Decentralized Execution of Constraint Handling Rules for Ensembles

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

Outline

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:

$$\overbrace{r}^{\text{Rule Name}} \ @ \ \underbrace{P}_{\text{Propagated Heads}} \ \backslash \ \overbrace{S}^{\text{Simplified Heads}} \iff \underbrace{G}_{\text{Guards}} \mid \overbrace{B}^{\text{Body}}$$

- CHR rules are applied to a multiset of constraints, known as the *constraint store* St.
- Informally: If we have $P \uplus S$ in St, such that G is true, then replace S in St with B.
- Short forms:
  - If P is empty, we write $r \ @ \ S \iff G \mid B$.
  - If S is empty, we write $r \ @ \ P \Longrightarrow G \mid B$.

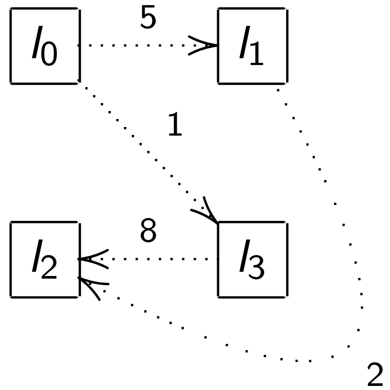
Constraint Handling Rules (CHR), Traditionally

- Example of CHR Program: Find all shortest path

$$base \;@\; edge(X, Y, D) \Longrightarrow \underline{path(X, Y, D)}$$

$$elim \;@\; path(X, Y, D1) \setminus path(X, Y, D2) \Longleftrightarrow D1 \leq D2 \mid true$$

$$trans \;@\; edge(X, Y, D1), path(Y, Z, D2) \Longrightarrow X \neq Z \mid \underline{path(X, Z, D1 + D2)}$$



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

$$\{\!\{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8)\}\!\}$$

$$\rightarrowtail_{base} \{\!\{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), \underline{p(l_0, l_1, 5)}\}\!\}$$

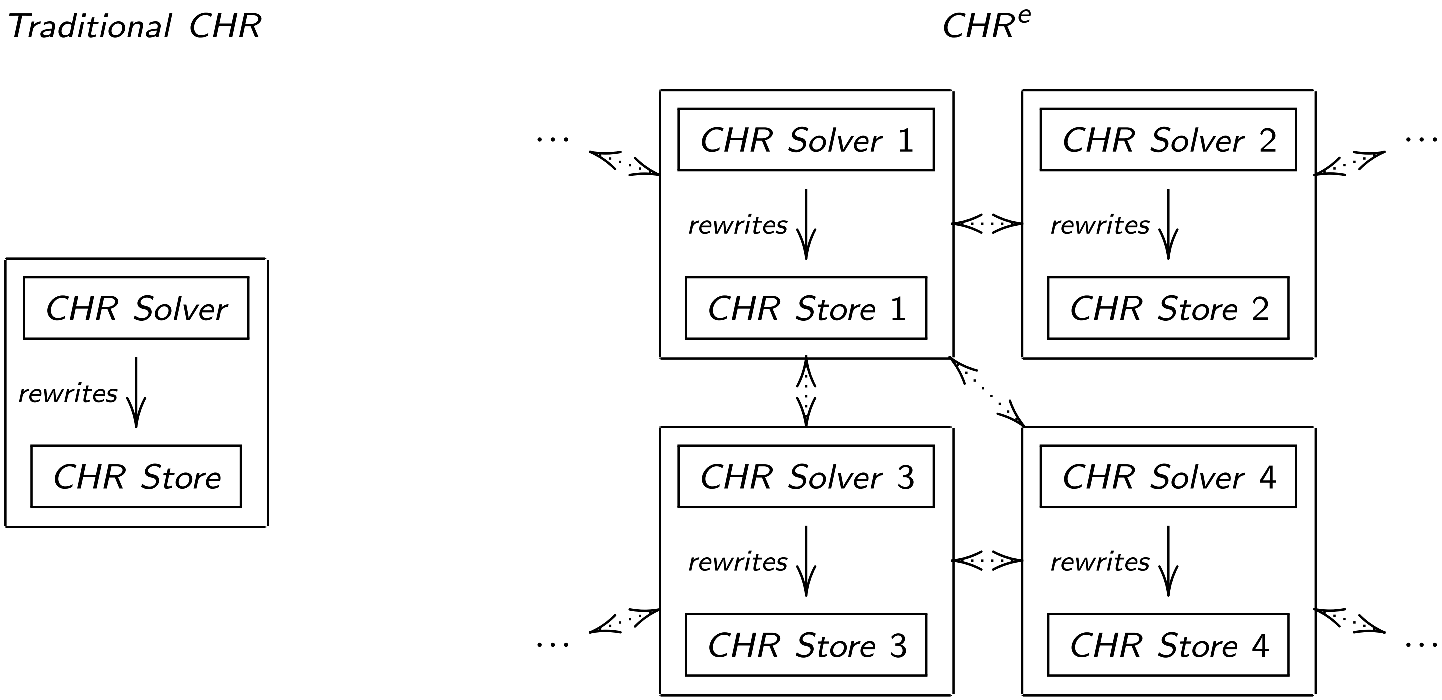
$$\rightarrowtail^{*}_{baseX3} \{\!\{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), \underline{p(l_0, l_3, 1)}, \underline{p(l_1, l_2, 2)}, \underline{p(l_3, l_2, 8)}\}\!\}$$

$$\rightarrowtail_{trans} \{\!\{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8), \underline{p(l_0, l_2, 7)}\}\!\}$$

$$\rightarrowtail_{trans} \{\!\{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8), p(l_0, l_2, 7), \underline{p(l_0, l_2, 9)}\}\!\}$$

$$\rightarrowtail_{elim} \{\!\{e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_2, 2), e(l_3, l_2, 8), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_2, 2), p(l_3, l_2, 8), p(l_0, l_2, 7)\}\!\}$$

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)

