e) \int_{I \leftarrow \wr X, Y \int} \end{bmatrix}$$

Commit rewriting

$$\forall \left(\wr[Z]Trans(e) \int, \ [Z]Done(e) \multimap \varnothing \right)$$

Clean up

$$\forall \left(\wr[Y]Trans(e') \int_{e' \to es} \setminus [Y]Req^{swp}_{Y}(e, X, Ns, P) \mid e \mathrel{\dot{\lll}} es \multimap [X]Abort(e) \right)$$

where $e \mathrel{\dot{\lll}} es$ iff $es = \varnothing$ or for some $e' \in es$ and $e < e'$

Roll back

Locking facts Transaction facts Staging facts

Choreographic Compilation of Comprehensions

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns} \\ \wr [Y]data(M) \mid M \geq P \int_{M \to Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \int_{M \leftarrow Ms} \\ \wr [Y]data(N) \int_{N \leftarrow Ns} \end{bmatrix}$$

$\downarrow$ Choreographic Compilation

$$\forall \begin{bmatrix} [X]swap(Y,P), \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns}, \\ [X]Free^{data}, \ [X]Next(n) \end{bmatrix} \multimap \begin{bmatrix} [Y]Req_Y^{swp}(e,X,Ns,P), \\ [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Trans(e), \ [X]Next(n') \end{bmatrix}$$

where $e = H(X,n)$ and $n' = n+1$.

Local rewriting at X

$$\forall \begin{bmatrix} [Y]okSwap, \quad \wr [Y]data(M) \mid M \geq P \int_{M \to Ms}, \\ [Y]Free^{data}, \ [Y]Req_Y^{swp}(e,X,Ns,P) \end{bmatrix} \multimap \begin{bmatrix} [Y]Trans(e), \\ [X]Ans_Y^{swp}(e,Y,Ms) \end{bmatrix}$$

Local rewriting at Y

$$\forall \begin{bmatrix} [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Ans_Y^{swp}(e,Y,Ms) \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \int_{M \leftarrow Ms}, \quad \wr [Y]data(N) \int_{N \leftarrow Ns}, \\ \wr [I]Free^{data} \int_{I \leftarrow \wr X,Y \int}, \ \wr [I]Done(e) \int_{I \leftarrow \wr X,Y \int} \end{bmatrix}$$

Commit rewriting

$$\forall \left(\wr [Z]Trans(e) \int, \ [Z]Done(e) \multimap \varnothing \right)$$

Clean up

$$\forall \left(\wr [Y]Trans(e') \int_{e' \to es} \backslash \ [Y]Req_Y^{swp}(e,X,Ns,P) \mid e \mathrel{\dot{\lll}} es \multimap [X]Abort(e) \right)$$

where $e \mathrel{\dot{\lll}} es$ iff $es = \varnothing$ or for some $e' \in es$ and $e < e'$

Roll back

Locking facts — Transaction facts — Staging facts

Choreographic Compilation of Comprehensions

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr[X]data(N) \mid N \leq P \int_{N\to Ns} \\ \wr[Y]data(M) \mid M \geq P \int_{M\to Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr[X]data(M)\int_{M\leftarrow Ms} \\ \wr[Y]data(N)\int_{N\leftarrow Ns} \end{bmatrix}$$

$\downarrow$ Choreographic Compilation

$$\forall \begin{bmatrix} [X]swap(Y,P), \\ \wr[X]data(N) \mid N \leq P\int_{N\to Ns}, \\ [X]Free^{data}, \ [X]Next(n) \end{bmatrix} \multimap \begin{bmatrix} [Y]Req^{swp}_{Y}(e,X,Ns,P), \\ [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Trans(e), \ [X]Next(n') \end{bmatrix}$$

where $e = H(X,n)$ and $n' = n+1$.

Local rewriting at X

$$\forall \begin{bmatrix} [Y]okSwap, \quad \wr[Y]data(M) \mid M \geq P\int_{M\to Ms}, \\ [Y]Free^{data}, \ [Y]Req^{swp}_{Y}(e,X,Ns,P) \end{bmatrix} \multimap \begin{bmatrix} [Y]Trans(e), \\ [X]Ans^{swp}_{Y}(e,Y,Ms) \end{bmatrix}$$

Local rewriting at Y

$$\forall \begin{bmatrix} [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Ans^{swp}_{Y}(e,Y,Ms) \end{bmatrix} \multimap \begin{bmatrix} \wr[X]data(M)\int_{M\leftarrow Ms}, \quad \wr[Y]data(N)\int_{N\leftarrow Ns}, \\ \wr[I]Free^{data}\int_{I\leftarrow \wr X,Y\int}, \ \wr[I]Done(e)\int_{I\leftarrow \wr X,Y\int} \end{bmatrix}$$