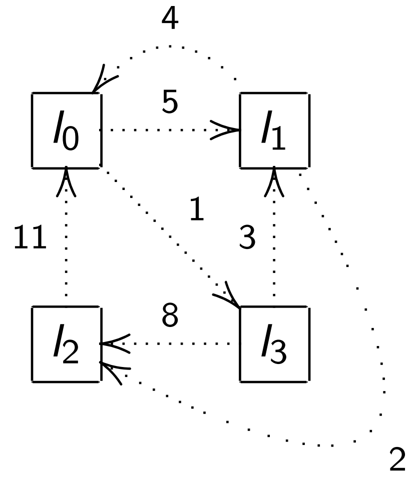
Outline

Recalling "Find Shortest Path" in CHR

$$base\ @\ edge(X, Y, D) \Longrightarrow path(X, Y, D)$$

$$elim\ @\ path(X, Y, D1) \setminus path(X, Y, D2) \Longleftrightarrow D1 \leq D2 \mid true$$

$$trans\ @\ edge(X, Y, D1), path(Y, Z, D2) \Longrightarrow X \neq Z \mid path(X, Z, D1 + D2)$$



$$\lbag e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3) \rbag$$

$$\rightarrowtail^* \lbag e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_0, 4), p(l_1, l_2, 2), p(l_2, l_0, 11), p(l_3, l_2, 8), p(l_3, l_1, 3) \rbag$$

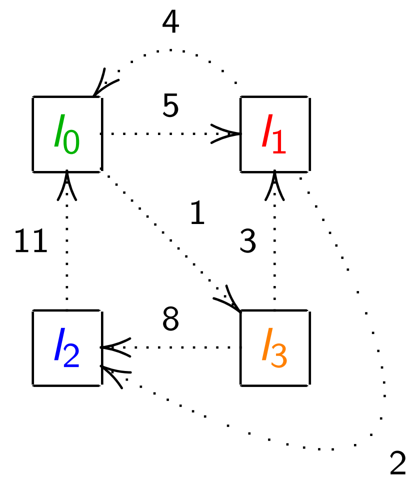
$$\rightarrowtail^* \lbag e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3), p(l_0, l_1, 5), p(l_0, l_3, 1), p(l_1, l_0, 4), p(l_1, l_2, 2), p(l_2, l_0, 11), p(l_3, l_2, 8), p(l_3, l_1, 3), p(l_0, l_2, 9), p(l_0, l_2, 7), p(l_1, l_0, 13), p(l_1, l_3, 5), p(l_2, l_1, 16), p(l_2, l_3, 12), p(l_3, l_0, 19), p(l_3, l_0, 7), p(l_3, l_2, 5) \rbag$$

$$\ldots$$

$$\rightarrowtail^* \lbag e(l_0, l_1, 5), e(l_0, l_3, 1), e(l_1, l_0, 4), e(l_1, l_2, 2), e(l_2, l_0, 11), e(l_3, l_2, 8), e(l_3, l_1, 3), p(l_0, l_1, 4), p(l_0, l_2, 6), p(l_0, l_3, 1), p(l_1, l_0, 4), p(l_1, l_2, 2), p(l_1, l_3, 5), p(l_2, l_0, 11), p(l_2, l_1, 15), p(l_2, l_3, 12), p(l_3, l_0, 19), p(l_3, l_1, 3), p(l_3, l_2, 5) \rbag$$

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

Explicit Location Annotations



$$base \ @ \ [\underline{X}]edge(Y, D) \Longrightarrow [\underline{X}]path(Y, D)$$

$$elim \ @ \ [\underline{X}]path(Y, D1) \setminus [\underline{X}]path(Y, D2) \Longleftrightarrow D1 \leq D2 \mid true$$

$$trans \ @ \ [\underline{X}]edge(Y, D1), [\underline{Y}]path(Z, D2) \Longrightarrow X \neq Z \mid [\underline{X}]path(Z, D1 + D2)$$

Location l_0	Location l_1	Location l_2	Location l_3
$\{e(l_1, 5), e(l_3, 1)\}$	$\{e(l_0, 4), e(l_2, 2)\}$	$\{e(l_0, 11)\}$	$\{e(l_2, 8), e(l_1, 3)\}$
$\rightarrowtail^* \{e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1)\}$	$\rightarrowtail^* \{e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2)\}$	$\rightarrowtail^* \{e(l_0, 11), p(l_0, 11)\}$	$\rightarrowtail^* \{e(l_2, 8), e(l_1, 3), p(l_2, 8), p(l_1, 3)\}$
$\rightarrowtail^* \{e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1), p(l_2, 9), p(l_2, 7)\}$	$\rightarrowtail^* \{e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_0, 13), p(l_3, 5)\}$	...	...
...	...	$\rightarrowtail^* \{e(l_0, 11), p(l_0, 11), p(l_1, 15), p(l_3, 12)\}$	$\rightarrowtail^* \{e(l_2, 8), e(l_1, 3), p(l_0, 19), p(l_1, 3), p(l_2, 5)\}$
$\rightarrowtail^* \{e(l_1, 5), e(l_3, 1), p(l_1, 4), p(l_2, 6), p(l_3, 1)\}$	$\rightarrowtail^* \{e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_3, 5)\}$		

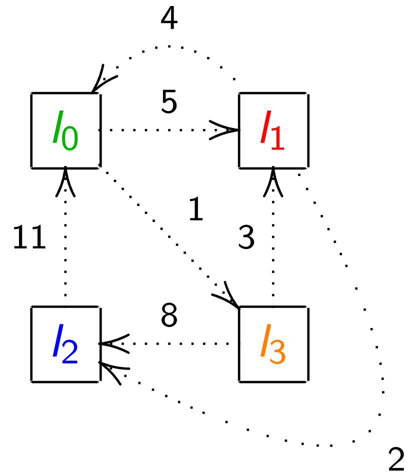
How to execute in decentralized manner, yet preserve soundness (w.r.t abstract CHR semantics)?

Multiset Matching in CHR^e Rules

$base$ @ $[\underline{X}]edge(Y, D) \Longrightarrow [\underline{X}]path(Y, D)$

$elim$ @ $[\underline{X}]path(Y, D1) \setminus [\underline{X}]path(Y, D2) \Longleftrightarrow D1 \leq D2 \mid true$

$trans$ @ $[\underline{X}]edge(Y, D1), [\underline{Y}]path(Z, D2) \Longrightarrow X \neq Z \mid [\underline{X}]path(Z, D1 + D2)$



Location l_0

$\lbag e(l_1, 5), e(l_3, 1) \rbag$

$\rightarrowtail^* \lbag e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1) \rbag$

$\rightarrowtail^* \lbag e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1), p(l_2, 9), p(l_2, 7) \rbag$

...

$\rightarrowtail^* \lbag e(l_1, 5), e(l_3, 1), p(l_1, 4), p(l_2, 6), p(l_3, 1) \rbag$

Location l_1

$\lbag e(l_0, 4), e(l_2, 2) \rbag$

$\rightarrowtail^* \lbag e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2) \rbag$

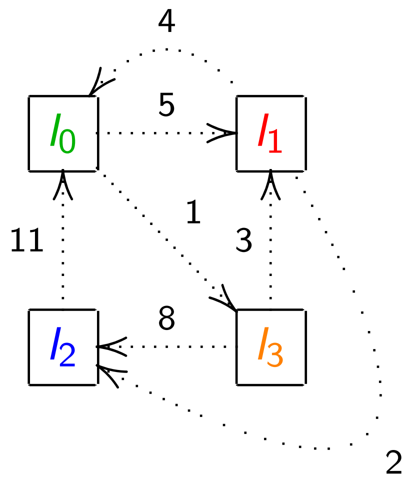
$\rightarrowtail^* \lbag e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_0, 13), p(l_3, 5) \rbag$

...

$\rightarrowtail^* \lbag e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_3, 5) \rbag$

Stark difference between *base* and *elim* rules, and the *trans* rule.

Localized Multiset Matching



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

elim @ $\boxed{[X]path(Y, D1)} \setminus \boxed{[X]path(Y, D2)} \Longleftrightarrow D1 \leq D2 \mid true$

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

Location $\boxed{l_0}$

$\wr e(l_1, 5), e(l_3, 1) \int$

$\rightarrowtail^* \quad \wr e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1) \int$

$\rightarrowtail^* \quad \wr e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1), \boxed{p(l_2, 9)}, \boxed{p(l_2, 7)} \int$

...

$\rightarrowtail^* \quad \wr e(l_1, 5), e(l_3, 1), p(l_1, 4), p(l_2, 6), p(l_3, 1) \int$

Location l_1

$\wr e(l_0, 4), e(l_2, 2) \int$

$\rightarrowtail^* \quad \wr e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2) \int$

$\rightarrowtail^* \quad \wr e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_0, 13), p(l_3, 5) \int$

...

$\rightarrowtail^* \quad \wr e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_3, 5) \int$

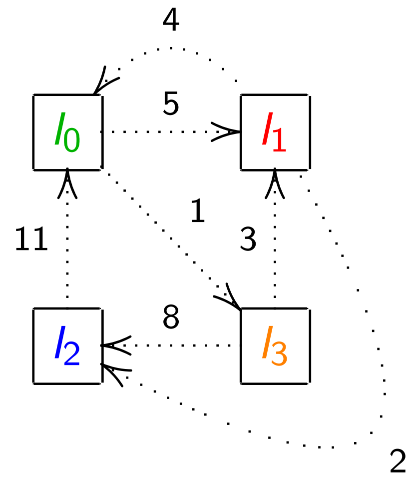
Rule-heads of *elim* rule are 'localized'. Same for *base* rule.

Distributed Multiset Matching

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

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

$trans\ @\ \boxed{[X]edge(Y, D1)}, \boxed{[Y]path(Z, D2)} \Longrightarrow X \neq Z \mid \underline{[X]path(Z, D1 + D2)}$



Location $\boxed{l_0}$

$\{\!| e(l_1, 5), e(l_3, 1) |\!\}$

$\rightarrowtail^* \{\!| \boxed{e(l_1, 5)}, e(l_3, 1), p(l_1, 5), p(l_3, 1) |\!\}$

$\rightarrowtail^* \{\!| e(l_1, 5), e(l_3, 1), p(l_1, 5), p(l_3, 1), p(l_2, 9), \underline{p(l_2, 7)} |\!\}$

...

$\rightarrowtail^* \{\!| e(l_1, 5), e(l_3, 1), p(l_1, 4), p(l_2, 6), p(l_3, 1) |\!\}$

Location $\boxed{l_1}$

$\{\!| e(l_0, 4), e(l_2, 2) |\!\}$

$\rightarrowtail^* \{\!| e(l_0, 4), e(l_2, 2), p(l_0, 4), \boxed{p(l_2, 2)} |\!\}$

$\rightarrowtail^* \{\!| e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_0, 13), p(l_3, 5) |\!\}$

...

$\rightarrowtail^* \{\!| e(l_0, 4), e(l_2, 2), p(l_0, 4), p(l_2, 2), p(l_3, 5) |\!\}$

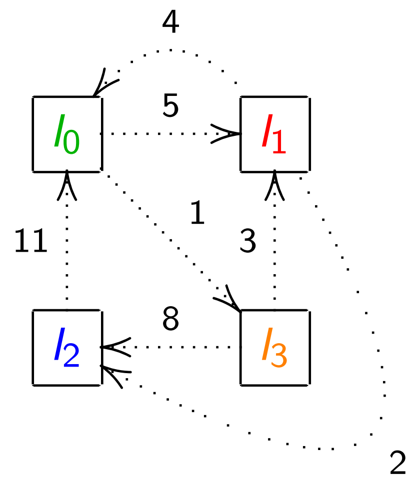
Rule-heads of *trans* rule are partially found in l_0 and l_1!