Commit rewriting

$$\forall \left(\wr[Z]Trans(e)\int, \ [Z]Done(e) \multimap \varnothing \right)$$

Clean up

$$\forall \left(\wr[Y]Trans(e')\int_{e'\to es} \setminus [Y]Req^{swp}_{Y}(e,X,Ns,P) \mid e \dot{\lll} es \multimap [X]Abort(e) \right)$$

where $e \dot{\lll} es$ iff $es = \varnothing$ or for some $e' \in es$ and $e < e'$

Roll back

Locking facts Transaction facts Staging facts

Choreographic Compilation of Comprehensions

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns} \\ \wr [Y]data(M) \mid M \geq P \int_{M \to Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \int_{M \leftarrow Ms} \\ \wr [Y]data(N) \int_{N \leftarrow Ns} \end{bmatrix}$$

$\downarrow$ Choreographic Compilation

$$\forall \begin{bmatrix} [X]swap(Y,P), \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns}, \\ [X]Free^{data}, \; [X]Next(n) \end{bmatrix} \multimap \begin{bmatrix} [Y]Req_Y^{swp}(e,X,Ns,P), \\ [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Trans(e), \; [X]Next(n') \end{bmatrix}$$

where $e = H(X,n)$ and $n' = n+1$.

Local rewriting at X

$$\forall \begin{bmatrix} [Y]okSwap, \quad \wr [Y]data(M) \mid M \geq P \int_{M \to Ms}, \\ [Y]Free^{data}, \; [Y]Req_Y^{swp}(e,X,Ns,P) \end{bmatrix} \multimap \begin{bmatrix} [Y]Trans(e), \\ [X]Ans_Y^{swp}(e,Y,Ms) \end{bmatrix}$$

Local rewriting at Y

$$\forall \begin{bmatrix} [X]Wait^{swp}(e,Y,Ns,P), \\ [X]Ans_Y^{swp}(e,Y,Ms) \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \int_{M \leftarrow Ms}, \quad \wr [Y]data(N) \int_{N \leftarrow Ns}, \\ \wr [I]Free^{data} \int_{I \leftarrow \wr X,Y \int}, \; \wr [I]Done(e) \int_{I \leftarrow \wr X,Y \int} \end{bmatrix}$$

Commit rewriting

$$\forall \left(\wr [Z]Trans(e) \int, \; [Z]Done(e) \multimap \varnothing \right)$$

Clean up

$$\forall \left(\wr [Y]Trans(e') \int_{e' \to es} \backslash \; [Y]Req_Y^{swp}(e,X,Ns,P) \mid e \dot{\lll} es \multimap [X]Abort(e) \right)$$

where $e \dot{\lll} es$ iff $es = \varnothing$ or for some $e' \in es$ and $e < e'$

Roll back

Locking facts — Transaction facts — Staging facts

Choreographic Compilation of Comprehensions

$$\forall \begin{bmatrix} [X]swap(Y,P), \quad [Y]okSwap \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns} \\ \wr [Y]data(M) \mid M \geq P \int_{M \to Ms} \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \int_{M \leftarrow Ms} \\ \wr [Y]data(N) \int_{N \leftarrow Ns} \end{bmatrix}$$

$\downarrow$ Choreographic Compilation

$$\forall \begin{bmatrix} [X]swap(Y,P), \\ \wr [X]data(N) \mid N \leq P \int_{N \to Ns}, \\ [X]Free^{data} \; , \; [X]Next(n) \end{bmatrix} \multimap \begin{bmatrix} [Y]Req^{swp}_{Y}(e, X, Ns, P) \; , \\ [X]Wait^{swp}(e, Y, Ns, P) \; , \\ [X]Trans(e) \; , \; [X]Next(n') \end{bmatrix}$$

where $e = H(X, n)$ and $n' = n + 1$.