Topology and Rule Heads

$$base\ @\ [\underline{X}]edge(Y, D) \Longrightarrow [\underline{X}]path(Y, D)$$

$$elim\ @\ [\underline{X}]path(Y, D1) \setminus [\underline{X}]path(Y, D2) \Longleftrightarrow D1 \leq D2 \mid true$$

$$trans\ @\ [\underline{X}]edge(Y, D1), [\underline{Y}]path(Z, D2) \Longrightarrow X \neq Z \mid [\underline{X}]path(Z, D1 + D2)$$



- 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

$$base\ @\ \boxed{[X]}\, edge(Y, D) \Longrightarrow [X]path(Y, D) \qquad \boxed{0 - neighbor}$$

$$elim\ @\ \boxed{[X]}\, path(Y, D1) \setminus \boxed{[X]}\, path(Y, D2) \Longleftrightarrow D1 \leq D2 \mid true \qquad \boxed{0 - neighbor}$$

$$trans\ @\ \boxed{[X]}\, edge(Y, D1), \boxed{[Y]}\, path(Z, D2) \Longrightarrow X \neq Z \mid [X]path(Z, D1 + D2) \qquad \boxed{1 - neighbor}$$

- 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

Directly connected

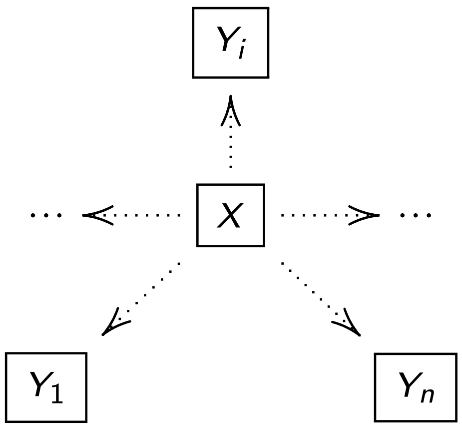
$$trans\ @\ \underbrace{\boxed{[X]}\,edge({\color{red}Y}, D1),}_{X\ \ match\ \ obligation}\ \underbrace{\boxed{[{\color{red}Y}]}\,path(Z, D2)}_{Y\ \ match\ \ obligation} \Longrightarrow X \neq Z \mid [X]path(Z, D1 + D2)$$

- 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

$$r\ @\ \overbrace{[X]P_x}^{\substack{X \text{ propagated} \\ \text{match obligation}}}\ ,\ \overbrace{\left(\biguplus_{i\in\mathcal{I}_n}[Y_i]P_i\right)}^{\substack{Y_i \text{ propagated} \\ \text{match obligation}}}\ \backslash\ \overbrace{[X]S_x}^{\substack{X \text{ simplified} \\ \text{match obligation}}}\ ,\ \overbrace{\left(\biguplus_{i\in\mathcal{I}_n}[Y_i]S_i\right)}^{\substack{Y_i \text{ simplified} \\ \text{match obligation}}} \iff G \mid B$$

- 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., $\forall i \in \mathcal{I}_n.\ Y_i \in FV(S_x \uplus P_x)$)
- We call X the *primary* location.
- "Star" Topology:



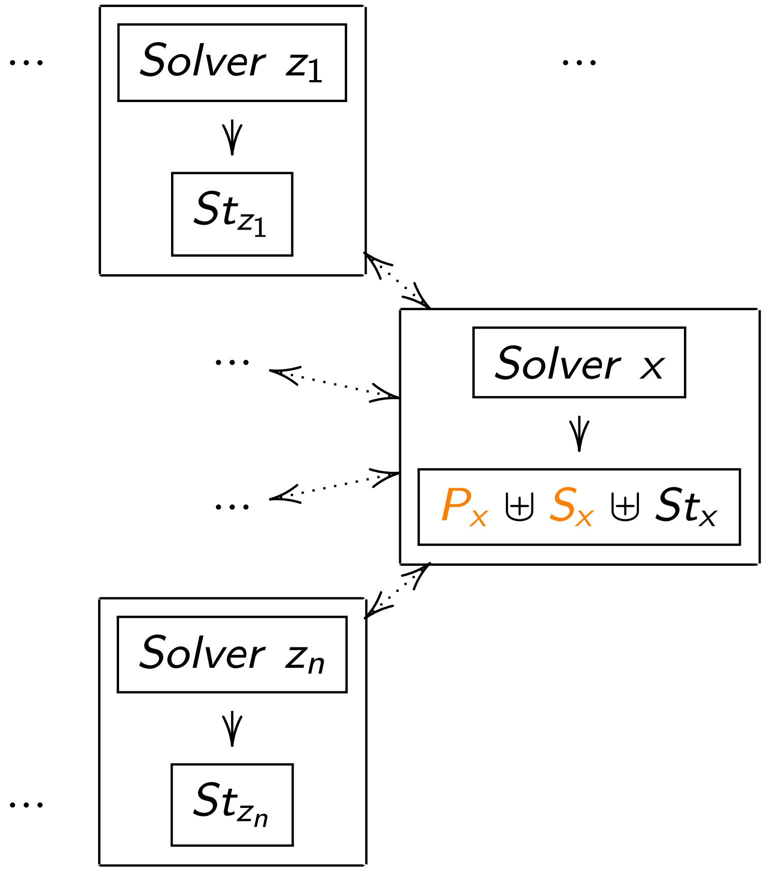
Outline

Executing 0-Neighbor Restricted Rules

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

$$r \; @ \; [X]P_x \backslash [X]S_x \Longleftrightarrow G \mid \boxed{[X]B_x, [Z_i]B_{z_i}, \exists D.[D_j]B_{d_j}} \qquad \textit{where } Z_i \in FV(P_x \uplus S_x) \textit{ and } D_j \in D$$

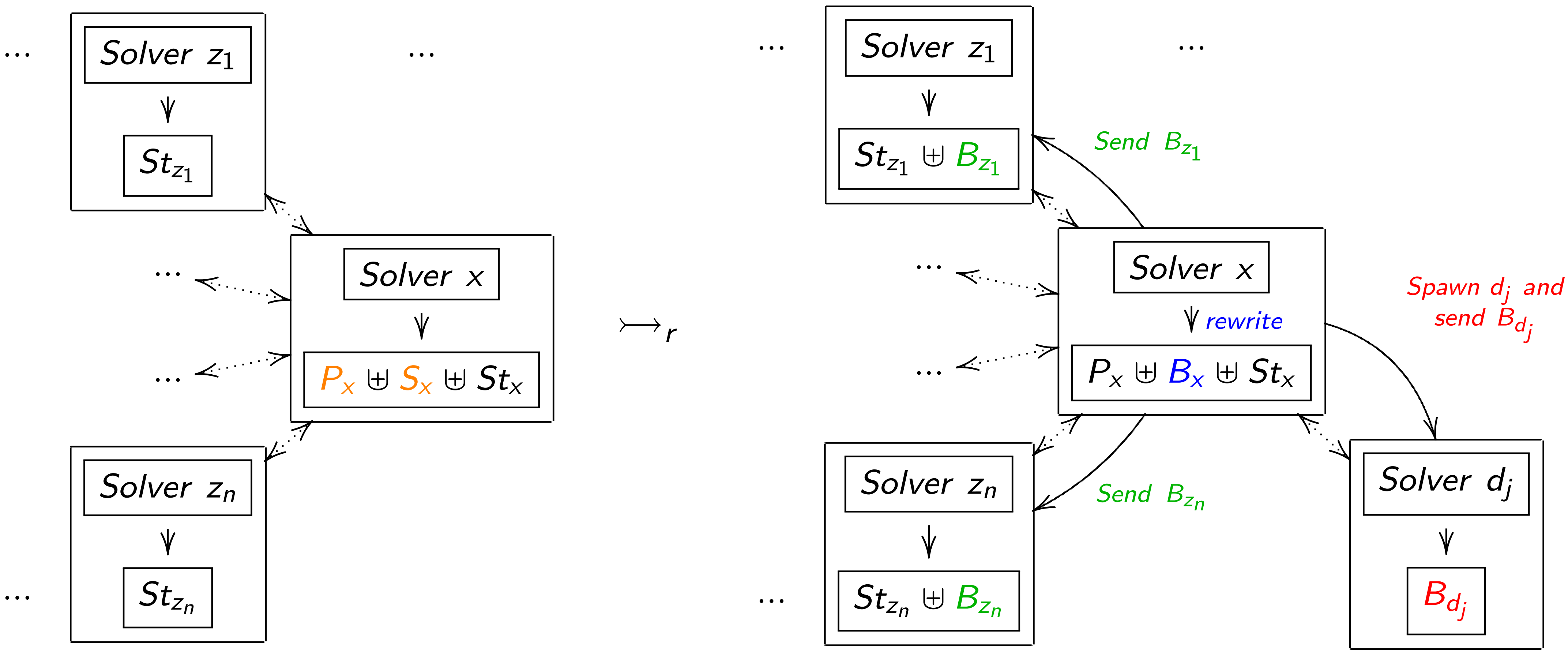
What happens when we apply r?



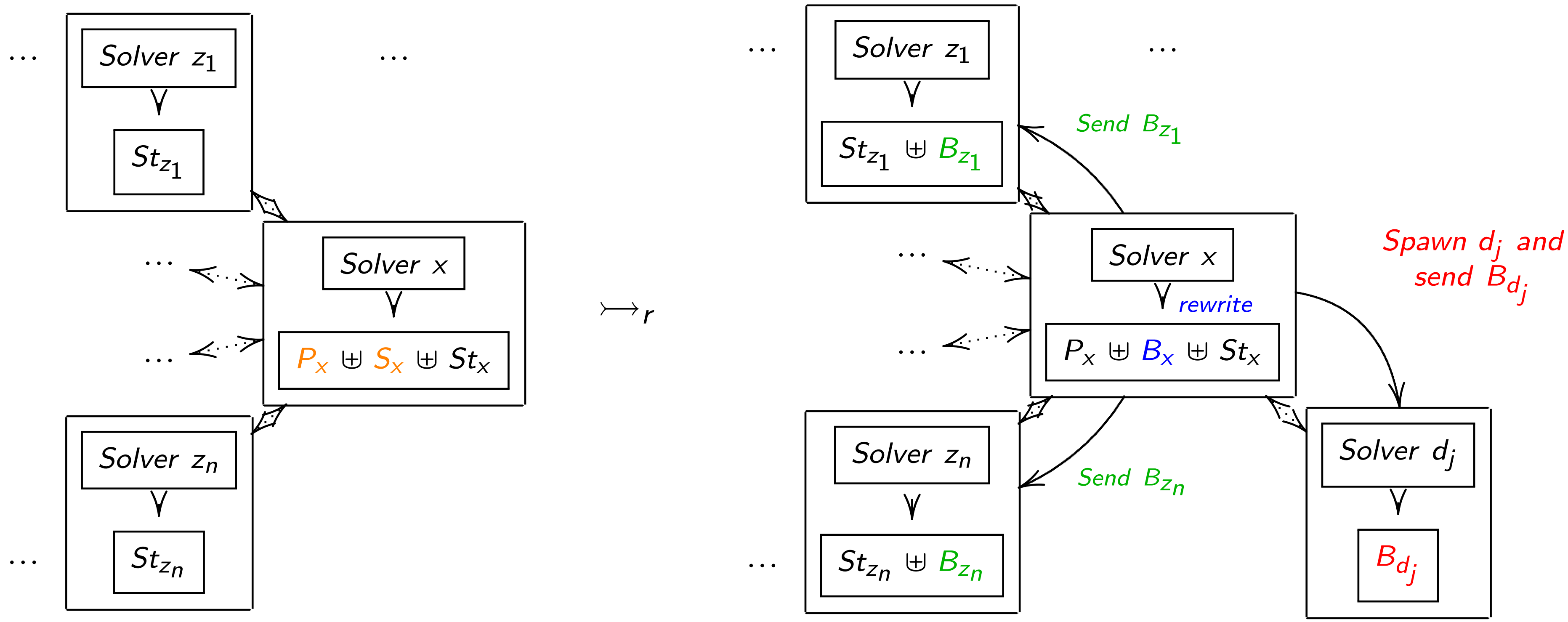
Execution 0-Neighbor Restricted Rules

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

$$r \;@\; [X]P_x \backslash [X]S_x \Longleftrightarrow G \mid \boxed{[X]B_x, [Z_i]B_{z_i}, \exists D.[D_j]B_{d_j}} \qquad \textit{where } Z_i \in FV(P_x \uplus S_x) \textit{ and } D_j \in D$$