Local rewriting at X

$$\forall \begin{bmatrix} [Y]okSwap, \quad \wr [Y]data(M) \mid M \geq P \int_{M \to Ms}, \\ [Y]Free^{data} \; , \; [Y]Req^{swp}_{Y}(e, X, Ns, P) \end{bmatrix} \multimap \begin{bmatrix} [Y]Trans(e) \; , \\ [X]Ans^{swp}_{Y}(e, Y, Ms) \end{bmatrix}$$

Local rewriting at Y

$$\forall \begin{bmatrix} [X]Wait^{swp}(e, Y, Ns, P) \; , \\ [X]Ans^{swp}_{Y}(e, Y, Ms) \end{bmatrix} \multimap \begin{bmatrix} \wr [X]data(M) \int_{M \leftarrow Ms}, \quad \wr [Y]data(N) \int_{N \leftarrow Ns}, \\ \wr [I]Free^{data} \int_{I \leftarrow \wr X, Y \int} \; , \; \wr [I]Done(e) \int_{I \leftarrow \wr X, Y \int} \end{bmatrix}$$

Commit rewriting

$$\forall \left(\wr [Z]Trans(e) \int \; , \; [Z]Done(e) \multimap \varnothing \right)$$

Clean up

$$\forall \left(\wr [Y]Trans(e') \int_{e' \to es} \setminus [Y]Req^{swp}_{Y}(e, X, Ns, P) \mid e \lll es \multimap [X]Abort(e) \right)$$

where $e \lll es$ iff $es = \varnothing$ or for some $e' \in es$ and $e < e'$

Roll back

Locking facts Transaction facts Staging facts

Choreographic Compilation of Comprehensions

In General.

- *Locking facts* $\rightarrow$ enforce maximality of comprehensions

Choreographic Compilation of Comprehensions

In General.

- *Locking facts* $\rightarrow$ enforce maximality of comprehensions
- *Transaction facts* $\rightarrow$ manage system-centric rule execution attempts

Choreographic Compilation of Comprehensions

In General.

- *Locking facts* $\rightarrow$ enforce maximality of comprehensions
- *Transaction facts* $\rightarrow$ manage system-centric rule execution attempts
- *Staging facts* $\rightarrow$ record stage of the consensus process

Choreographic Compilation of Comprehensions

In General.

- *Locking facts* $\rightarrow$ enforce maximality of comprehensions
- *Transaction facts* $\rightarrow$ manage system-centric rule execution attempts
- *Staging facts* $\rightarrow$ record stage of the consensus process
- See paper for generalized choreographic compilation scheme

$$
\begin{array}{l}
[\,x\,]\,H_x,\ [\,f_1\,]\,H_{f_1},\ldots,\ [\,f_n\,]\,H_{f_n} \mid g \\
\qquad \multimap\ [\,x\,]\,B_x,\ [\,f_1\,]\,B_{f_1},\ldots,\ [\,f_n\,]\,B_{f_n},\ [\,r_1\,]\,B_{r_1},\ldots,\ [\,r_m\,]\,B_{r_m}
\end{array}
$$

Outline

Formal Results

Theorem (Progress)

If $[\![\mathcal{P}]\!] \triangleright [\![St]\!]^{\mathcal{P}} \longmapsto^{*} Et$, *then* $[\![\mathcal{P}]\!] \triangleright Et \longmapsto^{*} Et'$ *for some* Et' *such that* $\mathit{obligations}(\mathcal{P}, Et') = \varnothing$.

Theorem (Soundness)

If $[\![\mathcal{P}]\!] \triangleright [\![St]\!]^{\mathcal{P}} \longmapsto^{*} Et$, *then* $\mathcal{P} \triangleright St \longmapsto^{*} \lfloor\!\lfloor Et \rfloor\!\rfloor^{\mathcal{P}}$

Theorem (Completeness)

If $\mathcal{P} \triangleright St \longmapsto^{*} St'$, *then* $[\![\mathcal{P}]\!] \triangleright [\![St]\!]^{\mathcal{P}} \longmapsto^{*} Et'$ *for some* Et' *such that* $\lfloor\!\lfloor Et' \rfloor\!\rfloor^{\mathcal{P}} = St'$.

Implementation

- Choreographic compilation discussed in this paper is implemented in our prototype system
- Available for download at:

$$https://github.com/sllam/comingle$$

- See (Coordination'15) and (WiMob'15) for proof-of-concept applications that utilize system-centric comprehension patterns

Outline

Conclusion

- CoMingle:
  - High-level rule-based language
  - Parametric on computing nodes + comprehension patterns
  - For coordinating ensembles of computing nodes

Conclusion

- CoMingle:
    - High-level rule-based language
    - Parametric on computing nodes + comprehension patterns
    - For coordinating ensembles of computing nodes
- Contributions:
    - Choreographic compilation of comprehensions
    - Progress + Soundness + Completeness
    - Prototype Implementation + Proof of Concepts

Thank you!

Questions?