The ω_0^e Operational Semantics



- *Decentralized* and *asynchronous* local CHR solvers, specifically ω_r [?].
- All 'send' and 'receive' are asynchronous, thanks to *monotonicity* of CHR! (i.e. if $St \rightarrowtail^* St'$ then $(St \uplus St'') \rightarrowtail^* (St' \uplus St'')$)
- Please see paper for technical details and results.

Executing 1-Neighbor Restricted Rules

$$swap\ @\ \underbrace{[X]neighbor(Y)\ \backslash\ [X]color(C1),}_{\text{primary matching obligation}}\ \underbrace{[Y]color(C2)}_{\text{neighbor matching obligation}} \iff \underbrace{[Y]color(C1)}_{\text{neighbor body constraints}}\ ,\ \underbrace{[X]color(C2)}_{\text{primary body constraints}}$$

$$\boxed{St_x \uplus \{n(y), c(pink)\}}^x \qquad \boxed{St_x \uplus \{n(y), c(brown)\}}^x$$

match $\qquad \rightarrowtail_{swap} \qquad$ add body constraints

$$\boxed{St_y \uplus \{n(x), c(brown)\}}^y \qquad \boxed{St_y \uplus \{n(x), c(pink)\}}^y$$

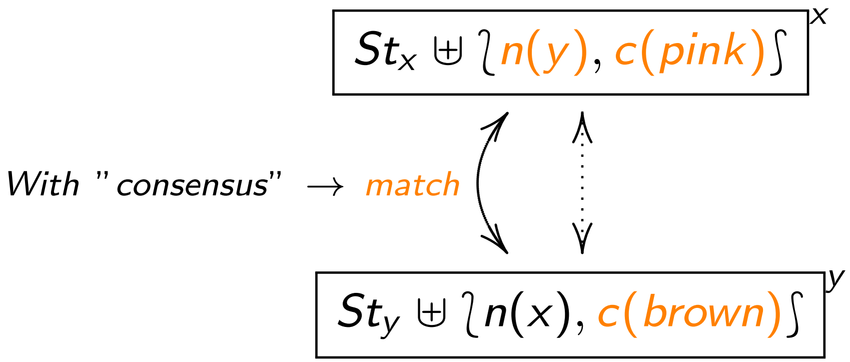
We need to deal with competing (over-lapping) rule applications!

$$\ldots \leftrightarrow \boxed{St_x \uplus \{n(y), c(pink)\}}^x \leftrightarrow \ldots$$

$$\updownarrow$$

$$\ldots \leftrightarrow \boxed{St_y \uplus \{n(x), c(brown)\}}^y \leftrightarrow \ldots$$

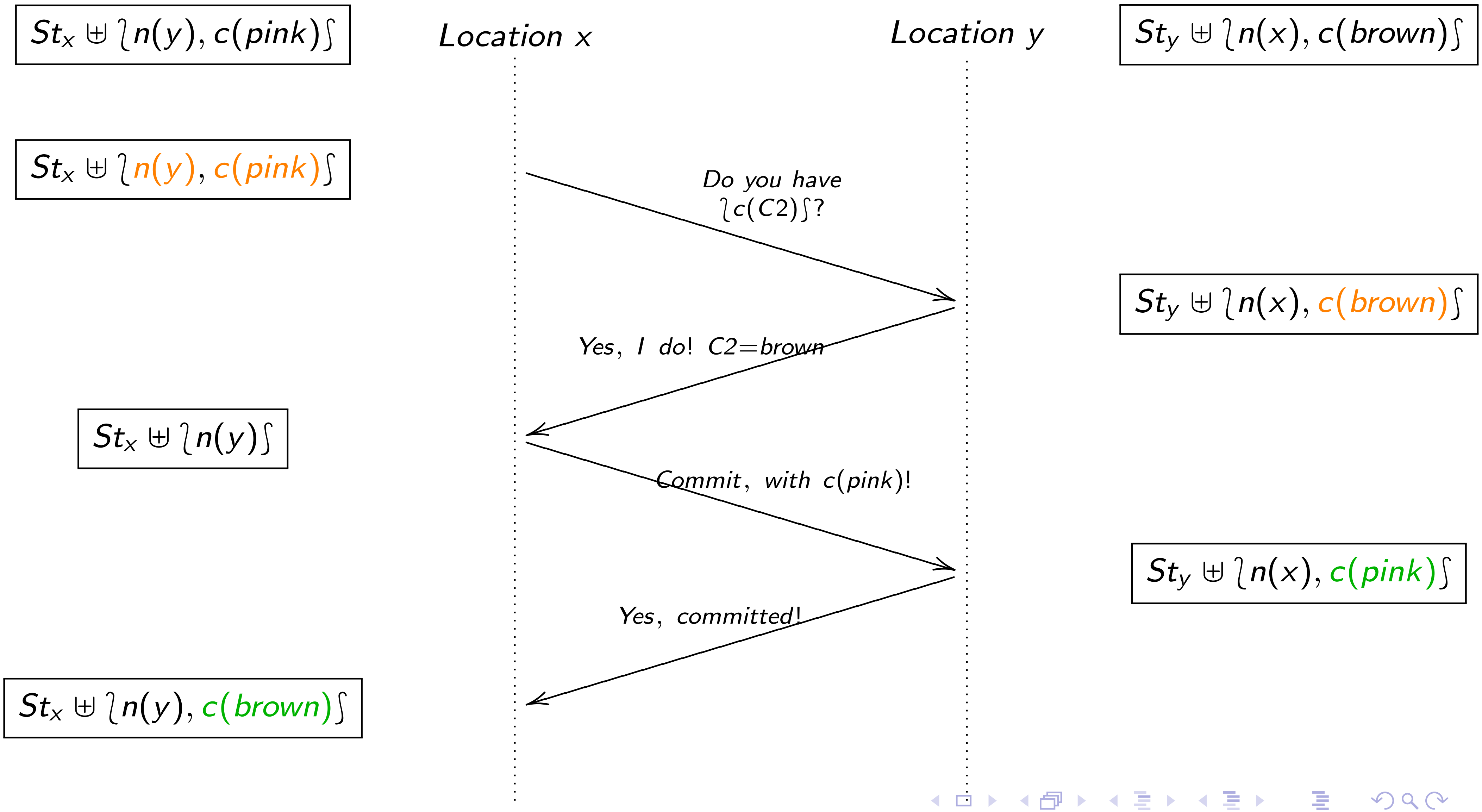
Executing 1-Neighbor Restricted Rules



- 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

$$swap \ @ \ [X]neighbor(Y) \setminus [X]color(C1), [Y]color(C2) \iff [Y]color(C1), [X]color(C2)$$



Implementing Consensus in ω_0^e

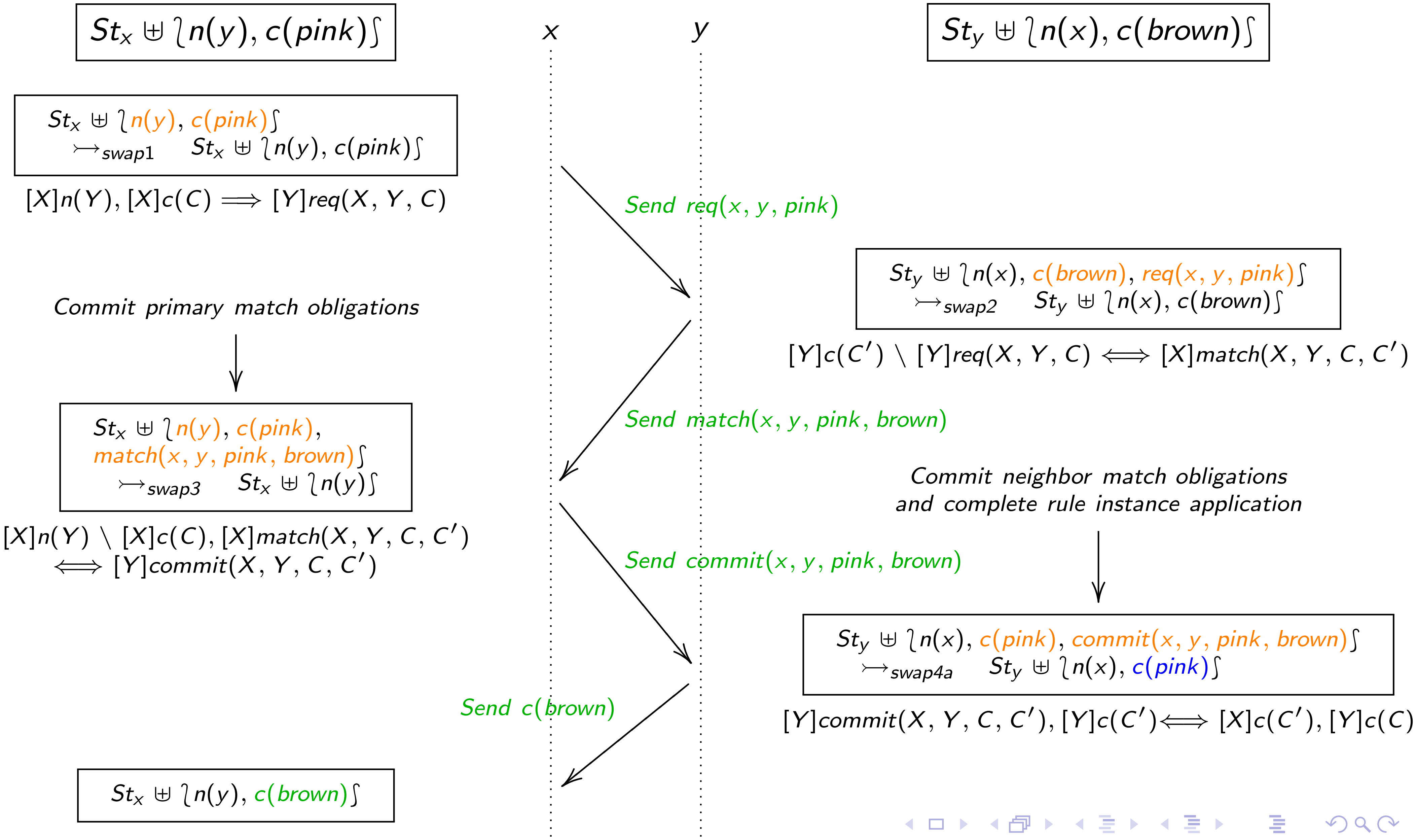
- We implement this consensus in ω_0^e, with source-to-source encoding $\leadsto^{\text{basic}}_{1\text{Nb}}$:

$$swap \; @ \; [X]neighbor(Y) \setminus [X]color(C1), [Y]color(C2) \iff [Y]color(C1), [X]color(C2)$$

$$\leadsto^{\text{basic}}_{1\text{Nb}} \left(\begin{array}{l} swap1 \; @ \; [X]neighbor(Y), [X]color(C) \Longrightarrow \underline{[Y]req(X, Y, C)} \\ swap2 \; @ \; [Y]color(C') \setminus \underline{[Y]req(X, Y, C)} \iff \underline{[X]match(X, Y, C, C')} \\ swap3 \; @ \; [X]neighbor(Y) \setminus [X]color(C), \underline{[X]match(X, Y, C, C')} \iff \underline{[Y]commit(X, Y, C, C')} \\ swap4a \; @ \; \underline{[Y]commit(X, Y, C, C')}, [Y]color(C') \iff [X]color(C'), [Y]color(C) \\ swap4b \; @ \; \underline{[Y]commit(X, Y, C, C')} \iff [X]color(C) \end{array} \right)$$

- *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 ω_0^e



Implementing Consensus in ω_0^e

x y

$St_x \uplus \{\!|n(y), c(pink)|\!\}$

$St_y \uplus \{\!|n(x), c(brown)|\!\}$

$St_x \uplus \{\!|n(y), c(pink)|\!\} \rightarrowtail_{swap1} St_x \uplus \{\!|n(y), c(pink)|\!\}$

$[X]n(Y), [X]c(C) \Longrightarrow [Y]req(X, Y, C)$

Send req(x, y, pink)

$St_y \uplus \{\!|n(x), c(brown), req(x, y, pink)|\!\} \rightarrowtail_{swap2} St_y \uplus \{\!|n(x), c(brown)|\!\}$

$[Y]c(C') \setminus [Y]req(X, Y, C) \Longleftrightarrow [X]match(X, Y, C, C')$

Send match(x, y, pink, brown)

Commit primary match obligations

$St_x \uplus \{\!|n(y), c(pink), match(x, y, pink, brown)|\!\} \rightarrowtail_{swap3} St_x \uplus \{\!|n(y)|\!\}$

$[X]n(Y) \setminus [X]c(C), [X]match(X, Y, C, C') \Longleftrightarrow [Y]commit(X, Y, C, C')$

$St_y \uplus \{\!|n(x), c(brown)|\!\} \rightarrowtail^* St'_y \uplus \{\!|n(x)|\!\}$ ← *Interleaving derivations*

Send commit(x, y, pink, brown)

$St'_y \uplus \{\!|n(x), commit(x, y, pink, brown)|\!\} \rightarrowtail_{swap4b} St'_y \uplus \{\!|n(x)|\!\}$

$[Y]commit(X, Y, C, C') \Longleftrightarrow [X]c(C)$

Roll back

Send c(pink)

$St_x \uplus \{\!|n(y), c(pink)|\!\}$

Optimizations and Soundness

- $\rightsquigarrow^{\text{basic}}_{1\text{Nb}}$ is the general encoding scheme.
- We can use optimized encoding schemes if primary or neighbor match obligation is known to be persistent. E.g.:

$$trans \ @ \ \underline{[X]edge(Y, D1)}, [Y]path(Z, D2) \Longrightarrow X \neq Z \mid [X]path(Z, D1 + D2)$$

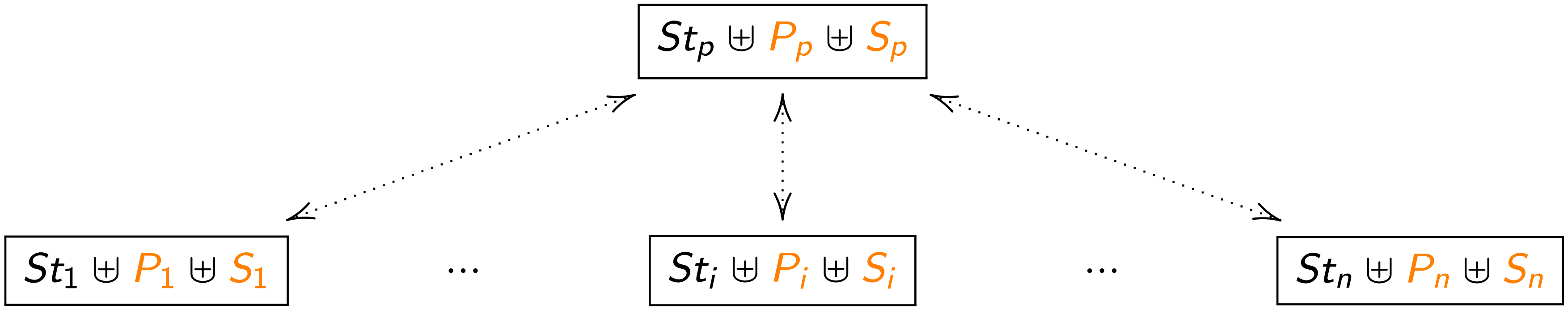
$$\rightsquigarrow^{\text{p-persist}}_{1\text{Nb}}$$

$$\left(\begin{array}{l} trans1 \ @ \ [X]edge(Y, D1) \Longrightarrow [Y]req(X, Y, D1) \\ trans2 \ @ \ [Y]req(X, Y, D1), [Y]path(Z, D2) \Longrightarrow X \neq Z \mid [X]path(Z, D1 + D2) \end{array} \right)$$

- $\rightsquigarrow^{p-persist}_{1\text{Nb}}$ is similar to link-restriction encoding in [**?**].
- We show soundness for composite encoding ($\rightsquigarrow_{1\text{Nb}}$):
  - Given $\mathcal{P}_1 \rightsquigarrow_{1\text{Nb}} \mathcal{P}_0$, derivations on $\mathcal{P}_0$ has a correspondence to derivations on $\mathcal{P}_1$.
  - See paper for details!

What about *n*-Neighbor Restriction?

- r @ $P_p, P_1, .., P_i, ..P_n \setminus S_p, S_1, .., S_i, ..S_n \iff G \mid B$ where location p is the *primary* and each i is the i^{th} *neighbor*.



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

Outline

Current Status

- Contributions, in this paper:
  - *n*-Neighbor Restriction
  - ω_0^e operational semantics for decentralized execution of CHR^e
  - Source-to-source encoding $\rightsquigarrow_{\mathrm{1Nb}}$
- See the paper for technical details!
- See technical report [?] for proofs and more technical details (e.g., $\rightsquigarrow_{\mathrm{nNb}}$)
- Prototype implementation in Python:
  - Available at GitHub, *https : //github.com/sllam/msre-py*
  - Built on top of MPI libraries
  - Implements ω_0^e with $\rightsquigarrow_{\mathrm{1Nb}}$.
  - 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!